Credit Card Fraud Detection - K-Nearest Neighbor(KNN)

Importing the Dependencies

```
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pylab as plt

%matplotlib inline

# loading the dataset to a Pandas DataFrame
df = pd.read_csv('/content/creditcard.csv') # uses pd which is the
alias for pandas df
```

Generate a smaller dataset for better processing

```
# Filter fraud cases (Class = 1)
fraud_cases = df[df['Class'] == 1]

fraud_cases
```

{"type":"dataframe","variable_name":"fraud_cases"}

```
# Filter non-fraud cases (Class = 0) and randomly sample 5000 rows
non_fraud_cases = df[df['Class'] == 0].sample(n=5000, random_state=42)

non_fraud_cases
```

{"type":"dataframe","variable_name":"non_fraud_cases"}

```
# Combine the two datasets
smaller_dataset = pd.concat([fraud_cases, non_fraud_cases])

smaller_dataset
```

{"type":"dataframe","variable_name":"smaller_dataset"}

```
# Shuffle the dataset
credit_card_data = smaller_dataset.sample(frac=1,
random_state=42).reset_index(drop=True)

credit_card_data
```

{"type":"dataframe","variable_name":"credit_card_data"}

Dataset info

```
credit_card_data.describe().transpose()
```


```
{"summary":"{\n  \"name\": \"credit_card_data\",\n  \"rows\": 31,\n
\"fields\": [\n    {\n      \"column\": \"count\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.0,\n        \"min\": 5492.0,\n        \"max\": 5492.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          5492.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"mean\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
16823.603924148127,\n        \"min\": -0.642816141147264,\n
\"max\": 93672.65640932265,\n        \"num_unique_values\": 31,\n
\"samples\": [\n          0.015646795469693147\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"std\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 8533.216197343192,\n
\"min\": 0.2856120769620389,\n        \"max\": 47519.474137857294,\n
\"num_unique_values\": 31,\n        \"samples\": [\n
0.5666777332177803\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"min\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 13.718084772209473,\n        \"min\": -43.5572415712451,\n
\"max\": 23.0,\n        \"num_unique_values\": 30,\n
\"samples\": [\n          -7.26348214633855\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"25%\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 9557.177308279628,\n
\"min\": -1.201457702246185,\n        \"max\": 53211.75,\n
\"num_unique_values\": 31,\n        \"samples\": [\n          -
0.06958052558712492\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"50%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 15186.044196557576,\n        \"min\": -0.323508637396354,\n
\"max\": 84553.0,\n        \"num_unique_values\": 31,\n
\"samples\": [\n          0.007780361595404449\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"75%\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 24803.118842769083,\n
\"min\": 0.0,\n        \"max\": 138101.0,\n
\"num_unique_values\": 31,\n        \"samples\": [\n
0.12286592844787475\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"max\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 30999.462165605302,\n        \"min\": 1.0,\n        \"max\":
172703.0,\n        \"num_unique_values\": 31,\n        \"samples\": [\
n          4.61093644143655\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```

```
# dataset informations
credit_card_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5492 entries, 0 to 5491
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Time    5492 non-null   float64
 1   V1      5492 non-null   float64
 2   V2      5492 non-null   float64
 3   V3      5492 non-null   float64
 4   V4      5492 non-null   float64
 5   V5      5492 non-null   float64
 6   V6      5492 non-null   float64
 7   V7      5492 non-null   float64
 8   V8      5492 non-null   float64
 9   V9      5492 non-null   float64
 10  V10     5492 non-null   float64
 11  V11     5492 non-null   float64
 12  V12     5492 non-null   float64
 13  V13     5492 non-null   float64
 14  V14     5492 non-null   float64
 15  V15     5492 non-null   float64
 16  V16     5492 non-null   float64
 17  V17     5492 non-null   float64
 18  V18     5492 non-null   float64
 19  V19     5492 non-null   float64
 20  V20     5492 non-null   float64
 21  V21     5492 non-null   float64
 22  V22     5492 non-null   float64
 23  V23     5492 non-null   float64
 24  V24     5492 non-null   float64
 25  V25     5492 non-null   float64
 26  V26     5492 non-null   float64
 27  V27     5492 non-null   float64
 28  V28     5492 non-null   float64
 29  Amount  5492 non-null   float64
 30  Class   5492 non-null   int64  
dtypes: float64(30), int64(1)
memory usage: 1.3 MB
```

