ML-MINOR-MAY

Problem:

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

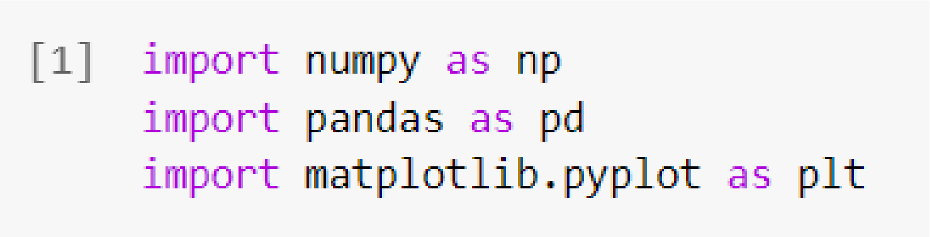
IDE: Google Collab

Packages: NumPy, Pandas, matplotlib.pyplot,sklearn

Approach:

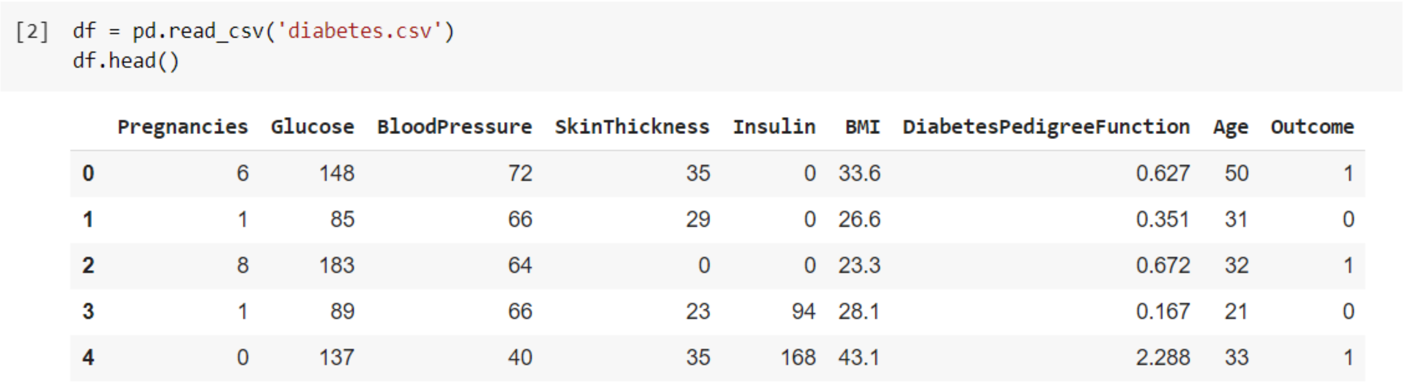
KNN (K-Nearest Neighbour) Classifier algorithm was used in this solution.
K-NN algorithm assumes the similarity between the new data and available data and puts the new case into the category that is most similar to the available categories.

Code:



