Machine Learning Assignment - III

September 30, 2021

Kannan Singaravelu

kannan.singaravelu@fitchlearning.com

* * *

Exercise 5

a) What is Gini Impurity?

b) When will the nodes record its highest and lowest impurity measure?

c) Plot the Gini Impurity Measure for two classes.

Solution

a) What is Gini Impurity?

Gini impurity metric is a measure of node impurity. It is equal to the probability of misclassifying an observation. For classification problems, decision tree will perform node splits that will result in reducing the Gini metric.

For a dataset D that contains samples from k classes, the probability of samples belonging to class i at a given node can be denoted as p_i and the Gini Index is then defined as

$$G = \sum_{k=1}^{K} p_{mk}(1 - p_{mk})$$

where p_{mk} is the fraction of observation of class k in node m and G is the Gini cost over all classes.

b) When will the tree node record its highest and lowest impurity measure?

Consider two scenarios where a node has 20 observations belonging to two classes. In scenario 1, we have equal representation of each class in the node and in scenario 2, all observations are recorded only in the first class.

Th Gini metric for both scenarios can then be represented as

- Scenario 1 : [10,10]

$$G = \frac{10}{20}\left(1 - \frac{10}{20}\right) + \frac{10}{20}\left(1 - \frac{10}{20}\right) = 0.5$$

- Scenario 1 : [20,0]

$$G = \frac{20}{20}\left(1 - \frac{20}{20}\right) + \frac{0}{20}\left(1 - \frac{0}{20}\right) = 0$$

As observed above, node with uniform class distribution has the highest impurity and the minimum impurity is recorded when all observations belong to the same class. This was evident from the graph as well.

c) Plot the Gini Impurity Measure

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
# Gini Impurity for two classes
p = np.linspace(0.01,1)
# gini = 1 - (p*p) - (1-p)*(1-p)

gini = p*(1-p) + (1-p)*p

plt.plot(p, gini)
plt.title('Gini Impurity')
plt.xlabel("Fraction of Class k ($p_k$)")
plt.ylabel("Gini Impurity Measure")
plt.xticks(np.arange(0,1.1,0.1))
plt.show()
```

Exercise 6

a) What is Anomaly Detection? List down the popular learning algorithms used for anomaly detection.

b) What is the Isolation Forest algorithm? How do Isolation Forests work?

c) For this exercise, use the credit card dataset from https://www.kaggle.com/mlg-ulb/creditcardfraud. The dataset contains thirty predictor variables and one binary response variable called class. Build a model to detect credit card fraud using the Isolation Forest Anomaly Detection algorithm.

Solution

a) What is Anomaly Detection? List down the popular learning algorithms used for anomaly detection.

Anomalies can be broadly categorized as a) Point anomalies; b) Contextual anomalies and c) Collective anomalies.

Anomaly detection is a technique used to identify unusual patterns that do not conform to expected behavior, called outliers. Anomaly detection is similar to noise removal and novelty detection.

While simple statistical methods can be applied to identify irregularities, machine learning-based approaches are proved to be effective. Learning algorithms such as K-NN, K-Means, Local Outlier Factor, and OneClassSVM are used for anomaly detection.

b) What is the Isolation Forest algorithm? How do Isolation Forests work?

One of the newest techniques to detect anomalies is called Isolation Forests. It is based on the fact that anomalies are points that are few/different and can be easily isolated. This method is highly useful and is fundamentally different from the existing methods. It introduces the use of isolation as a more effective and efficient means to detect anomalies than the usual distance and density measures.

The algorithm isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. Isolating anomaly observations are easier as there are only a few conditions needed to separate those cases from the normal observations.

The algorithm constructs the separation by first creating isolation trees (or random decision trees). Then, the score is calculated as the path length to isolate the observation.

[3]:
```
# Import Library
from sklearn.ensemble import IsolationForest
from sklearn.metrics import classification_report,accuracy_score
```

Dataset The dataset is sourced from Kaggle from the above link. The dataset has been collected and analyzed during a research collaboration of Worldline and the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection.

[4]:
```
df = pd.read_csv('creditcard.csv')
df.head()
```

[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	\
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	

	V8	V9	…	V21	V22	V23	V24	V25	\
0	0.098698	0.363787	…	-0.018307	0.277838	-0.110474	0.066928	0.128539	
1	0.085102	-0.255425	…	-0.225775	-0.638672	0.101288	-0.339846	0.167170	
2	0.247676	-1.514654	…	0.247998	0.771679	0.909412	-0.689281	-0.327642	
3	0.377436	-1.387024	…	-0.108300	0.005274	-0.190321	-1.175575	0.647376	
4	-0.270533	0.817739	…	-0.009431	0.798278	-0.137458	0.141267	-0.206010	

	V26	V27	V28	Amount	Class
0	-0.189115	0.133558	-0.021053	149.62	0
1	0.125895	-0.008983	0.014724	2.69	0
2	-0.139097	-0.055353	-0.059752	378.66	0
3	-0.221929	0.062723	0.061458	123.50	0
4	0.502292	0.219422	0.215153	69.99	0

