Dimensionality Reduction and Data Feature Selection

Prof. Shibdas Dutta,

Associate Professor,

DCG DATA CORE SYSTEMS INDIA PVT LTD

Kolkata

Data Feature Selection

In the previous chapter, we have seen in detail how to preprocess and prepare data for machine learning.

In this chapter, let us understand in detail data feature selection and various aspects involved in it.

Importance of Data Feature Selection

The performance of machine learning model is directly proportional to the data features used to train it.

The performance of ML model will be affected negatively if the data features provided to it are irrelevant.

On the other hand, use of relevant data features can increase the accuracy of your ML model especially linear and logistic regression.

Now the question arise that what is automatic feature selection? It may be defined as the process with the help of which we select those features in our data that are most relevant to the output or prediction variable in which we are interested. It is also called attribute selection.

The following are some of the benefits of automatic feature selection before modeling the data:

- Performing feature selection before data modeling will reduce the overfitting.

- Performing feature selection before data modeling will increases the accuracy of ML model.

- Performing feature selection before data modeling will reduce the training time

Feature Selection Techniques

The followings are automatic feature selection techniques that we can use to model ML data in Python:

Univariate Selection

This feature selection technique is very useful in selecting those features, with the help of statistical testing, having strongest relationship with the prediction variables.

We can implement univariate feature selection technique with the help of **`SelectKBest`** class of **`scikit-learn`** Python library.

Example:

In this example, we will use Pima Indians Diabetes dataset to select 4 of the attributes having best features with the help of chi-square statistical test.

```
from pandas import read_csv
from numpy import set_printoptions


from sklearn.feature_selection import SelectKBest
```

```
from sklearn.feature_selection import chi2

path = r'C:\pima-indians-diabetes.csv'

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age',
'class']

dataframe = read_csv(path, names=names)

array = dataframe.values
```

Next, we will separate array into input and output components:

```
X = array[:,0:8]
Y = array[:,8]
```

The following lines of code will select the best features from dataset:

```
test = SelectKBest(score_func=chi2, k=4)

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

We can also summarize the data for output as per our choice. Here, we are setting the precision to 2 and showing the 4 data attributes with best features along with best score of each attribute:

```
set_printoptions(precision=2)

print(fit.scores_)

featured_data = fit.transform(X)

print ("\nFeatured data:\n", featured_data[0:4])
```

Output

```
[111.52  1411.89      17.61        53.11 2175.57 127.67    5.39 181.3]

Featured data:

 [[148.   0.          33.650.]
  [85.    0.          26.631.]
  [183.   0.          23.332.]
  [89.    94.         28.121.]]
```

Recursive Feature Elimination (RFE)

As the name suggests, RFE (Recursive feature elimination) feature selection technique removes the attributes recursively and builds the model with remaining attributes.

We can implement RFE feature selection technique with the help of **`RFE`** class of **`scikit-learn`** Python library.

Example

In this example, we will use RFE with logistic regression algorithm to select the best 3 attributes having the best features from Pima Indians Diabetes dataset to.

```
from pandas import read_csv
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

path = r'C:\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age',
'class']

dataframe = read_csv(path, names=names)

array = dataframe.values
```

Next, we will separate the array into its input and output components:

```
X = array[:,0:8]
Y = array[:,8]
```

The following lines of code will select the best features from a dataset:

```
model = LogisticRegression()

rfe = RFE(model, n_features_to_select=3)

fit = rfe.fit(X, Y)

print("Number of Features: %d" %fit.n_features_)

print("Selected Features: %s" %fit.support_)

print("Feature Ranking: %s" %fit.ranking_)
```

Output

```
Number of Features: 3

Selected Features: [ True False False False False True True False]

Feature Ranking: [1 2 3 5 6 1 1 4]
```

We can see in above output, RFE choose preg, mass and pedi as the first 3 best features. They are marked as 1 in the output.

Principal Component Analysis (PCA)

PCA, generally called data reduction technique, is very useful feature selection technique as it uses linear algebra to transform the dataset into a compressed form.

We can implement PCA feature selection technique with the help of **`PCA`** class of **`scikit-learn`** Python library. We can select number of principal components in the output.

Example:

In this example, we will use PCA to select best 3 Principal components from Pima Indians Diabetes dataset.

```
from pandas import read_csv

from sklearn.decomposition import PCA

path = r'C:\pima-indians-diabetes.csv'

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age',

'class']

dataframe = read_csv(path, names=names)

array = dataframe.values
```

Next, we will separate array into input and output components:

```
X = array[:,0:8]
Y = array[:,8]
```

The following lines of code will extract features from dataset:

```
pca = PCA(n_components=3)

fit = pca.fit(X)

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

print(fit.components_)
```

Output

```
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]]
```

We can observe from the above output that 3 Principal Components bear little resemblance to the source data.

Feature Importance

As the name suggests, feature importance technique is used to choose the importance features. It basically uses a trained supervised classifier to select features.

We can implement this feature selection technique with the help of **`ExtraTreeClassifier`** class of **`scikit-learn`** Python library.

Example

In this example, we will use **`ExtraTreeClassifier`** to select features from Pima Indians Diabetes dataset.

```
from pandas import read_csv

from sklearn.ensemble import ExtraTreesClassifier

path = r'C:\Desktop\pima-indians-diabetes.csv'

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age',
'class']

dataframe = read_csv(data, names=names)

array = dataframe.values
```

Next, we will separate array into input and output components:

```
X = array[:,0:8]
Y = array[:,8]
```

The following lines of code will extract features from dataset:

```
model = ExtraTreesClassifier()

model.fit(X, Y)

print(model.feature_importances_)
```

Output

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

From the output, we can observe that there are scores for each attribute.

The higher the score, higher is the importance of that attribute.

Thank You