CREDIT CARD APPROVAL

Aathithya-CB.EN.U4ELC20001

Adarsh-CB.EN.U4ELC20005

Arun-CB.EN.U4ELC20010

AMRITA VISHWA VIDYAPEETHAM

References

- Credit Card Approval Dataset:
  https://www.kaggle.com/datasets/samuelcortinhas/credit-card-approval-clean-data
- Normalization:
  https://www.geeksforgeeks.org/normalization-vs-standardization/
- PCA Theory:
  https://towardsdatascience.com/theory-of-principal-component-analysis-pca-and-implementation-on-python-5d4839f9ae89

Problem Formulation

<u>**Objective**</u>: To classify people described by a set of attributes as good or bad credit risks for credit card applications.

<u>**Dataset details**</u>:

- No.of rows: 691
- No.of columns: 16
- No.of classes: 2
- Method of data collection is unknown

<u>**Link to full code mentioned in slides**</u>:

https://colab.research.google.com/drive/15muY91nzmtp0q0ODojKBhgd-I33CDVhb?usp=sharing

Assumptions:

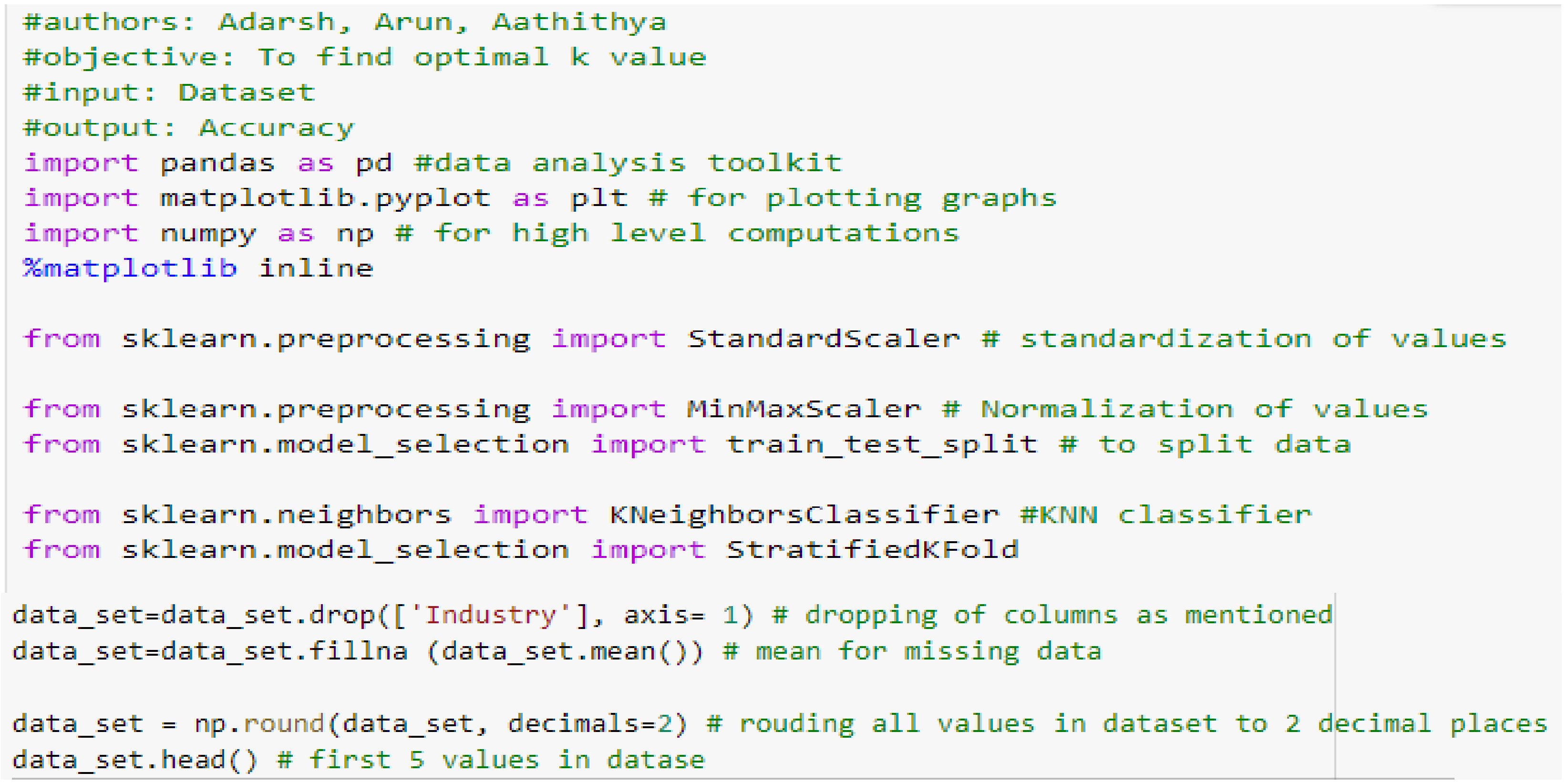
- Feature- “Industry” was ignored while computing due to the presence of multiple strings and requires the help of NLP for solving.
- The feature “Ethnicity” was numeralised where:
  - White=0/Black=1/Asian=2/Latino=3/Other=4
- The feature “Gender” was numeralised where:
  - Female=0/Male=1
- The feature “Citizen” was numeralised where:
  - ByOtherMeans=0/ByBirth=1/Temporary=2

