ASSIGNMENT 1

CSYE – 7245 – BIG DATA SYSTEMS & INTELLIGENCE ANALYTICS

a. ABSTRACT:

Credit Card Fraud detection dataset is used to predict anomaly detection. By applying Logistic Regression on the dataset we would be performing binary classification as we have two Classes Normal and fraud. The data has to be resampled and cross validated in order to prepare it for Logistic Regression. I will be using ROC (Receiver operating characteristic), AUC (Area under ROC curve) and Confusion metric for anomaly detection.

b. INTRODUCTION:

Anomaly detection is one of the most popular machine learning algorithms. The Credit card fraud detection is a type of anomaly detection where the prediction can be made based on **Logistic Regression model** if the transaction is fraud or normal. The dataset is made up of 284807 rows and 28 features. The dataset is highly unbalanced with about 99.82% of the transactions being normal and only 0.17% fraud transactions. The dataset has to be resampled before applying any machine learning algorithm.

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables (in our case 28 features) that determine an outcome i.e. Normal or false transaction. The outcome is measured with a dichotomous variable i.e. Class (in which there are only two possible outcomes).
In logistic regression, the dependent variable is binary or dichotomous, i.e. it only contains data coded as 1 (TRUE i.e. fraud) or 0 (FALSE i.e. Normal).
Hence, for this anomaly prediction the Logistic regression model would fit perfectly.
Since, it is binary classification, we can visualize the results using ROC curve.

c. CODE WITH DOCUMENTATION:

1. IMPORTING PACKAGES:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

%matplotlib inline
```

2. LOADING DATASET:

```
data = pd.read_csv("C:\Users\user\Downloads\creditcardfraud\creditcard.csv")
```

3. EXPLORATORY DATA ANALYSIS:

1. Get percentage of normal and fraud transactions:

```
Count_Normal_transaction = len(data[data["Class"]==0]) # normal transaction are repersented by 0
Count_Fraud_transaction = len(data[data["Class"]==1]) # fraud by 1
print(" Total Normal transactions: " ,Count_Normal_transaction)
print(" Total Fraud transactions: " ,Count_Fraud_transaction)
total_transactions = Count_Normal_transaction + Count_Fraud_transaction
print("Total transactions: ", total_transactions)
```

2. Plotting Normal and fraud transaction:

```
#amount of valid transactions and fraud transactions
Fraud_transacation = data[data["Class"]==1]
Normal_transacation= data[data["Class"]==0]
plt.figure(figsize=(10,6))
plt.subplot(121)
Fraud_transacation.Amount.plot.hist(title="Fraud Transacation")
plt.subplot(122)
Normal_transacation.Amount.plot.hist(title="Normal Transaction")
```

3. Data Summary statistics:

```
data.describe()
```

4. Plot each column:

The data is very sensitive. Hence, there are about 28 different unknown factors (columns) which contribute to the analysis. Since these features are unknown, we cannot comment on the correlation

```
import seaborn as sns
data_corr = data.corr()['Class'][:-1] # -1 because the latest row is Class
golden_features_list = data_corr[abs(data_corr) > 0.5].sort_values(ascending=False)
print("There is {} strongly correlated values with Class:\n{}".format(len(golden_features_list), golden_features_list))
for i in range(0, len(data.columns), 5):
   sns.pairplot(data=data,
                x_vars=data.columns[i:i+5],
                y_vars=['Class'])
```

5. Column correlation:

```
corr = data.drop('Class', axis=1).corr() # We already examined Class correlations
```

```
plt.figure(figsize=(20, 10))

sns.heatmap(corr[(corr <= 0.5) | (corr >= -0.004)],
        cmap='viridis', vmax=1.0, vmin=-1.0, linewidths=0.1,
        annot=True, annot_kws={"size":8}, square=True);
```

After Exploratory data analysis, we can conclude that none of the features are correlated to each other and hence all the features should be retained for further model building.

4. NORMALIZATON:

Since amount and time has not effect on the data, it can be normalized.

```
from sklearn.preprocessing import StandardScaler

data['normAmount'] = StandardScaler().fit_transform(data['Amount'].reshape(-1, 1))
data = data.drop(['Time','Amount'],axis=1)
data.head()
```

5. RESAMPLING:

The data has to be resampled because there is a very high unbalance in the dataset. About 99.3% of the data is normal while only 0.17% of the data is fraud.

