universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Permutation Feature Importance for Correlated Variables: A simulation study

verfasst von | submitted by

Severin Ableidinger BSc BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 840
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Psychologie
Betreut von | Supervisor:	Univ.-Prof. Dr. Frank Scharnowski MSc
Mitbetreut von | Co-Supervisor:	Dipl.-Ing. Dr.techn. David Steyrl

Contents

Abstract

There has been an increased use of Machine Learning (ML) practices in social science as well as other research domains. Many of the now used ML-models, however, are so called "black box models". For these, it is unclear how the model makes predictions, and it is difficult to single out the effect of any single variable, which would be important for model assessment and research. Various methods have been thought up to estimate the effect of single variables in black box models, one of which is permutation feature importance (PFI). This method, however, has been criticized to be biased in correlated variables. In this simulation study, the behavior of PFI was assessed in the context of uncorrelated and correlated predictor variables. Using a variance-covariance matrix three variables were created, one of which was used as the outcome variable and two as the predictor variables. The covariances between the three variables was systematically varied. It was found that in uncorrelated predictor variables, PFI accurately captured the underlying correlation between the predictor variable and the outcome and even in cases when the predictors were correlated and have an equal correlation to the outcome, PFI scores were predictable from the underlying correlation structure. PFI might even be able to accurately assess the underlying correlation when the model fails to grasp it. But in other cases, PFI was biased in correlated predictor variables, both under- and overestimating the underlying connection. There was further no clear connection between the bias and the magnitude of the correlation, as the same underlying data structure led to different estimations of PFI scores. Overall, the results show that PFI should not be used when predictor variables are correlated.

Zusammenfassung

In der Psychologie sowie in anderen wissenschaftlichen Disziplinen werden vermehrt Machine Learning Methoden verwendet. Bei vielen dieser Modelle handelt es sich um "black box models", bei welchen nicht nachvollziehbar ist, wie das Modell Vorhersagen trifft und welchen Einfluss einzelne Variablen auf das Modellverhalten haben. Dies ist relevant, um ein Modell bewerten zu können, aber auch um daraus Schlussfolgerungen ziehen zu können. Daher wurden verschiedene Methoden eingeführt, um auch bei diesen Modellen den Effekt einzelner Variablen abschätzen zu können, eine davon ist die Permutation Feature Importance (PFI). Frühere Studien zeigten allerdings, dass PFI bei korrelierten Variablen verzerrte Werte produziert. In der vorliegenden Arbeit wird mittels Simulationen das Verhalten von PFI bei unkorrelierten und korrelierten Variablen untersucht. Mittels Varianz-Kovarianz-Matrizen wurden 3 Variablen erstellt, eine Zielvariable sowie zwei Prädiktoren. Die Kovarianzen der drei Variablen wurden systematisch variiert. Bei unkorrelierten Prädiktoren gab PFI eine unverzerrte Einschätzung der Korrelationen zwischen den Prädiktoren und der Zielvariable. Auch bei korrelierten Prädiktoren konnte der PFI-Wert gut vorhergesagt werden, wenn die Prädiktoren gleichermaßen mit der Zielvariable korrelierten. Sogar wenn ein Modell nicht die zugrunde liegende Datenstruktur erfasste, konnte PFI die Korrelation abschätzen. In anderen Fällen waren die resultierenden PFI-Werte allerdings verzerrt und die zugrundeliegende Korrelation wurde unter- oder überschätzt. In diesen Fällen konnte kein systematischer Zusammenhang zwischen der Korrelation zwischen den Prädiktoren und dem Bias gefunden werden. Die Ergebnisse zeigen, dass PFI für die Verwendung bei korrelierten unabhängigen Variablen nicht geeignet ist.

1 Introduction

In social sciences, there has been an increase in the usage of new statistical methods (Rosenbusch, Soldner, Evans, & Zeelenberg, 2021), often outperforming traditional models in terms of accuracy and predictive value (Hindman, 2015). These methods have been summarized under the terms machine learning, data mining, statistical learning, applied mathematics, or data science (Hindman, 2015). Not only in social sciences, but also in other scientific disciplines, for example in chemistry (Shi et al., 2023), physics (Rodrigues, 2023), medicine (Sidey-Gibbons & Sidey-Gibbons, 2019), and genetics and genomics (Libbrecht & Noble, 2015; Alharbi & Rashid, 2022), the use of machine learning (ML) methods is increasing. One advantage of these methods is that they can tackle highly complex and high-dimensional data, for which traditional models are not well adopted (Verikas, Gelzinis, & Bacauskiene, 2011). The term "curse of dimensionality", as coined by Bellman (1966), is often used to describe that small increases in the number of predictor variables might lead to a large increase in needed sample size. In such high-dimensional cases, and in datasets, where there are more variables included than observations, the problem is undetermined and traditional statistical methods, like ordinary least squares (OLS) regression, fail to produce meaningful results, but some ML models can still be used (Hastie, Tibshirani, & Friedman, 2009). For many models, multicollinearity is also a problem. Multicollinearity describes when two or multiple predictor variables are correlated, and can lead to certain problems. For example, in OLS regression, the problem is no longer solvable or leads to large variance estimates for the estimated coefficients (Hoerl & Kennard, 1970). Many ML models can, however, handle such multicollinearity.

1.1 How ML and Other Statistical Methods Differ

ML practices are rooted in fields outside of statistics, like computer science. Early application areas were topics, in which traditional data models were not applicable, such as computer vision or speech recognition. Due to this, ML methods have a different approach compared to traditional statistics. There are hardly any assumptions about the underlying data generating process. The process is seen as a complex black box, which cannot be known, but only approached and estimated. This, however, does not mean, that ML does not use statistical models, quite the opposite, just the underlying background and approach is often different (Breiman, 2001b). Since its early development, ML now is closer related to traditional statistics, as more theoretical properties of ML methods are proposed and many statisticians work on ML topics (Hooker & Mentch, 2021). ML methods often have a different objective from tradi-

tional statistical models. They fundamentally aim at prediction accuracy as a benchmark, whereas in traditional methods, model-fit-testing and p-values are often the focus of attention (Breiman, 2001b). In psychology, it is common to build models that should validate a proposed theory and the derived hypotheses (Yarkoni & Westfall, 2017). Statistical tests are most often used to show the significance of a predictor variable in explaining the outcome variable. This has some essential shortcomings. First, given a large enough data set, almost any variable can be shown to have a significant effect, even if this effect is marginal (Combs, 2010; Lantz, 2013). Further, practices like "p-hacking" lead to an increase of false positive findings (Simmons, Nelson, & Simonsohn, 2011; Wicherts et al., 2016). Null-findings are often not published, especially journals with high-impact factors hardly accept such papers (Head, Holman, Lanfear, Kahn, & Jennions, 2015). This incentivizes p-hacking and selective reporting of found results. Such practices can be done with relative ease, as researchers can make many different decisions in the process of data generation and statistical analyses, which can influence the outcome of a test. This multitude of decisions has been called the "researcher degrees of freedom" (Wicherts et al., 2016). False positive findings are detrimental for a research field, as they are often hard to debunk, and can take up lots of resources funnelled into baseless research topics (Simmons et al., 2011). It has even been argued that most findings are in fact false (Ioannidis, 2005). Further, just because the data shows a significant relationship, the effect of this relationship might be negligible (Combs, 2010; Lantz, 2013). A significant predictor variable may explain almost nothing of the variance of the outcome variable. These problems have led to a call for the increased reporting of effect sizes (Lantz, 2013) as well as a rise in preregistration of planned studies (Wicherts et al., 2016). While this is certainly an improvement in transparency, some authors have argued that even in such practices similar problems might remain (Head et al., 2015). In ML, not significant results are the main goal, but prediction accuracy. For a model to be considered useful, it is generally not important for its components to reach significance, but for it to be an accurate predictor for future observations. ML models are often very complex and can learn very complex relationships to approach the data distribution. However, the models can be too complex and overfit the data (Hawkins, 2004). Overfitting occurs, when the model performs well on the training data, but performs poorly on new, previously unseen data (Ying, 2019). During the learning phase, when the model is trained on the training data, it may learn the structure of the observations almost perfectly, but it does not generalize well. To account for this the data is split into a training and a test set, called a train-test-split, and the test set used to estimate the models performance to check for overfitting. There are various reasons for overfitting:

The model might be too flexible, which might lead to the model learning random noise that is not part of the underlying data-structure. Also, the model might include too many predictors. In practice, all models do overfit, so using a test or evaluation set is important to adequately infer the performance of a model (Ying, 2019).

Various different approaches have been implemented to combat overfitting. Regularization is often used and includes various different techniques. For models trained over multiple iterations, one approach is to stop this process early on, so the model does not fit the training data too well. For tree-based models, pruning is often performed, which restricts the complexity of each tree. For linear models, Ridge (Hoerl & Kennard, 1970) and lasso regression (Tibshirani, 1996) are often used, which shrink the coefficients of the model.

1.2 Ridge Regression

Ridge regression (RR), was first introduced by Hoerl and Kennard (1970). It was proposed in order to handle within-predictor correlations. In normal OLS, when within-predictor correlations in the observed data X are present, the matrix X^TX is no longer invertible, making OLS impossible. Even if the matrix is still invertible, the results are unstable, and vary significantly with small changes in the data. By adding a penalty term λ multiplied by the identity matrix to the matrix X^TX, the matrix becomes invertible. The Ridge estimator is then given by the formula

$$\hat{\beta}_{\text{Ridge}} = (X^TX + \lambda I)^{-1}X^Ty. \tag{1}$$

Another way to view RR, is to think of λ as a shrinkage parameter for the coefficients. This is apparent, when writing Ridge as a minimization problem:

$$\hat{\beta}_{\text{Ridge}} = \underset{\beta}{\text{argmin}} \left\{ (y - X\beta)^T(y - X\beta) + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}. \tag{2}$$

In the case of orthonormal variables X, meaning that they are not correlated, the Ridge estimators are a scaled version of the OLS estimators

$$\hat{\beta}_{\text{Ridge}} = \frac{\hat{\beta}_{\text{OLS}}}{1 + \lambda}. \tag{3}$$

From this it is obvious, that the larger the shrinkage parameter λ is, the smaller the estimated coefficients are.

RR has been used to predict personality factors (Schwartz et al., 2013), depression (Ghandeharioun et al., 2017), and heart disease mortality (Eichstaedt et al., 2015).

1.3 Training a ML Model

As mentioned above, to counter overfitting and make reasonable estimates of the performance of a model, the data is often split into a training and a test-set. This split might significantly reduce the data available for training the model. In cases where not a lot of data is available, this may not be desirable. Further, due to the randomness of this split, the model might randomly perform poorly or extremely well on the test set, while the test set might not be representative for future observations. Because of that, this splitting procedure has been extended to k-fold cross-validation. The dataset is split into k groups of approximately equal size. Sequentially, each group is excluded once from the training set and used as the test set. The model is trained on the k-1 remaining groups and the desired metric is estimated on the excluded test-set. After each set has been excluded once, the estimated metric is averaged. While this approach reduces the randomness induced by the random splitting into train and test set, it can be computationally very expensive.

For many ML models, so called hyperparameters have to be given to the model. Most parameters are set by the model itself during the training process. These include such things as the size of the coefficients in regression models or the split chosen for decision trees and are learned from the data. But there are also hyperparameters, which have to be given to the model and influence the training process. These might be the penalty term in shrinkage regression models, such as lasso and RR, or the minimum amount of samples required to make a split in a decision tree, which control the regularization of the models, but can also be the specific kernel used in support vector machine. While there are some default hyperparameters given for most models, these might not be the best for each dataset. So, hyperparameters have to be tuned. This means finding the optimal setting for each hyperparameter in regard to a given dataset. However, there is no straightforward way to find this optimum, so there are many different approaches. One such approach is grid search. For each hyperparameter, a list of possible values is created. These values are then combined into an exhaustive grid, containing each possible combination. A model is trained using each possible combination and tested. The combination with the best performance is then chosen as the final hyperparameters. To prevent overfitting, this is done using a training and test set, or k-fold cross-validation. A major problem with this approach is that it is not efficient. Given the number of hyperparameters to be tuned and the extent of possible values for each, the grid can become excessively large, resulting in a lot of time being spent in this process. Further, it is often not clear how the list of possible values for the hyperparameters should be defined. There is hardly any empirical research done or suggestions given, so this

is mostly resolved by the user's experience or examples from other practitioners. Another approach is called random search. In this sets of hyperparameters are randomly drawn from a specified distribution. While this gives the user less control over which combinations of hyperparameters are tried, it can be more efficient than grid search while still achieving good performance. There are also other algorithms for optimizing hyperparameters, like Bayesian Optimization (Snoek, Larochelle, & Adams, 2012), Particle Swarm Optimization (Chuan & Quanyuan, 2007), and Genetic Algorithm (Wicaksono & Supianto, 2018), which also have their own advantages and disadvantages.

Hyperparameters have to be set for most ML models, such as RR, Random Forests, neural networks, or support vector machines.

1.4 Random Forest

Random Forest (RF) is a powerful and widely used model in ML introduced by Breiman (2001a). It is an ensemble of tree predictors and can be used both for classification and for regression. The trees are inspired by the Classification And Regression Tree (CART) methodology. These trees are naturally quite unstable, so bagging (bootstrap aggregating) is used in order to reduce the variance of a single tree, meaning that for training each tree, a bootstrap sample is made (Archer & Kimes, 2008). Individual learners, such as these trees, are often referred to as "weak learners", meaning they show low bias, but high variance (Verikas et al., 2011). Averaging between many weak learners can reduce the variance depending on the correlation between the weak learners. By using a random subset of variables at each node for the split, the correlation between the trees is decreased. For each variable, the best split, which reduces the Gini index the most, is calculated and the overall best split is then used in the node. Each tree is fully grown and not pruned (Altmann, Toloşi, Sander, & Lengauer, 2010). An advantage of choosing a random subset of variables for each split is that this enables RF to also include variables into the ensemble, that would otherwise be outperformed by other variables. This should allow for interaction effects to be picked up by RF, as each split depends on the splits above in the same tree-branch (Strobl, Boulesteix, Kneib, Augustin, & Zeileis, 2008). The algorithm of RF can be depicted as following:

1. From the original observations, B bootstrap samples $\widetilde{X}_b, b = 1, \ldots, B$ are generated.

2. Each tree is built with one of the samples $\widetilde{X}_b$.

3. To build a tree, at each node, a random subset m of the p predictor variables is selected. For each of those the best split is calculated and the

overall best split selected and used at that node.

4. Trees are fully grown without pruning, however, certain hyperparameters can be chosen, for example to fix how many samples must remain at each ending node (Archer & Kimes, 2008).

To make a new prediction the observation is passed down each tree, and each tree makes a prediction. In classification, a majority vote is done to predict the class, while in regression, the predictions are averaged. An advantage of RF is that the test error can be estimated from the out-of-bag (OOB) data, which is the data, which was not used on a tree to grow this tree.

The good performance of RF has been shown several times (Archer & Kimes, 2008), and it is often applied to select variables (Gregorutti, Michel, & Saint-Pierre, 2015). It is relatively robust against outliers and noise (Breiman, 2001a). RF has been used to predict romantic attraction (Joel, Eastwick, & Finkel, 2017), choice prediction (Plonsky, Erev, Hazan, & Tennenholtz, 2017), and suicide attempts (Walsh, Ribeiro, & Franklin, 2017).

Breiman (2001a) simultaneously introduced two measures for the importance of each variable for the prediction: the Permutation Feature Importance (PFI), and the gini importance, based on the gini index. Research has shown that RF are biased, preferring categorical variables with a large number of categories (Altmann et al., 2010) or in general variables with more possible split points (Strobl, Boulesteix, & Augustin, 2007). In continuous variables the number of possible splits can be up to $(n-1)$, with n being the number of observations, if each observation is different, while for categorical variables the number of possible splits can be $(2^{m-1}-1)$ where m is the number of categories. Similarly to multiple testing, when a variable has more possible split points, it is very likely to have at least one good performing split, simply by chance (Loh, 2014). The bias of RF then carries over to the assessment of variable importance via PFI or the gini index (Altmann et al., 2010; Strobl, Boulesteix, Zeileis, & Hothorn, 2007). To counter this, a different form of RF based on conditional inference trees was introduced (Hothorn, Hornik, & Zeileis, 2006; Strobl, Boulesteix, Zeileis, & Hothorn, 2007). RF were also shown to be biased in favor of variables with equally sized categories over unequally sized categories (Janitza, Strobl, & Boulesteix, 2013). Gini index also shows this bias (Boulesteix, Bender, Lorenzo Bermejo, & Strobl, 2011). Further, it has been argued that RF has a preference for selecting correlated variables in their tree building process (Strobl et al., 2008), but other simulation studies found that uncorrelated variables are preferred when choosing the split variable (Nicodemus & Malley, 2009). Gini index was also shown to be biased when predictor variables are correlated (Nicodemus & Malley, 2009). Compared to the Gini index,

PFI was argued to be more robust and stable when categories are unequally sized, when they vary in their scale of measurement, or under within-predictor correlation (Nicodemus & Malley, 2009; Nicodemus, 2011).

RF as well as other often used models, such as neural networks and support vector machines, are so called "black-box-models". For these, it is unclear how the models make predictions (Hassija et al., 2024; Molnar, 2025). These models do not have a straightforward way to examine how individual variables are used in making a prediction, as some traditional models do. In methods like linear and logistic regression the coefficient for each variable could be examined. Examining coefficients also can be done in some less traditional models, like lasso and RR, but black-box models, like RF or support vector machines, often outperform these linear models. Especially, when relationships between the predictor variables and the outcome variables are not linear, these models are more appropriate. For many purposes, however, it is essential to estimate the impact of each variable on the predictions made by a model. This is important in auditing the model, to see if there are obvious flaws and if the model could possibly be improved, and to ensure that the model adheres to regulatory guidelines and rules and to ensure the fairness of a model. It is also important in inference and theory building, which is crucial for research in general and especially in social sciences (Yarkoni & Westfall, 2017; Altmann et al., 2010). Inference is often done in the form of building models and examining their performance as well as the individual variables and their importance for the built model. While one might use the overall performance of the model, one might also be interested in the effect of one specific variable. For these purposes, it is therefore important to be able to single out the effect of one particular variable.

1.5 Variable Selection

Inspecting the importance of each variable is also necessary for variable selection. Variable selection is mainly used for two different purposes: building powerful predictive models and identifying all important connected variables. In many cases, including too many variables, which do not carry information useable for prediction, might hinder model performance, so these should be excluded from the built model (Hawkins, 2004; Gregorutti, Michel, & Saint-Pierre, 2017). The term "Occam's Razor" is often used to describe the rationale, that simpler models should be preferred, using an appropriate minimal number of variables, while still achieving good predictive performance (Breiman, 2001b). Contrary to this guideline, it was emphasized that including more variables naturally increases the information provided to the model (Breiman, 2001b) and it has been shown that including randomly generated variables in a model can boost performance

(Mentch & Zhou, 2022). Note that this is also a form of regularization which is used to limit overfitting and enhance generalization.

Three different approaches for variable selection have been described in literature: Filter, embedded, and wrapper methods (Pascoal, Oliveira, Pacheco, & Valadas, 2017; Blum & Langley, 1997).

Filter methods are implemented before a model is built. Using univariate methods, e.g. correlation coefficients and mutual information, important predictor variables are selected and then used to build a model (Blum & Langley, 1997; Chowdhury & Turin, 2020; Pascoal et al., 2017). These types of variable selection criteria do not consider interactions between various predictor variables and it has been argued that it should be avoided (Heinze, Wallisch, & Dunkler, 2018).

Embedded methods are implemented in the model building process. Algorithms like Lasso or decision trees only include a subset of the variables into their prediction process. Decision trees do so by selecting only useful variables at each split, while Lasso sets the coefficients of some variables to zero by using a penalty term (Tibshirani, 1996; Hastie et al., 2009).

Wrapper methods are mostly iterated methods, where one tries to approach the optimal set of variables, optimizing a chosen criteria. Examples for this are best subset selection, as well as forward and backward stepwise selection, which either try to optimize the error or some other criteria, like the AIC (Gregorutti et al., 2017; Blum & Langley, 1997; Guyon & Elisseeff, 2003). Best subset selects a subset of a smaller number of variables to fit a model. This, however, can get computationally expensive very fast. For example, choosing 6 predictors out of 15 possible variables already gives a total amount of 5005 possible subsets. Choosing 10 predictors out of 120 possible variables results in over 116 trillion possible subsets. If we checked 1000 subsets each second, it would still take over 35 centuries in order to check each possible subset. Another approach is to at first build a model and then estimate, how important each feature is for this model and choose variables based on this importance. In black box models, this is especially difficult, as there is no clear possible way to estimate the influence of one variable (Molnar, 2025). To solve this problem, various methods have been introduced to calculate the effect of each variable on the model performance and predictions. In RF, the gini index is often used. There are also so called model agnostic methods, which can be used for any model. These methods include Sobol Indices, Partial Dependence Plots, Shapley Values, and PFI, as well as many others (Molnar, 2025).

An issue with all of these approaches for variable selection is instability, as small perturbations in the training sample might change which variables are selected. It has been shown that this instability is increased with higher di-

mensions and correlated predictors (Gregorutti et al., 2017; Tolosi & Lengauer, 2011). Bootstrap sampling methods are often performed to increase and measure stability. This is connected to the "Rashomon effect", which is the occurrence that multiple different models using different variables can perform equally well. In this case, various different sets of variables can be selected and models using these different sets can still perform similarly well. This makes it especially difficult to infer and estimate which variables are of importance and which are not.