Feature Description

- Gender-Determines the Gender(Male/Female).
- Age-Age of the applicant.
- Debt-Total unpaid debt under the applicants name.
- Married-Checks if married or not.
- Bank Customer-Checks if the applicant is a customer of the bank.
- Industry-The industry in which the applicant works(Ignored).
- Ethnicity-Checks the Ethnicity of the applicant.
- Years Employed-Checks the number of years of employment.
- Prior Default-Checks prior default in returning loans.

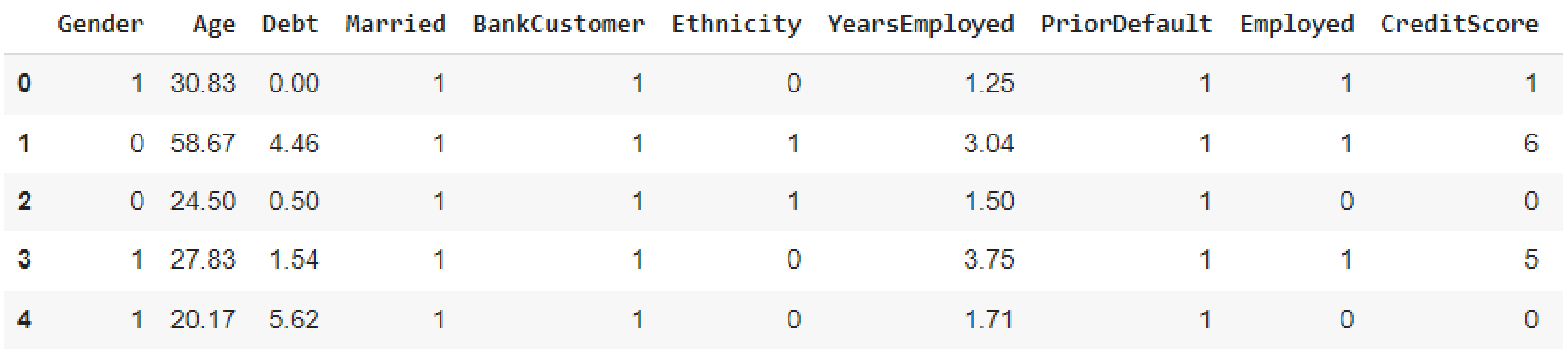
- Employed-Whether employed or unemployed.
- Credit Score- Shows the credit score of the applicant.
- Drivers License-Checks if applicant possesses a drivers license.
- Citizen-Checks if the applicant is a citizen of the country.
- Zip Code-Zip code of the applicant.
- Income- Income of the applicant.
- Approved- Shows whether the loan is approved or not.

