Feature Selection via Mutual Information based on Boruta

Why feature selection?

Feature selection is the process of identifying and selecting a subset of features that are most relevant to the target variable. In terms of Information Theory, we can disregard features which don't decrease model uncertainty.

Feature selection, In Machine learning works on a simple rule — If there's noise in data their will be noise in output. The problem is, in real life, training data could have hundreds of thousands of features and many not fit for any models at all. For example, you want to train a model to predict customer churn based on current 100+ customer properties, like age, location, country, tech stack used, last visit and so on. Now, the question is will all those feature provide relevant information? Does dropping some features have any impact on model performance? If the answer is no, then there is a need to drop these irrelevant features.[7]

Some Advantages of Feature selections are:

1. It enables the machine learning algorithm to reduce training time.
2. It reduces the complexity of a model and makes it easier to interpret.
3. It improves the accuracy of a model if the right subset is chosen.
4. It reduces overfitting by generalizing the model on optimal features.

Large no. of Features → Feature selection (any logic) → Fitting model with selected features → output

Types of feature selection:

We have basically 3 types of feature selection strategy [7][9][10]

1. Filter Methods
2. Wrapper Method (Forward, Backward Elimination)
3. Embedded Methods (Lasso-L1, Ridge-L2 Regression)

In subspace of Filter methods we have:

1. No/less Quadi-constant variables
2. No Duplicate Rows
3. High correlation with the target variable
4. Low correlation with another independent variable
5. Higher information gain or mutual information of the independent variable

So I will select Mutual Information as metric to calculate and apply feature selection

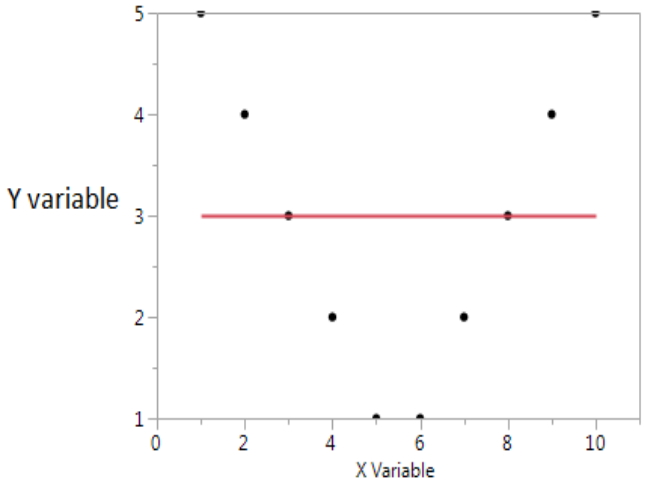
Why mutual info? why not correlation & KL divergence?

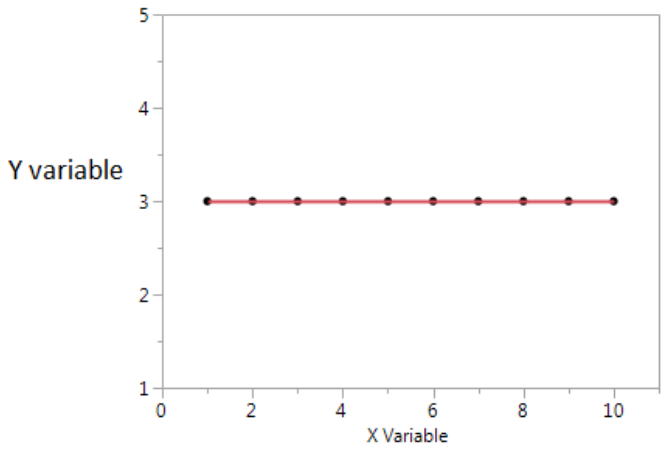
Mutual information is a distance between two probability distributions. Correlation is a ***linear*** distance between two random variables while mutual information tend to discover non linearity of data. Mutual Information measures the entropy drops under the condition of the target value You can have a mutual information between any two probabilities defined for a set of symbols, while you cannot have a correlation between symbols that cannot naturally be mapped into an N space.

Mutual information is a special case of KL divergence, where KL divergence is applied to measure the difference between the actual joint distribution of two variables (which captures their dependence) and the hypothetical joint distribution of the same variables, were they to be independent. KL_divergence is good for 2 distributions but what if we have multiple dimensions this is where MI come. [11]

Here's an example:

In these two plots the correlation coefficient is zero. But we can get high shared mutual information even when the correlation is zero.
In the first, I see that with change of X, y changes. For extreme value of X we have extreme y value but if the value of X is moderate then I have a low value of Y. The first plot holds information about the mutual information shared by X and Y. In the second plot, X tells me nothing about Y. [1][6]





Why Boruta:

