Feature Selection

Automatic Feature Selection

- It can be a good idea to reduce the number of features to only the most useful ones

- Simpler models that generalize better (less overfitting)
  - Curse of dimensionality (e.g. kNN)
  - Even models such as RandomForest can benefit from this
  - Sometimes it is one of the main methods to improve models

- Faster prediction and training
  - Training time can be quadratic (or cubic) in number of features

- Easier data collection, smaller models (less storage)

- More interpretable models: fewer features to look at

Automatic Feature Selection

- Feature selection methods mainly consists of the following:
- **Unsupervised learning** methods: don't use the target variable
  - Variance threshold

- **Supervised learning** methods: use the target variable
  - **Filter methods**: select subsets of features based on their relationship with the target.
    - Correlation coefficient, chi-square, F-test, mutual information

  - **Wrapper methods**: measure the usefulness of features based on the classifier performance
    - RFE
    - Forward/backward selection

  - **Embedded methods**: algorithms that perform automatic feature selection during training.
  - L1 regression
  - Tree-based mtehods: Random forest

- **PCA**: Project input data into a lower-dimensional feature space

Variance Threshold

- **Unsupervised learning methods**
- If the variance of a features is low or close to zero, then a feature is approximately constant and will not improve the performance of the model. In that case, it should be removed.
- Or if only a few observations differ from a constant value, the variance will also be very low
- This method removes all features whose variance doesn't meet some threshold.

- As an example, suppose that we have a dataset with boolean features, and we want to remove all features that are either one or zero (on or off) in more than 80% of the samples.
- Boolean features are Bernoulli random variables, and the variance of such variables is given by

$$Var(x)=p(1-p)$$

so we can select using the threshold=0.8*(1-0.8)

Variance Threshold in sklearn

```
from sklearn.feature_selection import VarianceThreshold

# Load Iris dataset
iris_data = load_iris()
X = iris_data.data

# Apply Variance Thresholding
selector = VarianceThreshold(threshold=0.2)
X_filtered = selector.fit_transform(X)


X = [[0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 1, 1], [0, 1, 0], [0, 1, 1]]
sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
sel.fit_transform(X)
```

Filter Methods

- The filter method is to evaluate each individual feature's relationship with the target variable and select the ones that have the strongest correlation.
- This process is repeated for each feature and the best ones are selected based on defined criteria, such as the correlation or statistical significance.
- As such, they are sometimes referred to as univariate feature selection.

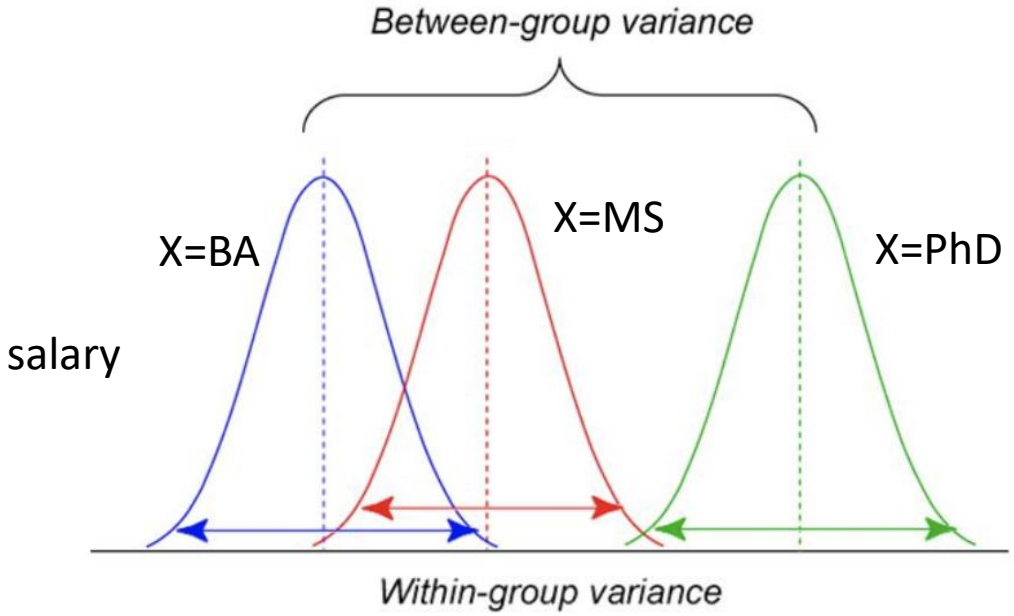
- Simple and straightforward, but it does not take into account any interactions between features.
- Useful when working with a large number of features
- Also useful in cases where the relationship between the target variable and individual features is not complex
- Filter methods:
  - Anova F-test
  - Pearson correlation coefficient
  - Chi-square
  - Mutual information

ANOVA F-test

- The main purpose of ANOVA (Analysis of Variance) is to test if two or more groups differ from each other significantly in one or more characteristics.
- F-test is another name for ANOVA that only compares the statistical means in groups.