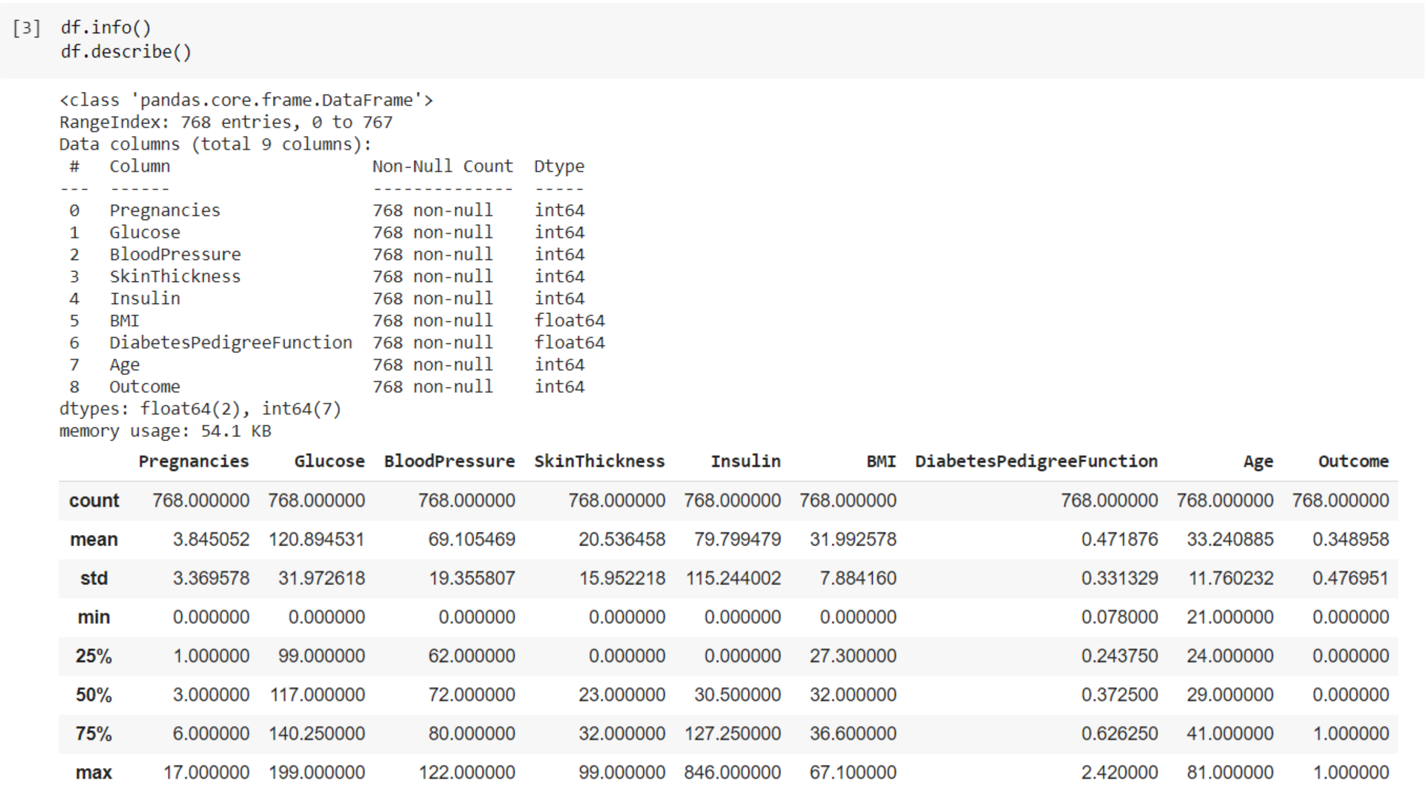
```
[1]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
```

Importing the Libraries



```
[2]  df = pd.read_csv('diabetes.csv')
     df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

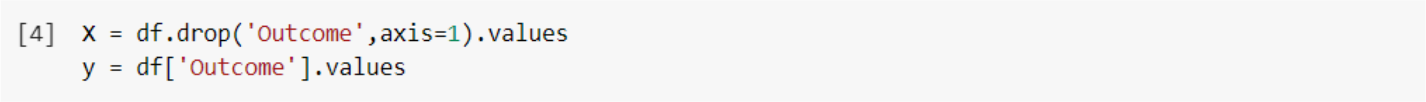


```
[3]  df.info()
     df.describe()
```

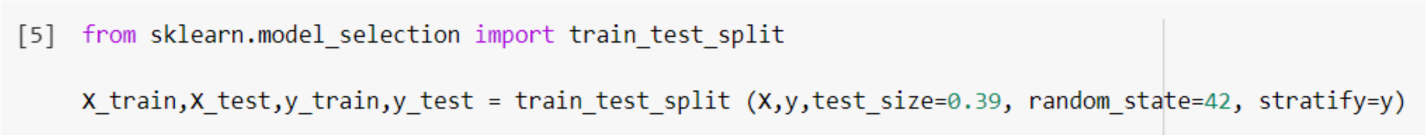
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

The 9th column represents the label.