```
import sys
X = data.ix[:,data.columns != "Class"]
Y = data.ix[:, data.columns == "Class"]
```

UNDERSAMPLING:

We need to under sample the normal data in order to get the data to be in the ratio of 50:50.

```
#Number of data points in the minority class i.e fraud class
no_fraud = len(data[data.Class == 1])
fraud_indices = np.array(data[data.Class == 1].index)

# Picking the indices of the normal classes
normal_indices = data[data.Class == 0].index

# Out of the indices we picked, randomly select "x" number (number_records_fraud)
```

```
random_normal_indices = np.random.choice(normal_indices, no_fraud, replace = False)
random_normal_indices = np.array(random_normal_indices)

# Appending the 2 indices
under_sample_indices = np.concatenate([fraud_indices,random_normal_indices])

# Under sample dataset
under_sample_data = data.iloc[under_sample_indices,:]

X_undersample = under_sample_data.iloc[:, under_sample_data.columns != 'Class']
y_undersample = under_sample_data.iloc[:, under_sample_data.columns == 'Class']

# Showing ratio
print("Percentage of normal transactions: ", (float)(len(under_sample_data[under_sample_data.Class == 0]))/len(under_sample_data))
print("Percentage of fraud transactions: ", (float)(len(under_sample_data[under_sample_data.Class == 1]))/len(under_sample_data))
print("Total number of transactions in resampled data: ", len(under_sample_data))
```

6. **CROSS VALIDATION:**

It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform.
In K-fold, the sample is partitioned into K-folds of equal subsamples.
C parameter is a trade-off between training error and the flatness of the solution. The larger C is the less the final training error will be.

7. **LOGISTICS REGRESSION:**

Binary classification between Normal and fraud transactions can be visualized using ROC and AUC. The logistic regression model can be predicted based on the recall metric based on confusion matrix as well as ROC and AUC.

1. Get the best C parameter for K-fold:

```
best_c = printing_Kfold_scores(X_train_undersample,y_train_undersample)
```

2. Confusion matrix function:

```
# Use this C_parameter to build the final model with the whole training dataset and predict the classes in the test
```

```
# dataset
lr = LogisticRegression(C = best_c, penalty = 'l1')
lr.fit(X_train_undersample,y_train_undersample.values.ravel())
y_pred_undersample = lr.predict(X_test_undersample.values)

# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test_undersample,y_pred_undersample)
np.set_printoptions(precision=2)

print("Recall metric in the testing dataset: ", (float)(cnf_matrix[1,1])/(cnf_matrix[1,0]+cnf_matrix[1,1]))

# Plot non-normalized confusion matrix
class_names = [0,1]
plt.figure()
plot_confusion_matrix(cnf_matrix
                      , classes=class_names
                      , title='Confusion matrix')
plt.show()
```

3. **ROC Curve:**

```
lr = LogisticRegression(C = best_c, penalty = 'l1')
y_pred_undersample_score = lr.fit(X_train_undersample,y_train_undersample.values.ravel()).decision_function(X_test_undersample.values)

fpr, tpr, thresholds = roc_curve(y_test_undersample.values.ravel(),y_pred_undersample_score)
roc_auc = auc(fpr,tpr)

# Plot ROC
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b',label='AUC = %0.2f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.0])
plt.ylim([-0.1,1.01])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

d. RESULTS:

An ROC curve is the used to visualize the performance of a binary classifier, and AUC is used to summarize its performance in a single number. ROC is a curve created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. AUC is the area under the ROC curve. Maximum area means very good prediction. Logistic Regression uses this in order to evaluate performance as it is a binary classifier algorithm. The AUC of the model is very high i.e. 0.97 depicting that the prediction for binary classification is very good.

The recall metric for the whole testing data came out to be about 0.9416 which is very high.

e. REFERENCES:

https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/
https://www.kaggle.com/ekami66/detailed-exploratory-data-analysis-with-python
https://www.kaggle.com/dalpozz/creditcardfraud
https://www.kaggle.com/joparga3/in-depth-skewed-data-classif-93-recall-acc-now
https://en.wikipedia.org/wiki/Cross-validation_(statistics)
http://www.dataschool.io/roc-curves-and-auc-explained/
https://www.medcalc.org/manual/logistic_regression.php