- For feature selection, we specifically use one Way ANOVA test, and normally the test is applied on a categorical feature and numeric target

- For example, given a categorical feature “degree” which has three categories BA, MS, PhD, and a numeric target “Salary”
- We want to know does feature “degree” has any predictive power for target “Salary”?
- ANOVA solution is to compare the mean salary values in each feature category (BA,MS,PhD).



ANOVA F-test

- Procedure of ANOVA F-test:

1) Set up the following hypnosis assumption: Begins with a null hypothesis that the means of the groups are equal.

- Null Hypothesis H_0 : All categories mean values are same (i.e. $\mu_A = \mu_B = \mu_C$)
- Alternative Hypothesis H_a : At least one of the categories mean values differ

2) Calculate F-score: The F-score/statistic is calculated as the ratio of the variance between the groups to the variance within the groups.

3) If F-score > critical value of F-distribution, reject H_0 and accept the feature X
(it concludes there is a significant difference between the means of the groups)
Otherwise, drop the feature X

Test Statistic for a One-Way ANOVA

$$F = \frac{variance\ betwn\ groups}{variance\ within\ groups} = \frac{var(\overline{x_i})}{var(x_i)} = \frac{\sum_i n_i(\overline{x_i} - \bar{\bar{x}})^2/(k-1)}{\sum_i (n_i - 1)s_i^2/(N-k)}$$

N : the overall sample size

k : the number of groups

n_i : the number of observations in the i-th group

$\overline{x_i}$: the sample mean in the i-th group

$\bar{\bar{x}}$: the overall mean of the data

s_i^2 : the variance of the i-th group

Example of F-test

Feature X	x1	x2	x3
target	12 15 17 12	16 14 21 15 19	14 17 20 15
	$\overline{x_1} = \frac{56}{4} = 14$	$\overline{x_2} = \frac{85}{5} = 17$	$\overline{x_3} = \frac{66}{4} = 16.5$
	$s_1^2 = 6$	$s_2^2 = 8.5$	$s_3^2 = 7$

k = 3 (3 feature values; 3 groups)

$N = n_1 + n_2 + n_3 = 4 + 5 + 4 = 13$ (total data size)

Solution: Performing a One-Way ANOVA

$$F = \frac{variance\ betwn\ groups}{variance\ within\ groups} = \frac{var(\bar{x}_i)}{var(x_i)} = \frac{\sum_i n_i(\bar{x}_i - \bar{\bar{x}})^2/(k-1)}{\sum_i (n_i - 1)s_i^2/(N-k)}$$

$$N = 13, k = 3$$

$$n_1 = 4, n_2 = 5, n_3 = 4$$

$$\bar{\bar{x}} = \frac{\sum x}{N} = \frac{56 + 85 + 66}{13} \approx 15.9$$

$$var(\bar{x}_i) = \frac{\sum n_i(\bar{x}_i - \bar{x})^2}{k-1} = \frac{4(14 - 15.9)^2 + 5(17 - 15.9)^2 + 4(16.5 - 15.9)^2}{3-1} \approx 10.9$$

$$var(x_i) = \frac{\sum (n_i - 1)s_i^2}{N-k} = \frac{(4-1)6 + (5-1)8.5 + (4-1)7}{13-3} \approx 7.3$$

$$F = \frac{var(\bar{x}_i)}{var(x_i)} = \frac{10.9}{7.3} \approx 1.5$$

Solution: Performing a One-Way ANOVA

- H_0: $\mu_1 = \mu_2 = \mu_3$ (μ_i *:* true mean of xi)
- H_a: at least one mean is different
- F statistic: $F = \frac{MS_B}{MS_W} \approx 1.5$
- $\alpha = 0.01$
- (degree of freedom) $df_N = k - 1 = 2$, $df_D = N - k = 10$
- Critical F value with df_N=2 & df_D =10 is 7.56
- Decision: F statistic is less than critical F value
  - Therefore, fail to reject H_0
  - There is not enough evidence at the 1% level of significance to conclude that there is a difference in the means

Correlation Analysis (Numeric Data)

- Correlation is a statistical term which refers to how close two variables are to having a linear relationship with each other.
- For example, two variables which are linearly dependent (say, x and y which depend on each other as x = 2y) will have a higher correlation
- Correlation coefficient (also called Pearson's coefficient) is computed as follows

$$r_{A,B} = \frac{cov(A,B)}{\sigma(A)\sigma(B)} = \frac{\frac{1}{N-1}\sum_i (a_i - \bar{A})(b_i - \bar{B})}{\sigma(A)\sigma(B)}$$

where N is the number of tuples, $\bar{A}$ and $\bar{B}$ are the respective means of A and B, $\sigma(A)$ and $\sigma(B)$ are the respective standard deviation of A and B

Correlation Analysis (Numeric Data)

- Correlation coefficient $r_{A,B}$ ranges between -1 and 1
- If $r_{A,B} > 0$, A and B are positively correlated (A's values increase as B's). The higher, the stronger correlation.
- $r_{A,B} = 0$: independent; $r_{A,B} < 0$: negatively correlated
- When two features have high correlation, we can drop one of the two features.

