COMP809 – Data Mining and Machine Learning

Lab 5 – Feature Selection

This lab will cover some of the feature selection methods supported by Python. All of the methods discussed in the lectures will form part of the lab. Use the sklearn documentation online to configure the methods. Above all, make sure you understand what you are doing; simply configuring the methods is not enough without an understanding of how they work. The basic code appears below.

1) Feature Extraction with Univariate Statistical Tests – importing libraraies

```
import pandas
import numpy as np
from sklearn.model_selection import train_test_split,
cross_val_score
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.decomposition import PCA
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.feature_selection import SelectFromModel
```

2) Load data

```
url =
"https://raw.githubusercontent.com/jbrownlee/Datasets/master
/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass',
'pedi', 'age', 'class']
rawdata = pandas.read_csv(url, names=names)
array = rawdata.values
nrow, ncol = rawdata.shape
X = array[:, 0:8]
Y = array[:, 8]
```

3) **Generate the model and get accuracy**

```
def get_accuracy(target_train, target_test,
predicted_test,predicted_train):
    clf = MLPClassifier(activation='logistic', solver='sgd',
learning_rate_init=0.1, alpha=1e-5,hidden_layer_sizes=(5,
2), random_state=1)
    clf.fit(predicted_train, np.ravel(target_train,
order='C'))
    predictions = clf.predict(predicted_test)
    return accuracy_score(target_test, predictions)

pred_train, pred_test, tar_train, tar_test =
train_test_split(X, Y, test_size=.3, random_state=4) #
shuffle the dataset using a random seed of 4 and split into
train/test segments
print("Accuracy score of our model without feature selection
: %.2f" % get_accuracy(tar_train, tar_test, pred_test,
pred_train))
```

4) Feature extraction – Chi Square

By referring the documentation of SelectKBest at https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html complete the following line of code and extract the K(K=3) top ranked features according to the chi squared test.

```
test = SelectKBest()
fit = test.fit(X, Y)
```

Summarize scores

```
np.set_printoptions(precision=3)
print(fit.scores_)
features = fit.transform(X)
```

Summarize selected features

```
print(features[0:3, :])
print()
```

Now apply only the K most significant features according to the chi square method.

```
pred_features = features[ : , … ]     #complete on your own
pred_train, pred_test, tar_train, tar_test =
train_test_split(pred_features, Y, test_size=.3,
random_state=2)
print("Accuracy score of our model with chi square feature
selection : %.2f" % get_accuracy(tar_train, tar_test,
pred_test,pred_train))
print()
```

5) Feature Extraction with - Recursive Feature Extraction (RFE)

```
model = LogisticRegression()  # Logistic regression is
the Wrapper classifier here
```

By referring the documentation of RFE at https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html complete the following line of code and extract the K,(K=3) top ranked features according to the RFE wrapper.

```
rfe = RFE( )          #complete on your own
fit = rfe.fit(X, Y)
print("Num Features: %d" % (fit.n_features_))
print("Selected Features: %s" % (fit.support_))
print("Feature Ranking: %s" % (fit.ranking_))
```

Now apply only the K most significant features according to the RFE feature selection method.

```
features = fit.transform(X)
pred_features = features[:, …. ]    #complete on your own
pred_train, pred_test, tar_train, tar_test =
train_test_split(pred_features, Y, test_size=.3,
random_state=2)
print("Accuracy score of our model with RFE selection :
%.2f" % get_accuracy(tar_train, tar_test,
pred_test,pred_train))
print()
```

6) **Feature Extraction –Principal Components Analysis (PCA)**

PCA computes a new set of features as a linear combination of the original set of features. The new features created can be ordered by the amount of variance they explain in the data - using features that capture a high degree of variance can improve classification accuracy.

By referring the documentation of PCA at https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
complete the following line of code and extract the K, (K=3) top ranked features according to PCA.

```
pca = PCA( )                    #complete on your own
fit = pca.fit(X)
```

Summarize components

```
print("Explained Variance: %s" %
(fit.explained_variance_ratio_))
print(fit.components_)
```

Now apply only the K most significant features (components) according to the PCA feature selection method.

```
features = fit.transform(X)
pred_features = features[:, ….]     #complete on your own
pred_train, pred_test, tar_train, tar_test =
train_test_split(pred_features, Y, test_size=.3,
random_state=2)
print("Accuracy score of our model with PCA selection :
%.2f" % get_accuracy(tar_train, tar_test,
pred_test,pred_train))
print()
```

7) **Feature Selection/Importance - Extra Trees Classifier**

The Extra trees classifier is a variation on Random Forest and differs from it in two ways: Firstly it drops the idea of bootstrap sampling. Secondly, it uses a full randomised choice for splitting nodes - unlike RF that uses a hybrid of Information Gain and random choice.

By referring the documentation of ET at https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html
Configure ET in the following line of code to set max_depth to 3, min_samples_leaf to 2 .

```
model = ExtraTreesClassifier( ) #complete
fit = model.fit(X, Y)
print(model.feature_importances_)
print()
t = SelectFromModel(fit, prefit=True)    #extra step required
as we are using an ensemble classifier here
features = t.transform(X)
pred_features = features[:, ….] #complete on your own

pred_train, pred_test, tar_train, tar_test =
train_test_split(pred_features, Y, test_size=.3,
random_state=2)
print("Accuracy score of our model with Extra Trees
selection : %.2f" % get_accuracy(tar_train, tar_test,
pred_test, pred_train))
```