Abhishek Patwardhan

D17A - 57

ADS Experiment 6

==> Importing Libraries

```
import pandas as pd
import numpy as np
from sklearn.neighbors import NearestNeighbors
import matplotlib.pyplot as plt
import warnings
from google.colab import files
import io
warnings.filterwarnings('ignore')
```

```
uploaded = files.upload()
df = pd.read_csv(io.BytesIO(uploaded['creditcard.csv']))
df.head()
```

Choose Files creditcard.csv

- **creditcard.csv**(text/csv) - 150828752 bytes, last modified: 3/20/2023 - 100% done

Saving creditcard.csv to creditcard (1).csv

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...

5 rows × 31 columns

Sampling the dataset

1% of the entire populace

```
# Select a random 1% of the data
df1 = df.sample(frac=0.01, random_state=42)
```

==> Instantiating the KNN Model

```
knn_model = NearestNeighbors(n_neighbors=5)
knn_model.fit(df1)
```

NearestNeighbors

NearestNeighbors()

```
# Find the distance of each point to its k-th nearest neighbor
distances, indices = knn_model.kneighbors()
k_distances = distances[:, -1]
```

==> Defining Outlier Score and marking Outliers

```
# Calculate the outlier score for each point
outlier_score = k_distances / np.mean(k_distances)
```

```
# Identify outliers based on the threshold
outliers = np.where(outlier_score > 1.5)[0]
```

```
df1 = df1.reset_index(drop=True)
```

```
df1['Outlier'] = np.NaN
```

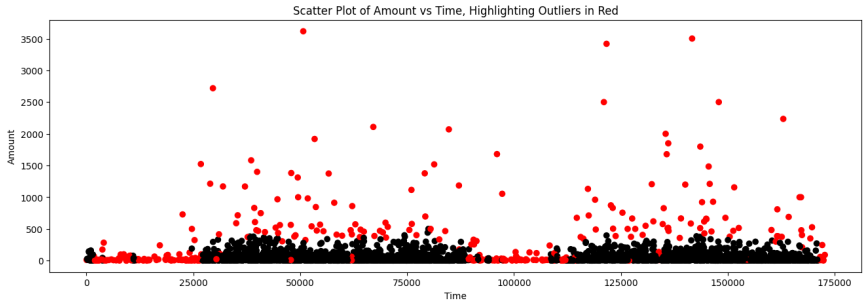
```
outlier_count = 0
for i in df1.index:
  if i in outliers:
    df1['Outlier'][i] = 1
    outlier_count += 1
  else:
    df1['Outlier'][i] = 0
print(outlier_count)
```

```
334
```

```
print("The total entries in dataframe are", df1.shape[0])
print("The number of outliers are", outlier_count)
```

```
The total entries in dataframe are 2848
The number of outliers are 334
```

```
plt.figure(figsize=(16,5))
# Create a color map for the Class values
color_map = {0: 'black', 1: 'red'}
# Create a scatter plot of Amount vs Time, with color based on Class
plt.scatter(df1['Time'], df1['Amount'], c=df1['Outlier'].map(color_map))
# Set the axis labels and title
plt.xlabel('Time')
plt.ylabel('Amount')
plt.title('Scatter Plot of Amount vs Time, Highlighting Outliers in Red')
# Display the plot
plt.show()
```



==> Implementing DBSCAN

```
from sklearn.cluster import DBSCAN
```

DBSCAN that will be performed on the data and returns the cluster labels. A cluster label of -1 is considered as outlier

```
outlier_detection = DBSCAN(eps = 500, min_samples = 5)
```

```
clusters = outlier_detection.fit_predict(df1)
```

```
df1['Cluster'] = clusters
```

```
print("The number of clusters thus formed using DBSCAN are : ", len(df1['Cluster']))
```

```
     The number of clusters thus formed using DBSCAN are :  2848
```

==> Dissolving the rest of the classes in favour of Outlier class

```
for i in df1.index:
  if df1['Cluster'][i] == -1:
    pass
  else:
    df1['Cluster'][i] = 0
```

```
plt.figure(figsize=(16,5))
# Create a color map for the Class values
color_map = {0: 'black', -1: 'red'}
# Create a scatter plot of Amount vs Time, with color based on Cluster
plt.scatter(df1['Time'], df1['Amount'], c=df1['Cluster'].map(color_map))
# Set the axis labels and title
plt.xlabel('Time')
plt.ylabel('Amount')
plt.title('Scatter Plot of Amount vs Time, Highlighting Outliers in Red using DBS')
# Display the plot
plt.show()
```