Feature Selection For Machine Learning in Python

by **Jason Brownlee** on May 20, 2016 in **Python Machine Learning**

Last Updated on December 18, 2019

The data features that you use to train your machine learning models have a huge influence on the performance you can achieve.

Irrelevant or partially relevant features can negatively impact model performance.

In this post you will discover automatic feature selection techniques that you can use to prepare your machine learning data in python with scikit-learn.

Discover how to prepare data with pandas, fit and evaluate models with scikit-learn, and more in my new book, with 16 step-by-step tutorials, 3 projects, and full python code.

Let's get started.

- **Update Dec/2016**: Fixed a typo in the RFE section regarding the chosen variables.
- **Update Mar/2018**: Added alternate link to download the dataset.
- **Update Sep/2019**: Fixed code to be compatible with Python 3.
- **Update Dec/2019**: Updated univariate selection to use ANOVA.

Feature Selection For Machine Learning in Python
Photo by Baptiste Lafontaine, some rights reserved.

Feature Selection

Feature selection is a process where you automatically select those features in your data that contribute most to the prediction variable or output in which you are interested.

Having irrelevant features in your data can decrease the accuracy of many models, especially linear algorithms like linear and logistic regression.

Three benefits of performing feature selection before modeling your data are:

- **Reduces Overfitting**: Less redundant data means less opportunity to make decisions based on noise.
- **Improves Accuracy**: Less misleading data means modeling accuracy improves.
- **Reduces Training Time**: Less data means that algorithms train faster.

You can learn more about feature selection with scikit-learn in the article Feature selection.

Feature Selection for Machine Learning

This section lists 4 feature selection recipes for machine learning in Python

This post contains recipes for feature selection methods.

Each recipe was designed to be complete and standalone so that you can copy-and-paste it directly into you project and use it immediately.

Recipes uses the Pima Indians onset of diabetes dataset to demonstrate the feature selection method . This is a binary classification problem where all of the attributes are numeric.

- Dataset File.
- Dataset Details.

1. Univariate Selection

Statistical tests can be used to select those features that have the strongest relationship with the output variable.

The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features.

Many different statistical test scan be used with this selection method. For example the ANOVA F-value method is appropriate for numerical inputs and categorical data, as we see in the Pima dataset. This can be used via the f_classif() function. We will select the 4 best features using this method in the example below.

```
# Feature Selection with Univariate Statistical Tests
from pandas import read_csv
from numpy import set_printoptions
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
# load data
filename = 'pima-indians-diabetes.data.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
# feature extraction
test = SelectKBest(score_func=f_classif, k=4)
fit = test.fit(X, Y)
# summarize scores
set_printoptions(precision=3)
print(fit.scores_)
features = fit.transform(X)
# summarize selected features
print(features[0:5,:])
```

For help on which statistical measure to use for your data, see the tutorial:

- How to Choose a Feature Selection Method For Machine Learning

You can see the scores for each attribute and the 4 attributes chosen (those with the highest scores). Specifically features with indexes 0 (*preq*), 1 (*plas*), 5 (*mass*), and 7 (*age*).

```
[ 39.67  213.162   3.257   4.304  13.281  71.772  23.871  46.141]

[[  6.  148.   33.6  50. ]
 [  1.   85.   26.6  31. ]
 [  8.  183.   23.3  32. ]
 [  1.   89.   28.1  21. ]
 [  0.  137.   43.1  33. ]]
```

2. Recursive Feature Elimination

The Recursive Feature Elimination (or RFE) works by recursively removing attributes and building a model on those attributes that remain.

It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute.

You can learn more about the RFE class in the scikit-learn documentation.

The example below uses RFE with the logistic regression algorithm to select the top 3 features. The choice of algorithm does not matter too much as long as it is skillful and consistent.

```
# Feature Extraction with RFE
from pandas import read_csv
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
# load data
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
# feature extraction
model = LogisticRegression(solver='lbfgs')
rfe = RFE(model, 3)
fit = rfe.fit(X, Y)
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

You can see that RFE chose the the top 3 features as *preg*, *mass* and *pedi*.

These are marked True in the *support_* array and marked with a choice “1” in the *ranking_* array.

```
Num Features: 3
Selected Features: [ True False False False False  True  True False]
Feature Ranking: [1 2 3 5 6 1 1 4]
```

3. Principal Component Analysis

Principal Component Analysis (or PCA) uses linear algebra to transform the dataset into a compressed form.

Generally this is called a data reduction technique. A property of PCA is that you can choose the number of dimensions or principal component in the transformed result.

In the example below, we use PCA and select 3 principal components.

Learn more about the PCA class in scikit-learn by reviewing the PCA API. Dive deeper into the math behind PCA on the Principal Component Analysis Wikipedia article.

```
# Feature Extraction with PCA
import numpy
from pandas import read_csv
from sklearn.decomposition import PCA
# load data
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(url, names=names)
array = dataframe.values
```

```
X = array[:,0:8]
Y = array[:,8]
# feature extraction
pca = PCA(n_components=3)
fit = pca.fit(X)
# summarize components
print("Explained Variance: %s" % fit.explained_variance_ratio_)
print(fit.components_)
```

You can see that the transformed dataset (3 principal components) bare little resemblance to the source data.

```
Explained Variance: [ 0.88854663  0.06159078  0.02579012]
[[ -2.02176587e-03   9.78115765e-02   1.60930503e-02   6.07566861e-02
    9.93110844e-01   1.40108085e-02   5.37167919e-04  -3.56474430e-03]
 [  2.26488861e-02   9.72210040e-01   1.41909330e-01  -5.78614699e-02
   -9.46266913e-02   4.69729766e-02   8.16804621e-04   1.40168181e-01]
 [ -2.24649003e-02   1.43428710e-01  -9.22467192e-01  -3.07013055e-01
    2.09773019e-02  -1.32444542e-01  -6.39983017e-04  -1.25454310e-01]]
```

4. Feature Importance

Bagged decision trees like Random Forest and Extra Trees can be used to estimate the importance of features.

In the example below we construct a ExtraTreesClassifier classifier for the Pima Indians onset of diabetes dataset. You can learn more about the ExtraTreesClassifier class in the scikit-learn API.

```
# Feature Importance with Extra Trees Classifier
from pandas import read_csv
from sklearn.ensemble import ExtraTreesClassifier
# load data
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
# feature extraction
model = ExtraTreesClassifier(n_estimators=10)
model.fit(X, Y)
print(model.feature_importances_)
```

You can see that we are given an importance score for each attribute where the larger score the more important the attribute. The scores suggest at the importance of *plas*, *age* and *mass*.

```
[ 0.11070069  0.2213717   0.08824115  0.08068703  0.07281761  0.14548537 0.12654214  0.15415431]
```

Summary

In this post you discovered feature selection for preparing machine learning data in Python with scikit-learn.

You learned about 4 different automatic feature selection techniques:

- Univariate Selection.
- Recursive Feature Elimination.

- Principle Component Analysis.
- Feature Importance.

If you are looking for more information on feature selection, see these related posts:

- Feature Selection with the Caret R Package
- Feature Selection to Improve Accuracy and Decrease Training Time
- An Introduction to Feature Selection
- Feature Selection in Python with Scikit-Learn

Do you have any questions about feature selection or this post? Ask your questions in the comment and I will do my best to answer them.