```
# checking the number of missing values in each column
credit_card_data.isnull().sum()
```

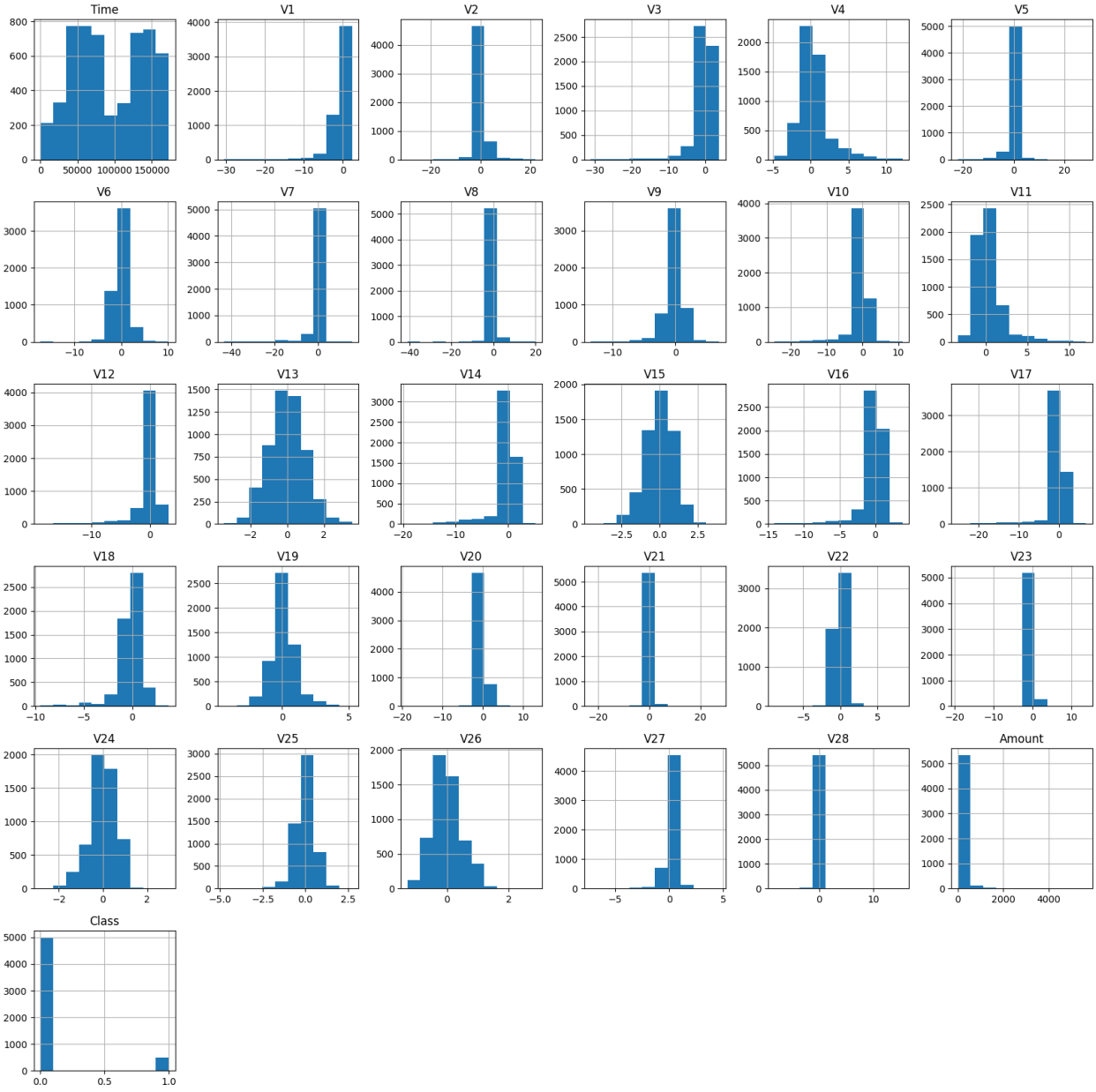
```
Time      0
V1        0
V2        0
V3        0
V4        0
```

```
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

```
credit_card_data.hist(figsize=(20,20))
```

```
array([[<Axes: title={'center': 'Time'}>, <Axes: title={'center':
'V1'}>,
        <Axes: title={'center': 'V2'}>, <Axes: title={'center':
'V3'}>,
        <Axes: title={'center': 'V4'}>, <Axes: title={'center':
'V5'}>],
       [<Axes: title={'center': 'V6'}>, <Axes: title={'center':
'V7'}>,
        <Axes: title={'center': 'V8'}>, <Axes: title={'center':
'V9'}>,
        <Axes: title={'center': 'V10'}>, <Axes: title={'center':
'V11'}>],
       [<Axes: title={'center': 'V12'}>, <Axes: title={'center':
'V13'}>,
        <Axes: title={'center': 'V14'}>, <Axes: title={'center':
'V15'}>,
        <Axes: title={'center': 'V16'}>, <Axes: title={'center':
'V17'}>],
       [<Axes: title={'center': 'V18'}>, <Axes: title={'center':
'V19'}>,
```

```
        <Axes: title={'center': 'V20'}>, <Axes: title={'center':
'V21'}>,
        <Axes: title={'center': 'V22'}>, <Axes: title={'center':
'V23'}>],
       [<Axes: title={'center': 'V24'}>, <Axes: title={'center':
'V25'}>,
        <Axes: title={'center': 'V26'}>, <Axes: title={'center':
'V27'}>,
        <Axes: title={'center': 'V28'}>,
        <Axes: title={'center': 'Amount'}>],
       [<Axes: title={'center': 'Class'}>, <Axes: >, <Axes: >, <Axes:
>,
        <Axes: >, <Axes: >]], dtype=object)
```