1.6 Variable Importance Measures

Methods that try to estimate how important a variable is for a given model are often called variable importance measures (VIM). A broader movement to tackle variable importance and interpretability in ML and AI is explainable AI (XAI). Goals of XAI methods and VIM are to audit a model and debug it, as well as to learn how a model makes a specific decision (Ribeiro, Singh, & Guestrin, 2016; Freiesleben & König, 2023). Another goal is to learn something about the modeled phenomenon, make statistical interference and try to find "ground truth" about the underlying data generating process (Molnar, Freiesleben, et al., 2023; Freiesleben & König, 2023). The notion of ground truth is, however, debated. In an influential paper, Breiman (2001b) argued against focusing on the underlying data generation process, as it is arguably unknowable. One can only approach the behavior of it, using predictive accuracy to validate models as appropriate. This skepticism of finding "truth" in statistical modelling has been shared by other researchers (Heinze et al., 2018). Trying to only make estimations about the model itself, without the notion of ground truth, also has some drawbacks as it is often difficult to isolate the effect of a single variable in a given model. Many methods retrain the model after deleting or altering one variable, but it can be argued that they compare different models, which might not be reflective of the importance of a variable for either of the models. There has been further substantial criticism that the goals in XAI are often not well-defined and many explanation models are presented without a specific purpose, or clear application (Freiesleben & König, 2023). Similarly, it was criticized that it is often unclear what VIM actually measure, what they mean theoretically, as well as how they should behave (Debeer & Strobl, 2020). Some authors have, however, tried to formulate some requirements for VIM. Tolosi and Lengauer (2011) stated that VIM should reflect the effect of a variable on the outcome and should not depend on the size of the groups of correlated variables. It has also been shown that various methods aiming to provide explanations can be misled and be fooled into giving any arbitrary explanation (Lakkaraju & Bastani, 2020;

Kindermans et al., 2019; Slack, Hilgard, Jia, Singh, & Lakkaraju, 2020).

In this paper, the behavior of PFI will be further explored, in particular in connection with RF and RR.

1.7 Permutation Feature Importance

This method was first introduced for RF (Breiman, 2001a), but has since been used as a model agnostic tool, meaning that it can be used for any model (Fisher, Rudin, & Dominici, 2019; Hooker, Mentch, & Zhou, 2021; Molnar, 2025; Zacarias & Boström, 2013). The goal is to break up the association between the permuted variable and the outcome and then compare the prediction error (Gregorutti et al., 2015). A variable is randomly permuted, and the reduction in prediction accuracy or rise in prediction error is measured as an indicator for the importance of this specific variable. A large increase in prediction error means that the model performs much worse when the variable is permuted, so the variable seems to be important for the predictions of the model. While Breiman (2001a) called this method variable importance, others have called it mean decrease in accuracy (Chavent, Lacaille, Mourer, & Olteanu, 2021), or Permutation Feature Importance (PFI) (Molnar, 2025). Throughout the rest of this paper, PFI will be used.

The original algorithm as proposed by Breiman (2001a) can be summarized as following:

1. For each tree, the out-of-bag (OOB) observations are identified. OOB observations are observations not used in building a tree.

2. Each tree makes a prediction based on the OOB observations, and a loss function is calculated describing the performance, mostly accuracy in classification and mean squared error in regression.

3. For each predictor variable $x_j, j = 1, \ldots, p$, the values of this variable are randomly permuted. p denotes the number of predictor variables. Again, each tree makes a prediction based on the OOB data in which the jth but only the jth variable is permuted. The loss function is calculated and the two performance measures are subtracted from each other.

4. The average difference between these is calculated and estimates the variable importance measure for x_j (Archer & Kimes, 2008).

The OOB-PFI can therefore be defined as:

$$\hat{I}(x_j) = \frac{1}{M} \sum_{m=1}^{M} \left[\hat{L}(\hat{f}_m, X^{mj}_{\text{oob}}) - \hat{L}(\hat{f}_m, X^{m}_{\text{oob}}) \right], \tag{4}$$

where $\hat{I}(x_j)$ is the estimated PFI for the jth variable, M denotes the number of trees in the forest, $m = (1, \ldots, M)$ specifies a specific tree, $\hat{L}(a, b)$ is the calculated loss function, for the function a using dataset b, $\hat{f}_m$ denotes the model, in this case the specific trained tree, X^{mj}_{oob} is the OOB data for the mth tree, where the jth variable is randomly permuted, and X^{m}_{oob} the oob data for the mth tree. Using the OOB data, there is no need for a train-test-split, as all estimations can be done on them, which is especially in small data sets an advantage.

In the model-agnostic version, instead of using the OOB observation in each tree, the training set or the test set can be used and predictions are made for the whole model (Hooker et al., 2021). It is generally recommended to estimate the PFI score using the test set, to avoid elevated results due to overfitting (Molnar, 2025). The algorithm for the model-agnostic version can be described as following:

1. The model makes predictions on the train or test data set.
2. The loss function is calculated, again for classification mostly accuracy, and mean squared error for regression, but other measures are possible as well.
3. For each predictor variable $x_j, j = 1, \ldots, p$, the observations of this variable are randomly permuted. The model makes predictions for the dataset X^j, where the entries of the jth variable are randomly permuted. Again the loss function is calculated.
4. The difference between these two measures indicates the PFI score for the jth variable.

The PFI can be stated as:

$$\hat{I}(x_j) = L(\hat{f}, X^j) - L(\hat{f}, X), \tag{5}$$

where X^j is the dataset X, where the jth variable is randomly permuted. Again either the train or the test set can be used here, but for generalization, it is recommended to use the test data. To increase stability, the permutation and subsequent prediction and performance measure are performed and calculated multiple times, and the resulting difference is averaged (Wang, Yang, & Luo, 2016; Molnar et al., 2022).

Theoretically, PFI is often stated as the difference in the quadratic expected risk $\mathbb{E}[(Y - f(X))^2]$, and can be defined as:

$$I(x_j) := \mathbb{E}[(Y - f(X^j))^2] - \mathbb{E}[(Y - f(X))^2]. \tag{6}$$

It has been shown, that PFI should then be equal to

$$I(x_j) := 2\rho^2, \tag{7}$$

where ρ is the correlation between variable x_j and Y (Gregorutti et al., 2017). To include correlation between variables this formula was extended to:

$$I(x_j) = 2\left(\frac{\rho}{1+c}\right)^2, \tag{8}$$

where c is the correlation between x_j and another variable x_k. It is assumed that the correlation between x_k and Y is also ρ. In these cases, both variables x_j and x_k might show a lower PFI then another variable, even when this third variable shows a lower correlation to the outcome variable Y.

Since its introduction, PFI has been widely used (Díaz-Uriarte & de Andrés, 2006; Datta, Sen, & Zick, 2016), especially in variable selection(Gregorutti et al., 2017; Genuer, Poggi, & Tuleau-Malot, 2010), and its functionality was often shown (Archer & Kimes, 2008; Breiman, 2001b). It was emphasized that this approach can detect interactions, which is not possible in univariate scanning (Auret & Aldrich, 2011; Rodenburg et al., 2008; Strobl & Zeileis, 2008; Casalicchio, Molnar, & Bischl, 2019)

However, PFI measures have also been criticized. They have been shown to be unreliable when predictors vary in their scale of measurement or the number of categories (Strobl, Boulesteix, Zeileis, & Hothorn, 2007), or when variables are correlated (Strobl et al., 2008). Some have also argued that the error-rate-based PFI, for example, by using accuracy, is biased when classes are imbalanced (Janitza et al., 2013). It has both been shown that the importance of correlated variables can be overestimated (Strobl et al., 2008; Nicodemus, Malley, Strobl, & Ziegler, 2010) and underestimated (Gregorutti et al., 2017; Genuer et al., 2010; Tolosi & Lengauer, 2011) when using PFI. In one simulation study, correlated variables appeared more important in small sample sizes, but not in larger samples (Hooker et al., 2021). In the same study, linear models were found to return more accurate estimates, regardless of the correlation structure. Other researchers, however, have found no bias for PFI when variables are correlated (Nicodemus & Malley, 2009), and have argued that PFI is reliable, when predictors vary in their scale of measurement or number of categories (Nicodemus, 2011).

For underestimation, some researchers argue that correlated features probably share predictive value, possibly through a common underlying factor, and information that is normally lost due to permutation is covered by another variable, leading to lower PFI estimates (Tolosi & Lengauer, 2011; Pereira, Stroes,

Zwinderman, & Levin, 2022).

For overestimation, it has been argued that permuting the variable can lead to deviations in the joint distribution of the predictor variables, and that this might artificially increase the importance of the permuted variable (Strobl et al., 2008). Hooker and colleagues (Hooker, 2007; Hooker et al., 2021) argued that permuting features breaks the dependencies between features, which creates data in an unlikely or even impossible region of the feature space, forcing the model to extrapolate. ML models perform poorly when forced to extrapolate, explaining why this results in an unlikely high PFI score. Other XAI methods, like partial dependence plots, use a similar framework, in which variables get permuted and predictions are done on the permuted dataset. All these models are likely to over-emphasize correlated features. Therefore, it was argued to not use regular permutation, but to remove or permute a predictor variable and retrain the model, which can be computationally much more expensive than PFI (Hooker et al., 2021).

It was shown that PFI can be unstable due to data perturbations (Calle & Urrea, 2010). Wang et al. (2016) studied the stability of PFI. The number of variables per observation was found to be negatively correlated to the intrinsic stability. This was also shown by another study in which also was shown, that the variable importance generally decreases with increases in the number of variables (Genuer et al., 2010). Higher variance in PFI and variable selection was also shown, when a small number of trees in RF are used (Verikas et al., 2011).

It should be noted that in most of the studies mentioned above, the OOB version of PFI was used, with only Pereira et al. (2022) and Hooker et al. (2021) using the model agnostic version, with Hooker et al. (2021) even using both the model agnostic and the OOB version. The OOB version is closely related to the behavior of RF, and might therefore reflect behavior of the RF and not of the importance measure. It was emphasized that the model agnostic version and the OOB version are different, have different meaning, and lead to different conclusions (Bénard, Da Veiga, & Scornet, 2022).

Despite the above mentioned criticism, permutation importance is still widely used (Cappelli & Grimaldi, 2023; Wu et al., 2023; Verma, 2022; Zhang, Su, Altaf, Wik, & Gros, 2023; Hier, Do, & Obafemi-Ajayi, 2024; Azarmi, Rezaei, Wang, & Arabian, 2024; Card, Simpson, Aryal, Gupta, & Islam, 2024; Kucukosmanoglu, Garcia, Brooks, & Bansal, 2024).

After its initial introduction, different extensions and various alterations were created. Various researchers have expended PFI to include various variance estimates, confidence intervals and p-values for both the oob version (Altmann et al., 2010; Ishwaran & Lu, 2019) as well as for the model-agnostic version

(Hapfelmeier, Hornung, & Haller, 2023; Nembrini, 2018). Molnar, Freiesleben, et al. (2023) tried to establish some statistical properties, find bias-variance decompositions, and propose theorems of unbiasedness, standard estimators and confidence intervals. They also argued that additional variance in PFI comes from the learning process of the model. To overcome this, they introduce a learner PFI, which averages over several model fits for the same learning algorithm using different data samples. There have also been efforts to connect PFI to classical sensitivity analysis and to Sobol indices (Gregorutti, 2015; Bénard et al., 2022).

To counteract unreliability due to correlated data, researchers have proposed to conditionally permute variables, depending on other variables (Strobl et al., 2008). This method was computationally much more expensive than normal PFI (Antoniadis, Lambert-Lacroix, & Poggi, 2021), but has since been revised to reduce computation time (Debeer & Strobl, 2020) and has been extended to the model agnostic version of PFI (Molnar, König, Bischl, & Casalicchio, 2023). It was emphasized that conditional PFI is different from classical PFI, and answers a different question, and therefore should not be used as a substitute (Molnar et al., 2022). Conditional PFI has also been shown to perform poorly when many features are important (Molnar, König, et al., 2023). To account for underestimation of PFI due to correlated variables, another approach tries to correct PFI based on covered information (Pereira et al., 2022). König, Molnar, Bischl, and Grosse-Wentrup (2021) created a general framework called Relative Feature Importance, of which PFI and conditional PFI are special cases.

As stated above, different loss functions can be used for PFI, like accuracy and mean squared error, but also AUC has been used (Janitza et al., 2013), as well as likelihood and entropy (Wood, Papamarkou, Benatan, & Allmendinger, 2024). It was argued that using entropy, the resulting measure does not suffer from correlation bias (Wood et al., 2024). Meng, Yu, Cupples, Farrer, and Lunetta (2009) used an altered version of PFI, which only averaged over the number of trees, in which a variable was actually used in a split, while the original version averages over all trees (Breiman, 2001a). Another proposed method averages over all possible permutations (Fisher et al., 2019). In the same paper the authors argue that many different models might perform reasonably well but yield different PFI estimates. Therefore, they introduce model class reliance, as the range of PFI values across various well-performing models. Fumagalli, Muschalik, Hüllermeier, and Hammer (2023) have argued, that PFI should be scaled by $\frac{N}{N-1}$, with N describing the number of observations. Different authors have extended the PFI for groups of variables (Gregorutti et al., 2015; Au, Herbinger, Stachl, Bischl, & Casalicchio, 2022). Hapfelmeier, Hothorn, Ulm, and Strobl (2014) proposed a method to deal with missing data in PFI.

PFI has also been implemented for temporal models (Sood & Craven, 2022), for incremental learning models, where data is progressively sampled, and the model progressively trained (Fumagalli et al., 2023), as well as for clustering (Scholbeck, Funk, & Casalicchio, 2023). PFI has also been combined with other XAI methods, such as counterfactuals (Pfeifer et al., 2023). Based on PFI, a new type of learning trees was proposed, using PFI for each split (Ruoqing Zhu & Kosorok, 2015).

In this thesis, the behavior of PFI is examined in the presence of correlated features. As it is unclear, how exactly the PFI behaves in the presence of correlated predictors, as seemingly both over- and underestimation can occur, the correlation between the predictor variables, as well as the correlation between the predictor variables and the outcome variable, are systematically varied to estimate the effect these correlations have on the PFI scores. To see whether PFI behaves similarly in different models and if there are differences between linear models and non-linear models, this is done in RF as well as in RR models.

1.8 Propositions

1. When the predictor variable and the outcome variable are not correlated or connected, the resulting PFI should be 0.

2. In uncorrelated variables, the PFI score should be 2 times the squared correlation between the predictor and the outcome variable as seen in formula 7.

3. When two predictor variables carry the same information and one is permuted, the other variable can still provide this information, and the resulting PFI should be lower by a factor depending on the correlation between the two predictor variables, as seen in formula 8.

4. When two predictor variables are correlated but carry slightly different information for the outcome, extrapolation can occur, and the resulting PFI score is higher than expected.

2 Methods

All the simulations were done in python (Version 3.10; Van Rossum & Drake Jr, 1995) using Jupyter Notebooks (Version 6.5.2; Kluyver et al., 2016), as well as the libraries pandas (Version 1.5; The pandas development team, 2020), numpy (Version 1.23; Harris et al., 2020), and scikit-learn (Version 1.1; Pedregosa et al., 2011). For the graphs, the library matplotlib (Version 3.10; Hunter, 2007) was used.

2.1 Data

All the data used was simulated for this study.

For data generation, a multivariate normal distribution was used by defining a variance-covariance (VC) matrix. For simplicity, two predictor variables and one outcome variable were created. A 3 by 3 matrix was defined, where the first generated variable, as defined by the first column, was used as the outcome variable, and the other two variables as the predictor variables, which will be called variable 1 and variable 2 from here on. Variable 1 describes the variable, described by the second column of the VC matrix, while the third column describes variable 2. So, the outcome variable was not defined as a linear combination of variable 1 and variable 2, but was related to those two predictor variables by the covariances defined by the VC matrix.

The covariance between variable 1 and variable 2 is denoted as COV_l, the covariance between variable 1 and the outcome is denoted as COV_k, and the covariance between variable 2 and the outcome as COV_j. The three covariances were varied individually and set to the values $0, 0.3, 0.5, 0.7, 0.9$, resulting in 125 possible combinations. But since the order of variable 1 and 2 did not matter, only 75 possible combinations are of interest. The resulting VC matrix had the structure

$$\begin{pmatrix} 1 & COV_k & COV_j \\ COV_k & 1 & COV_l \\ COV_j & COV_l & 1 \end{pmatrix}. \tag{9}$$

Due to all the variances being set to 1, the correlation coefficients between two variables are equal to the covariances, due to

$$\rho_{a,b} = \frac{COV_{a.b}}{VAR_a \cdot VAR_b} = \frac{COV_{a.b}}{1 \cdot 1} = COV_{a.b}. \tag{10}$$

For extreme values, not all so created matrices were VC matrices, and so could not be used for data generation. For example, when COV_k was set to 0.5 and COV_j was set to 0.9, COV_l could not be set to 0, as the determinant for this matrix would then be negative, and it could not be used as a VC matrix. For these, the third covariance, in this case COV_l had to be changed accordingly. If, for example, COV_k was 0.5 and COV_j was 0.9, COV_l was set to 0.1, for $COV_k = 0.7$ and $COV_j = 0.9$, COV_l was set 0.4, and for $COV_k = 0.9$ and $COV_j = 0.9$, COV_l was set to 0.7. Equivalently, the same was done when other pairs of those variables reached these values. So, not all possible combinations could be performed. This lead to a reduction of the number of unique possible combinations to 66, resulting in 132 unique PFI values.

2.2 Statistical Analysis

For model building, RF and RR were used. Variable 1 and variable 2 were used to predict the outcome. For each simulation the following steps were performed:

1. The VC matrix was defined.

2. 5000 datapoints for each variable were generated using the VC matrix and used for hyperparameter tuning. The datapoints, as defined by the first column of the VC matrix were used as the outcomes and the datapoints defined by the other two columns as predictors. Grid search, random search and cross-validation were performed using the scikit-learn library. For RF, a grid search was performed as the possible values of hyperparameters was limited, for example for maximum features used for each split, only the values 1 or 2 were possible and for other hyperparameters only natural numbers up to a point were possible to be used, while for RR, for the penalty term any positive rational number can be chosen. The hyperparameters defining the maximum features used for each split, the minimum samples per leaf, the minimum samples per split, and the number of trees in the forest were tuned with 5 fold-cross-validation for RF. For RR, the regularization term λ, noted α in the scikit-learn library, was tuned using random search with 5 fold-cross-validation and 100 iterations.

3. Using the best set of hyperparameters for each set of generated data, 5000 new datapoints were generated, and the model was trained on a training set using 75% of the data, with 25% being the test set.

4. R^2 and PFI were then calculated using the test set. For PFI, the predictor variables were randomly permutated and the increase in prediction error was calculated, comparing when none of the variables was permuted and when one was permuted. This process was repeated 50 times in order to counteract any random fluctuations due to the permutation and the average of these increases was used to indicate the feature importance.

Steps 3 and 4 were repeated 100 times for each VC matrix, each time using the same hyperparameters gained in step 2. For these 100 repetitions, mean scores and standard deviations for R^2 and PFI were calculated.

3 Results

The results will be presented as follows: At first, the behavior of PFI will be examined when there is no correlation between the two predictor variables, setting COV_l to 0. This is at first examined when only one variable is connected to the

outcome, and then when both variables are connected to the outcome. Then, correlation between the two predictor variables is introduced. At first, cases are presented where the PFI underestimates the connection between the predictor variables and the outcome. Finally, cases are presented where PFI estimates higher scores than would be expected given the corresponding covariance.

In general, the PFI scores were quite stable. The standard deviations of the PFI score using a RF model ranged from 0 to 0.22, with 88% smaller than 0.05. Only in cases, where the two variables are highly correlated and the PFI score gets very high the standard deviation rises, and in the case of it reaching 0.22, the PFI score is 4.56. Similarly, using a RR model, the standard deviations of the PFI score ranged from 0 to 0.44, with 82% smaller than 0.05. In the case of the standard deviation being 0.44, the PFI score was estimated as 9.3. The highest PFI score and highest standard deviation were achieved in the same case for RF and RR.

3.1 Uncorrelated Predictors

As mentioned, at first the behavior of PFI is examined when only one variable is connected to the outcome and there is no correlation between the two predictor variables. As an example, the results of the VC matrix

$$\begin{pmatrix} 1 & 0.5 & 0 \\ 0.5 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{11}$$

are presented. The mean resulting score of the 100 simulations ± the standard deviation over these 100 simulations are presented in Table 1.

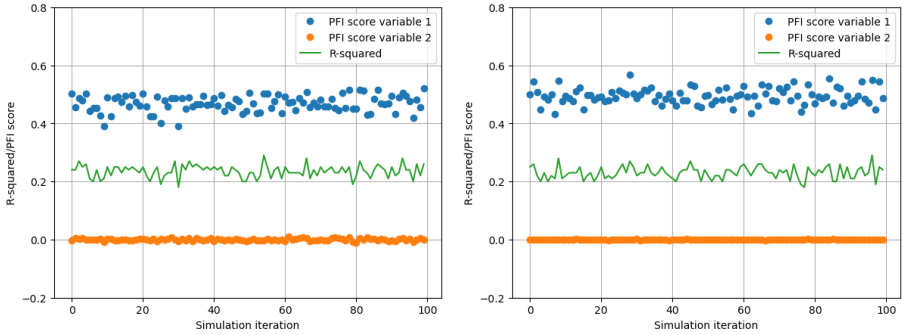
Table 1

The R^2 and PFI scores for Random Forest and Ridge Regression models

	R^2	PFI score 1	PFI score 2
RF	0.24 ± 0.02	0.47 ± 0.03	0 ±<0.01
RR	0.25 ± 0.02	0.5 ± 0.03	0 ±<0.01

Note. PFI score 1 is the PFI score for variable 1 and PFI score 2 the PFI score for variable 2. RF stands for Random Forest and RR for Ridge Regression. The number in the tables are the resulting mean score over 100 iterations ± the standard deviation.

As seen in the VC matrix, variable 1 is set to have a correlation of 0.5 to the outcome, which should result in a maximum possible R^2 of 0.25. Both models approximately reached the upper limit of the possible R^2 on the test set. As was noted in formula 7, the PFI score should be equal to two times the correlation



(a) Random Forest *(b)* Ridge Regression

Figure 1. Shows the PFI scores and R^2 over the 100 iterations of the simulation using Random Forest and Ridge Regression

Table 2

The R^2 and PFI scores for an untuned Random Forest with 100 trees with varying sizes of COV_k

	R^2	PFI score 1	PFI score 2
$COV_k = 0.3$	-0.08 ± 0.03	0.18 ± 0.03	0 ± 0.02
$COV_k = 0.5$	0.11 ± 0.03	0.5 ± 0.04	0 ± 0.02
$COV_k = 0.7$	0.39 ± 0.03	0.98 ± 0.04	0 ± 0.01
$COV_k = 0.9$	0.77 ± 0.03	1.62 ± 0.03	0 ±<0.01

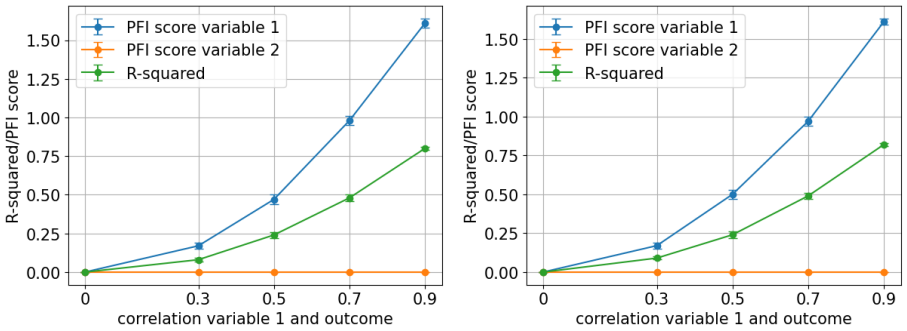
Note. PFI score 1 is the PFI score for variable 1 and PFI score 2 the PFI score for variable 2. The number in the tables are the resulting mean score over 100 iterations ± the standard deviation.

squared. In both cases, when using RF or RR, this was approximately reached. Also, as expected, for the variable not correlated to the outcome, the PFI score is 0. Figure 1 shows the resulting R^2 as a green line, and the PFI scores for variable 1 and 2 as blue and orange points for both RF and RR models.