Common in all methods/calculations



```
#authors: Adarsh, Arun, Aathithya
#objective: To find optimal k value
#input: Dataset
#output: Accuracy
import pandas as pd #data analysis toolkit
import matplotlib.pyplot as plt # for plotting graphs
import numpy as np # for high level computations
%matplotlib inline

from sklearn.preprocessing import StandardScaler # standardization of values

from sklearn.preprocessing import MinMaxScaler # Normalization of values
from sklearn.model_selection import train_test_split # to split data

from sklearn.neighbors import KNeighborsClassifier #KNN classifier
from sklearn.model_selection import StratifiedKFold
```

```
data_set=data_set.drop(['Industry'], axis= 1) # dropping of columns as mentioned
data_set=data_set.fillna (data_set.mean()) # mean for missing data

data_set = np.round(data_set, decimals=2) # rouding all values in dataset to 2 decimal places
data_set.head() # first 5 values in datase
```



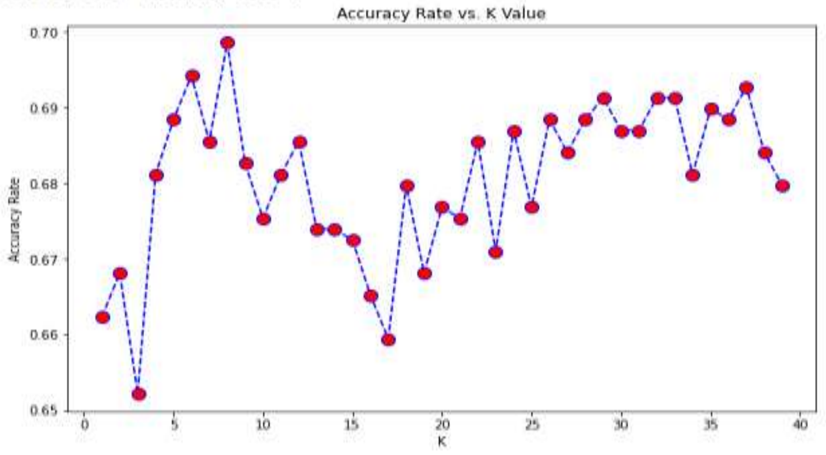
	Gender	Age	Debt	Married	BankCustomer	Ethnicity	YearsEmployed	PriorDefault	Employed	CreditScore
0	1	30.83	0.00	1	1	0	1.25	1	1	1
1	0	58.67	4.46	1	1	1	3.04	1	1	6
2	0	24.50	0.50	1	1	1	1.50	1	0	0
3	1	27.83	1.54	1	1	0	3.75	1	1	5
4	1	20.17	5.62	1	1	0	1.71	1	0	0