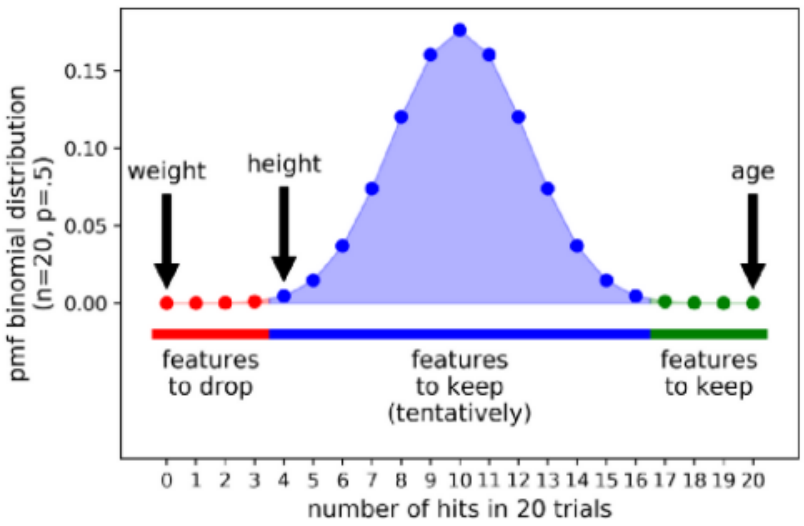
Boruta is a random forest-based method, so it works for tree models like Random Forest or XGBoost but is also valid with other classification models like Logistic Regression or SVM.

Boruta iteratively removes features that are statistically less relevant than a random probe (artificial noise variables introduced by the Boruta algorithm). In each iteration, rejected variables are removed from consideration in the next iteration. It generally ends up with a good global optimization for feature selection [2]

Reason for selecting Boruta:

Basically, boruta is an algorithm designed to solve what the paper calls the "all-relevant problem": finding a subset of features from the dataset which are relevant to a given task. This is different from the "minimal-optimal problem", which is the problem of finding the minimal subset of features which are performant in a model. While machine learning models in production should ultimately target building on minimal-optimal selections of the data, the unique point of Boruta is that for the purposes of exploration minimum-optimality goes too far.

Boruta attempts to curate features down to the "all-relevant" stopping point, not the "minimal-optimal" stopping point. What does "all-relevant" mean? The paper defines a variable as being relevant if there is a subset of attributes in the dataset among which the variable is not redundant when used for prediction. To illustrate this, suppose that we have two variables, A and B. These variables are correlated with one another and correlated with the objective variable. However, A is a stronger signal than B is. In a regular decision tree the A feature will be given high importance while the B feature will be mostly ignored in A's favor. If we were minimizing our variable count, then we would remove B. [3]



Binomial distribution and positioning of the features

Implementation of Boruta:

1. shadow features (creating features based on original features)
2. concatenate all the features (original & imputed)
3. apply tree based model on all features (Boruta)
4. repeat process for certain no. of iterations
5. score all the features based on feature importance score
6. create binomial distribution to select/reject features

What basically happens is that randomly shuffled shadow attributes are defined to establish a baseline performance for prediction against the target variable. Then a hypothesis test is used to determine, with a certain level of risk, if each variable is correlated only randomly. Variables that fail to be rejected are removed from consideration.

Recall that we had a feature B that was redundant to feature A, but still related to the objective variable. By training a random forest Boruta will, by sheer chance, generate decision trees having the B feature but lacking the A feature. In these cases, in the absence of feature A, feature B will be given high importance instead. This importance will be the average importance assigned to that variable and prevent it from failing the test and being removed.

Simultaneously, as more and more uninformative variables are removed, the feature importance of the remaining informative variables will improve. Noisier (more random) related variables will see larger gains, as they will be decorrelated from the noise being pruned from the dataset [3][4]

Apart from Boruta I also implemented:

1. *Feature selection based on mean decrease in impurity*

   Not a good fit for high cardinality features. Score features based on tree based mean scores.

2. *Feature importance based on feature permutation*

   Permutation feature importance overcomes limitations of the impurity-based feature selection: they do not have a bias toward high-cardinality feature. Here I have selected feature based on max score of feature importance of shadow features.

Implementation of Boruta:

```
def MI_via_boruta(X, y, iter, model, seed=7, estimator=10, depth=None, boruta=True, mean_dec=False, feat_perm=False, learning_rate=0.01):
  """
  implementation of generalized Boruta idea for feature selection based on MI & Entropy via ensemble tree based models
  """

  """
  This function supports following models with: [following parameters that can be modified]:
  RandomForestClassifier          : [estimator, depth]
  RandomForestRegressor           : [estimator, depth]
  ExtraTreesClassifier            : [estimator, depth]
  AdaBoostClassifier              : [estimator]
  AdaBoostRegressor               : [estimator]
  GradientBoostingClassifier      : [estimator, depth, learning_rate]
  GradientBoostingRegressor       : [estimator, depth, learning_rate]
  HistGradientBoostingClassifier  : [estimator/max_iter, depth, learning_rate]
  HistGradientBoostingRegressor   : [estimator/max_iter, depth, learning_rate]

  #TODO: add configration for more parameters and more tree based models
  #TODO Later: Further optimization of tree based models via GridSearchCV for optimal parameters and model selection
  """

  """
  3 optionals are given here for selection of method for feature selection
  boruta : implemented below, Set to True by default
  mean_dec : based on mean decrease in impurity : via Scikit-Learn , False by default set to True for using this and set [boruta and feat_perm] = False
  feat_perm : based on feature permutation : via Scikit-Learn , False by default set to True for using this and set [boruta and mean_dec] = False
  """

  """
  X    : will take dataframe of features
  y    : will take label (both classification and regression "select model accordingly")
  iter : how many times for loop run for selected method of feature selection
  seed : random seed to start feature permutation
  ** estimator, depth, learning_rate can be used as default or selected as needed
  """

  """
  TODO: add checking of input data errors(type, size, shape)
  TODO Later: improve boruta feature selection strategy
  """
```