Varying COV_k shows the same trends regarding the R^2 and the PFI scores. When varying COV_k from 0 to 0.9, using the set {0,0.3,0.5,0.7,0.9}, the R^2 is equal to the COV_k squared, while the PFI scores are about two times this for both RF and RR. Figure 2 shows the resulting R^2 and the PFI scores for both RF and RR.

During the simulation process, it was also found, that even when a RF model did not accurately capture the underlying datastructure, the PFI score still was approximately equal to two times the correlation squared. This is shown in Table 2. As can be seen, while the R^2 is lower than the COV_k squared, the PFI scores still are about two times the COV_k squared.

Next, a second influencing variable is introduced. COV_j is no longer set to



(a) Random Forest *(b)* Ridge Regression

Figure 2. Shows the PFI score and R^2, when only one variable is connected to the outcome, and there is no correlation between the variables using Random Forest and Ridge Regression. The correlation between the predictor variable and the outcome is varied.

0 and variable 2 is correlated to the outcome. For example, COV_k and COV_j are set to 0.5, resulting in the following VC matrix:

$$\begin{pmatrix} 1 & 0.5 & 0.5 \\ 0.5 & 1 & 0 \\ 0.5 & 0 & 1 \end{pmatrix}. \tag{12}$$

The results are presented in table 3.

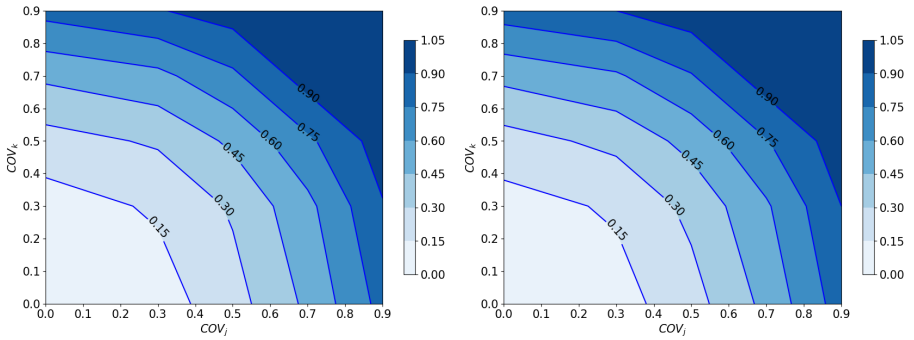
Table 3

The R^2 and PFI scores for Random Forest and Ridge Regression models

	R^2	PFI score 1	PFI score 2
RF	0.48 ± 0.02	0.47 ± 0.03	0.47 ± 0.03
RR	0.5 ± 0.02	0.5 ± 0.02	0.5 ± 0.02

Note. PFI score 1 is the PFI score for variable 1 and PFI score 2 the PFI score for variable 2. RF stands for Random Forest and RR for Ridge Regression. The number in the tables are the resulting mean score over 100 iterations ± the standard deviation.

The resulting PFI scores are equal to two times the covariance squared, and the resulting R^2 is equal to the sum of the two covariances COV_k and COV_j squared. Varying both COV_k and COV_j, while leaving COV_l at 0, leads to similar results. However, when both covariances COV_k and COV_j get higher, COV_l can no longer be set to 0, as the resulting matrix could not be used as a VC matrix to generate data. When one covariance is set to 0.9, and the other



(a) Random Forest *(b)* Ridge Regression

Figure 3. Shows the R^2, when there is no correlation between the variables and the correlation to the outcome is varied using Random Forest and Ridge Regression. At the top-right corner, above 0.5 - 0.9, the correlation between the two variables could not be set to 0. At 0.5-0.9 it is 0.1, at 0.7-0.9 it is 0.4, at 0.9-0.9 it is 0.7.

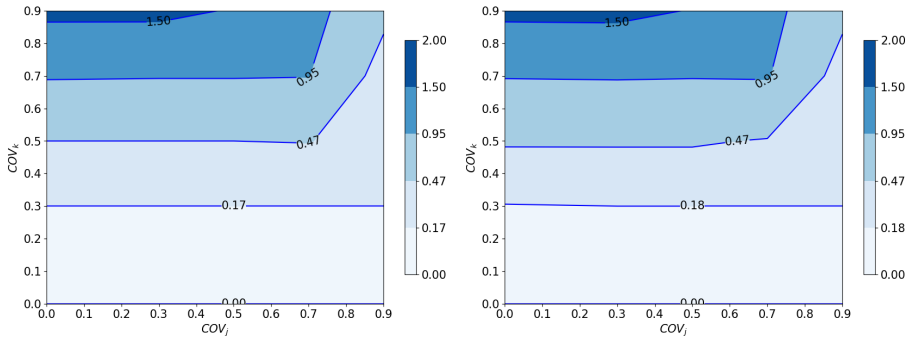
covariance is either 0.5, 0.7, or 0.9, the final covariance, in this case COV_l, the covariance between the two predictors, has to be set to at least 0.1, 0.4, or 0.7 respectively. As in these cases the two variables are no longer uncorrelated, they will not be discussed here. The resulting behavior when the two variables are correlated, is discussed below, when COV_l is deliberately varied. Figure 3 shows the behavior of the R^2 value depending on the covariances COV_k and COV_J. Figure 4 shows the behavior of the resulting PFI score, which is approximately equal to two times the correlation squared, when COV_l is set to 0.

In the above cases, when there is no correlation, the PFI scores behave as expected, yielding accurate estimations of the connection between the predictor variables and the outcome.

3.2 Underestimation in Correlated Predictors

Cases in which PFI underestimates the connection between a predictor variable and the outcome are examined. This occurs, when the information of one variable is covered by the other when it is permuted.

At first, we set both covariances COV_k and COV_j to 0.5 and vary the co-



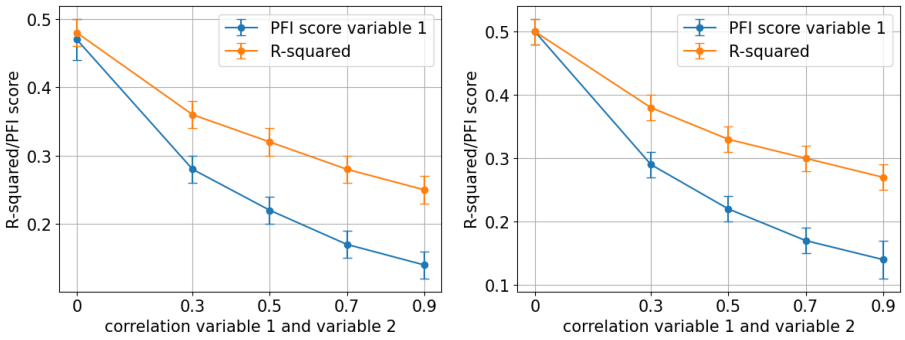
(a) Random Forest *(b)* Ridge Regression

Figure 4. Shows the PFI scores, when there is no correlation between the variables and the correlation to the outcome is varied using Random Forest and Ridge Regression. At the top-right corner, above 0.5 - 0.9, the correlation between the two variables could not be set to 0. At 0.5-0.9 it is 0.1, at 0.7-0.9 it is 0.4, at 0.9-0.9 it is 0.7.

variance between the two variables, COV_l, resulting in the VC matrix

$$\begin{pmatrix} 1 & 0.5 & 0.5 \\ 0.5 & 1 & COV_l \\ 0.5 & COV_l & 1 \end{pmatrix}. \tag{13}$$

When there is no correlation between the features, the PFI scores equal two times the covariance squared, approximately 0.5, and the R^2 is the sum of the squared covariances, which also approximately equals 0.5. However, when increasing COV_l, the achieved R^2, as well as the PFI scores shrink, with the PFI scores shrinking faster. When COV_l is set to 0.9, the resulting R^2 score is equal to 0.25 ± 0.2 for RF, and 0.27 ± 0.2 for RR, which indicates that, when both variables highly correlate, they likely carry mostly the same information about the outcome. The resulting PFI score is then equal to 0.14 ± 0.2 in RF, and 0.14 ± 0.3 in RR. So, it seems that in this case, both variables carry the same information, and when one is permuted, the other variable covers the information. The resulting PFI score of 0.14 would indicate a lower covariance than is given by the VC matrix, in this case 0.26. The connection seen above, where R^2 can be depicted as the sum of all PFI scores halved, does no longer work in this case. The resulting PFI scores, however, do coincide with formula 8. Figure 5 shows the PFI scores as well as the R^2, when COV_l is varied, with both COV_k and COV_j being constant at 0.5.



(a) Random Forest *(b)* Ridge Regression

Figure 5. Shows the PFI score and R^2, when both variable are equally connected to the outcome with a correlation of 0.5, and the correlation between the variables is varied

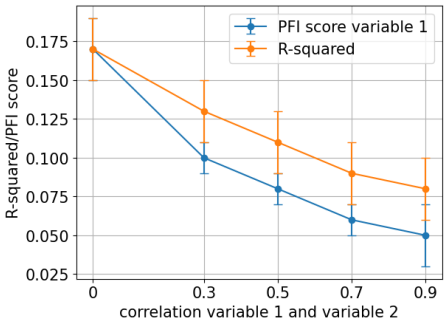
Similar behavior is seen when COV_k and COV_j are set to 0.3, 0.7, or 0.9, as depicted in Figure 6. Increasing the correlation between these variables leads to them carrying the same information, which results in a lower R^2 and lower PFI scores.

3.3 Mixed Cases

Underestimation from PFI can also happen, when the two variables do not have the same correlation to the outcome. But in these cases, PFI can underestimate or overestimate the importance of a variable. As a first example, COV_k is set to 0.7, COV_j to 0.3 and COV_l is varied, resulting the VC matrix

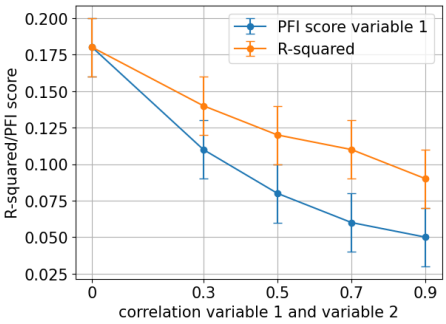
$$\begin{pmatrix} 1 & 0.7 & 0.3 \\ 0.7 & 1 & COV_l \\ 0.3 & COV_l & 1 \end{pmatrix}. \tag{14}$$

When COV_l is set to 0.9, COV_j was set from 0.3 to 0.4. The results can be seen in Table 4 and in Figure 7. It can be seen, as COV_l is set to higher values, at first the R^2 and PFI scores drop, especially for the variable with a lower correlation to the outcome. However, as COV_l is set to higher values, the R^2 as well as the PFI scores rise. Using PFI in RR yields higher scores than in RF when it overestimates the importance of a variable, but in both models, the PFI score is unreasonably high. As the R^2 rises with higher correlation between the variables, there is possibly an interaction effect, due to peculiarities in the data-generating process. While this interaction probably also is reflected in the



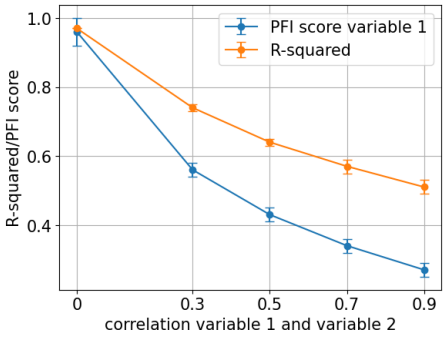
(a) Random Forest

$COV_k = COV_j = 0.3$



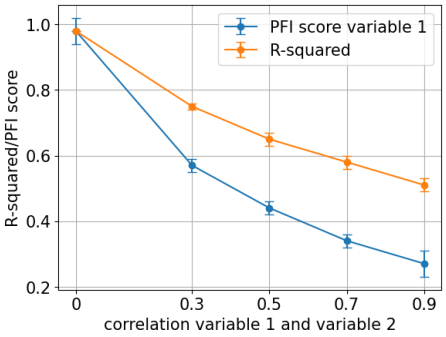
(b) Ridge Regression

$COV_k = COV_j = 0.3$



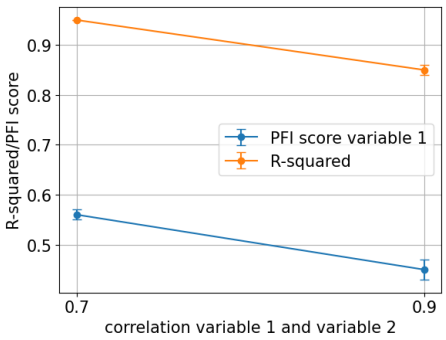
(c) Random Forest

$COV_k = COV_j = 0.7$



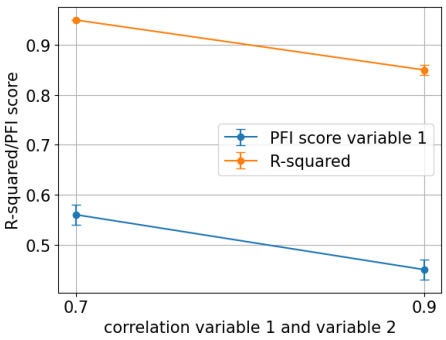
(d) Ridge Regression

$COV_k = COV_j = 0.7$



(e) Random Forest

$COV_k = COV_j = 0.9$



(f) Ridge Regression

$COV_k = COV_j = 0.9$

Figure 6. Shows the PFI score and R^2, when both variable are equally connected to the outcome, and the correlation between the variables is varied.

Table 4

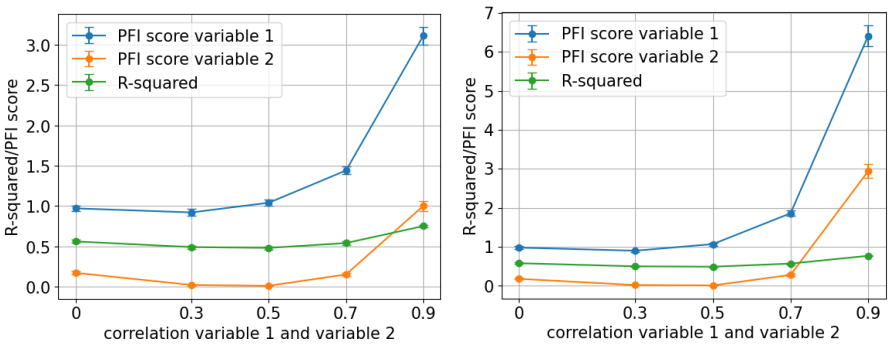
The R^2 and PFI scores for Random Forest and Ridge Regression models

RF			
COV_l	R^2	PFI score 1	PFI score 2
0	0.56 ± 0.02	0.97 ± 0.03	0.17 ± 0.02
0.3	0.49 ± 0.02	0.92 ± 0.04	$0.02 \pm <0.01$
0.5	0.48 ± 0.02	1.04 ± 0.04	$0.01 \pm <0.01$
0.7	0.54 ± 0.02	1.44 ± 0.05	0.15 ± 0.02
0.9	0.75 ± 0.01	3.11 ± 0.11	1 ± 0.06
RR			
COV_l	R^2	PFI score 1	PFI score 2
0	0.56 ± 0.02	0.98 ± 0.04	0.18 ± 0.02
0.3	0.5 ± 0.02	0.9 ± 0.03	$0.02 \pm <0.01$
0.5	0.49 ± 0.02	1.07 ± 0.04	$0.01 \pm <0.01$
0.7	0.57 ± 0.02	1.86 ± 0.07	0.28 ± 0.03
0.9	0.77 ± 0.01	6.41 ± 0.27	2.94 ± 0.186

Note. PFI score 1 is the PFI score for variable 1 and PFI score 2 the PFI score for variable 2. RF stands for Random Forest and RR for Ridge Regression. The number in the tables are the resulting mean score over 100 iterations ± the standard deviation. When the correlation between the two variables is set to 0.9, the correlation of variable 2 to the outcome is set to 0.4.

PFI score, leading to higher scores, the resulting values do not seem to carry any reasonable information, especially given that they differ for the two models.

Similar behavior is shown for other combinations, when the correlation between the two variables is increased. When the correlation between the two variables is low but not 0, the PFI estimates lower scores for the variables. Notably, the variable with a lower correlation to the outcome is estimated to be quite unimportant, with the PFI score sometimes reaching 0. This can be seen to make sense, as the information in this variable is probably covered by the other variable, and the variable is probably not needed for the model. When correlations between the two variables are high, very high PFI scores are found, but also higher values for R^2, suggesting an increase in information. However, the PFI scores do not seem to be in any sensible relationship with this increase in information, exemplified by PFI scores in RR models being higher than in RF models. In this simulation study, only in cases where the correlation between the two variables, COV_l was higher than either COV_k or COV_l, these effects were found. In these cases, the data generation process has to highly correlate the two variables, but only one of them is also highly correlated to the outcome. This is seemingly the cause of the found interaction effect and the rise in R^2 and PFI scores.



(a) Random Forest *(b)* Ridge Regression

Figure 7. Shows the PFI score and R^2, when the two predictor variable are connected in a different magnitude to the outcome, and the correlation between the variables is varied. Variable 1 is correlated to the outcome at 0.7, and variable 2 at 0.3. When the correlation between the two variables is set to 0.9, the correlation of variable 2 to the outcome is set to 0.4

Figure 8 shows other combinations, when both variables are connected to the outcome.

During the simulation process, some of the variants have been done two times. For example, the results should be the same, given the VC-matrices

$$\begin{pmatrix} 1 & 0.7 & 0.3 \\ 0.7 & 1 & 0.7 \\ 0.3 & 0.7 & 1 \end{pmatrix} \text{ or } \begin{pmatrix} 1 & 0.3 & 0.7 \\ 0.3 & 1 & 0.7 \\ 0.7 & 0.7 & 1 \end{pmatrix},$$

with the PFI scores switched, of course. It was found in 5 cases in RF, that these switches did not yield the same PFI scores. This was the case for the VC-matrices given in 15, and their switched counterparts. This was only found in cases where the PFI scores were lower or higher than expected, but not in uncorrelated predictor variables. For now, only one of the resulting PFI scores is shown in the diagrams, but in the appendix, altered versions of the diagrams are shown to compare the results. Note that the resulting R^2 was the same or very similar in all these cases. Possible causes for the different PFI results will be discussed later in the results section.

$$\begin{pmatrix} 1 & 0.7 & 0 \\ 0.7 & 1 & 0.7 \\ 0 & 0.7 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0.9 & 0.7 \\ 0.9 & 1 & 0.5 \\ 0.7 & 0.5 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0.7 & 0.3 \\ 0.7 & 1 & 0.7 \\ 0.3 & 0.7 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0.9 & 0.5 \\ 0.9 & 1 & 0.7 \\ 0.5 & 0.7 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0.5 & 0.3 \\ 0.5 & 1 & 0.7 \\ 0.3 & 0.7 & 1 \end{pmatrix} \tag{15}$$

3.4 Overestimation in Correlated Predictors

Overestimation is further exemplified when only one variable is correlated to the outcome and the correlation between the two variables is increased. As an example, COV_k is set to 0.5, COV_j to 0 and COV_l then varied, leading to the VC matrix

$$\begin{pmatrix} 1 & 0.5 & 0 \\ 0.5 & 1 & COV_l \\ 0 & COV_l & 1 \end{pmatrix}. \tag{16}$$

Only the first variable is therefore correlated to the outcome. When a correlation between the two predictor variables is introduced, the resulting R^2 increases as well as the PFI scores for both variables. The results can be seen in Table 5 and Figure 9.

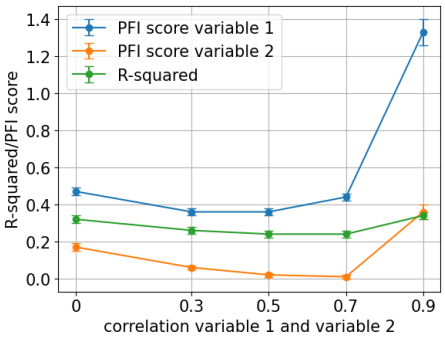
Similarly to above, very high PFI scores are found, which are in no relation to the correlation the variable has to the outcome, or the R^2. Again, R^2 also rises as the PFI scores rise, however, the PFI scores rise to such extremes, that any logical connection between the correlation or explained variance of the variable and the PFI score is destroyed, especially seen as RR and RF yield different results in such cases.

Also, the second variable shows a higher PFI score than 0, even though it should have a correlation of 0 with the outcome. When looking at the higher R^2 it seems likely, that variable 2 also carries some information about the outcome variable, so it makes sense, that the PFI score also rises. However, the PFI score rises as high as 6.79, which is questionably high.

The same trend can be seen, with different values for COV_k. Figure 10 shows when COV_k is set to 0.3, 0.7, or 0.9.

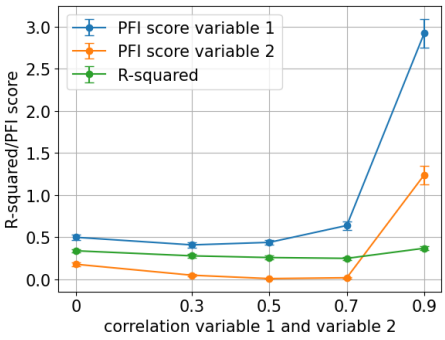
3.5 Overview

Figures 11 and 12 summarize how R^2 changes depending on the different correlations, for RF and for RR respectively. For each graph the covariance between the variable 1 and variable 2, COV_l, is fixed, and COV_k and COV_j are varied.