Knn classifier

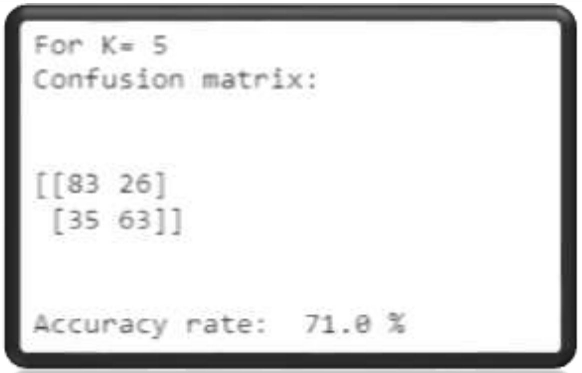
- Distance metric used for computation is Minkowski distance (default_metric)
- Splitting of dataset into testing and training; one_train, one_test, two_train, two_test with 70% for training and 30% for testing.
- Cross validation: Re-sampling procedure used to evaluate a model
  - cv set to 5 (generally considered 5 or 10 based on data)
  - In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration, 2nd fold is used as the testing set while the rest serve as the training set. This process is repeated until each fold of the 5 folds have been used as the testing set.
- Extra code after running common code is as follows.
- Other information is mentioned in comments of code for better understanding.

```python
dset_modified = data_set.drop('Approved', axis=1) # dataset without class feature
data_set_feat = pd.DataFrame(dset_modified, columns=data_set.columns[:-1]) #dataset without class feature
data_set_feat = np.round(data_set_feat, decimals=2) # rouding all values to 2 decimal places

#test_train split with test size 30% and train size -70%
one_train, one_test, two_train, two_test = train_test_split(data_set_feat, data_set['Approved'],test_size=0.30)

#Computation of accuracy rates for various neighbour values
Accurate_rates = []
for i in range(1,40):

  k_nearest_neighbour = KNeighborsClassifier(n_neighbors=i)
  final_score = cross_val_score(k_nearest_neighbour, data_set_feat, data_set['Approved'], cv=5)
  Accurate_rates.append(final_score.mean())

plt.figure(figsize=(10,6))
plt.plot(range(1,40), Accurate_rates, color='blue', linestyle='dashed', marker='o', markerfacecolor='red', markersize=10)
plt.title('Accuracy Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Accuracy Rate')
```

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



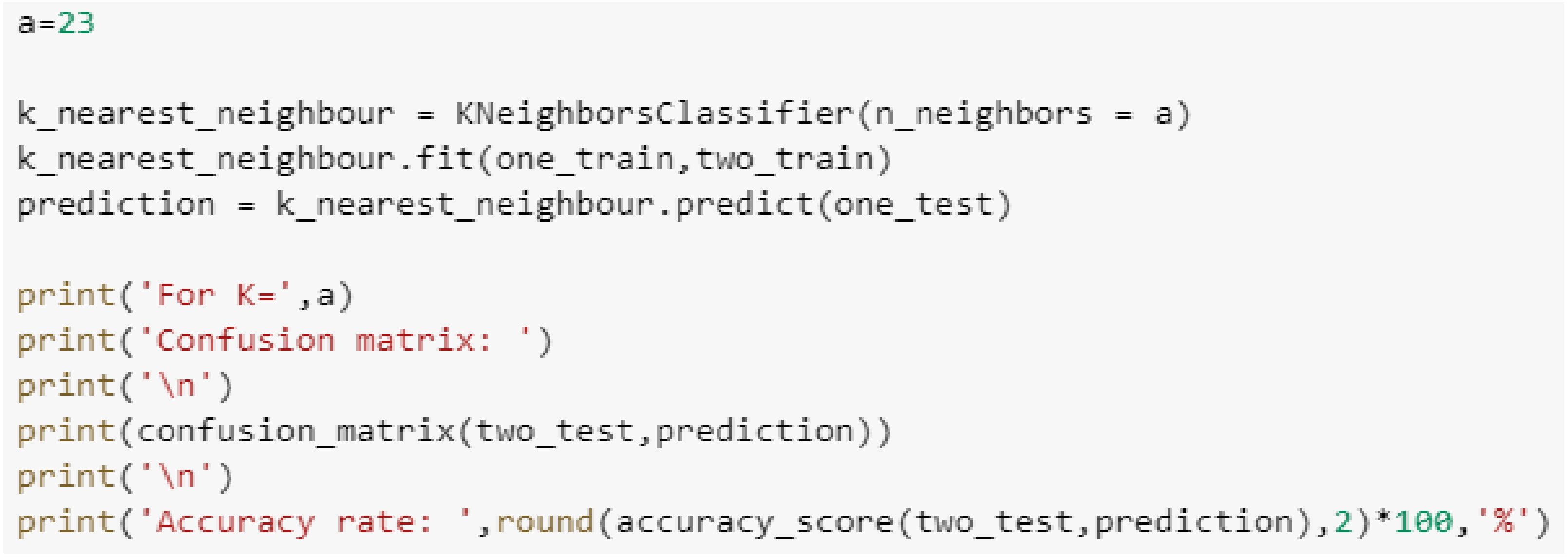
```
max_index  = Accurate_rates.index(max(Accurate_rates))

k_nearest_neighbour = KNeighborsClassifier(n_neighbors = max_index)
k_nearest_neighbour.fit(one_train,two_train)
prediction = k_nearest_neighbour.predict(one_test)

print('For K=',max_index)
print('Confusion matrix: ')
print('\n')
print(confusion_matrix(two_test,prediction))
print('\n')
print('Accuracy rate: ',round(accuracy_score(two_test,prediction),2)*100,'%')
```