```
[4]  X = df.drop('Outcome',axis=1).values
     y = df['Outcome'].values
```

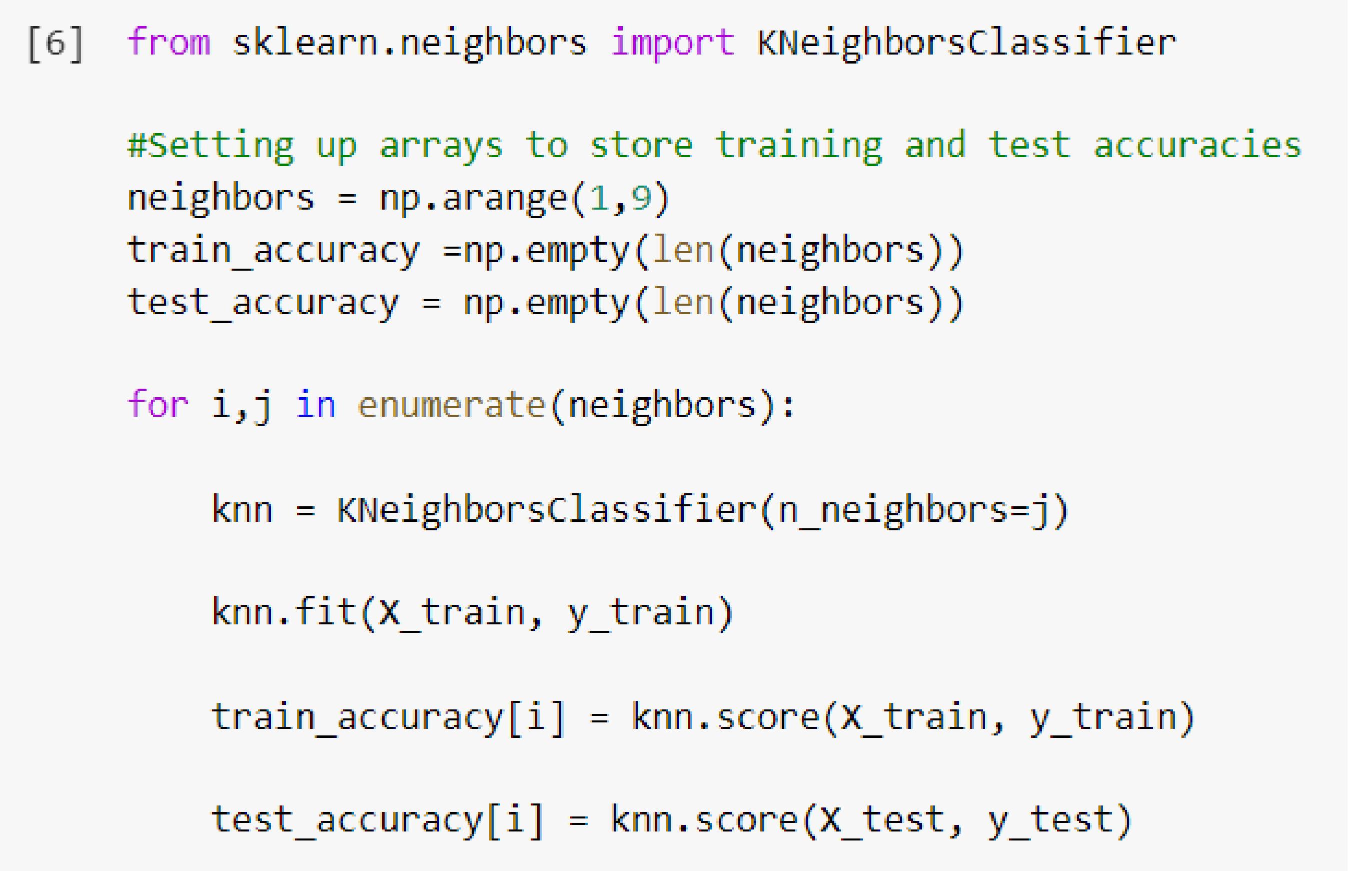