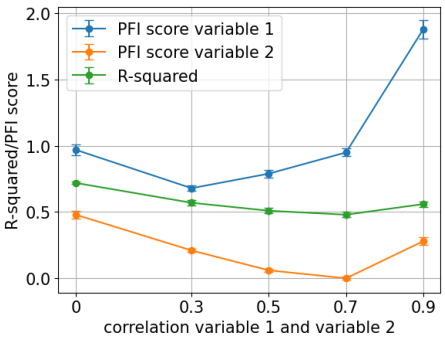
(a) Random Forest

$COV_k = 0.5, COV_j = 0.3$



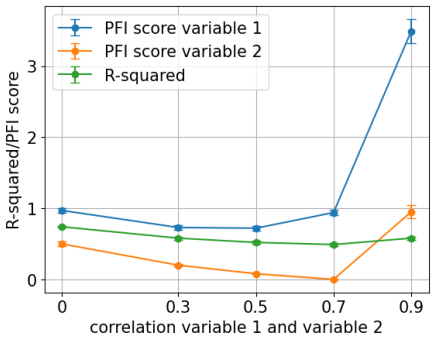
(b) Ridge Regression

$COV_k = 0.5, COV_j = 0.3$



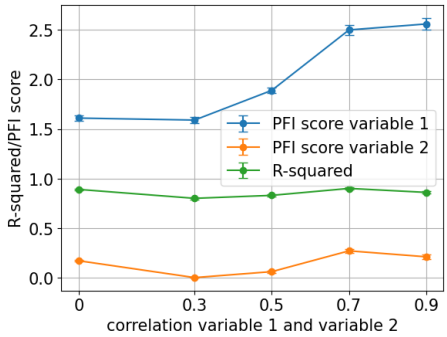
(c) Random Forest

$COV_k = 0.7, COV_j = 0.5$



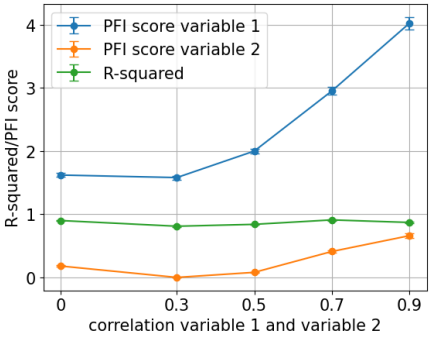
(d) Ridge Regression

$COV_k = 0.7, COV_j = 0.5$



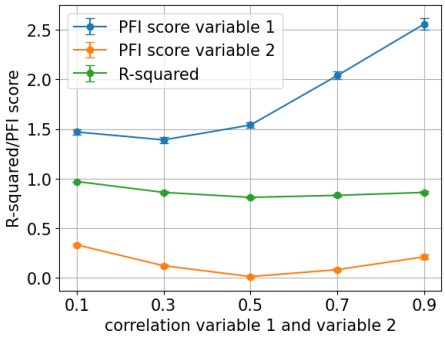
(e) Random Forest

$COV_k = 0.9, COV_j = 0.3$



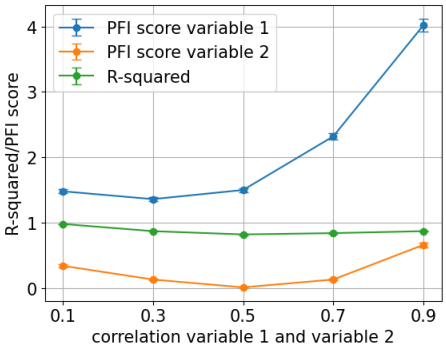
(f) Ridge Regression

$COV_k = 0.9, COV_j = 0.3$



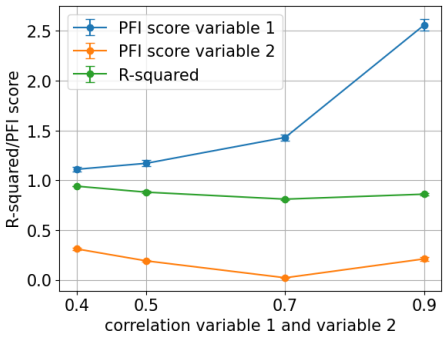
(g) Random Forest

$COV_k = 0.9, COV_j = 0.5$



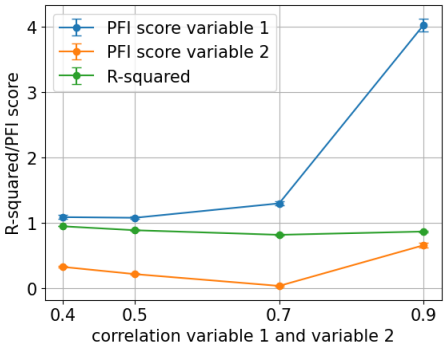
(h) Ridge Regression

$COV_k = 0.9, COV_j = 0.5$



(i) Random Forest

$COV_k = 0.9, COV_j = 0.7$



(j) Ridge Regression

$COV_k = 0.9, COV_j = 0.7$

Figure 8. Shows the PFI score and R^2, when the two predictor variable are connected in a different magnitude to the outcome, and the correlation between the variables is varied. At (e) and (f), when the correlation between the two variables is set to 0.7, the correlation between variable 2 and the outcome is set to 0.4, at 0.9, it is set to 0.7. At (g) and (h), when the correlation between the two variables is set to 0.9, the correlation between variable 2 and the outcome is set to 0.7.

Table 5

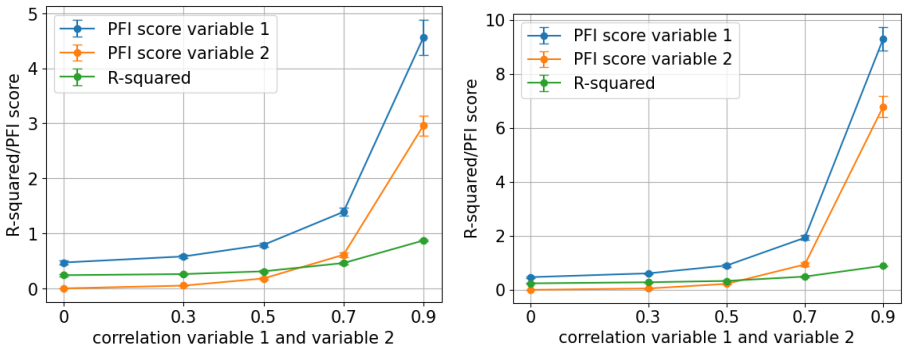
The R^2 and PFI scores for Random Forest and Ridge Regression models

RF			
COV_l	R^2	PFI score 1	PFI score 2
0	0.24 ± 0.02	0.47 ± 0.03	$0 \pm <0.01$
0.3	0.26 ± 0.02	0.58 ± 0.03	0.05 ± 0.01
0.5	0.31 ± 0.02	0.79 ± 0.04	0.18 ± 0.02
0.7	0.46 ± 0.02	1.39 ± 0.07	0.61 ± 0.04
0.9	0.87 ± 0.01	4.56 ± 0.32	2.96 ± 0.18
RR			
COV_l	R^2	PFI score 1	PFI score 2
0	0.25 ± 0.02	0.50 ± 0.03	$0 \pm <0.01$
0.3	0.28 ± 0.02	0.61 ± 0.03	0.05 ± 0.01
0.5	0.33 ± 0.02	0.9 ± 0.05	0.22 ± 0.02
0.7	0.49 ± 0.02	1.93 ± 0.09	0.94 ± 0.07
0.9	0.89 ± 0.01	9.3 ± 0.44	6.79 ± 0.39

Note. PFI score 1 is the PFI score for variable 1 and PFI score 2 the PFI score for variable 2. RF stands for Random Forest and RR for Ridge Regression. The number in the tables are the resulting mean score over 100 iterations ± the standard deviation. When the correlation between the two variables is set to 0.9, the correlation between the second variable and the outcome is set to 0.1.

When both variables are correlated to the outcome and to each other, they might carry redundant information, lowering the explained variance of the outcome. Interaction can also happen, leading to higher explained variance of the outcome. Both methods work almost equally well, with RR sometimes performing slightly better, which is to be expected, as the correlation is linear and so the task is also linear.

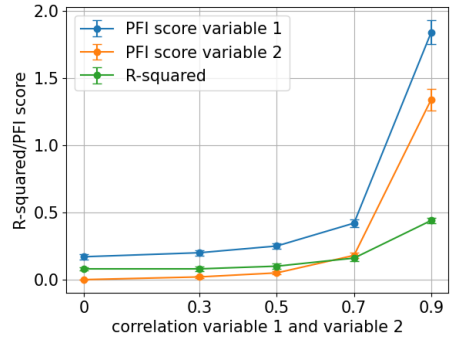
Figures 13 and 14 summarize the PFI scores, depending on the different correlations, for RF and for RR respectively. For each graph, COV_l is again fixed, and COV_k and COV_j are varied. Depicted is the PFI score for variable 1, for which the correlation to the outcome, COV_k, is depicted on the y-axis. Without any correlation between the two variables, the PFI score is equal to 2 times the correlation coefficient squared. With rising correlation between the two variables, PFI scores can either be higher or lower. It is lower, when both variables are equally correlated to the outcome, or when the correlation between the two variables is lower than the correlation of either variable to the outcome. It is higher, when the correlation between the variables is higher than one of the correlations of the variables to the outcome. However, in these cases one variable can still have a lower PFI score than would be expected. Also, RR generally produces higher PFI scores when such an overestimation takes place.



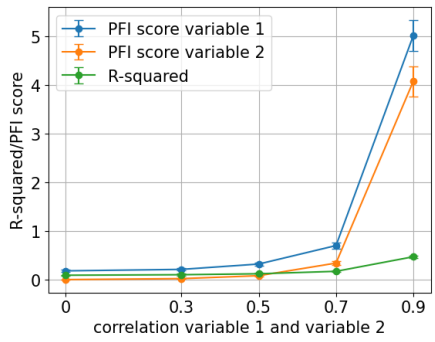
(a) Random Forest *(b)* Ridge Regression

Figure 9. Shows the PFI score and R^2, when only one variable is connected to the outcome with a correlation of 0.5, and the correlation between the variables is varied. When the correlation between the two variables is set to 0.9, the correlation between the second variable and the outcome is set to 0.1.

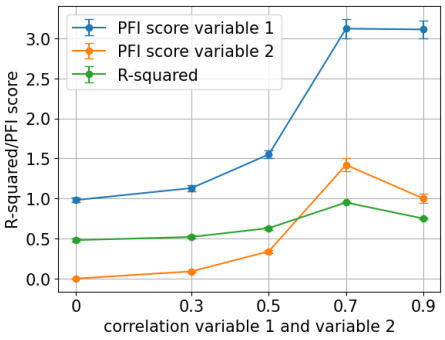
As noted above, in some cases, the RF model produced different PFI scores in different simulations, when COV_k and COV_l were switched. Since PFI scores were found to be quite stable, this was somewhat surprising, as in both simulations, the PFI scores seemed very stable, but they still differed. It was suspected, that the chosen set of hyperparameters in RF was the reason for the different PFI scores. To explore this, the case $COV_k = 0.7$, $COV_j = 0$, and $COV_l = 0.7$ was picked up again. The simulation using the according VC matrix was run multiple times, but this time a randomized search was used to extend the space of possible hyperparameter sets. The hyperparameters setting the maximum features used for each split, the minimum samples per leaf, and the minimum samples per split were tuned for this simulation. In all of the 7 simulation runs, different sets of hyperparameters were used. The R^2 ranged from 0.93 ± 0.01 to $0.95 \pm <0.01$, while the PFI scores for variable 1 ranged from 2.59 ± 0.09 to 3.12 ± 0.11, and for variable 2 from 1.09 ± 0.06 to 1.43 ± 0.09. Interestingly, when one PFI score was higher, the other was also higher, so both the highest PFI score for variables 1 and 2 were found in the same simulation run and the lowest PFI score for variables 1 and 2 were found in the same simulation run. Using only simulations with a R^2 of 0.95, the lowest PFI scores were 2.89 ± 0.1 and 1.27 ± 0.07. As seen, the PFI scores are stable in a set of hyperparameters but can differ between sets of hyperparameters.



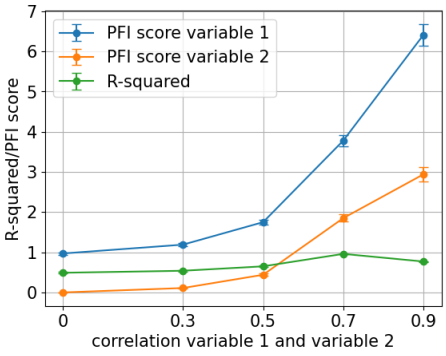
(a) Random Forest, $COV_k = 0.3$



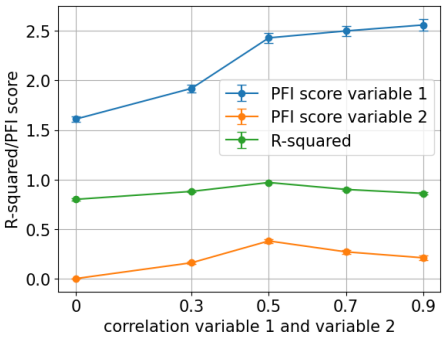
(b) Ridge Regression, $COV_k = 0.3$



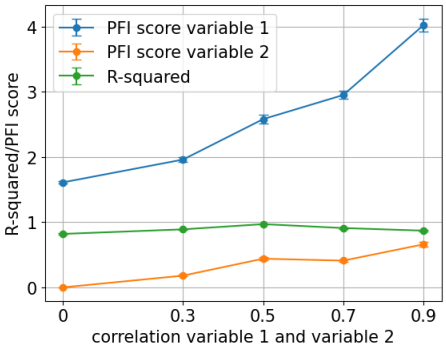
(c) Random Forest, $COV_k = 0.7$



(d) Ridge Regression, $COV_k = 0.7$

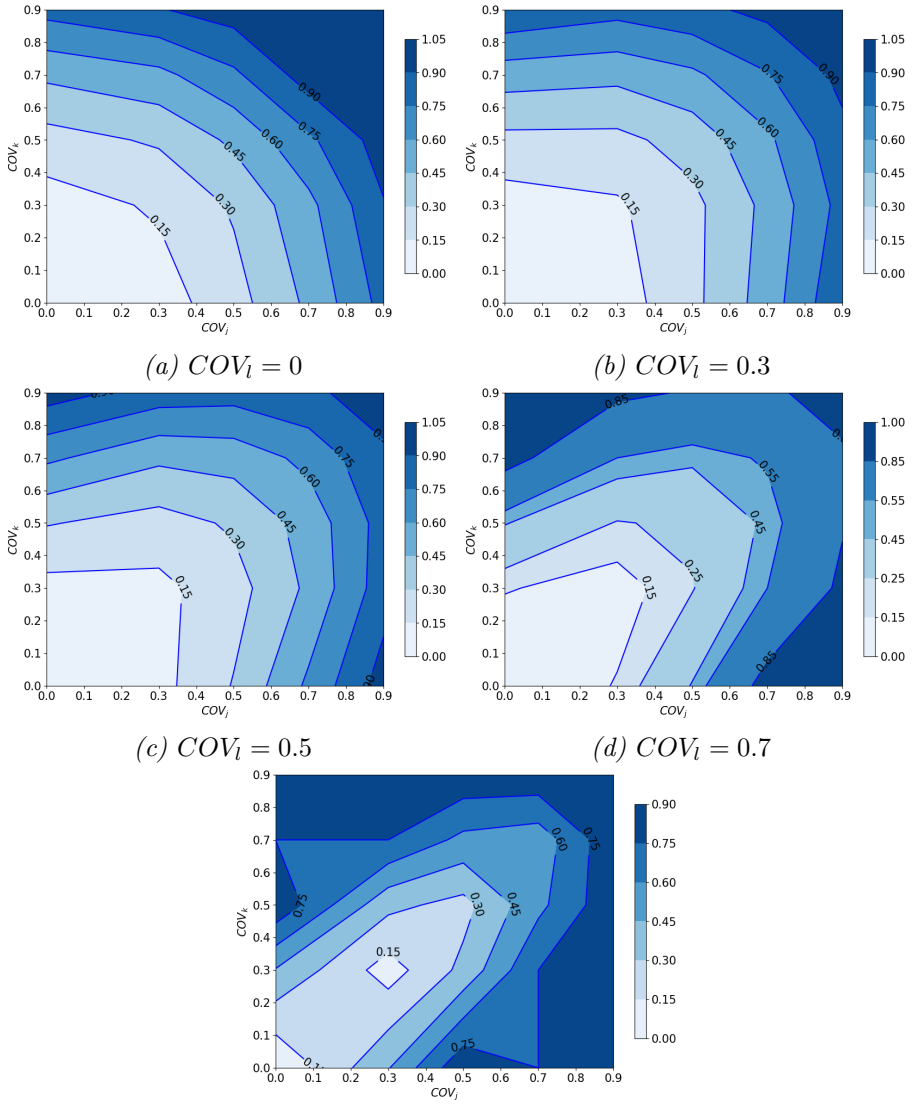


(e) Random Forest, $COV_k = 0.9$



(f) Ridge Regression, $COV_k = 0.9$

Figure 10. Shows the PFI score and R^2, when only one variable is connected to the outcome, and the correlation between the variables is varied. In (c) and (d), when COV_l is 0.9, COV_j is set to 0.4, in (e) and (f), when COV_l is 0.5, COV_j is set to 0.1, when COV_l is 0.7, COV_j is set to 0.4, when COV_l is 0.9, COV_j is set to 0.7.

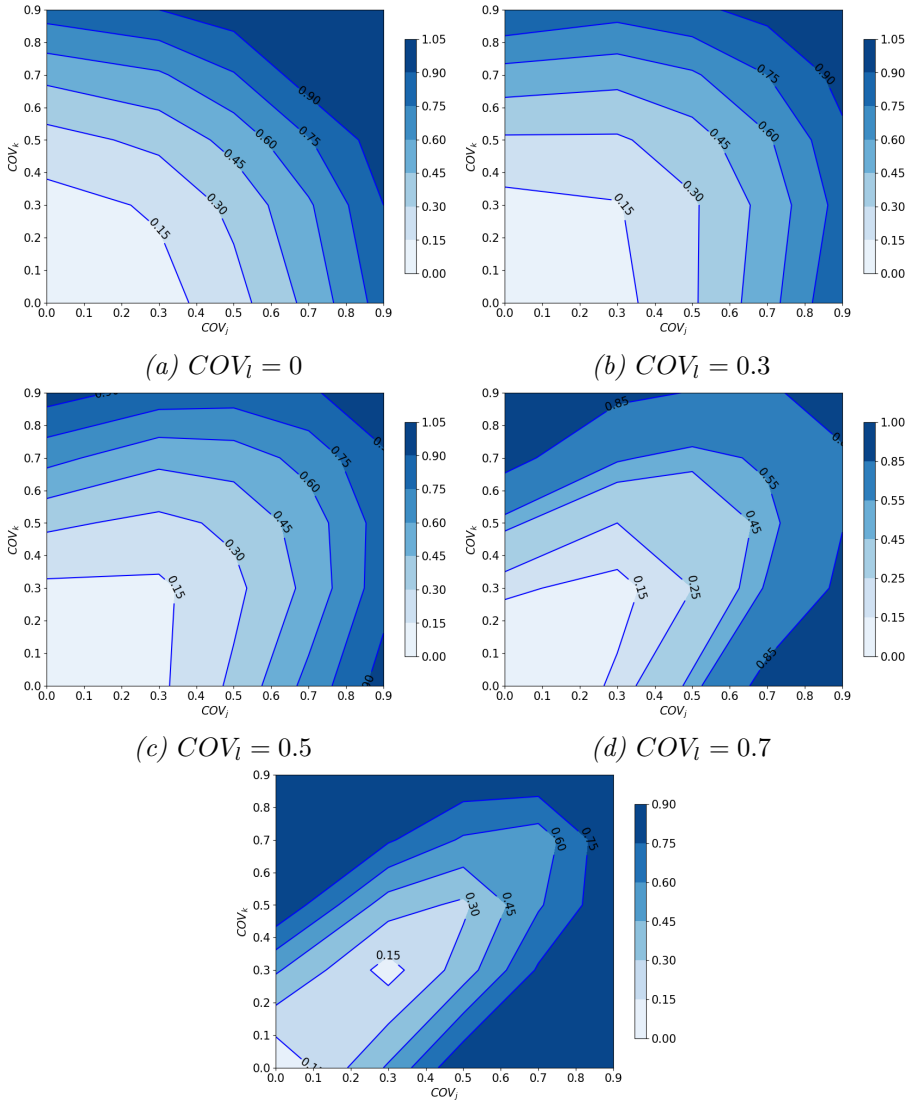


(a) $COV_l = 0$ *(b)* $COV_l = 0.3$

(c) $COV_l = 0.5$ *(d)* $COV_l = 0.7$

(e) $COV_l = 0.9$

Figure 11. Shows the R^2, with a set COV_l and varying both COV_k and COV_l using Random Forest. Note that when two of the correlations reach a certain high, the third correlation had to be changed accordingly. When two of the correlations are set to 0.5-0.9, the third had to be at least 0.1, at 0.7-0.9 it had to be at least 0.4, at 0.9-0.9 at least 0.7.



(a) $COV_l = 0$ *(b)* $COV_l = 0.3$

(c) $COV_l = 0.5$ *(d)* $COV_l = 0.7$

(e) $COV_l = 0.9$

Figure 12. Shows the R^2, with a set COV_l and varying both COV_k and COV_l using Ridge regression. Note that when two of the correlations reach a certain high, the third correlation had to be changed accordingly. When two of the correlations are set to 0.5-0.9, the third had to be at least 0.1, at 0.7-0.9 it had to be at least 0.4, at 0.9-0.9 at least 0.7.