Output of classification task:

```
clf_bruta = MI_via_boruta(X_clf, y_clf, iter=20, model='RandomForestClassifier', estimator=10)
```

```
clf_bruta
```

	Feautre	score
0	age	20.0
1	menopause	20.0
2	tumor-size	20.0
3	inv-nodes	20.0
4	node-caps	20.0
5	deg-malig	20.0
6	breast	20.0
7	breast-quad	20.0
8	irradiat	16.0

Comparison with BorutaPy:

I have had 3 distinct implementations: Boruta, feature impurity & feature permutations and compare the test dataset's output to the state-of-the-art implementation of Pybruta. [12]

Score comparison table (based on 2 regression and 1 classification dataset)

dataset	Boruta (implemented)	feature impurity (implemented)	Feature permutation (implemented)	PyBoruta
Cancer (classification)	All 9 selected	All 9 selected	All 9 selected	All 9 selected
Insurance (regression)	6 selected out of 6	6 selected out of 6	6 selected out of 6	4 out of 6 selected
Forest (regression)	11 out of 12 selected	12 out of 12 selected	11 out of 12 selected	0 out of 12 selected

Based on the findings, our implementation has matching score for classification task but for regression they are not coinciding well.

Classification score of BorutaPy:

```
BorutaPy finished running.

Iteration:      13 / 100
Confirmed:      9
Tentative:      0
Rejected:       0
BorutaPy(estimator=RandomForestClassifier(max_depth=5, n_estimators=84,
                                          n_jobs=-1,
                                          random_state=RandomState(MT19937) at 0x7EFDBE4B8270),
         n_estimators='auto',
         random_state=RandomState(MT19937) at 0x7EFDBE4B8270, verbose=2)
```

```
feat_selector.ranking_
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1])
```

regression score of Boruta:

```
reg_bruta = MI_via_boruta(X_reg, y_reg, iter=20, model='RandomForestRegressor', estimator=10)
#reg_meandec = MI_via_boruta(xt, yt, iter=20, model='RandomForestRegressor', estimator=10, boruta=False, mean_dec=True)
```

```
reg_bruta
```

	Feautre	score
0	age	20.0
1	sex	20.0
2	bmi	20.0
3	children	20.0
4	smoker	20.0
5	region	20.0

Future work:

We can further improve our score by implementing Feature selection via kernel density estimation of Gaussian mixture models [16] [18]

Another way could be Feature selection via kernel canonical correlation analysis [17]

Code implementation is available at: https://github.com/SaadMuhammad/Feature_Selection_via_Boruta

References:

[1] https://stats.stackexchange.com/questions/81659/mutual-information-versus-correlation

[2] https://towardsdatascience.com/simple-example-using-boruta-feature-selection-in-python-8b96925d5d7a

[3] https://www.kaggle.com/residentmario/automated-feature-selection-with-boruta

[4] https://towardsdatascience.com/boruta-explained-the-way-i-wish-someone-explained-it-to-me-4489d70e154a

[5] https://github.com/scikit-learn-contrib/boruta_py

[6] https://machinelearningmastery.com/information-gain-and-mutual-information/

[7] https://medium.com/mlearning-ai/feature-selection-using-filter-method-python-implementation-from-scratch-375d86389003

[8] https://towardsdatascience.com/select-features-for-machine-learning-model-with-mutual-information-534fe387d5c8

[9] https://medium.datadriveninvestor.com/feature-selection-techniques-1a99e61da222

[10] https://www.kaggle.com/prashant111/comprehensive-guide-on-feature-selection

[11] https://stats.stackexchange.com/questions/13389/information-gain-mutual-information-and-related-measures

[12] https://github.com/scikit-learn-contrib/boruta_py

[13] https://github.com/danielhomola/mifs

[14] https://github.com/jupiters1117/mico

[15] https://scikitlearn.org/stable/modules/generated/sklearn.feature_selection.SelectPercentile.html#sklearn.feature_selection.SelectPercentile

[16] https://pypi.org/project/infoselect/ (inprogress for implementation)

[17] https://www.sciencedirect.com/science/article/pii/S2590188519300149

[18] Eirola, E., Lendasse, A., & Karhunen, J. (2014, July). Variable selection for regression problems using Gaussian mixture models to estimate mutual information. In 2014 International Joint Conference on Neural Networks (IJCNN) (pp. 1606-1613). IEEE.