```
[5]  from sklearn.model_selection import train_test_split

     X_train,X_test,y_train,y_test = train_test_split (X,y,test_size=0.39, random_state=42, stratify=y)
```

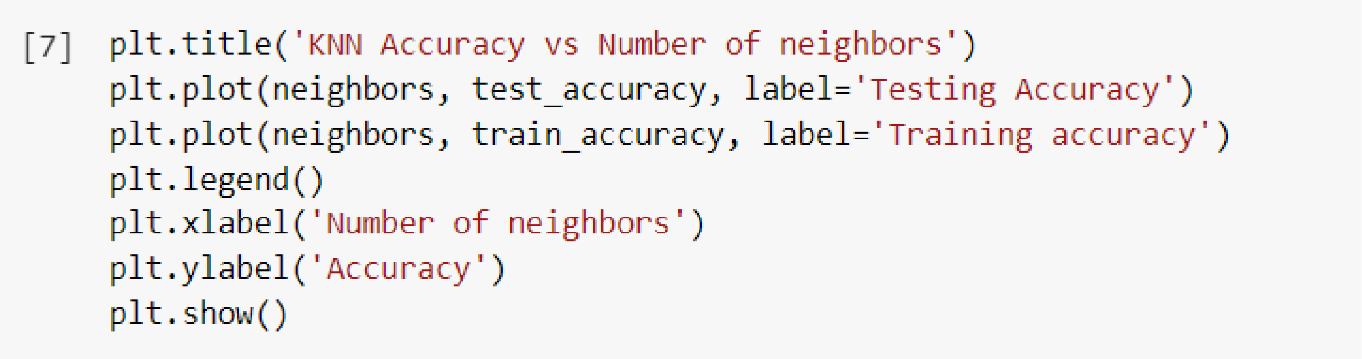
Splitting the data randomly into training and test set.

We will fit a classifier on the training set and make predictions on the test set. Then we will compare the predictions with the known labels.

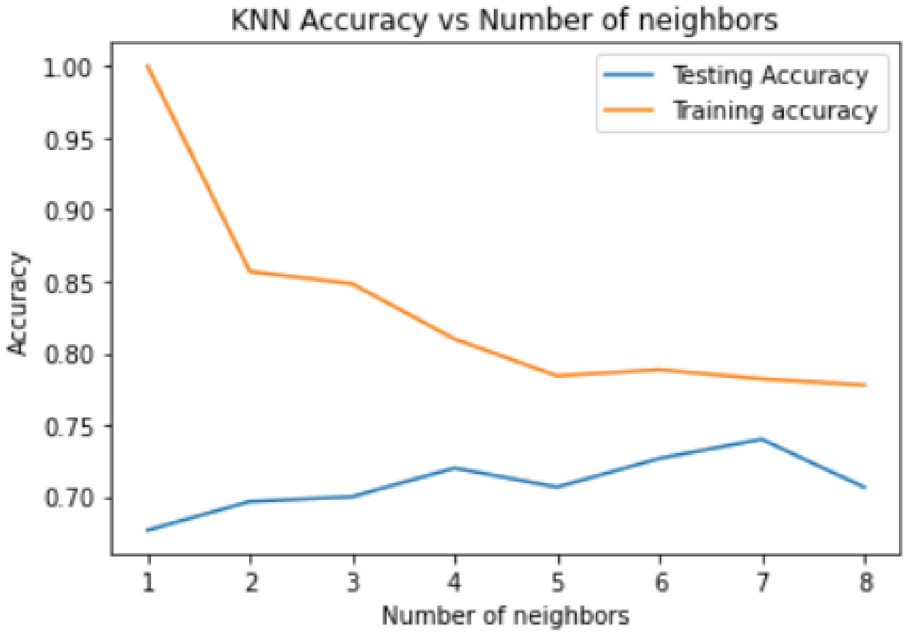
A test set of size of 39% of the dataset has been created.



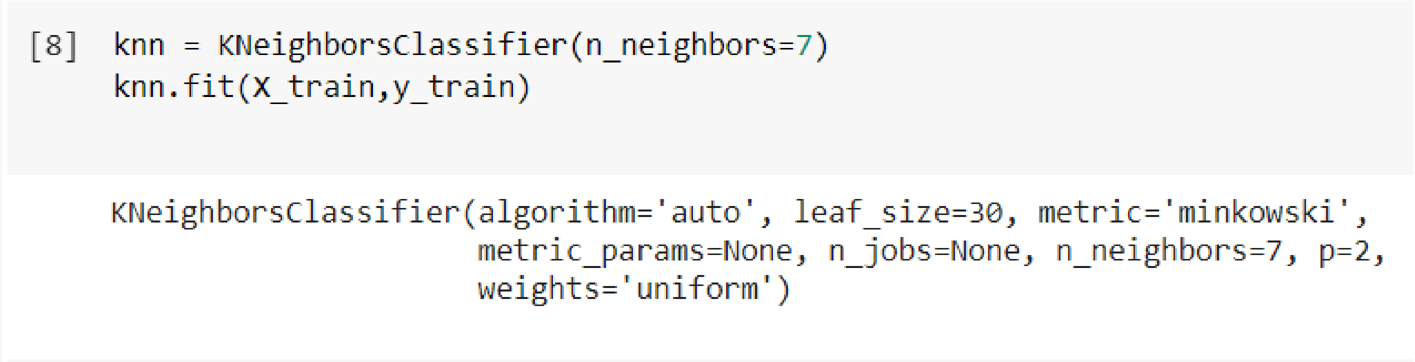
```
[6]  from sklearn.neighbors import KNeighborsClassifier