Chi-Square Method

- Alternatively, we can use the χ^2 test (only for categorical features)

	Play chess	Not play chess	Sum (row)
Like science fiction	250(90)	200(360)	450
Not like science fiction	50(210)	1000(840)	1050
Sum(col.)	300	1200	1500

- X^2 (chi-square) calculation (numbers in parenthesis are expected counts calculated based on the data distribution in the two categories)

$$\chi^2 = \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{(1000-840)^2}{840} = 507.93$$

- If X^2 value is greater than the critical value in X^2 distribution, we can say that two features are strongly related to each other.
- It shows that ‘like_science_fiction’ and ‘play_chess’ are correlated in the group
- (Refer to discretization method in preprocessing chapter)

Mutual information

- Mutual information method is the application of information gain to feature selection.
- Mutual Information (MI) of two random variables is a measure of the mutual dependence between the two variables (aka Information Gain).
  - Calculated between two variables and measures the reduction in uncertainty for one variable given a known value of the other variable.
  - Measures how much information X gives about the target Y. In terms of entropy H:

$$MI(X,Y) = H(Y) - H(Y|X)$$

  - $H(Y)$ is entropy of Y and measures the degree of uncertainty in variable Y
  - $MI(X,Y)$ represents how much target uncertainty(entropy) reduced if we know feature X
- *MI* is equal to zero if two random variables are independent, and higher values mean higher dependency.
- For feature selection, we can use *MI* to measure the dependency of a feature variable and target variable
- (For more information, refer to decision tree algorithm)
- The scikit-learn machine learning library provides an implementation of mutual information for feature selection via the mutual_info_classif() or mutual_info_regression() function.
- (Refer to discretization method in preprocessing chapter)

Feature Selection in sklearn

- SelectKBest removes all but the k highest scoring features
- SelectPercentile removes all but a user-specified highest scoring percentage of features

```
from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif

X, y = load_iris(return_X_y=True)    # If True, returns (data, target)
X.shape
X_new = SelectKBest( f_classif, k=2).fit_transform(X, y)
X_new.shape
```

(150, 4)

(150,2)

- For regression: f_regression, r_regression, mutual_info_regression
- For classification: chi2, f_classif, mutual_info_classif

Feature Selection in sklearn

- f_classif : f_classif, which is used for classification problems.
- The f_classif scoring function calculates the F-value between each feature and the target variable, and the features with the highest F-values are selected as the top k features.

- f_regression : f_regression, which is used for regression problems.
- The f_regression scoring function calculates the F-value between each feature and the target variable, and the features with the highest F-values are selected as the top k features.
- f_regression is used when the target variable is numerical, while f_classif is used for the categorical target variable

- r_regression: compute Pearson's r for each features and the target. Pearson's r is also known as the Pearson correlation coefficient.

- chi2: This test is used to determine whether there is a significant association between two categorical variables. The test calculates the difference between the expected frequency of occurrences and the observed frequency of occurrences.

Categorical Input Data and a Categorical Target Variable

- Two popular feature selection techniques for categorical input data and a categorical target variable.
  Chi-Squared Statistic.
  Mutual Information Statistic.

```
from sklearn.feature_selection import SelectKBest, chi2, mutual_info_classif

# use chi-square method
fs = SelectKBest(score_func=chi2, k='all')
fs.fit(X_train, y_train)
X_train_fs = fs.transform(X_train)
X_test_fs = fs.transform(X_test)

# what are scores for the features
for i in range(len(fs.scores_)):
    print('Feature %d: %f' % (i, fs.scores_[i]))
```

Chi2	Mutual Information
Feature 0: 0.4725	Feature 0: 0.0035
Feature 1: 0.02913	Feature 1: 0.0000
Feature 2: 2.1376	Feature 2: 0.0259
Feature 3: 29.3810	Feature 3: 0.0714
Feature 4: 8.2226	Feature 4: 0.0000
Feature 5: 8.1001	Feature 5: 0.0389
Feature 6: 1.2738	Feature 6: 0.0647
Feature 7: 0.9506	Feature 7: 0.0030
Feature 8: 3.6999	Feature 8: 0.0000

- This clearly shows that feature 3 might be the most relevant (according to chi-squared) and that perhaps four of the nine input features are the most relevant.
- We could set k=4 when configuring the SelectKBest to select these top four features.
- “score_func=mutual_info_classif” uses Mutual Information method

Wrapper Method

- Measure the usefulness of features based on the performance of a classifier
- Add best new feature or remove worst one, repeat (forward or backward)

Forward Selection:

- Forward selection starts training a model with no feature.
- Then, it goes over all the features to find the one best feature to add.
- It repeats this until cross-validation score improvement by adding a feature does not exceed a tolerance level or a desired number of features are selected.