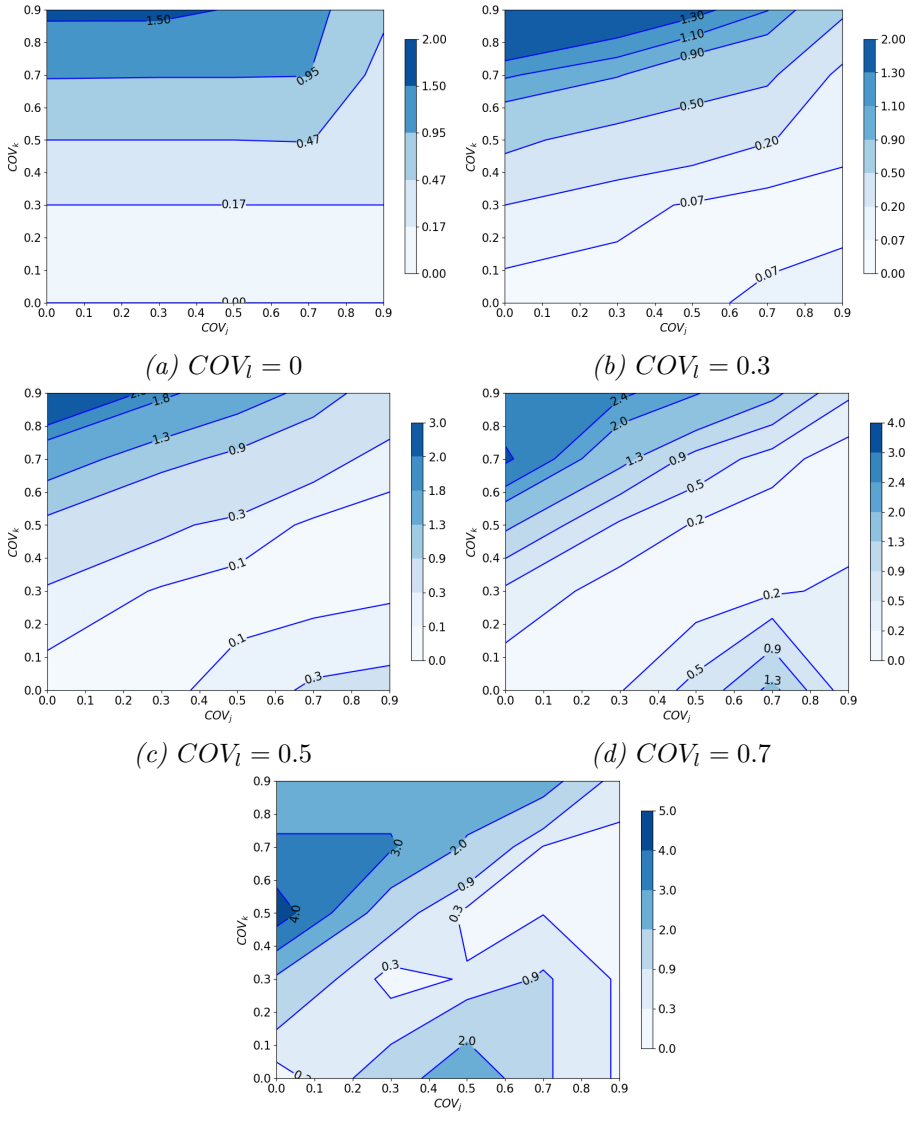
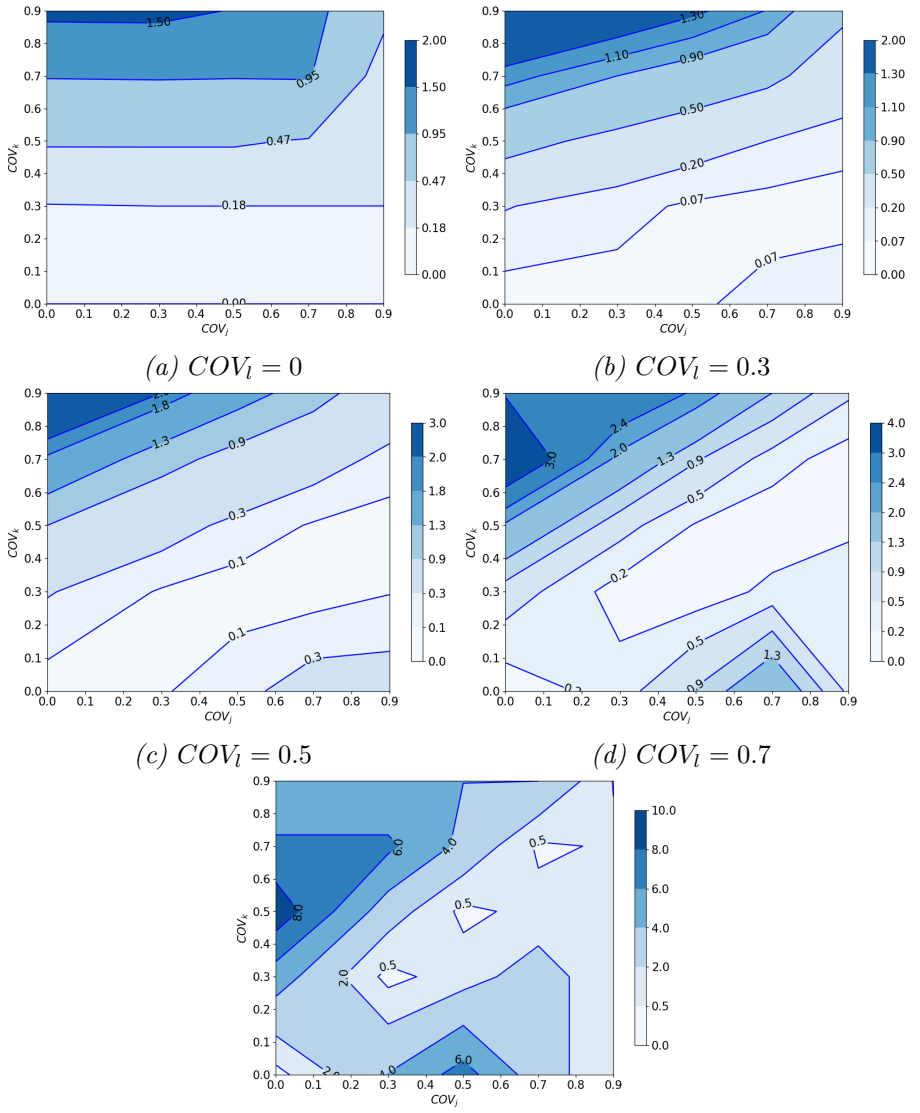


Figure 13. Shows the PFI scores for variable 1, for which the correlation to the outcome is depicted on the y-axis, with a set COV_l and varying both COV_k and COV_l using Random Forest. Note that when two of the correlations reach a certain high, the third correlation had to be changed accordingly. When two of the correlations are set to 0.5-0.9, the third had to be at least 0.1, at 0.7-0.9 it had to be at least 0.4, at 0.9-0.9 at least 0.7.



(a) $COV_l = 0$ *(b)* $COV_l = 0.3$

(c) $COV_l = 0.5$ *(d)* $COV_l = 0.7$

(e) $COV_l = 0.9$

Figure 14. Shows the PFI scores for variable 1, for which the correlation to the outcome is depicted on the y-axis, with a set COV_l and varying both COV_k and COV_l using Ridge regression. Note that when two of the correlations reach a certain high, the third correlation had to be changed accordingly. When two of the correlations are set to 0.5-0.9, the third had to be at least 0.1, at 0.7-0.9 it had to be at least 0.4, at 0.9-0.9 at least 0.7.

4 Discussion

Different cases were found, in which PFI behaved differently. In some, PFI accurately captures the underlying data structure, in others, the PFI scores are higher or lower than the underlying correlation would suggest. The findings reinforce earlier criticism of the PFI, namely that it is biased in correlated variables (Strobl et al., 2008; Nicodemus et al., 2010; Gregorutti et al., 2017; Genuer et al., 2010; Tolosi & Lengauer, 2011; Hooker et al., 2021).

4.1 When PFI does Work

When predictor variables were uncorrelated, the PFI score accurately depicted the correlation between the predictor variables and the outcome. The resulting PFI was equal to two times the squared correlation. Even when the model could not adequately capture the relationship between the predictor and the outcome, the PFI score still accurately depicted the underlying correlation. It is unclear why this is the case and should be examined in future research. PFI also was applicable when both predictor variables were equally correlated to the outcome and were correlated to each other. In this case the PFI score was lower than two times the squared correlation, but the resulting score was equal to the formula given in 8. It is assumed that when one variable is permuted, the other can cover the lost information. This was also found in previous research (Tolosi & Lengauer, 2011; Pereira et al., 2022). It is assumed that the two variables carry the same information, which is also indicated by the lower overall R^2 shown in the simulations.

4.2 When PFI does Underestimate the Importance of a Predictor

As mentioned, in cases, when multiple predictors are correlated to the outcome and to each other, the resulting PFI score is lower than expected. In some cases, the difference can still be depicted by the given formula. In other instances, PFI score was also lower than two times the squared correlation but could not be depicted in any clear formula. This occurred, when one predictor variable was correlated stronger with the outcome than the other predictor variable, and the predictors were correlated. In some cases, one of the variables reached a PFI score of 0 despite being connected to the outcome. It is probable that in the tree building process at every split the variable with a lower correlation simply was outdone in Gini reduction by the other variable, so it was hardly used at all during the tree building process. So, even when the variable was permuted, some splits might result in going down the wrong branch of each tree, but since the

other variable was still more important, the information given was still enough to make adequate predictions. Similarly, in RR, the model might have only used the predictor variable with the stronger correlation to the outcome during the model training process and ignored the other predictor.

4.3 When PFI does Overestimate the Importance of a Predictor

In some instances, when one predictor has a higher correlation to the outcome than the other predictor, but they are correlated to each other, seemingly an interaction effect arises, as the achieved R^2 was higher than expected, and at the same time the PFI scores were higher than expected for one or both variables. This is also similar to previous studies, which found overestimation in correlated variables (Strobl et al., 2008; Nicodemus et al., 2010). In these cases, it is assumed that extrapolation occurs, and because of this, the model performs very poorly, leading to high PFI scores. It is also likely that an interaction occurs between the two predictors, and it has been argued previously that PFI can detect interactions (Auret & Aldrich, 2011; Rodenburg et al., 2008; Strobl & Zeileis, 2008; Casalicchio et al., 2019). However, it is unclear how exactly this interaction is reflected in the PFI score, as in this study, when an interaction was present, the PFI scores were often much higher than reasonable, even when accounting for some interaction effect.

Contrary to previous research, we did not find that overestimation disappeared in larger samples (Hooker et al., 2021), but sample size was not varied in this simulation, so nothing conclusive can be stated. Based on the simulation, it is also unclear, if extrapolation is the reason for the overestimation in PFI scores, as previously suspected (Hooker et al., 2021). However, contrary to earlier criticism, we did not find, that linear models suffer less from the extrapolation bias (Hooker, 2007; Hooker et al., 2021), as RR produced even higher PFI scores than RF in most cases.

4.4 Further Criticism of PFI

Previous research also pointed at instability of PFI due to data perturbations (Calle & Urrea, 2010). In this study, however, PFI was found to be quite stable, producing a standard deviation of 0.05 or less in most cases over 100 iterations. This dissimilarity might also reflect that RF may be intrinsically unstable (Wang et al., 2016), and different RF models might place different emphasis on different variables due to data perturbations. Therefore, the suspected instability of PFI might only reflect the instability of the RF, without leading to any conclusions about the behavior of PFI. This is further exemplified by the dependency of the

PFI score on hyperparameters in correlated variables. This instability might be more pronounced in the presence of more variables, but in this simulation only two variables were used. It was also found that this instability decreases with rising number of trees in a forest but does not disappear (Wang et al., 2016).

As PFI studies are closely connected to RF, it is possible that much of the criticism related to the behavior of PFI might reflect the behavior of RF. Most studies so far on PFI are done on a RF model and most use the OOB-version of PFI, which is only applicable in RF models (Strobl, Boulesteix, Zeileis, & Hothorn, 2007; Strobl et al., 2008; Janitza et al., 2013; Nicodemus et al., 2010; Gregorutti et al., 2015; Genuer et al., 2010; Tolosi & Lengauer, 2011; Nicodemus & Malley, 2009; Nicodemus, 2011; Calle & Urrea, 2010; Wang et al., 2016; Verikas et al., 2011). It was argued that PFI might be preferring variables with a large number of categories (Strobl, Boulesteix, Zeileis, & Hothorn, 2007), but similar criticism was connected to RF (Altmann et al., 2010). It was also argued that PFI are biased in imbalanced classes, but the same criticism is extended to RF in general (Janitza et al., 2013) Further, while it was found that PFI can both over and underestimate the importance of correlated variables (Nicodemus et al., 2010; Gregorutti et al., 2017), it was also found that the tree building algorithm in RF might prefer correlated variables (Strobl et al., 2008), or even uncorrelated variables (Nicodemus & Malley, 2009). Therefore, the bias of PFI might just reflect the bias of RF. The simulation in this paper, however, has shown, that the over and underestimation of correlated variables is not exclusive to RF, as a similar bias was shown in RR. Other mentioned biases were not assessed in the present simulations.

4.5 Applications of PFI

Possible applications for PFI are auditing and debugging a model. Despite the shown bias of PFI, this might still be a useful application, if one has some previous knowledge about the relationship captured by the model and thereby can use PFI to expose some obvious flaws, for example when a variable that is known to be important is not seen as important by the model, or a variable known to not be important is seen as important. This is however still very limited in correlated data. PFI is further not useful for giving insight into how a model makes a specific decision, as it always uses the whole dataset and can only be used to infer how important a variable is for the general performance of the model.

Another application for PFI is testing and as mentioned above, many different attempts to make tests for PFI have been made (Hapfelmeier et al., 2023; Nembrini, 2018; Molnar, Freiesleben, et al., 2023). The possibility of using PFI

for tests is however limited, as PFI scores are potentially misleading, with both over and underestimation happening. Especially in correlated features, PFI is not reliable, so neither is any test and test statistic derived from the PFI score. Further, it is often unclear what is even tested. In linear regression, it is assumed that the data generating model is a linear model, which is then estimated given the observed data. Then a coefficient and therefore the variable as a whole can be tested if it influences the outcome given the linear model and the observed data. When using ML methods, hardly any assumptions are made for the data and the underlying data generating process. This makes them widely applicable, but also significantly limits the scope of possible conclusions which can be drawn from the model. Since there are no assumptions made about the underlying data generating process, it is also not assumed, that the trained ML model resembles this process. All that is expected of the model, is that it makes accurate predictions. So, even when understanding the ML model and which variables are placed more emphasis on than others (which is also a difficult goal to achieve), it cannot be argued that the true underlying data generation process also places importance on this variable. In fact, it was shown that various different models might work equally well, but place emphasis on different variables, this being called the "Rashomon-effect" (Breiman, 2001b). To sidestep this, some researchers have argued, that while singular models might place emphasis on different variables than the underlying data generation process, using a multitude of different models to estimate which variables are used in the underlying data generation process might lead to selecting the most important variables (Fisher et al., 2019). This, however, also relies on some assumptions, which might not be given. For example, it relies on the assumption that models are more likely to choose the "real" important variable. The authors themselves also recognize this problem and only claim that their method can show how important or unimportant the variable in question can be to any well-performing model, disregarding any connection to the true data generating process.

4.6 Alternatives for PFI

A possible alternative to PFI, which was developed to deal with correlated variables, is conditional PFI (Strobl et al., 2008; Debeer & Strobl, 2020; Molnar, König, et al., 2023), but was not explored in this simulation. While it has been argued, that conditional PFI leads to different conclusions than regular PFI (Molnar et al., 2022), conditional PFI might still be a reasonable VIM for correlated variables, although it should not be used as a simple substitute.

There are also other VIM which suffer from similar problems as PFI, especially when they are also based on permutations. SHAP is one possible alter-

native to PFI, but in many implementations it is also based on permutations, as it would not be feasible to calculate the exact SHAP value, but only to approximate it (Mase, Owen, & Seiler, 2019).

Other methods that retrain the model without the variable or with a permuted variable could give a remedy but are computationally much more expensive (Hooker et al., 2021). There is also more instability and uncertainty introduced by retraining a model, especially in models that have an inherent randomness built in their algorithm, such as RF. These methods might then compare two different models, which might be built very differently and lead to different conclusions when compared.

4.7 Limitations

In this simulation, as well as in previous works, only linear correlations between the predictors and between the predictors and the outcome were examined. Different types of correlation might lead to different behaviors of the PFI score. In the given regression task, the PFI score depicts the squared correlation, when the predictors are uncorrelated, however, it was not examined, what the PFI score depicts in classification tasks. Also, the used predictors were linear as well, and PFI might behave differently for categorical predictors. For example, the PFI has been argued to be biased towards variables with many different categories (Strobl, Boulesteix, Zeileis, & Hothorn, 2007).

As the loss function, the measure R^2 was used, but different measures can be used to estimate PFI. Using different loss functions, PFI might behave differently, as for example it was argued that using entropy, PFI does not suffer from correlation bias (Wood et al., 2024).

For this simulation a different data-generating process was used compared to other studies, where the outcome variable is generated via a linear combination of the predictors and random noise (Hooker et al., 2021). The defined VC matrices could lead to weirdly structured data, as the data-generating process is forced to make the predictor variables highly correlated but one is hardly or not at all correlated to the outcome. This might have led to the observed interaction effect, increasing the R^2 and the PFI scores.

In the present simulation study, RF models often achieved slightly lower R^2 than RR models. While RR might be more appropriate in this context, as the relationship between the predictor variables and the outcome was linear, and RF can only approach a linear relationship by making smaller and smaller steps, it could also be due to not ideal hyperparameters. In the simulation, a grid search was performed to choose the best hyperparameters for RF, while a random search was performed to choose the best penalty term for RR. Due

to the strict predefinition of possible values in grid search, the optimal or even near optimal combination of hyperparameters might not have been in the set of possible hyperparameters, as defined by the grid. Using a different approach for hyperparameter optimization might have led to a better performance of the RF.

4.8 Conclusions

The question remains whether PFI can be useful in applications. From the results seen, only in cases where the variables are not correlated, PFI should be used. In cases where predictor variables are correlated, PFI is biased, and can produce highly unrealistic values. So, either for model auditing, variable selection, or testing, the use of PFI is highly restricted based on the distribution of the predictor variables used in a model. Alternatives either suffer from similar problems or cannot be counted as a simple substitute but rather provide different information.

It seems that new methods are needed to find solid variable importance tools, however, as often stated "There is no free lunch" (Wolpert & Macready, 1997), and no single method is best for all use cases. This might also be the case for variable importance, and for different applications, different methods might be used and useable. In this regard, PFI might still have a place in certain applications, but only under the restriction that the predictor variables are not correlated.

References

Alharbi, W. S., & Rashid, M. (2022). A review of deep learning applications in human genomics using next-generation sequencing data. *Human Genomics*, *16*(1), 26.

Altmann, A., Toloşi, L., Sander, O., & Lengauer, T. (2010). Permutation importance: a corrected feature importance measure. *Bioinformatics*, *26*(10), 1340–1347. doi: 10.1093/bioinformatics/btq134

Antoniadis, A., Lambert-Lacroix, S., & Poggi, J.-M. (2021). Random forests for global sensitivity analysis: A selective review. *Reliab. Eng. Syst. Saf.*, *206*, 107312.

Archer, K. J., & Kimes, R. V. (2008). Empirical characterization of random forest variable importance measures. *Computational Statistics & Data Analysis*, *52*(4), 2249-2260. doi: https://doi.org/10.1016/j.csda.2007.08.015

Au, Q., Herbinger, J., Stachl, C., Bischl, B., & Casalicchio, G. (2022). Grouped feature importance and combined features effect plot. *Data Mining and Knowledge Discovery*, *36*(4), 1401–1450. doi: 10.1007/s10618-022-00840-5

Auret, L., & Aldrich, C. (2011). Empirical comparison of tree ensemble variable importance measures. *Chemometrics and Intelligent Laboratory Systems*, *105*(2), 157-170. doi: https://doi.org/10.1016/j.chemolab.2010.12.004

Azarmi, M., Rezaei, M., Wang, H., & Arabian, A. (2024). Feature importance in pedestrian intention prediction: A context-aware review. *arXiv preprint arXiv:2409.07645*.

Bellman, R. (1966). Dynamic programming. *science*, *153*(3731), 34–37.

Blum, A. L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, *97*(1), 245-271. doi: https://doi.org/10.1016/S0004-3702(97)00063-5

Boulesteix, A.-L., Bender, A., Lorenzo Bermejo, J., & Strobl, C. (2011). Random forest gini importance favours snps with large minor allele frequency: impact, sources and recommendations. *Briefings in Bioinformatics*, *13*(3), 292-304. doi: 10.1093/bib/bbr053

Breiman, L. (2001a). Random forests. *Machine learning*, *45*, 5–32. doi: https://doi.org/10.1023/A:1010933404324

Breiman, L. (2001b). Statistical Modeling: The Two Cultures. *Statistical Science*, *16*(3), 199 – 231. doi: 10.1214/ss/1009213726

Bénard, C., Da Veiga, S., & Scornet, E. (2022). Mean decrease accuracy for random forests: inconsistency, and a practical solution via the Sobol-MDA. *Biometrika*, *109*(4), 881-900. doi: 10.1093/biomet/asac017

Calle, M. L., & Urrea, V. (2010). Letter to the editor: Stability of random forest importance measures. *Briefings in Bioinformatics*, *12*(1), 86–89. doi: 10.1093/bib/bbq011

Cappelli, F., & Grimaldi, S. (2023). Feature importance measures for hydrological applications: insights from a virtual experiment. *Stochastic Environmental Research and Risk Assessment*, *37*, 1-19. doi: 10.1007/s00477-023-02545-7

Card, Q., Simpson, D., Aryal, K., Gupta, M., & Islam, S. R. (2024). Explainable Deep Learning Models for Dynamic and Online Malware Classification . In *2024 ieee international conference on smart computing (smartcomp)*

(p. 182-189). Los Alamitos, CA, USA: IEEE Computer Society. doi: 10.1109/SMARTCOMP61445.2024.00045

Casalicchio, G., Molnar, C., & Bischl, B. (2019). Visualizing the feature importance for black box models. In M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, & G. Ifrim (Eds.), *Machine learning and knowledge discovery in databases* (pp. 655–670). Cham: Springer International Publishing.

Chavent, M., Lacaille, J., Mourer, A., & Olteanu, M. (2021). Handling correlations in random forests: which impacts on variable importance and model interpretability? *ESANN 2021 proceedings*.

Chowdhury, M. Z. I., & Turin, T. C. (2020). Variable selection strategies and its importance in clinical prediction modelling. *Family medicine and community health*, *8*(1).