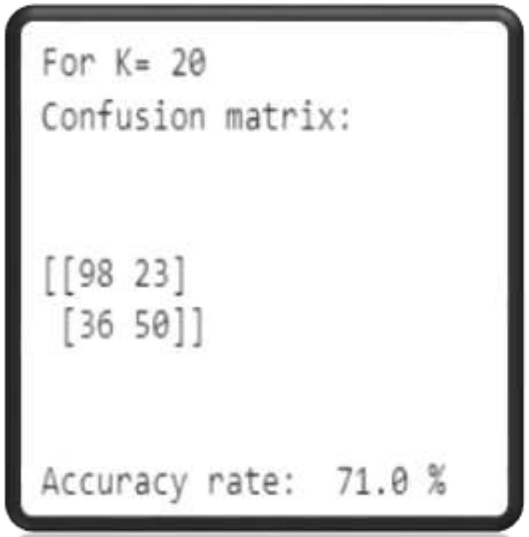
```
For K= 5
Confusion matrix:


[[83 26]
 [35 63]]


Accuracy rate:   71.0 %
```

- Therefore, for the given data the maximum accuracy using K-nearest neighbors method was found as 71% (BEST CASE) for k= 7 neighbors.
- The corresponding confusion matrix has been printed.

- For a different K value :
- Same accuracy rate has been obtained as in graph

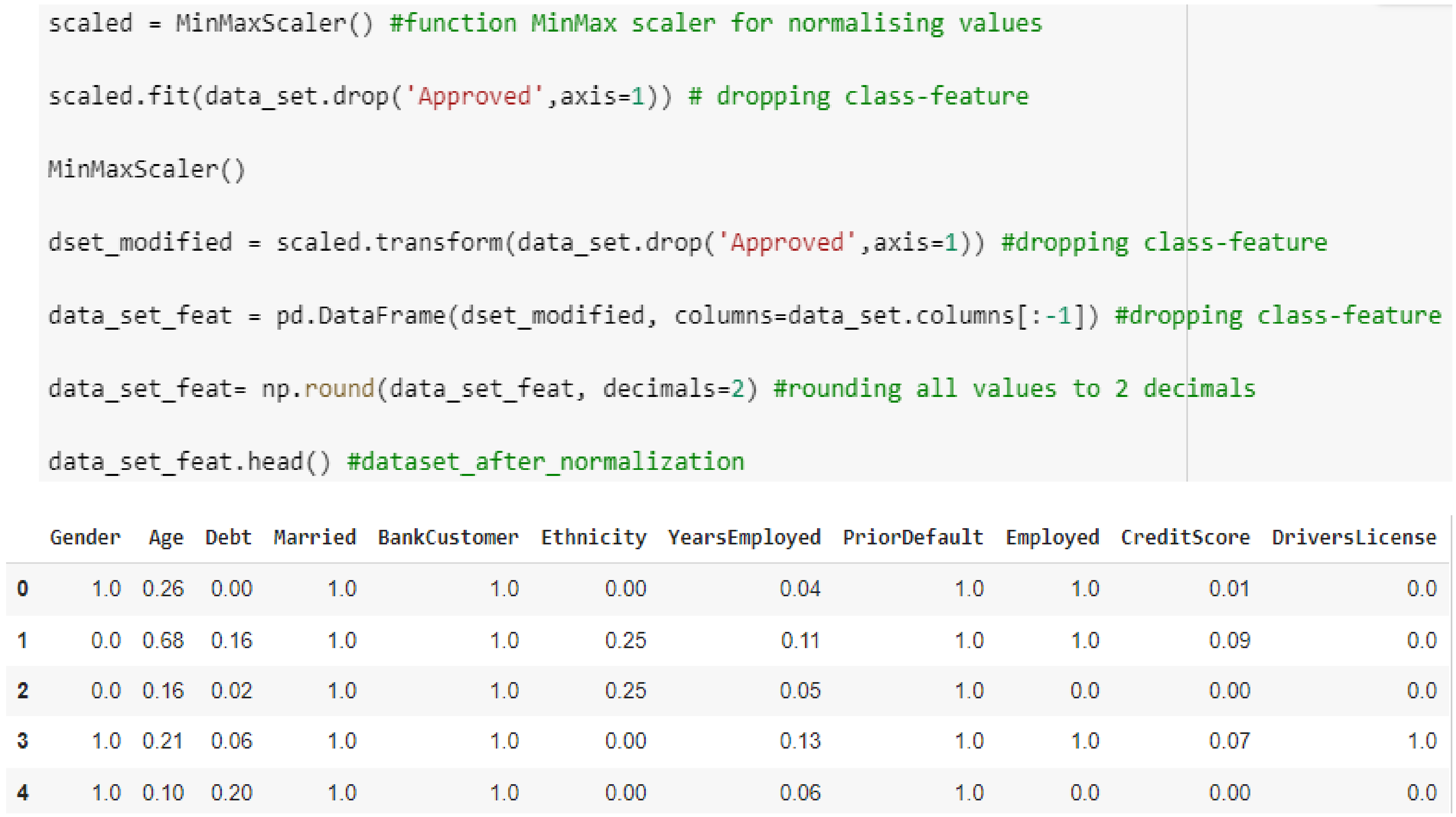


```
a=23

k_nearest_neighbour = KNeighborsClassifier(n_neighbors = a)
k_nearest_neighbour.fit(one_train,two_train)
prediction = k_nearest_neighbour.predict(one_test)

print('For K=',a)
print('Confusion matrix: ')
print('\n')
print(confusion_matrix(two_test,prediction))
print('\n')
print('Accuracy rate: ',round(accuracy_score(two_test,prediction),2)*100,'%')
```



```
For K= 20
Confusion matrix:


[[98 23]
 [36 50]]


Accuracy rate:  71.0 %
```

