Week11 Tutorial Feature Selection

September 27, 2022

1 Introduction

Feature selection is also known as Variable selection or Attribute selection. Essentially, it is the process of selecting the most important/relevant features of a dataset.

The importance of feature selection can best be recognized when you are dealing with a dataset that contains a vast number of features. This type of dataset is often referred to as a high dimensional dataset. Now, with this high dimensionality, comes a lot of problems such as - this high dimensionality will significantly increase the training time of your machine learning model, it can make your model very complicated which in turn may lead to Overfitting.

Often in a high dimensional feature set, there remain several features which are redundant meaning these features are nothing but extensions of the other essential features. These redundant features do not effectively contribute to the model training as well. So, clearly, there is a need to extract the most important and the most relevant features for a dataset in order to get the most effective predictive modeling performance.

Let me summarize the importance of feature selection:

- It enables the machine learning algorithm to train faster.
- It reduces the complexity of a model and makes it easier to interpret.
- It improves the accuracy of a model if the right subset is chosen.
- It reduces Overfitting.

2 Feature Selection Methods

In this section, you will study the different types of general feature selection methods

- Filter methods
- Wrapper methods

2.1 Filter methods

The filter method uses the principal criteria of ranking technique and uses the rank ordering method for feature selection. The reason for using the ranking method is simplicity, produce excellent and relevant features. The ranking method will filter out irrelevant features before classification process starts.

Filter methods are generally used as a data preprocessing step. The selection of features is independent of any machine learning algorithm. Features give rank on the basis of statistical scores

which tend to determine the features' correlation with the outcome variable. Some examples of some filter methods include the Chi-squared test and correlation coefficient scores.

2.2 An example on Python code

For this example, we will use the Pima Indians Diabetes dataset. The description of the dataset can be found here. https://www.kaggle.com/uciml/pima-indians-diabetes-database

The dataset corresponds to classification tasks on which you need to predict if a person has diabetes based on 8 features.

2.2.1 Import and Loading dataset

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
# load dataset
diabetes_dataset = pd.read_csv("diabetes.csv", sep = ",")
```

[3]:
```
diabetes_dataset.head()
```

[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

[4]:
```
diabetes_dataset.shape
```

[4]:
```
(768, 9)
```

[5]:
```
# Import the necessary libraries first
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

2.2.2 Feature extraction

[6]:
```
feature_columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',␣
↪'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']

X = diabetes_dataset[feature_columns] # Features
y = diabetes_dataset.Outcome # Target
```

[7]:
```
X.head()
```

[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

[8]:
```
y.unique()
```

[8]:
```
array([1, 0], dtype=int64)
```

2.2.3 Chi-Squared statistical test

Chi-squared stats of non-negative features for classification tasks. See more details on https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html#sklearn.feature_selection.chi2

We now implement a Chi-Squared statistical test for non-negative features to select 4 of the best features from the dataset. You have already seen Chi-Squared test belongs the class of filter methods. If anyone's curious about knowing the internals of Chi-Squared, see the video here https://www.youtube.com/watch?v=VskmMgXmkMQ.

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, in this case, it is Chi-Squared.

[9]:
```
best_features  = SelectKBest(score_func=chi2, k=3)
best_features.fit(X, y)
```

[9]:
```
SelectKBest(k=3, score_func=<function chi2 at 0x000001CDC9EACE50>)
```

2.2.4 Summarize scores

[10]:
```
print(best_features.scores_)
```

```
[ 111.51969064 1411.88704064   17.60537322   53.10803984 2175.56527292
  127.66934333    5.39268155  181.30368904]
```

2.2.5 Obtaining features

[11]:
```
features = best_features.transform(X)
# Summarize selected features
print(features[0:5,:])
```

```
[[148.   0.  50.]
 [ 85.   0.  31.]
 [183.   0.  32.]
 [ 89.  94.  21.]
 [137. 168.  33.]]
```

2.2.6 Print features and their scores

[12]:
```
df_scores = pd.DataFrame(best_features.scores_)
df_columns = pd.DataFrame(X.columns)

# concatenate dataframes
feature_scores = pd.concat([df_columns, df_scores],axis=1)
feature_scores.columns = ['Features','Schi-square Score']
print(feature_scores.nlargest(8,'Schi-square Score'))
```

```
                   Features  Schi-square Score
4                   Insulin        2175.565273
1                   Glucose        1411.887041
7                       Age         181.303689
5                       BMI         127.669343
0               Pregnancies         111.519691
3             SkinThickness          53.108040
2             BloodPressure          17.605373
6  DiabetesPedigreeFunction           5.392682
```

2.2.7 Interpretation

You can see the scores for each attribute and the 3 attributes chosen (those with the highest scores):

- Insulin
- Glucose
- Age

This scores will help you further in determining the best features for training your model.