```
[5 rows x 31 columns]
```

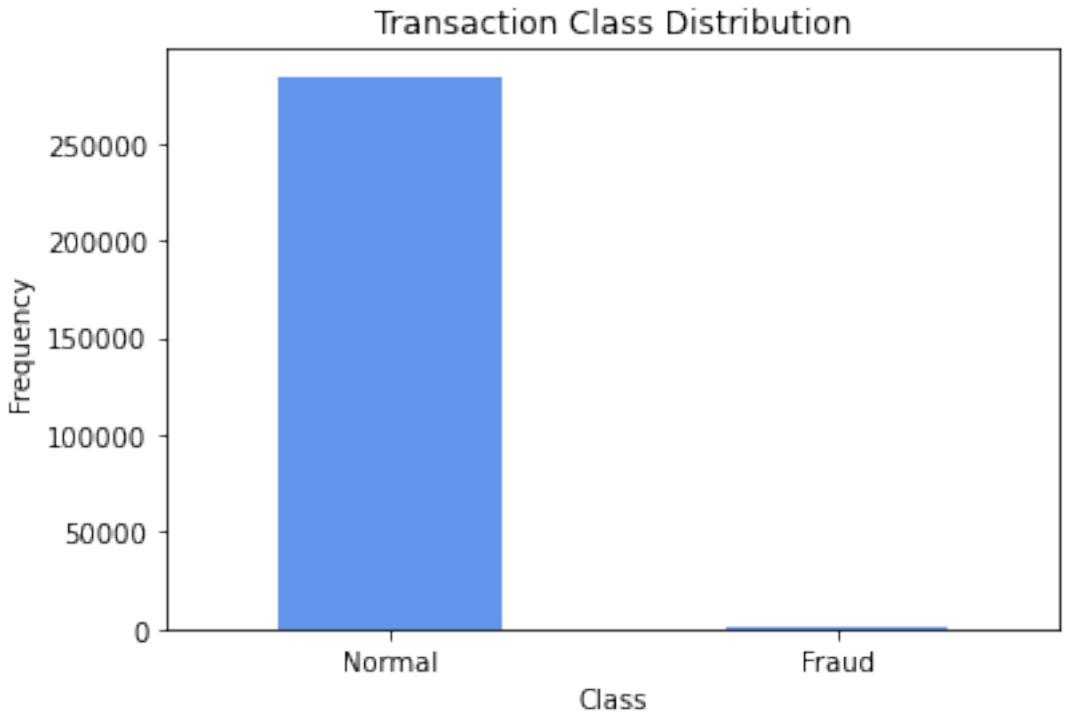
The dataset consists of numerical values from the 28 PCA transformed features, namely V1 to V28. Further, metadata about the original features are not provided and hence we cannot perform any feature study. The Time and Amount features are not transformed data and there is no missing value as this is a pre-cleaned dataset.

Exploratory Data Analysis

[5]:
```
# check for missing values
df.isnull().values.any()
```

[5]:
```
False
```

[6]:
```
# plot classes
count_classes = pd.value_counts(df['Class'], sort = True)
count_classes.plot(kind = 'bar', rot=0, color='cornflowerblue')
plt.title("Transaction Class Distribution")
plt.xticks(range(2),["Normal", "Fraud"])
plt.xlabel('Class')
plt.ylabel('Frequency')
plt.show()
```



[7]:
```
# split normal and fraud transactions
fraud = df[df['Class']==1]
normal = df[df['Class']==0]
outlier_fraction = len(fraud)/len(normal)

print(fraud.shape, normal.shape, outlier_fraction)
```

```
(492, 31) (284315, 31) 0.0017304750013189597
```

[8]:
```
# using only 1% of the data for this exercise
df1 = df.sample(frac = 0.1, random_state=1)
df1.shape
```

[8]:
```
(28481, 31)
```

Let's build the model from this sub sample dataset.

[9]:
```
cols = df1.drop('Class', axis=1).columns
target = 'Class'
```

[10]:
```
# specify X, y and random state
state = np.random.RandomState(42)
X = df1[cols].values
y = df1[target].values
```

[11]:
```
y
```

[11]:
```
array([0, 0, 0, …, 0, 0, 0])
```

[12]:
```
# define the outlier detection method
iso = IsolationForest(n_estimators=100, max_samples=len(X),
                      contamination=outlier_fraction,random_state=state)

iso.fit(X)
```

[12]:
```
IsolationForest(contamination=0.0017304750013189597, max_samples=28481,
                random_state=RandomState(MT19937) at 0x1A20B68CA8)
```

[13]:
```
scores = iso.decision_function(X)
y_pred = iso.predict(X)
```

[14]:
```
scores
```

[14]:
```
array([0.22880271, 0.25793869, 0.25845758, …, 0.24077742, 0.22001207,
       0.24963303])
```

[15]:
```
y_pred
```

[15]:
```
array([1, 1, 1, …, 1, 1, 1])
```

[16]:
```
# reshape the prediction values to 0 for normal and 1 for fraudulent␣
 ↪transactions
y_pred[y_pred == 1] = 0
y_pred[y_pred == -1] = 1
n_errors = (y_pred != y).sum()
ac = accuracy_score(y,y_pred)
cr = classification_report(y,y_pred)

# classification metrics
print(f"No of Errors \t {n_errors}")
print(f"Accuracy Score \t {100.*ac:0.4}%")
print()
print(f"Classification Report")
print(cr)
```

```
No of Errors     77
Accuracy Score   99.73%

Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28432
           1       0.22      0.22      0.22        49

    accuracy                           1.00     28481
   macro avg       0.61      0.61      0.61     28481
weighted avg       1.00      1.00      1.00     28481
```

The Isolation Forest model detected 77 errors with a 99.73% accuracy.

References

- Scikit-learn Isolation Forest
- Python resources

Kannan Singaravelu
Certificate in Quantitative Finance, June 2021