Scatter Plot

```
sns.pairplot(credit_card_data, hue='Class')
```

```
<seaborn.axisgrid.PairGrid at 0x7bddce9a1ba0>
```

Standardize the Variables

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X =
pd.DataFrame(scaler.fit_transform(credit_card_data.drop(["Class"],axis
= 1)))
y = credit_card_data.Class

X.head()
```

{"type":"dataframe","variable_name":"X"}

Train Test Split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size=0.30)
```

Using KNN

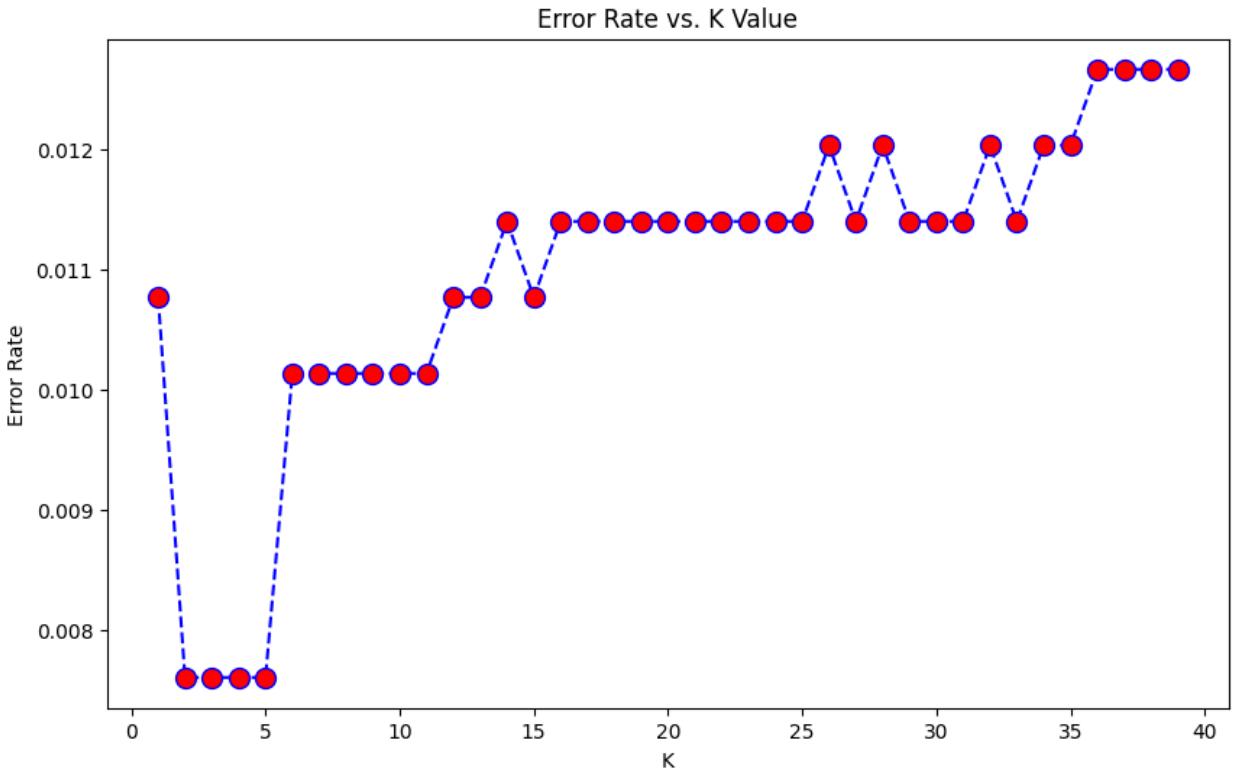
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(X_train,y_train)

KNeighborsClassifier(n_neighbors=1)

pred = knn.predict(X_test)
```

Predictions and Evaluations

```
from sklearn.metrics import classification_report,confusion_matrix

print(confusion_matrix(y_test,pred))

[[1498    6]
 [  11   64]]

print(classification_report(y_test,pred))

              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99      1504
         1.0       0.91      0.85      0.88        75

    accuracy                           0.99      1579
   macro avg       0.95      0.92      0.94      1579
weighted avg       0.99      0.99      0.99      1579


error_rate = []

# Will take some time
for i in range(1,40):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))

plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed',
```

```
marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

```
Text(0, 0.5, 'Error Rate')
```




```
#Orginal K=3
knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(X_train,y_train)
pred = knn.predict(X_test)

print('WITH k=3')
print('\n')
print(confusion_matrix(y_test,pred))
print('\n')
print(classification_report(y_test,pred))
```

```
WITH k=3


[[1498    6]
 [  11   64]]
```

```
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99      1504
         1.0       0.91      0.85      0.88        75

    accuracy                           0.99      1579
   macro avg       0.95      0.92      0.94      1579
weighted avg       0.99      0.99      0.99      1579
```

```
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt

conf_matrix = confusion_matrix(y_test, pred)
vis = ConfusionMatrixDisplay(confusion_matrix =
conf_matrix,display_labels = [True,False])
vis.plot()
plt.grid(False)
plt.show()
```