Backward Selection:

- Backward selection is a reverse of forward selection.
- It starts from all features and iteratively removes 1 feature at a time that is the least significant
- And just like forward selection, it repeats until the cv score improvement by removing a feature does not exceed a tolerance level or it's left with a desired number of features

RFE

- similar in spirit to backward selection, but Instead of relying on a model performance, RFE makes its decision based on feature importance extracted from the model

Forward/Backward Selection Method

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn_evaluation import plot

rfc = RandomForestClassifier()

# forward selection
select_features = SequentialFeatureSelector( rfc, direction='forward', n_features_to_select="auto“ )
select_features.fit(X_clf_train, y_clf_train)

features = select_features.get_feature_names_out()
rfc.fit(X_clf_train[features], y_clf_train)
plot.feature_importances(rfc)

# backward selection
# instead of ‘forward’, use ‘backward’
select_features = SequentialFeatureSelector( rfc, direction=‘backward', n_features_to_select="auto“ )
```

Recursive Feature Elimination(RFE)

- The Recursive Feature Elimination (RFE) method is a feature selection approach. It works by recursively removing attributes and building a model on those attributes that remain.
- RFE is similar in spirit to backward selection. It also starts with a full model and iteratively eliminates the features one by one.
- The difference is, instead of relying on a model performance metric, RFE makes its decision based on feature importance extracted from the model.
- This could be feature weights in linear models, impurity decrease in tree-based models, or permutation importance (which is applicable to any model type)

- First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through any specific attribute (such as coef_, feature_importances_)
- Then, the least important features are pruned from current set of features.
- That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

RFE in sklearn

```
# Recursive Feature Elimination
from sklearn import datasets
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

# load the iris datasets
dataset = datasets.load_iris()

# create a base classifier used to evaluate a subset of attributes
model = LogisticRegression()

# create the RFE model and select 3 attributes
rfe = RFE(model, n_features_to_select=3)
rfe = rfe.fit(dataset.data, dataset.target)

# summarize the selection of the attributes
print(rfe.support_)
print(rfe.ranking_)
```

Model-based Feature Selection

- Feature selection using SelectFromModel
- SelectFromModel is a meta-transformer that can be used alongside any estimator that assigns importance to each feature through a specific attribute (such as coef_, feature_importances_)
- The features are considered unimportant and removed if the corresponding importance of the feature values are below the provided threshold parameter.

- Use a tuned supervised model to compute the importance of each feature
- Linear models (Ridge, Lasso, LinearSVM,...): features with highest weights (coefficients)
- Tree–based models: features used in first nodes (high information gain)
- Selection model can be different from the one you use for final modelling

Model-based Feature Selection

- Like the RFE, SelectFromModel from Scikit-Learn is based on a Machine Learning Model estimation for selecting the features.
- The differences are that SelectFromModel feature selection is based on the importance of attribute (often is coef_ or feature_importances_ but it could be any callable) threshold, no iteration involved.
- By default, the threshold is the mean.
- Like RFE, you could use any Machine Learning model to select the feature, as long as it was callable to estimate the attribute importance. You could try it out with the Random Forest model or XGBoost

SelectFromModel from Scikit-Learn

```
from sklearn.feature_selection import SelectFromModel
from sklearn.svm import LinearSVC
from sklearn.datasets import load_iris

X, y = load_iris(return_X_y=True)
X.shape

# Selecting the Best important features according to Logistic Regression using SelectFromModel
sfm = SelectFromModel(estimator=LogisticRegression())
sfm.fit(X, y)
sfm.estimator_.coef_
X.columns[sfm.get_support()]  # Get a mask, or integer index, of the features selected.
sfm.transform(X)
```

SelectFromModel from Scikit-Learn

```
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier

# Selecting the Best important features according to Random Forest
select = SelectFromModel(RandomForestClassifier(n_estimators=100, random_state=7),
                                    threshold='median')
select.fit(X_train, y_train)
X_train_selected = select.transform(X_train)
print(X_train.shape)
print(X_train_selected.shape)
```