Coding Exercise, Week 3, Dec 8

Feature Selection

CSML1010 Fall 2019

Pete Gray

ptgray@my.yorku.ca (mailto:ptgray@my.yorku.ca) , YorkU # 217653247

Using code and educational resources at https://machinelearningmastery.com/feature-selection-machine-learning-python/ (https://machinelearningmastery.com/feature-selection-machine-learning-python/)

Feature Extraction with Univariate Statistical Tests (Chi-squared for classification)

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.

The example below uses the chi squared (chi^2) statistical test for non-negative features to select 4 of the best features from the Pima Indians onset of diabetes dataset.

In [1]:
```
import pandas
import numpy
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

In [2]:
```
# load data
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-d
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
```

In [3]:
```
# feature extraction
test = SelectKBest(score_func=chi2, k=4)
fit = test.fit(X, Y)
```

In [4]:
```
# summarize scores
numpy.set_printoptions(precision=3)
print(fit.scores_)
features = fit.transform(X)
# summarize selected features
print(features[0:5,:])
```

```
[ 111.52  1411.887   17.605   53.108 2175.565  127.669    5.393  181.304]
[[148.    0.   33.6  50. ]
 [ 85.    0.   26.6  31. ]
 [183.    0.   23.3  32. ]
 [ 89.   94.   28.1  21. ]
 [137.  168.   43.1  33. ]]
```

You can see the scores for each attribute and the 4 attributes chosen (those with the highest scores): plas, test, mass and age.

(Note: I briefly had trouble interpreting this interpretation. I noted that the last, third-last and fourth-last columns got the high scores - but this doesn't line up with the last, third-last and fourth-last values for names. Ah, because the last value in names is the class variable, and it isn't included in array for evaluation. Got it. It's 5, 6, and 8 (and 2, of course))

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.

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.

In [5]:
```
from pandas import read_csv
from sklearn.feature_selection import RFE
```

In [6]:
```
from sklearn.linear_model import LogisticRegression
```

In [7]:
```
# load data
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-d
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]
```

In [8]:
```
# feature extraction

# Note that we increase max_iter from the default value up to 200
# in order to avoid the warning:
#
# ConvergenceWarning: lbfgs failed to converge.
# Increase the number of iterations.
#  "of iterations.", ConvergenceWarning)

model = LogisticRegression(solver='lbfgs', max_iter=200)
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_)
```

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

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.

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.

In [9]:
```
from sklearn.decomposition import PCA
```

In [10]:
```
# load data
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-d
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]
```

In [11]:
```
# feature extraction
pca = PCA(n_components=3)
fit = pca.fit(X)
# summarize components
print("Explained Variance: %s" % fit.explained_variance_ratio_)
print(fit.components_)
```

```
Explained Variance: [0.889 0.062 0.026]
[[-2.022e-03  9.781e-02  1.609e-02  6.076e-02  9.931e-01  1.401e-02
   5.372e-04 -3.565e-03]
 [-2.265e-02 -9.722e-01 -1.419e-01  5.786e-02  9.463e-02 -4.697e-02
  -8.168e-04 -1.402e-01]
 [-2.246e-02  1.434e-01 -9.225e-01 -3.070e-01  2.098e-02 -1.324e-01
  -6.400e-04 -1.255e-01]]
```

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

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.

In [12]:
```
from sklearn.ensemble import ExtraTreesClassifier
```

In [13]:
```
# load data
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-d
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]
```

In [14]:
```
# feature extraction
model = ExtraTreesClassifier(n_estimators=10)
model.fit(X, Y)
print(model.feature_importances_)
```

```
[0.118 0.236 0.091 0.086 0.076 0.142 0.116 0.134]
```

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.

This concludes the weekly programming exercise for Dec 8 2019.