2.3 Wrapper methods

A wrapper method needs one machine learning algorithm and uses its performance as evaluation criteria. This method searches for a feature which is best-suited for the machine learning algorithm and aims to improve the performance. To evaluate the features, the predictive accuracy used for classification tasks.

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 here https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html.

2.4 An example on Python code

In this example, we will continue to work with the diabetes dataset and a classification model to select features based on the RFE.

2.4.1 Import libraries

[13]:
```
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE

from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
```

[14]:
```
classifiers = {
#reason: these models has feature importances
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Logistic Regression": LogisticRegression(max_iter = 1000)
}
```

2.4.2 Comparison between various classifiers

[15]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    train_size = 0.8)
```

[16]:
```
best_model_score = 0
```

[17]:
```
for name, classifier in classifiers.items():
    print("classifier is ", name)
    model = classifier.fit(X_train, y_train)
```

```
    y_pred = model.predict(X_test)

    if (best_model_score < metrics.accuracy_score(y_test, y_pred)):
        best_model = classifier
        best_model_score = metrics.accuracy_score(y_test, y_pred)

    print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
#    print("Precision:",metrics.precision_score(y_test, y_pred, average =␣
 ↪'weighted'))
#    print("Recall:",metrics.recall_score(y_test, y_pred, average = 'weighted'))
    print("F1-score:",metrics.f1_score(y_test, y_pred, average = 'weighted'))
    print('---------------------------------')
```

```
classifier is  Decision Tree
Accuracy: 0.7207792207792207
F1-score: 0.7193073202985346
---------------------------------
classifier is  Random Forest
Accuracy: 0.7402597402597403
F1-score: 0.7347282347282348
---------------------------------
classifier is  Logistic Regression
Accuracy: 0.7077922077922078
F1-score: 0.7007679166976141
---------------------------------
```

[18]:
```
best_model
```

[18]:
```
RandomForestClassifier()
```

2.4.3 Adding the RFE method

You will use RFE with the various classifiers to select the top 3 features. The choice of algorithm does not matter too much as long as it is skillful and consistent.

[19]:
```
best_model_score = 0
```

[20]:
```
for n_features in range(1, 9, 1):
    print("Number of features is ", n_features)

    rfe = RFE(best_model, n_features_to_select=n_features)
    model = rfe.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    if (best_model_score < metrics.f1_score(y_test, y_pred, average =␣
 ↪'weighted')):
        n_best_features = n_features
```

```
        rfe_best = rfe
        best_model_score = metrics.f1_score(y_test, y_pred, average =␣
 ↪'weighted')

    print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
#    print("Precision:",metrics.precision_score(y_test, y_pred, average =␣
 ↪'weighted'))
#    print("Recall:",metrics.recall_score(y_test, y_pred, average = 'weighted'))
    print("F1-score:",metrics.f1_score(y_test, y_pred, average = 'weighted'))
    print('---------------------------------')
```

```
Number of features is  1
Accuracy: 0.6493506493506493
F1-score: 0.6273661480029404
---------------------------------
Number of features is  2
Accuracy: 0.6558441558441559
F1-score: 0.641012548459815
---------------------------------
Number of features is  3
Accuracy: 0.7012987012987013
F1-score: 0.6894711988943857
---------------------------------
Number of features is  4
Accuracy: 0.7012987012987013
F1-score: 0.6949374699374701
---------------------------------
Number of features is  5
Accuracy: 0.7207792207792207
F1-score: 0.7124346917450366
---------------------------------
Number of features is  6
Accuracy: 0.7402597402597403
F1-score: 0.7332725186265256
---------------------------------
Number of features is  7
Accuracy: 0.7337662337662337
F1-score: 0.7287952930385277
---------------------------------
Number of features is  8
Accuracy: 0.7207792207792207
F1-score: 0.7124346917450366
---------------------------------
```

[21]:
```
n_best_features
```

[21]: 6

[22]:
```
rfe_best
```

[22]:
```
RFE(estimator=RandomForestClassifier(), n_features_to_select=6)
```

2.4.4 Summarize scores

[23]:
```
print("Num Features: %s" % (rfe_best.n_features_))
print("Selected Features: %s" % (rfe_best.support_))
print("Feature Ranking: %s" % (rfe_best.ranking_))
```

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

2.4.5 Print the features

[24]:
```
df_scores = pd.DataFrame(rfe_best.ranking_)
df_columns = pd.DataFrame(X.columns)

# concatenate dataframes
feature_scores = pd.concat([df_columns, df_scores],axis=1)
feature_scores.columns = ['Features','Ranking']
print(feature_scores.nsmallest(8,'Ranking'))
```

```
                  Features  Ranking
0              Pregnancies        1
1                  Glucose        1
2            BloodPressure        1
5                      BMI        1
6  DiabetesPedigreeFunction        1
7                      Age        1
4                  Insulin        2
3            SkinThickness        3
```

These are marked True in the support array and marked with a choice “1” in the ranking array. This, in turn, indicates the strength of these features.

[ ]:
```

```