     #Setting up arrays to store training and test accuracies
     neighbors = np.arange(1,9)
     train_accuracy =np.empty(len(neighbors))
     test_accuracy = np.empty(len(neighbors))

     for i,j in enumerate(neighbors):

         knn = KNeighborsClassifier(n_neighbors=j)

         knn.fit(X_train, y_train)

         train_accuracy[i] = knn.score(X_train, y_train)

         test_accuracy[i] = knn.score(X_test, y_test)
```



```
[7]  plt.title('KNN Accuracy vs Number of neighbors')
     plt.plot(neighbors, test_accuracy, label='Testing Accuracy')
     plt.plot(neighbors, train_accuracy, label='Training accuracy')
     plt.legend()
     plt.xlabel('Number of neighbors')
     plt.ylabel('Accuracy')
     plt.show()
```

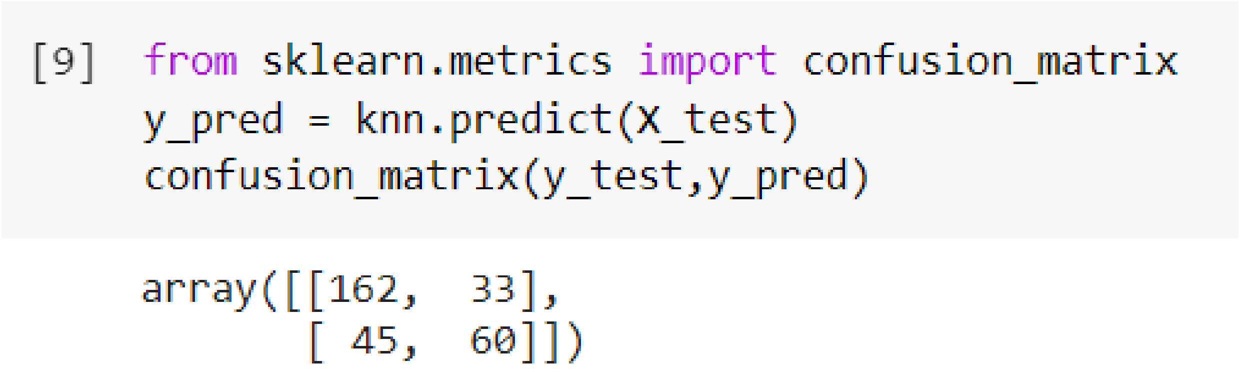


The testing accuracy is greatest for 7 neighbours. Thus, we will go with number of neighbours as 7.



```
[8]  knn = KNeighborsClassifier(n_neighbors=7)
     knn.fit(X_train,y_train)

     KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                          metric_params=None, n_jobs=None, n_neighbors=7, p=2,
                          weights='uniform')
```

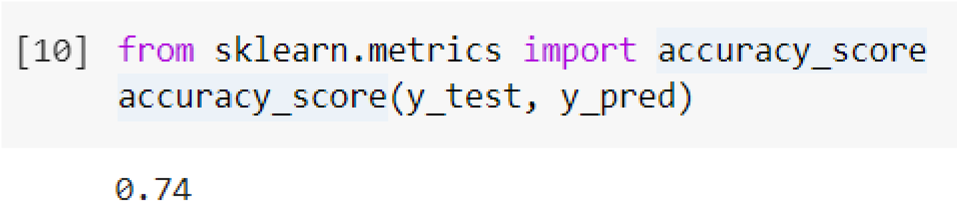
Setting up a knn classifier with 7 neighbours



```
[9]  from sklearn.metrics import confusion_matrix
     y_pred = knn.predict(X_test)
     confusion_matrix(y_test,y_pred)

     array([[162,  33],
            [ 45,  60]])
```

From the above Confusion Matrix,

- True negative = 162
- False positive = 33
- False negative = 45
- True positive = 60



```
[10]  from sklearn.metrics import accuracy_score
      accuracy_score(y_test, y_pred)

      0.74
```

The accuracy comes out to be **74%**.

(~Dev Agrawal)