Chuan, L., & Quanyuan, F. (2007). The standard particle swarm optimization algorithm convergence analysis and parameter selection. In *Third international conference on natural computation (icnc 2007)* (Vol. 3, p. 823-826). doi: 10.1109/ICNC.2007.746

Combs, J. G. (2010). Big samples and small effects: Let's not trade relevance and rigor for power. *Academy of Management Journal*, *53*(1), 9-13. doi: 10.5465/amj.2010.48036305

Datta, A., Sen, S., & Zick, Y. (2016). Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 ieee symposium on security and privacy (sp)* (p. 598-617). doi: 10.1109/SP.2016.42

Debeer, D., & Strobl, C. (2020). Conditional permutation importance revisited. *BMC Bioinformatics*, *21*(1), 307. doi: 10.1186/s12859-020-03622-2

Díaz-Uriarte, R., & de Andrés, S. A. (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, *7*, 3 - 3.

Eichstaedt, J. C., Schwartz, H. A., Kern, M. L., Park, G., Labarthe, D. R., Merchant, R. M., ... Seligman, M. E. P. (2015). Psychological language on twitter predicts county-level heart disease mortality. *Psychological Science*, *26*(2), 159-169. doi: 10.1177/0956797614557867

Fisher, A., Rudin, C., & Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, *20*(177), 1–81.

Freiesleben, T., & König, G. (2023). Dear xai community, we need to talk! In L. Longo (Ed.), *Explainable artificial intelligence* (pp. 48–65). Cham: Springer Nature Switzerland.

Fumagalli, F., Muschalik, M., Hüllermeier, E., & Hammer, B. (2023). Incremental permutation feature importance (ipfi): towards online explanations on data streams. *Machine Learning*, *112*(12), 4863–4903. doi: 10.1007/s10994-023-06385-y

Genuer, R., Poggi, J.-M., & Tuleau-Malot, C. (2010). Variable selection using random forests. *Pattern Recognition Letters*, *31*(14), 2225-2236. doi: https://doi.org/10.1016/j.patrec.2010.03.014

Ghandeharioun, A., Fedor, S., Sangermano, L., Ionescu, D., Alpert, J., Dale, C., ... Picard, R. (2017). Objective assessment of depressive symptoms with machine learning and wearable sensors data. In *2017 seventh international conference on affective computing and intelligent interaction (acii)* (p. 325-332). doi: 10.1109/ACII.2017.8273620

Gregorutti, B. (2015). *Random forests and variable selection : analysis of the flight data recorders for aviation safety* (PhD thesis). Université Pierre et Marie Curie - Paris VI.

Gregorutti, B., Michel, B., & Saint-Pierre, P. (2015). Grouped variable importance with random forests and application to multiple functional data analysis. *Computational Statistics & Data Analysis*, *90*, 15-35. doi: https://doi.org/10.1016/j.csda.2015.04.002

Gregorutti, B., Michel, B., & Saint-Pierre, P. (2017). Correlation and variable importance in random forests. *Statistics and Computing*, *27*. doi: 10 .1007/s11222-016-9646-1

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, *3*, 1157–1182.

Hapfelmeier, A., Hornung, R., & Haller, B. (2023). Efficient permutation testing of variable importance measures by the example of random forests. *Computational Statistics & Data Analysis*, *181*, 107689. doi: https://doi.org/10.1016/j.csda.2022.107689

Hapfelmeier, A., Hothorn, T., Ulm, K., & Strobl, C. (2014). A new variable importance measure for random forests with missing data. *Statistics and Computing*, *24*(1), 21–34. doi: 10.1007/s11222-012-9349-1

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. doi: 10.1038/s41586-020-2649-2

Hassija, V., Chamola, V., Mahapatra, A., Singal, A., Goel, D., Huang, K., ... Hussain, A. (2024). Interpreting black-box models: a review on explainable artificial intelligence. *Cognitive Computation*, *16*(1), 45–74.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (Vol. 2). Springer.

Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, *44*(1), 1–12.

Head, M. L., Holman, L., Lanfear, R., Kahn, A. T., & Jennions, M. D. (2015). The extent and consequences of p-hacking in science. *PLoS biology*, *13*(3), e1002106.

Heinze, G., Wallisch, C., & Dunkler, D. (2018). Variable selection–a review and recommendations for the practicing statistician. *Biometrical journal*, *60*(3), 431–449.

Hier, D. B., Do, T. S., & Obafemi-Ajayi, T. (2024). When less is not more: Large language models normalize less-frequent terms with lower accuracy. *arXiv preprint arXiv:2409.13746*.

Hindman, M. (2015). Building better models: Prediction, replication, and machine learning in the social sciences. *The ANNALS of the American Academy of Political and Social Science*, *659*(1), 48-62. doi: 10.1177/ 0002716215570279

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55–67. doi: 10.2307/ 1267351

Hooker, G. (2007). Generalized functional anova diagnostics for high-dimensional functions of dependent variables. *Journal of Computational and Graphical Statistics*, *16*(3), 709–732.

Hooker, G., & Mentch, L. (2021). Bridging breiman's brook: From algorithmic modeling to statistical learning. *Observational Studies*, *7*, 107-125. doi:

10.1353/obs.2021.0027

Hooker, G., Mentch, L., & Zhou, S. (2021). Unrestricted permutation forces extrapolation: variable importance requires at least one more model, or there is no free variable importance. *Statistics and Computing*, *31*, 1–16.

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, *15*(3), 651–674.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. doi: 10.1109/MCSE.2007.55

Ioannidis, J. P. (2005). Why most published research findings are false. *PLoS medicine*, *2*(8), e124.

Ishwaran, H., & Lu, M. (2019). Standard errors and confidence intervals for variable importance in random forest regression, classification, and survival. *Statistics in Medicine*, *38*(4), 558-582. doi: https://doi.org/10.1002/sim.7803

Janitza, S., Strobl, C., & Boulesteix, A.-L. (2013). An auc-based permutation variable importance measure for random forests. *BMC bioinformatics*, *14*, 119. doi: 10.1186/1471-2105-14-119

Joel, S., Eastwick, P. W., & Finkel, E. J. (2017). Is romantic desire predictable? machine learning applied to initial romantic attraction. *Psychological Science*, *28*(10), 1478-1489. doi: 10.1177/0956797617714580

Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., ... Kim, B. (2019). The (un)reliability of saliency methods. In W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, & K.-R. Müller (Eds.), *Explainable ai: Interpreting, explaining and visualizing deep learning* (pp. 267–280). Cham: Springer International Publishing. doi: 10.1007/978-3-030-28954-6_14

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., ... Willing, C. (2016). Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides & B. Schmidt (Eds.), *Positioning and power in academic publishing: Players, agents and agendas* (p. 87 - 90).

Kucukosmanoglu, M., Garcia, J. O., Brooks, J., & Bansal, K. (2024). Cognitive networks and performance drive fmri-based state classification using dnn models. *arXiv preprint arXiv:2409.00003*.

König, G., Molnar, C., Bischl, B., & Grosse-Wentrup, M. (2021). Relative feature importance. In *2020 25th international conference on pattern recognition (icpr)* (p. 9318-9325). doi: 10.1109/ICPR48806.2021.9413090

Lakkaraju, H., & Bastani, O. (2020). "how do i fool you?": Manipulating user trust via misleading black box explanations. In *Proceedings of the aaai/acm conference on ai, ethics, and society* (p. 79–85). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3375627.3375833

Lantz, B. (2013). The large sample size fallacy. *Scandinavian Journal of Caring Sciences*, *27*(2), 487-492. doi: https://doi.org/10.1111/j.1471-6712.2012.01052.x

Libbrecht, M. W., & Noble, W. S. (2015). Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, *16*(6), 321–332.

Loh, W.-Y. (2014). Fifty years of classification and regression trees. *International Statistical Review*, *82*(3), 329–348.

Mase, M., Owen, A. B., & Seiler, B. (2019). Explaining black box decisions by shapley cohort refinement. *CoRR*, *abs/1911.00467*.

Meng, Y., Yu, Y., Cupples, L., Farrer, L., & Lunetta, K. (2009). Performance of random forest when snps are in linkage disequilibrium. *BMC bioinformatics*, *10*, 78. doi: 10.1186/1471-2105-10-78

Mentch, L., & Zhou, S. (2022). Getting better from worse: Augmented bagging and a cautionary tale of variable importance. *Journal of Machine Learning Research*, *23*(224), 1–32.

Molnar, C. (2025). *Interpretable machine learning* (3rd ed.). Retrieved from `https://christophm.github.io/interpretable-ml-book`

Molnar, C., Freiesleben, T., König, G., Herbinger, J., Reisinger, T., Casalicchio, G., ... Bischl, B. (2023). Relating the partial dependence plot and permutation feature importance to the data generating process. In L. Longo (Ed.), *Explainable artificial intelligence* (pp. 456–479). Cham: Springer Nature Switzerland.

Molnar, C., König, G., Herbinger, J., Freiesleben, T., Dandl, S., Scholbeck, C. A., ... Bischl, B. (2022). General pitfalls of model-agnostic interpretation methods for machine learning models. In A. Holzinger, R. Goebel, R. Fong, T. Moon, K.-R. Müller, & W. Samek (Eds.), *xxai - beyond explainable ai: International workshop, held in conjunction with icml 2020, july 18, 2020, vienna, austria, revised and extended papers* (pp. 39–68). Cham: Springer International Publishing. doi: 10.1007/978-3-031-04083-2_4

Molnar, C., König, G., Bischl, B., & Casalicchio, G. (2023). Model-agnostic feature importance and effects with dependent features: a conditional subgroup approach. *Data Mining and Knowledge Discovery*, *38*, 1-39. doi: 10.1007/s10618-022-00901-9

Nembrini, S. (2018). On what to permute in test-based approaches for variable importance measures in random forests. *Bioinformatics*, *35*. doi: 10.1093/bioinformatics/bty1025

Nicodemus, K. K. (2011). Letter to the editor: On the stability and ranking of predictors from random forest variable importance measures. *Briefings in bioinformatics*, *12 4*, 369-73.

Nicodemus, K. K., & Malley, J. D. (2009). Predictor correlation impacts machine learning algorithms: implications for genomic studies. *Bioinformatics*, *25*(15), 1884–1890. doi: 10.1093/bioinformatics/btp331

Nicodemus, K. K., Malley, J. D., Strobl, C., & Ziegler, A. (2010). The behaviour of random forest permutation-based variable importance measures under predictor correlation. *BMC Bioinformatics*, *11*, 110 - 110.

Pascoal, C., Oliveira, M. R., Pacheco, A., & Valadas, R. (2017). Theoretical evaluation of feature selection methods based on mutual information. *Neurocomputing*, *226*, 168-181. doi: https://doi.org/10.1016/j.neucom.2016.11.047

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pereira, J. P. B., Stroes, E. S. G., Zwinderman, A. H., & Levin, E. (2022). Covered information disentanglement: Model transparency via unbiased permutation importance. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(7), 7984-7992. doi: 10.1609/aaai.v36i7.20769

Pfeifer, B., Krzyzinski, M., Baniecki, H., Saranti, A., Holzinger, A., & Biecek, P. (2023). Explaining and visualizing black-box models through counterfactual paths. *arXiv preprint arXiv:2307.07764*.

Plonsky, O., Erev, I., Hazan, T., & Tennenholtz, M. (2017). Psychological forest: Predicting human behavior. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1). doi: 10.1609/aaai.v31i1.10613

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (p. 1135–1144). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/2939672.2939778

Rodenburg, W., Heidema, A. G., Boer, J. M. A., Bovee-Oudenhoven, I. M. J., Feskens, E. J. M., Mariman, E. C. M., & Keijer, J. (2008). A framework to identify physiological responses in microarray-based gene expression studies: selection and interpretation of biologically relevant genes. *Physiological Genomics*, *33*(1), 78-90. doi: 10.1152/physiolgenomics.00167.2007

Rodrigues, F. A. (2023). Machine learning in physics: a short guide. *Europhysics Letters*, *144*(2), 22001.

Rosenbusch, H., Soldner, F., Evans, A. M., & Zeelenberg, M. (2021). Supervised machine learning methods in psychology: A practical introduction with annotated r code. *Social and Personality Psychology Compass*, *15*(2), e12579. doi: https://doi.org/10.1111/spc3.12579

Ruoqing Zhu, D. Z., & Kosorok, M. R. (2015). Reinforcement learning trees. *Journal of the American Statistical Association*, *110*(512), 1770–1784. doi: 10.1080/01621459.2015.1036994

Scholbeck, C. A., Funk, H., & Casalicchio, G. (2023). Algorithm-agnostic feature attributions for clustering. In L. Longo (Ed.), *Explainable artificial intelligence* (pp. 217–240). Cham: Springer Nature Switzerland.

Schwartz, H. A., Eichstaedt, J. C., Kern, M. L., Dziurzynski, L., Ramones, S. M., Agrawal, M., ... Ungar, L. H. (2013). Personality, gender, and age in the language of social media: The open-vocabulary approach. *PLOS ONE*, *8*(9), 1-16. doi: 10.1371/journal.pone.0073791

Shi, Y.-F., Yang, Z.-X., Ma, S., Kang, P.-L., Shang, C., Hu, P., & Liu, Z.-P. (2023). Machine learning for chemistry: Basics and applications. *Engineering*, *27*, 70-83. doi: https://doi.org/10.1016/j.eng.2023.04.013

Sidey-Gibbons, J. A., & Sidey-Gibbons, C. J. (2019). Machine learning in medicine: a practical introduction. *BMC medical research methodology*, *19*, 1–18.

Simmons, J. P., Nelson, L. D., & Simonsohn, U. (2011). False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological science*, *22*(11), 1359–1366.

Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2020). Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the aaai/acm conference on ai, ethics, and society* (p. 180–186). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3375627.3375830

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 25). Curran Associates, Inc.

Sood, A., & Craven, M. (2022). Feature importance explanations for temporal black-box models. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 36, pp. 8351–8360).

Strobl, C., Boulesteix, A.-L., & Augustin, T. (2007). Unbiased split selection for classification trees based on the gini index. *Computational Statistics & Data Analysis*, *52*(1), 483-501. doi: https://doi.org/10.1016/j.csda.2006.12.030

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., & Zeileis, A. (2008). Conditional variable importance for random forests. *BMC Bioinformatics*, *9*(307), 1 – 11. doi: 10.1186/1471-2105-9-307

Strobl, C., Boulesteix, A.-L., Zeileis, A., & Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics*, *8*, 1–21.

Strobl, C., & Zeileis, A. (2008). *Danger: High power! - exploring the statistical properties of a test for random forest variable importance* (WorkingPaper No. 17). doi: 10.5282/ubm/epub.2111

The pandas development team. (2020). *pandas-dev/pandas: Pandas.* Zenodo. doi: 10.5281/zenodo.3509134

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267-288. doi: https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Tolosi, L., & Lengauer, T. (2011). Classification with correlated features: unreliability of feature ranking and solutions. *Bioinformatics (Oxford, England)*, *27*(14), 1986—1994. doi: 10.1093/bioinformatics/btr300

Van Rossum, G., & Drake Jr, F. L. (1995). *Python reference manual.* Centrum voor Wiskunde en Informatica Amsterdam.

Verikas, A., Gelzinis, A., & Bacauskiene, M. (2011). Mining data with random forests: A survey and results of new tests. *Pattern Recognition*, *44*(2), 330-349. doi: https://doi.org/10.1016/j.patcog.2010.08.011

Verma, S. (2022). Development of interpretable machine learning models to detect arrhythmia based on ecg data. *arXiv preprint arXiv:2205.02803*.

Walsh, C. G., Ribeiro, J. D., & Franklin, J. C. (2017). Predicting risk of suicide attempts over time through machine learning. *Clinical Psychological Science*, *5*(3), 457-469. doi: 10.1177/2167702617691560

Wang, H., Yang, F., & Luo, Z. (2016). An experimental study of the intrinsic stability of random forest variable importance measures. *BMC Bioinformatics*, *17*. doi: 10.1186/s12859-016-0900-5

Wicaksono, A. S., & Supianto, A. A. (2018). Hyper parameter optimization using genetic algorithm on machine learning methods for online news popularity prediction. *International Journal of Advanced Computer Science and Applications*, *9*(12). doi: 10.14569/IJACSA.2018.091238

Wicherts, J. M., Veldkamp, C. L., Augusteijn, H. E., Bakker, M., Van Aert, R. C., & Van Assen, M. A. (2016). Degrees of freedom in planning, running, analyzing, and reporting psychological studies: A checklist to avoid p-hacking. *Frontiers in psychology*, *7*, 1832.

Wolpert, D., & Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67-82. doi: 10.1109/4235.585893

Wood, D., Papamarkou, T., Benatan, M., & Allmendinger, R. (2024). Model-agnostic variable importance for predictive uncertainty: an entropy-based

approach. *Data Mining and Knowledge Discovery*, *38*, 4184-4216. doi: 10.1007/s10618-024-01070-7

Wu, H., Ruan, W., Wang, J., Zheng, D., Liu, B., Geng, Y., ... Helal, S. (2023). Interpretable machine learning for covid-19: An empirical study on severity prediction task. *IEEE Transactions on Artificial Intelligence*, *4*(4), 764–777. doi: 10.1109/tai.2021.3092698

Yarkoni, T., & Westfall, J. (2017). Choosing prediction over explanation in psychology: Lessons from machine learning. *Perspectives on Psychological Science*, *12*(6), 1100-1122. doi: 10.1177/1745691617693393

Ying, X. (2019). An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, *1168*(2), 022022. doi: 10.1088/1742-6596/1168/2/022022

Zacarias, O. P., & Boström, H. (2013). Comparing support vector regression and random forests for predicting malaria incidence in mozambique. In *2013 international conference on advances in ict for emerging regions (icter)* (p. 217-221). doi: 10.1109/ICTer.2013.6761181

Zhang, H., Su, Y., Altaf, F., Wik, T., & Gros, S. (2023). Interpretable battery cycle life range prediction using early cell degradation data. *IEEE Transactions on Transportation Electrification*, *9*(2), 2669–2682. doi: 10.1109/tte.2022.3226683

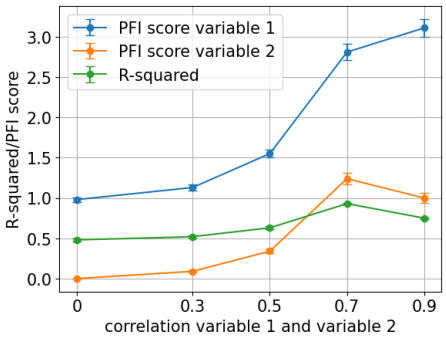
A List of Figures

List of Figures

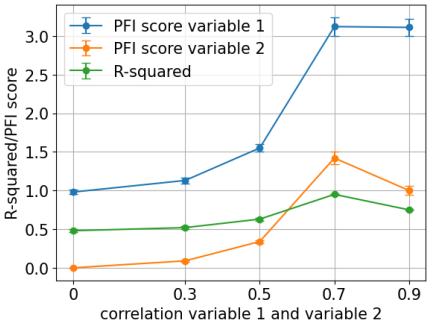
B List of Tables

List of Tables

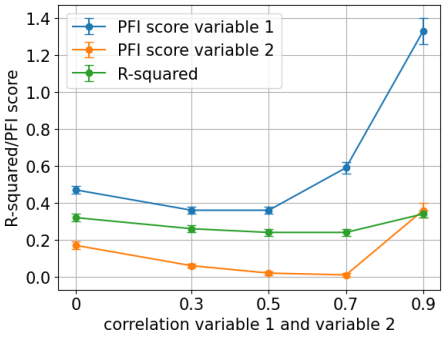
C Alternate plots



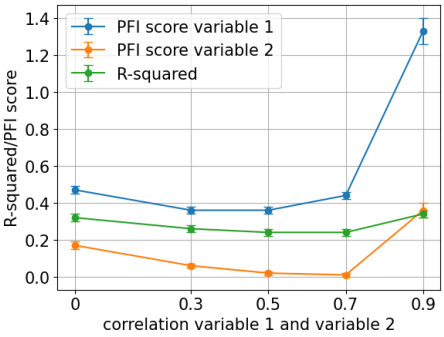
(a) Alternate plot



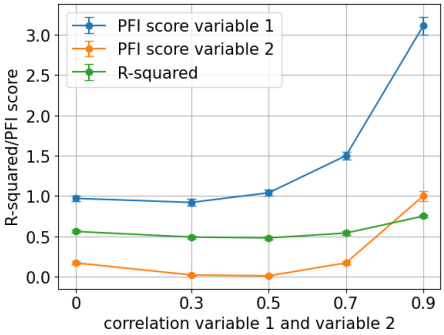
(b) Used plot



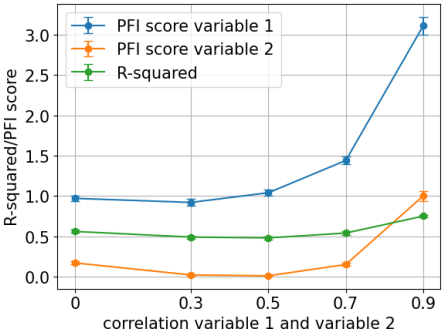
(c) Alternate plot



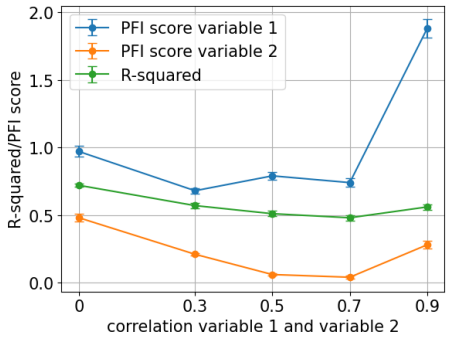
(d) Used plot



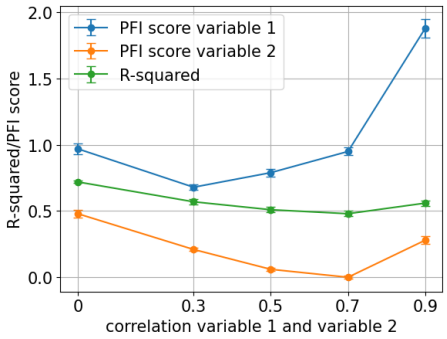
(e) Alternate plot



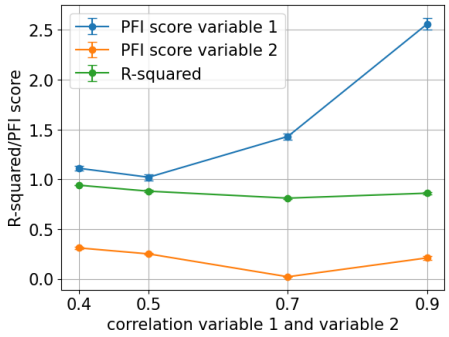
(f) Used plot



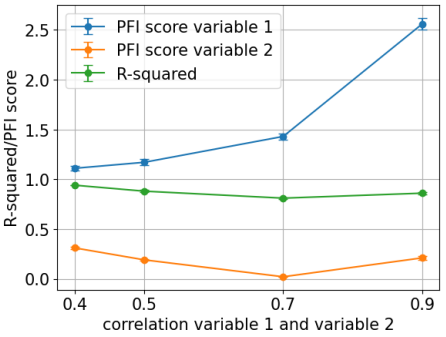
(g) Alternate plot



(h) Used plot



(i) Alternate plot



(j) Used plot

Figure 15. Compares the plots shown in the results section, with plots created with the results found, when a redone simulation did not find the same resulting PFI scores as the initial simulation. In all the plots the correlation between the two variables was varied. In (a) and (b) $COV_k = 0.7$ and $COV_j = 0.0$, in (c) and (d) $COV_k = 0.5$ and $COV_j = 0.3$, in (e) and (f) $COV_k = 0.7$ and $COV_j = 0.3$, in (g) and (h) $COV_k = 0.7$ and $COV_j = 0.5$, in (c) and (d) $COV_k = 0.9$ and $COV_j = 0.7$.

D Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rcParams
import matplotlib.cm as cm
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.inspection import permutation_importance
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.utils.fixes import loguniform
from sklearn.linear_model import Ridge
from sklearn.model_selection import RandomizedSearchCV

# Due to simplicity, all possible combinations are calculated at once
# It is recommended to split up the matrices
# Especially in RF, this will take a very long time otherwise

# define all possible VC-matrices
mats = []
# choose the possible covariances
k = [0,0.3,0.5,0.7,0.9]
j = [0,0.3,0.5,0.7,0.9]
l = [0,0.3,0.5,0.7,0.9]
for q in j:
    for i in k:
        for m in l:
            matrix = [[1, i, q], [i, 1, m],[q, m, 1]]
            if all(np.linalg.eig(matrix)[0]>=0):
                mats.append(matrix)
            else:
                values = np.array([q, i, m])
                # Find the minimum index
                min_index = np.argmin(values)
                values[min_index] = 0.1
                matrix = [[1, values[1], values[0]],
                [values[1], 1, values[2]],
                [values[0], values[2], 1]]
                if all(np.linalg.eig(matrix)[0]>=0):
                    mats.append(matrix)
                else:
                    values[min_index] = 0.4
                    matrix = [[1, values[1], values[0]],
                    [values[1], 1, values[2]],
                    [values[0], values[2], 1]]
                    if all(np.linalg.eig(matrix)[0]>=0):
                        mats.append(matrix)
```

```
                    else:
                        values[min_index] = 0.7
                        matrix = [[1, values[1], values[0]],
                        [values[1], 1, values[2]],
                        [values[0], values[2], 1]]
                        mats.append(matrix)
            print(matrix)

# define the number of datapoints to create n
# and the number of iterations for each simulation n
n = 5000
m = 100
#set parameters for plots
plt.rcParams.update({'font.size': 15})

#For Random Forest
#create empty lists to store
# the best estimator
# the R-squared for each iteration and simulation
# the PFI scores for both variables
best_estimator_RF = []
r2_list_RF = [[0]*m for _ in range(len(mats))]
imp1_list_RF = [[0]*m for _ in range(len(mats))]
imp2_list_RF = [[0]*m for _ in range(len(mats))]
# loop through all the matrices
for q in range(len(mats)):
    # define the means and the matrix
    mean = [1,1,1]
    matrix = mats[q]
    # create the datapoints for the hyperparameter tuning
    # split them into X and y
    datamat = np.random.multivariate_normal(mean, matrix, n)
    y = datamat[:,0]
    X = pd.DataFrame(datamat[:,(1,2)])
    # define the distribution for the hyperparameter
    parameters = {'max_features': [1, 2],
                  'min_samples_leaf': [1, 10, 20, 30, 40, 50],
                  'min_samples_split': [2, 5, 10, 15, 20],
                  'n_estimators': [100, 250, 400, 550, 700]}
    rf_regr = RandomForestRegressor()
    CV = GridSearchCV(estimator = rf_regr,
    param_grid = parameters, cv = 5)
    CV.fit(X, y)
    best_estimator_RF.append(CV.best_estimator_)
    # If desired print the VC matrix and the found best estimator
    print(matrix)
    print('best estimator:', CV.best_estimator_)

    for i in range(m):
        # create new datapoints for each iteratio
```

```
        datamat = np.random.multivariate_normal(mean, matrix, n)
        y = datamat[:,0]
        X = pd.DataFrame(datamat[:,(1,2)])
        # split the datapoints into a train and a test set
        X_train, X_test, y_train, y_test = train_test_split(X, y)
        # fit a ridge regression model
        rf_regr = CV.best_estimator_
        rf_regr.fit(X_train, y_train.values.ravel())
        # calculate the R-squared and the PFI scores
        result = permutation_importance(rf_regr, X_test,
        y_test, n_repeats=50)
        r2 = round(rf_regr.score(X_test, y_test), 2)
        #insert them into the above created list
        r2_list_RF[q][i] = r2
        imp1_list_RF[q][i] = result.importances_mean[0]
        imp2_list_RF[q][i] = result.importances_mean[1]
    # If desired, the mean PFI score and R-squared
    # the standard deviation
    # plus a diagram can be displayed
    # after each simulation
    mean_imp1 = round(np.mean(imp1_list_RF[q]),2)
    std_imp1 = round(np.std(imp1_list_RF[q]),2)
    mean_imp2 = round(np.mean(imp2_list_RF[q]),2)
    std_imp2 = round(np.std(imp2_list_RF[q]),2)
    mean_r2 = round(np.mean(r2_list_RF[q]),2)
    std_r2 = round(np.std(r2_list_RF[q]),2)
    print('mean_r2: {}, standard deviation: {}'
    .format(mean_r2, std_r2))
    print('feature1 importances: {}, standard deviation: {}'
    .format(mean_imp1, std_imp1))
    print('feature2 importances: {}, standard deviation: {}'
    .format(mean_imp2, std_imp2))
    a= plt.figure()
    axes= a.add_axes([0.1,0.1,0.8,0.8])
    maxylim = max(1, max(max(imp1_list_RF[q]),
    max(imp2_list_RF[q]))+ 0.2)
    minylim = min(1, min(min(imp1_list_RF[q]),
    min(imp2_list_RF[q])) - 0.2)
    axes.set_ylim([minylim,maxylim])
    plt.plot(imp1_list_RF[q], marker='o', linestyle='None')
    plt.plot(imp2_list_RF[q], marker='o', linestyle='None')
    plt.plot(r2_list_RF[q], linestyle = '-')
    plt.xlabel("Simulation iteration")
    plt.ylabel("R-squared/PFI score")
    plt.legend(labels = ["PFI score variable 1",
    "PFI score variable 2", "R-squared"])
    plt.grid()
    plt.show()

#For Ridge Regression
```

```
#create empty lists to store the best estimator
# the R-squared for each iteration and simulation,
# and the PFI scores for both variables
best_estimator_RR = []
r2_list_RR = [[0]*m for _ in range(len(mats))]
imp1_list_RR = [[0]*m for _ in range(len(mats))]
imp2_list_RR = [[0]*m for _ in range(len(mats))]
# loop through all the matrices
for q in range(len(mats)):
    # define the means and the matrix
    mean = [1,1,1]
    matrix = mats[q]
    # create the datapoints for the hyperparameter tuning
    # split them  into X and y
    datamat = np.random.multivariate_normal(mean, matrix, n)
    y = datamat[:,0]
    X = pd.DataFrame(datamat[:,(1,2)])
    # define the distribution for the hyperparameter
    parameters = {'alpha': loguniform(1e-10,1e10)}
    # tune the hyperparameter
    regr = Ridge()
    RcV = RandomizedSearchCV(estimator = regr,
    param_distributions = parameters, cv = 5, n_iter = 100)
    RcV.fit(X, y)
    best_estimator_RR.append(RcV.best_estimator_)
    # If desired print the VC matrix and the found best estimator
    print(matrix)
    print('best estimator:', RcV.best_estimator_)

    for i in range(m):
        # create new datapoints for each iteratio
        datamat = np.random.multivariate_normal(mean, matrix, n)
        y = datamat[:,0]
        X = pd.DataFrame(datamat[:,(1,2)])
        # split the datapoints into a train and a test set
        X_train, X_test, y_train, y_test = train_test_split(X, y)
        # fit a ridge regression model
        reg = RcV.best_estimator_
        reg.fit(X_train, y_train.ravel())
        # calculate the R-squared and the PFI scores
        result = permutation_importance(reg, X_test,
        y_test, n_repeats=50)
        r2 = round(reg.score(X_test, y_test), 2)
        # insert them into the above created list
        r2_list_RR[q][i] = r2
        imp1_list_RR[q][i] = result.importances_mean[0]
        imp2_list_RR[q][i] = result.importances_mean[1]
    # If desired, the mean PFI score and R-squared
    # the standard deviation
    # plus a diagram can be displayed
```

```
    # after each simulation
    mean_imp1 = round(np.mean(imp1_list_RR[q]),2)
    std_imp1 = round(np.std(imp1_list_RR[q]),2)
    mean_imp2 = round(np.mean(imp2_list_RR[q]),2)
    std_imp2 = round(np.std(imp2_list_RR[q]),2)
    mean_r2 = round(np.mean(r2_list_RR[q]),2)
    std_r2 = round(np.std(r2_list_RR[q]),2)
    print('mean_r2: {}, standard deviation: {}'
    .format(mean_r2, std_r2))
    print('feature1 importances: {}, standard deviation: {}'
    .format(mean_imp1, std_imp1))
    print('feature2 importances: {}, standard deviation: {}'
    .format(mean_imp2, std_imp2))
    a= plt.figure()
    axes= a.add_axes([0.1,0.1,0.8,0.8])
    maxylim = max(1, max(max(imp1_list_RR[q]),
    max(imp2_list_RR[q]))+0.2)
    minylim = min(1, min(min(imp1_list_RR[q]),
    min(imp2_list_RR[q])-0.2)
    axes.set_ylim([minylim,maxylim])
    plt.plot(imp1_list_RR[q], marker='o', linestyle='None')
    plt.plot(imp2_list_RR[q], marker='o', linestyle='None')
    plt.plot(r2_list_RR[q], linestyle = '-')
    plt.xlabel("Simulation iteration")
    plt.ylabel("R-squared/PFI score")
    plt.legend(labels = ["PFI score variable 1",
    "PFI score variable 2", "R-squared"])
    plt.grid()
    plt.show()

#The plot shows the R-square for RF model, when COVj = 0
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.08, 0.24, 0.48, 0.8],
     [0.08, 0.17, 0.32, 0.56, 0.89],
     [0.24, 0.32, 0.48, 0.72, 0.97],
     [0.48, 0.56, 0.72, 0.97, 0.94],
     [0.8, 0.89, 0.97, 0.94, 0.95]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
plt.rcParams.update({'font.size': 15})
fig = plt.figure()
ax = fig.add_subplot(111)
cpf = ax.contourf(X,Y,z,  cmap=cm.Blues)
cp = ax.contour(X, Y, z,  colors = 'blue')
ax.clabel(cp, fontsize=15, colors = 'k')
ax.set_xlabel(r'$COV_{j}$', fontsize = 15)
_ = ax.set_ylabel(r'$COV_{k}$', fontsize = 15)
CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)
```

```
#The plot shows the R-square for RR model, when COVj = 0
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.09, 0.24, 0.49, 0.82],
     [0.09, 0.17, 0.34, 0.58, 0.9],
     [0.24, 0.34, 0.5, 0.74, 0.98],
     [0.49, 0.58, 0.74, 0.98, 0.95],
     [0.82, 0.9, 0.98, 0.95, 0.95]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)
cpf = ax.contourf(X,Y,z,  cmap=cm.Blues)
cp = ax.contour(X, Y, z,  colors = 'blue')
ax.clabel(cp, colors = 'k')
ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')
CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the R-square for RF model, when COVj = 0.3
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.08, 0.26, 0.52, 0.88],
     [0.08, 0.13, 0.26, 0.49, 0.8],
     [0.26, 0.26, 0.36, 0.57, 0.86],
     [0.52, 0.49, 0.57, 0.74, 0.94],
     [0.88, 0.8, 0.86, 0.94, 0.95]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)


cpf = ax.contourf(X,Y,z, cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,  colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the R-square for RR model, when COVj = 0.3
x = [0,0.3,0.5,0.7,0.9]
```

```
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.1, 0.28, 0.54, 0.89],
     [0.1, 0.14, 0.28, 0.5, 0.81],
     [0.28, 0.28, 0.38, 0.58, 0.87],
     [0.54, 0.5, 0.58, 0.75, 0.95],
     [0.89, 0.81, 0.87, 0.95, 0.95]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)


cpf = ax.contourf(X,Y,z, cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,  colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the R-square for RF model, when COVj = 0.5
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.1, 0.31, 0.63, 0.97],
     [0.1, 0.11, 0.24, 0.48, 0.83],
     [0.31, 0.24, 0.32, 0.51, 0.81],
     [0.63, 0.48, 0.51, 0.64, 0.88],
     [0.97, 0.83, 0.81, 0.88, 0.95]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)


cpf = ax.contourf(X,Y,z,  cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,  colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')
```

```
CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the R-square for RR model, when COVj = 0.5
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.12, 0.33, 0.65, 0.97],
     [0.12, 0.12, 0.26, 0.49, 0.84],
     [0.33, 0.26, 0.33, 0.52, 0.82],
     [0.65, 0.49, 0.52, 0.65, 0.89],
     [0.97, 0.84, 0.82, 0.89, 0.95]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)


cpf = ax.contourf(X,Y,z,  cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,  colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the R-square for RF model, when COVj = 0.7
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.16, 0.46, 0.95, 0.9],
     [0.16, 0.09, 0.24, 0.55, 0.9],
     [0.46, 0.24, 0.28, 0.48, 0.83],
     [0.95, 0.55, 0.48, 0.57, 0.81],
     [0.9, 0.9, 0.83, 0.81, 0.95]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)


cpf = ax.contourf(X,Y,z,
levels = [0,0.15,0.25,0.45,0.55, 0.85,1],
cmap=cm.Blues)


# Make plot and customize axes
```

```
cp = ax.contour(X, Y, z,
levels = [0,0.15,0.25,0.45,0.55, 0.85,1],
colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the R-square for RR model, when COVj = 0.7
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.17, 0.49, 0.96, 0.91],
     [0.17, 0.11, 0.25, 0.57, 0.91],
     [0.49, 0.25, 0.3, 0.49, 0.84],
     [0.96, 0.57, 0.49, 0.58, 0.82],
     [0.91, 0.91, 0.84, 0.82, 0.95]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)


cpf = ax.contourf(X,Y,z,
levels = [0,0.15,0.25,0.45,0.55, 0.85,1],
cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,
levels = [0,0.15,0.25,0.45,0.55, 0.85,1],
colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the R-square for RF model, when COVj = 0.9
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.44, 0.87, 0.75, 0.86],
     [0.44, 0.08, 0.34, 0.75, 0.86],
     [0.87, 0.34, 0.25, 0.56, 0.86],
     [0.75, 0.75, 0.56, 0.51, 0.86],
     [0.86, 0.86, 0.86, 0.86, 0.85]]
```

```
rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)


cpf = ax.contourf(X,Y,z,  cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,  colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the R-square for RR model, when COVj = 0.9
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.47, 0.89, 0.77, 0.87],
     [0.47, 0.09, 0.37, 0.77, 0.87],
     [0.89, 0.37, 0.27, 0.58, 0.87],
     [0.77, 0.77, 0.58, 0.51, 0.87],
     [0.87, 0.87, 0.87, 0.87, 0.85]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)


cpf = ax.contourf(X,Y,z,  cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,  colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)


#The plot shows the PFI for RF model, when COVj = 0
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
```

```
z = [[0, 0, 0, 0, 0],
     [0.17, 0.17, 0.17, 0.17, 0.17],
     [0.47, 0.47, 0.47, 0.48, 0.33],
     [0.98, 0.97, 0.97, 0.96, 0.31],
     [1.61, 1.61, 1.47, 1.11, 0.56]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)


cpf = ax.contourf(X,Y,z,
levels = [0,0.17,0.47,0.95,1.5, 2],  cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,
levels = [0,0.17,0.47,0.95,1.5, 2],colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the PFI for RR model, when COVj = 0
z = [[0, 0, 0, 0, 0],
     [0.17, 0.18, 0.18, 0.18, 0.18],
     [0.5, 0.5, 0.5, 0.45, 0.34],
     [0.97, 0.98, 0.97, 0.98, 0.31],
     [1.61, 1.62, 1.48, 1.09, 0.56]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)


cpf = ax.contourf(X,Y,z,
levels = [0,0.18,0.47,0.95,1.5, 2],
cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,
levels = [0,0.18,0.47,0.95,1.5, 2],
colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')
```

```
CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the PFI for RF model, when COVj = 0.3
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.02, 0.05, 0.09, 0.16],
     [0.2, 0.1, 0.06, 0.02, 0],
     [0.58, 0.36, 0.29, 0.21, 0.12],
     [1.13, 0.92, 0.68, 0.56, 0.13],
     [1.92, 1.59, 1.39, 1.11, 0.56]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)


cpf = ax.contourf(X,Y,z,
levels = [0,0.07,0.2, 0.5,  0.9, 1.1,1.3,2],
cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,
levels = [0,0.07,0.2,0.5,  0.9, 1.1,1.3,2],
colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the PFI for RR model, when COVj = 0.3
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.02, 0.05, 0.11, 0.18],
     [0.21, 0.11, 0.05, 0.02, 0],
     [0.61, 0.41, 0.29, 0.2, 0.13],
     [1.19, 0.9, 0.73, 0.57, 0.33],
     [1.96, 1.58, 1.36, 1.09, 0.56]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)


cpf = ax.contourf(X,Y,z,
```

```
levels = [0,0.07,0.2, 0.5,  0.9, 1.1,1.3,2],
cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,
levels = [0,0.07,0.2,0.5,  0.9, 1.1,1.3,2],
colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the PFI for RF model, when COVj = 0.5
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.05, 0.18, 0.34, 0.38],
     [0.25, 0.08, 0.02, 0.01, 0.06],
     [0.79, 0.36, 0.22, 0.06, 0.01],
     [1.55, 1.04, 0.79, 0.43, 0.19],
     [2.43, 1.89, 1.54, 1.17, 0.56]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)

#levels = [0.07,0.2, 0.5,  0.9, 1.1,1.3,2],

cpf = ax.contourf(X,Y,z,
    levels = [0,0.1,0.3,0.9,1.3,1.8,2,3], cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,
levels = [0,0.1,0.3,  0.9, 1.3,1.8,2,3],
        colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the PFI for RR model, when COVj = 0.5
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
```

```
z = [[0, 0.08, 0.22, 0.44, 0.44],
     [0.32, 0.08, 0.01, 0.01, 0.09],
     [0.9, 0.44, 0.22, 0.08, 0.01],
     [1.75, 1.07, 0.72, 0.44, 0.22],
     [2.58, 2, 1.5, 1.08, 0.56]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)

#levels = [0.07,0.2, 0.5,  0.9, 1.1,1.3,2],

cpf = ax.contourf(X,Y,z,
levels = [0,0.1,0.3,  0.9, 1.3,1.8,2,3],
cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,
levels = [0,0.1,0.3,  0.9, 1.3,1.8,2,3],
colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the PFI for RF model, when COVj = 0.7
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.18, 0.61, 1.42, 0.27],
     [0.42, 0.06, 0.01, 0.15, 0.27],
     [1.39, 0.44, 0.17, 0.03, 0.08],
     [3.12, 1.44, 0.74, 0.33, 0.02],
     [2.5, 2.5, 2.04, 1.43, 0.56]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)

cpf = ax.contourf(X,Y,z,
levels = [0,0.2, 0.5,  0.9, 1.3,2,2.4,3,4], cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,
levels = [0,0.2, 0.5,  0.9, 1.3,2,2.4,3,4],
colors = 'blue')
```

```
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the PFI for RR model, when COVj = 0.7
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 0.34, 0.94, 1.85, 0.41],
     [0.7, 0.06, 0.02, 0.28, 0.41],
     [1.93, 0.64, 0.17, 0, 0.13],
     [3.78, 1.86, 0.94, 0.34, 0.04],
     [2.95, 2.95, 2.32, 1.3, 0.56]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)

cpf = ax.contourf(X,Y,z,
levels = [0,0.2, 0.5,  0.9, 1.3,2,2.4,3,4],
cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,
levels = [0,0.2, 0.5,  0.9, 1.3,2,2.4,3,4],
colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the PFI for RF model, when COVj = 0.9
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 1.34, 2.96, 1, 0.21],
     [1.84, 0.05, 0.36, 1, 0.21],
     [4.56, 1.33, 0.14, 0.28, 0.21],
     [3.11, 3.11, 1.88, 0.27, 0.21],
     [2.56, 2.56, 2.56, 2.56, 0.45]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)
```

```
cpf = ax.contourf(X,Y,z,
    levels = [0,0.3,0.9,2,3,4,5], cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,
    levels = [0,0.3,0.9,2,3,4,5], colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#The plot shows the PFI for RF model, when COVj = 0.9
x = [0,0.3,0.5,0.7,0.9]
y = [0,0.3,0.5,0.7,0.9]
X, Y = np.meshgrid(x,y)
z = [[0, 4.08, 6.79, 2.94, 0.66],
     [5.02, 0.05, 1.24, 2.94, 0.66],
     [9.3, 2.92, 0.14, 0.95, 0.66],
     [6.41, 6.41, 3.49, 0.27, 0.66],
     [4.02, 4.02, 4.02, 4.02, 0.45]]

rcParams['figure.figsize'] = 10, 7 # sets plot size
fig = plt.figure()
ax = fig.add_subplot(111)

#levels = [0.07,0.2, 0.5,  0.9, 1.1,1.3,2],

cpf = ax.contourf(X,Y,z,
    levels = [0,0.5,2,4,6,8,10], cmap=cm.Blues)


# Make plot and customize axes
cp = ax.contour(X, Y, z,
    levels = [0,0.5,2,4,6,8,10], colors = 'blue')
ax.clabel(cp, colors = 'k')

ax.set_xlabel(r'$COV_{j}$')
_ = ax.set_ylabel(r'$COV_{k}$')

CBI = plt.colorbar(cpf, orientation='vertical', shrink=0.8)