Normalization

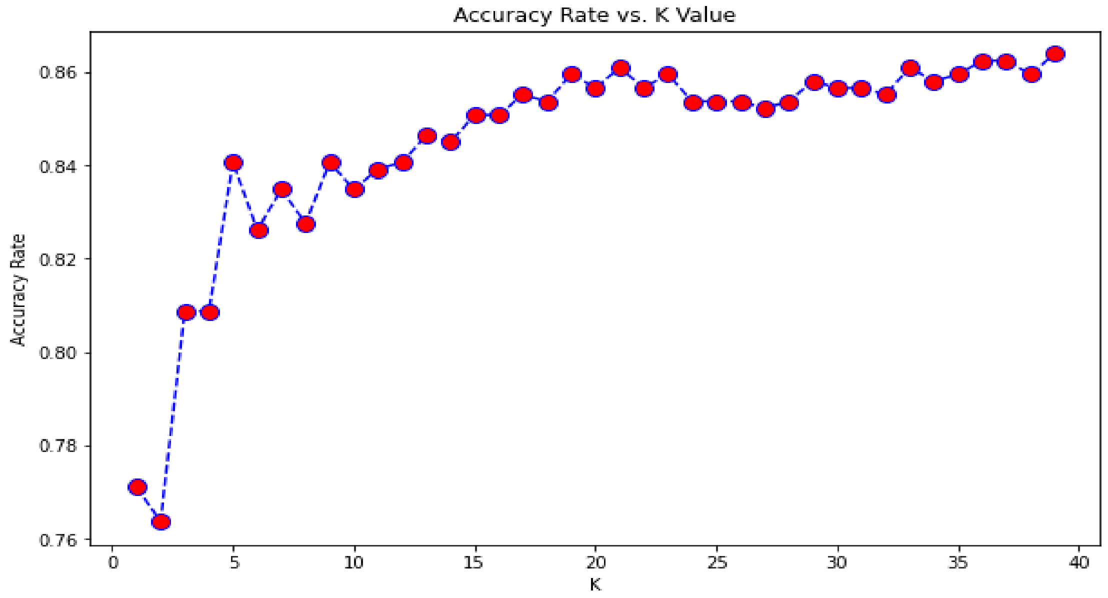


```
scaled = MinMaxScaler() #function MinMax scaler for normalising values

scaled.fit(data_set.drop('Approved',axis=1)) # dropping class-feature

MinMaxScaler()

dset_modified = scaled.transform(data_set.drop('Approved',axis=1)) #dropping class-feature

data_set_feat = pd.DataFrame(dset_modified, columns=data_set.columns[:-1]) #dropping class-feature

data_set_feat= np.round(data_set_feat, decimals=2) #rounding all values to 2 decimals

data_set_feat.head() #dataset_after_normalization
```

	Gender	Age	Debt	Married	BankCustomer	Ethnicity	YearsEmployed	PriorDefault	Employed	CreditScore	DriversLicense
0	1.0	0.26	0.00	1.0	1.0	0.00	0.04	1.0	1.0	0.01	0.0
1	0.0	0.68	0.16	1.0	1.0	0.25	0.11	1.0	1.0	0.09	0.0
2	0.0	0.16	0.02	1.0	1.0	0.25	0.05	1.0	0.0	0.00	0.0
3	1.0	0.21	0.06	1.0	1.0	0.00	0.13	1.0	1.0	0.07	1.0
4	1.0	0.10	0.20	1.0	1.0	0.00	0.06	1.0	0.0	0.00	0.0

```
#test_train split with test size 30% and train size -70%
one_train, one_test, two_train, two_test = train_test_split(data_set_feat, data_set['Approved'],test_size=0.30)

#Computation of accuracy rates for various neighbour values
Accurate_rates = []

for i in range(1,40):

  k_nearest_neighbour = KNeighborsClassifier(n_neighbors=i)
  final_score = cross_val_score(k_nearest_neighbour, data_set_feat, data_set['Approved'], cv=5)
  Accurate_rates.append(final_score.mean())


plt.figure(figsize=(10,6))

plt.plot(range(1,40), Accurate_rates, color='blue', linestyle='dashed', marker='o', markerfacecolor='red', markersize=10)
plt.title('Accuracy Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Accuracy Rate')
```

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



Inference

- Minkowski Distance uses both Manhattan and Euclidean distance in a generalized form for calculation.
- For various values of K the accuracy rates changes and through plotting all the values, the best case was found.
- The accuracy rates for other K values can b inferred from graph
- Confusion matrix shows the predicted vs actual values.

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

Miscellaneous

- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- CV value should be equal to or less than the number of features present in our dataset. If the cv value exceeds the number of features it would show an error. Hence cv value from 5-10