#plots with errorbars
# COV_j = 0, COV_l = 0
# varying COV_k
# RF
mylist = [0, 0.3, 0.5, 0.7, 0.9]
```

```
r2 = [0, 0.08, 0.24, 0.48, 0.8]
r2err = [0, 0.01, 0.02, 0.02, 0.01]
featureimp1 = [0, 0.17, 0.47, 0.98, 1.61]
featureimp1err = [0,0.02,0.03,0.03,0.03]
featureimp2 = [0, 0, 0, 0, 0]
featureimp2err = [0,0,0,0,0]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and outcome")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

# COV_j = 0, COV_l = 0
# varying COV_k
# RR
mylist = [0, 0.3, 0.5, 0.7, 0.9]
r2 = [0, 0.09, 0.24, 0.49, 0.82]
r2err = [0, 0.01, 0.02, 0.02, 0.01]
featureimp1 = [0, 0.17, 0.5, 0.97, 1.61]
featureimp1err = [0,0.02,0.03,0.03,0.02]
featureimp2 = [0, 0, 0, 0, 0]
featureimp2err = [0,0,0,0,0]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and outcome")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()


# COV_j = 0, COV_k = 0.5
# varying COV_l
# RF
r2 = [0.25, 0.26, 0.31, 0.46, 0.87]
featureimp1 = [0.5, 0.58, 0.79, 1.39, 4.56]
featureimp2 = [0, 0.05, 0.18, 0.61, 2.96]
r2err = [0.02, 0.02, 0.02, 0.02, 0.01]
```

```
featureimp1err = [0.03,0.03,0.04,0.07,0.32]
featureimp2err = [0,0.01,0.02,0.04,0.18]


plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

# COV_j = 0, COV_k = 0.5
# varying COV_l
# RR
r2 = [0.24, 0.28, 0.33, 0.49, 0.89]
featureimp1 = [0.47, 0.61, 0.9, 1.93, 9.3]
featureimp2 = [0, 0.05, 0.22, 0.94, 6.79]
r2err = [0.02, 0.02, 0.02, 0.02, 0.01]
featureimp1err = [0.03,0.03,0.05,0.09,0.44]
featureimp2err = [0,0.01,0.02,0.07,0.39]


plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err,
    marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

#COV_j = COV_k = 0.5
# varying COV_l
#RF

r2 = [0.48, 0.36, 0.32, 0.28, 0.25]
featureimp1 = [0.47, 0.28, 0.22, 0.17, 0.14]
featureimp2 = [0.47, 0.28, 0.22, 0.17, 0.14]
r2err = [0.02, 0.02, 0.02, 0.02, 0.02]
featureimp1err = [0.03,0.02,0.02,0.02,0.02]
featureimp2err = [0.03,0.02,0.02,0.02,0.02]
```

```
plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1", "R-squared"])
plt.grid()


#COV_j = COV_k = 0.5
# varying COV_l
#RF

r2 = [0.5, 0.38, 0.33, 0.3, 0.27]
featureimp1 = [0.5, 0.29, 0.22, 0.17, 0.14]
featureimp1 = [0.5, 0.29, 0.22, 0.17, 0.14]
r2err = [0.02, 0.02, 0.02, 0.02, 0.02]
featureimp1err = [0.02,0.02,0.02,0.02,0.03]
featureimp2err = [0.02,0.02,0.02,0.02,0.03]


plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1", "R-squared"])
plt.grid()

#COV_j = 0.3, COV_k = 0.7
# varying COV_l
#RF

r2 = [0.56, 0.49, 0.48, 0.54, 0.75]
featureimp1 = [0.97, 0.92, 1.04, 1.44, 3.11]
featureimp2 = [0.17, 0.02, 0.01, 0.15, 1]
r2err = [0.02, 0.02, 0.02, 0.02, 0.01]
featureimp1err = [0.03,0.04,0.04,0.05,0.11]
featureimp2err = [0.02,0.01,0,0.02,0.06]


plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
```

```
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()


#COV_j = 0.3, COV_k = 0.7
# varying COV_l
#RR

r2 = [0.58, 0.5, 0.49, 0.57, 0.77]
featureimp1 = [0.98, 0.9, 1.07, 1.86, 6.41]
featureimp2 = [0.18, 0.02, 0.01, 0.28, 2.94]
r2err = [0.02, 0.02, 0.02, 0.02, 0.01]
featureimp1err = [0.04,0.03,0.04,0.07,0.27]
featureimp2err = [0.02,0,0,0.03,0.18]


plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

# COV_j = COV_k = 0.3
# varying COV_l
#for RF

r2 = [0.17, 0.13, 0.11, 0.09, 0.08]
featureimp1 = [0.17, 0.1, 0.08, 0.06, 0.05]
r2err = [0.02, 0.02, 0.02, 0.02, 0.02]
featureimp1err = [0.02,0.01,0.01,0.01,0.02]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1", "R-squared"])
plt.grid()
```

```
# COV_j = COV_k = 0.3
# varying COV_l
#for RR

r2 = [0.18, 0.14, 0.12, 0.11, 0.09]
featureimp1 = [0.18, 0.11, 0.08, 0.06, 0.05]
r2err = [0.02, 0.02, 0.02, 0.02, 0.02]
featureimp1err = [0.02,0.02,0.02,0.02,0.02]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1", "R-squared"])
plt.grid()

# COV_j = COV_k = 0.7
# varying COV_l
#for RF

r2 = [0.97, 0.74, 0.64, 0.57, 0.51]
featureimp1 = [0.96, 0.56, 0.43, 0.34, 0.27]
r2err = [0, 0.01, 0.01, 0.02, 0.02]
featureimp1err = [0.04,0.02,0.02,0.02,0.02]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1", "R-squared"])
plt.grid()

# COV_j = COV_k = 0.7
# varying COV_l
#for RR

r2 = [0.98, 0.75, 0.65, 0.58, 0.51]
featureimp1 = [0.98, 0.57, 0.44, 0.34, 0.27]
r2err = [0, 0.01, 0.02, 0.02, 0.02]
featureimp1err = [0.04,0.02,0.02,0.02,0.04]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
```

```
plt.ylabel("R-squared/PFI-score")
plt.legend(labels = ["PFI-score-variable-1", "R-squared"])
plt.grid()

#COV_j = 0.3, COV_k = 0.5
# varying COV_l
#RF
r2 = [0.32, 0.26, 0.24, 0.24, 0.34]
featureimp1 = [0.47, 0.36, 0.36, 0.44, 1.33]
featureimp2 = [0.17, 0.06, 0.02, 0.01, 0.36]
r2err = [0.02, 0.02, 0.02, 0.02, 0.02]
featureimp1err = [0.02,0.02,0.02,0.02,0.07]
featureimp2err = [0.02,0.01,0.01,0.01,0.04]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation-variable-1-and-variable-2")
plt.ylabel("R-squared/PFI-score")
plt.legend(labels = ["PFI-score-variable-1",
"PFI-score-variable-2", "R-squared"])
plt.grid()

#COV_j = 0.3, COV_k = 0.5
# varying COV_l
#RR
r2 = [0.34, 0.28, 0.26, 0.25, 0.37]
featureimp1 = [0.5, 0.41, 0.44, 0.64, 2.92]
featureimp2 = [0.18, 0.05, 0.01, 0.02, 1.24]
r2err = [0.02, 0.02, 0.02, 0.02, 0.02]
featureimp1err = [0.03,0.03,0.03,0.05,0.17]
featureimp2err = [0.02,0.01,0.00,0.01,0.11]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation-variable-1-and-variable-2")
plt.ylabel("R-squared/PFI-score")
plt.legend(labels = ["PFI-score-variable-1",
"PFI-score-variable-2", "R-squared"])
plt.grid()

# COV_j = 0.5, COV_k = 0.7
# varying COV_l
```

```
#RF
mylist = [0, 0.3, 0.5, 0.7, 0.9]

r2 = [0.72, 0.57, 0.51, 0.48, 0.56]
featureimp1 = [0.97, 0.68, 0.79, 0.95, 1.88]
featureimp2 = [0.48, 0.21, 0.06, 0, 0.28]
r2err = [0.01, 0.02, 0.02, 0.02, 0.02]
featureimp1err = [0.04,0.02,0.03,0.03,0.07]
featureimp2err = [0.03,0.01,0.01,0.01,0.03]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

# COV_j = 0.5, COV_k = 0.7
# varying COV_l
#RR
mylist = [0, 0.3, 0.5, 0.7, 0.9]

r2 = [0.74, 0.58, 0.52, 0.49, 0.58]
featureimp1 = [0.97, 0.73, 0.72, 0.94, 3.49]
featureimp2 = [0.5, 0.2, 0.08, 0, 0.95]
r2err = [0.01, 0.02, 0.02, 0.02, 0.02]
featureimp1err = [0.03,0.03,0.03,0.04,0.17]
featureimp2err = [0.03,0.02,0.01,0.00,0.09]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

#COV_j = 0.5, COV_k = 0.9
# varying COV_l
#RF
mylist = [0.1, 0.3, 0.5, 0.7, 0.9]
```

```
r2 = [0.97, 0.86, 0.81, 0.83, 0.86]
featureimp1 = [1.47, 1.39, 1.54, 2.04, 2.56]
featureimp2 = [0.33, 0.12, 0.01, 0.08, 0.21]
r2err = [0.00, 0.01, 0.01, 0.01, 0.01]
featureimp1err = [0.03,0.03,0.03,0.04,0.06]
featureimp2err = [0.01,0.01,0.01,0.01,0.02]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()


#COV_j = 0.5, COV_k = 0.9
# varying COV_l
#RR
mylist = [0.1, 0.3, 0.5, 0.7, 0.9]

r2 = [0.98, 0.87, 0.82, 0.84, 0.87]
featureimp1 = [1.48, 1.36, 1.5, 2.32, 4.02]
featureimp2 = [0.34, 0.13, 0.01, 0.13, 0.66]
r2err = [0.00, 0.01, 0.01, 0.01, 0.01]
featureimp1err = [0.03,0.03,0.03,0.05,0.1]
featureimp2err = [0.02,0.01,0.00,0.01,0.04]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

#COV_j = 0.7, COV_k = 0.9
# varying COV_l
#RF
mylist = [0.4, 0.5, 0.7, 0.9]
```

```
r2 = [0.94,0.88, 0.81, 0.86]
featureimp1 = [1.11, 1.17, 1.43, 2.56]
featureimp2 = [0.31, 0.19, 0.02, 0.21]
r2err = [0.00, 0.01, 0.01, 0.01]
featureimp1err = [0.02,0.03,0.03,0.06]
featureimp2err = [0.01,0.01,0.00,0.02]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

#COV_j = 0.7, COV_k = 0.9
# varying COV_l
#RR
mylist = [0.4, 0.5, 0.7, 0.9]

r2 = [0.95,0.89, 0.82, 0.87]
featureimp1 = [1.09, 1.08, 1.3, 4.02]
featureimp2 = [0.33, 0.22, 0.04, 0.66]
r2err = [0.00, 0.01, 0.01, 0.01]
featureimp1err = [0.03,0.02,0.03,0.1]
featureimp2err = [0.01,0.01,0.01,0.04]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

#COV_j = 0.3, COV_k = 0.9
# varying COV_l
#RF
mylist = [0, 0.3, 0.5, 0.7, 0.9]

r2 = [0.89, 0.8, 0.83, 0.9, 0.86]
featureimp1 = [1.61, 1.59, 1.89, 2.5, 2.56]
```

```
featureimp2 = [0.17, 0, 0.06, 0.27, 0.21]
r2err = [0.01, 0.01, 0.01, 0.01,0.01]
featureimp1err = [0.03,0.03,0.03,0.05,0.06]
featureimp2err = [0.01,0.00,0.01,0.02,0.02]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

#COV_j = 0.3, COV_k = 0.9
# varying COV_l
#RR
mylist = [0, 0.3, 0.5, 0.7, 0.9]

r2 = [0.9, 0.81, 0.84, 0.91, 0.87]
featureimp1 = [1.62, 1.58, 2, 2.95, 4.02]
featureimp2 = [0.18, 0, 0.08, 0.41, 0.66]
r2err = [0.01, 0.01, 0.01, 0.01,0.01]
featureimp1err = [0.03,0.03,0.04,0.06,0.1]
featureimp2err = [0.01,0.00,0.01,0.02,0.04]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

#COV_j = 0, COV_k = 0.3
# varying COV_l
#RF
mylist = [0, 0.3, 0.5, 0.7, 0.9]

r2 = [0.08, 0.08, 0.1, 0.16, 0.44]
featureimp1 = [0.17, 0.2, 0.25, 0.42, 1.84]
featureimp2 = [0, 0.02, 0.05, 0.18, 1.34]
r2err = [0.01, 0.02, 0.02, 0.02,0.02]
```

```
featureimp1err = [0.02,0.02,0.02,0.03,0.09]
featureimp2err = [0.00,0.01,0.01,0.02,0.08]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

#COV_j = 0, COV_k = 0.3
# varying COV_l
#RR
mylist = [0, 0.3, 0.5, 0.7, 0.9]

r2 = [0.09, 0.1, 0.12, 0.17, 0.47]
featureimp1 = [0.18, 0.21, 0.32, 0.7, 5.02]
featureimp2 = [0, 0.02, 0.08, 0.34, 4.08]
r2err = [0.01, 0.02, 0.02, 0.02,0.02]
featureimp1err = [0.02,0.02,0.03,0.06,0.32]
featureimp2err = [0.00,0.01,0.02,0.04,0.31]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

#COV_j = 0, COV_k = 0.7
# varying COV_l
#RF
mylist = [0, 0.3, 0.5, 0.7, 0.9]

r2 = [0.48, 0.52, 0.63, 0.95, 0.75]
featureimp1 = [0.98, 1.13, 1.55, 3.12, 3.11]
featureimp2 = [0, 0.09, 0.34, 1.42, 1]
r2err = [0.02, 0.02, 0.02, 0.00,0.01]
featureimp1err = [0.03,0.04,0.05,0.12,0.11]
featureimp2err = [0.00,0.01,0.02,0.08,0.06]
```

```
plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

#COV_j = 0, COV_k = 0.7
# varying COV_l
#RR
mylist = [0, 0.3, 0.5, 0.7, 0.9]

r2 = [0.49, 0.54, 0.65, 0.96, 0.77]
featureimp1 = [0.97, 1.19, 1.75, 3.78, 6.41]
featureimp2 = [0, 0.11, 0.44, 1.85, 2.94]
r2err = [0.02, 0.02, 0.02, 0.00,0.01]
featureimp1err = [0.03,0.04,0.06,0.14,0.27]
featureimp2err = [0.00,0.01,0.03,0.09,0.18]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

#COV_j = 0, COV_k = 0.9
# varying COV_l
#RF
mylist = [0, 0.3, 0.5, 0.7, 0.9]

r2 = [0.8, 0.88, 0.97, 0.9, 0.86]
featureimp1 = [1.61, 1.92, 2.43, 2.5, 2.56]
featureimp2 = [0, 0.16, 0.38, 0.27, 0.21]
r2err = [0.01, 0.01, 0.00, 0.01,0.01]
featureimp1err = [0.03,0.04,0.05,0.05,0.06]
featureimp2err = [0.00,0.01,0.02,0.02,0.02]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
```

```
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"], bbox_to_anchor=(0.4, 0.45))
plt.grid()

#COV_j = 0, COV_k = 0.9
# varying COV_l
#RR
mylist = [0, 0.3, 0.5, 0.7, 0.9]

r2 = [0.82, 0.89, 0.97, 0.91, 0.87]
featureimp1 = [1.61, 1.96, 2.58, 2.95, 4.02]
featureimp2 = [0, 0.18, 0.44, 0.41, 0.66]
r2err = [0.01, 0.01, 0.00, 0.01,0.01]
featureimp1err = [0.02,0.04,0.07,0.06,0.1]
featureimp2err = [0.00,0.01,0.03,0.02,0.04]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

# COV_j = COV_k = 0.9
# varying COV_l
#RF
mylist = [0.7, 0.9]
r2 = [0.95, 0.85]
featureimp1 = [0.56, 0.45]
r2err = [0.00, 0.01]
featureimp1err = [0.01,0.02]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
```

```
plt.legend(labels = ["PFI score variable 1", "R-squared"])
plt.grid()

# COV_j = COV_k = 0.9
# varying COV_l
#RR
mylist = [0.7, 0.9]
r2 = [0.95, 0.85]
featureimp1 = [0.56, 0.45]
r2err = [0.00, 0.01]
featureimp1err = [0.02,0.02]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1", "R-squared"])
plt.grid()

#COV_j = 0.7, COV_k = 0.9 # alternate plot
# varying COV_l
#RF
mylist = [0.4, 0.5, 0.7, 0.9]

r2 = [0.94,0.88, 0.81, 0.86]
featureimp1 = [1.11, 1.02, 1.43, 2.56]
featureimp2 = [0.31, 0.25, 0.02, 0.21]
r2err = [0.00, 0.01, 0.01, 0.01]
featureimp1err = [0.02,0.03,0.03,0.06]
featureimp2err = [0.01,0.01,0.00,0.02]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

#COV_j = 0.3, COV_k = 0.7 #alternate plot
# varying COV_l
#RF
mylist = [0, 0.3, 0.5, 0.7, 0.9]
```

```
r2 = [0.56, 0.49, 0.48, 0.54, 0.75]
featureimp1 = [0.97, 0.92, 1.04, 1.5, 3.11]
featureimp2 = [0.17, 0.02, 0.01, 0.17, 1]
r2err = [0.02, 0.02, 0.02, 0.02, 0.01]
featureimp1err = [0.03,0.04,0.04,0.05,0.11]
featureimp2err = [0.02,0.01,0,0.02,0.06]


plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()


# COV_j = 0.5, COV_k = 0.7 # alternate plot
# varying COV_l
#RF
mylist = [0, 0.3, 0.5, 0.7, 0.9]

r2 = [0.72, 0.57, 0.51, 0.48, 0.56]
featureimp1 = [0.97, 0.68, 0.79, 0.74, 1.88]
featureimp2 = [0.48, 0.21, 0.06, 0.04, 0.28]
r2err = [0.01, 0.02, 0.02, 0.02, 0.02]
featureimp1err = [0.04,0.02,0.03,0.03,0.07]
featureimp2err = [0.03,0.01,0.01,0.01,0.03]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

#COV_j = 0.3, COV_k = 0.5 # alternate plot
# varying COV_l
#RF
r2 = [0.32, 0.26, 0.24, 0.24, 0.34]
featureimp1 = [0.47, 0.36, 0.36, 0.59, 1.33]
```

```
featureimp2 = [0.17, 0.06, 0.02, 0.01, 0.36]
r2err = [0.02, 0.02, 0.02, 0.02, 0.02]
featureimp1err = [0.02,0.02,0.02,0.03,0.07]
featureimp2err = [0.02,0.01,0.01,0.01,0.04]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

#COV_j = 0, COV_k = 0.7 #alternate plot
# varying COV_l
#RF
mylist = [0, 0.3, 0.5, 0.7, 0.9]

r2 = [0.48, 0.52, 0.63, 0.93, 0.75]
featureimp1 = [0.98, 1.13, 1.55, 2.81, 3.11]
featureimp2 = [0, 0.09, 0.34, 1.24, 1]
r2err = [0.02, 0.02, 0.02, 0.01,0.01]
featureimp1err = [0.03,0.04,0.05,0.1,0.11]
featureimp2err = [0.00,0.01,0.02,0.07,0.06]

plt.errorbar(mylist, featureimp1, yerr = featureimp1err,
    marker='o', capsize=4)
plt.errorbar(mylist, featureimp2, yerr = featureimp2err,
    marker='o', capsize=4)
plt.errorbar(mylist, r2, yerr = r2err, marker='o', capsize=4)
plt.xticks(mylist,mylist)
plt.xlabel("correlation variable 1 and variable 2")
plt.ylabel("R-squared/PFI score")
plt.legend(labels = ["PFI score variable 1",
"PFI score variable 2", "R-squared"])
plt.grid()

# additional simulations for COV_k = 0.7, COV_j = 0,
# and COV_l = 0.7 in RF
m = 100
n = 5000
k = 7
r2_list_add = [[None]*100 for _ in range(k)]
imp1_list_add = [[None]*100 for _ in range(k)]
imp2_list_add = [[None]*100 for _ in range(k)]
```

```
for q in range(k):
    mean = [1,1,1]
    matrix = [[1, 0.7, 0], [0.7, 1, 0.7],[0, 0.7, 1]]
    datamat = np.random.multivariate_normal(mean, matrix, n)
    y = datamat[:,0]
    X = pd.DataFrame(datamat[:,(1,2)])

    parameters = {'max_features': [1,2],
                  'min_samples_leaf': range(1, 256),
                  'min_samples_split': range(2, 256)}
    rf_regr = RandomForestRegressor(n_estimators = 500)
    CV = RandomizedSearchCV(estimator = rf_regr,
    param_distributions = parameters, n_iter = 100, cv = 5)
    CV.fit(X, y)
    print(CV.best_estimator_)
    print(CV.best_score_)

    for i in range(m) :
        datamat = np.random.multivariate_normal(mean, matrix, n)
        y = datamat[:,0]
        X = pd.DataFrame(datamat[:,(1,2)])

        X_train, X_test, y_train, y_test = train_test_split(X, y)
        rf_regr = CV.best_estimator_
        rf_regr.fit(X_train, y_train)
        result = permutation_importance(rf_regr,
            X_test, y_test, n_repeats=50)
        r2 = round(rf_regr.score(X_test, y_test), 2)
        r2_list_add[q][i] = r2
        imp1_list_add[q][i] = result.importances_mean[0]
        imp2_list_add[q][i] = result.importances_mean[1]
    mean_imp1 = round(np.mean(imp1_list_add[q]),2)
    std_imp1 = round(np.std(imp1_list_add[q]),2)
    mean_imp2 = round(np.mean(imp2_list_add[q]),2)
    std_imp2 = round(np.std(imp2_list_add[q]),2)
    mean_r2 = round(np.mean(r2_list_add[q]),2)
    std_r2 = round(np.std(r2_list_add[q]),2)
    print(CV.best_estimator_)
    print('mean_r2:-{},-standard-deviation:-{}'
    .format(mean_r2, std_r2))
    print('feature1-importances:-{},-standard-deviation:-{}'
    .format(mean_imp1, std_imp1))
    print('feature2-importances:-{},-standard-deviation:-{}'
    .format(mean_imp2, std_imp2))
```