Table of Contents

Detection of Parkinson’s Disease

Introduction

Parkinson’s disease is an age-related degenerative brain condition, meaning it causes parts of your brain to deteriorate. It’s known for causing slowed movements, tremors, balance problems and more. Most cases happen for unknown reasons, but some are inherited. The condition isn’t curable, but there are many different treatment options.

Aim and Objective of the Project

For this project, We have decided to focus on the field of medicine and classify whether or not a patient has Parkinson’s disease based on their vocalization data. For more context, Parkinson’s is a progressive disease that causes the degeneration of the brain, leading to both motor and cognitive problems. It is thus reasonable to assume a correlation between a patient’s ability to speak and their progression into Parkinson’s as these capabilities regress. The data set we have worked with was obtained through a 2008 study by the journal, IEEE Transactions on Biomedical Engineering, of how various parameters of voice frequency can help classify if a patient is suffering from Parkinson’s. By performing a classification on this data, we hope to prove that vocalization tests are indeed a well suited way to diagnose a patient for this disease.

Problem Statement

This project attempts to prove that vocalization data from a patient can help diagnose whether or not they suffer from Parkinson’s. As such, it is initially assumed that there is a relationship between the two. I will attempt to run various Data mining algorithms on the data in hopes to reach a high predictability rate that is matched with a reasonable runtime.

Data Set

https://archive.ics.uci.edu/ml/datasets/parkinsons

Title: Parkinson's Disease Data Set

Data Set Information:

This dataset is composed of a range of biomedical voice measurements from 31 people, 23 with Parkinson's disease (PD). Each column in the table is a particular voice measure, and each row corresponds to one of 195 voice recordings from these individuals ("name" column). The main aim of the data is to discriminate healthy people from those with PD, according to the "status" column which is set to 0 for healthy and 1 for PD. The data is in ASCII CSV format. The rows of the CSV file contain an instance corresponding to one voice recording. There are around six recordings per patient, the name of the patient is identified in the first column.

Abstract: Oxford Parkinson's Disease Detection Dataset

- Data Set Characteristics: Multivariate
- Number of Instances: 195
- Area: Life
- Attribute Characteristics: Real
- Number of Attributes: 23
- Date Donated: 2008-06-26
- Associated Tasks: Classification
- Missing Values? N/A

Attribute Information:

- Matrix column entries (attributes):
- name - ASCII subject name and recording number
- MDVP:Fo(Hz) - Average vocal fundamental frequency
- MDVP:Fhi(Hz) - Maximum vocal fundamental frequency
- MDVP:Flo(Hz) - Minimum vocal fundamental frequency
- MDVP:Jitter(%),MDVP:Jitter(Abs),MDVP:RAP,MDVP:PPQ,Jitter:DDP - Several
- measures of variation in fundamental frequency
- MDVP:Shimmer,MDVP:Shimmer(dB),Shimmer:APQ3,Shimmer:APQ5,MDVP:APQ,Shimmer:DDA - Several measures of variation in amplitude
- NHR,HNR - Two measures of ratio of noise to tonal components in the voice

- status - Health status of the subject (one) - Parkinson's, (zero) - healthy
- RPDE,D2 - Two nonlinear dynamical complexity measures
- DFA - Signal fractal scaling exponent
- spread1,spread2,PPE - Three nonlinear measures of fundamental frequency variation

Implementation

Code

Importing Library

```
# Importing Libraries

import requests

import pandas as pd

from imblearn.over_sampling import SMOTE

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings("ignore")

from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import train_test_split

from sklearn.model_selection import GridSearchCV

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics

from sklearn.metrics import classification_report

from sklearn.metrics import confusion_matrix

from sklearn.metrics import accuracy_score
```

from sklearn.model_selection import cross_val_score

from sklearn.datasets import make_classification

from sklearn.metrics import plot_confusion_matrix

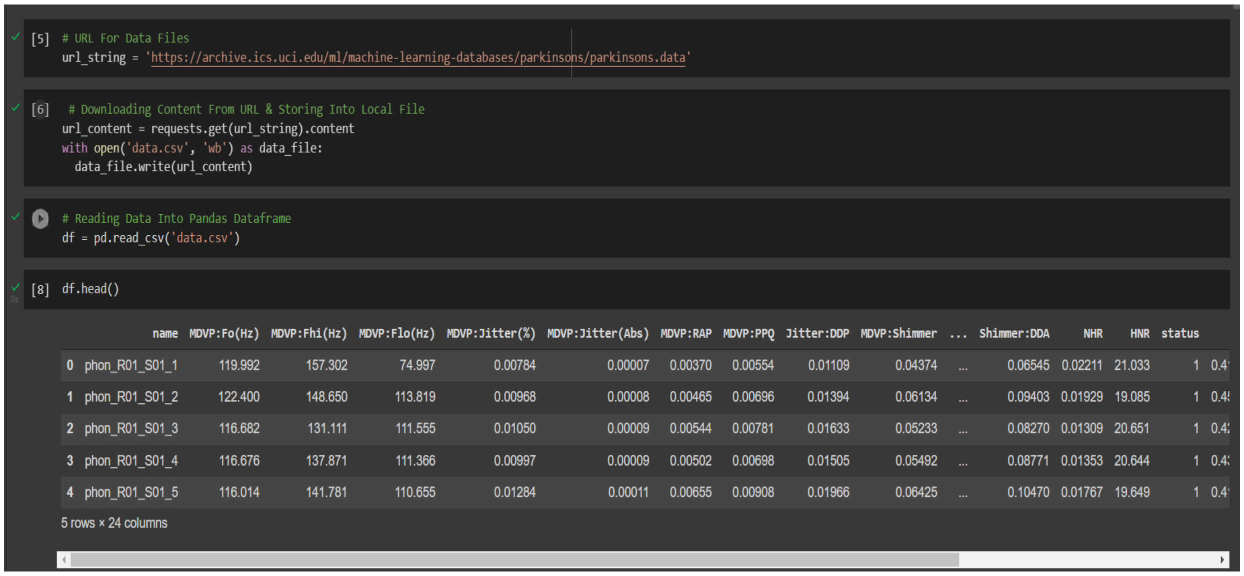
import joblib

from IPython.display import display

Dataset importing from the URL

'https://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/parkinsons.data'

Output



```
[5] # URL For Data Files
    url_string = 'https://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/parkinsons.data'
```

```
[6]  # Downloading Content From URL & Storing Into Local File
    url_content = requests.get(url_string).content
    with open('data.csv', 'wb') as data_file:
      data_file.write(url_content)
```

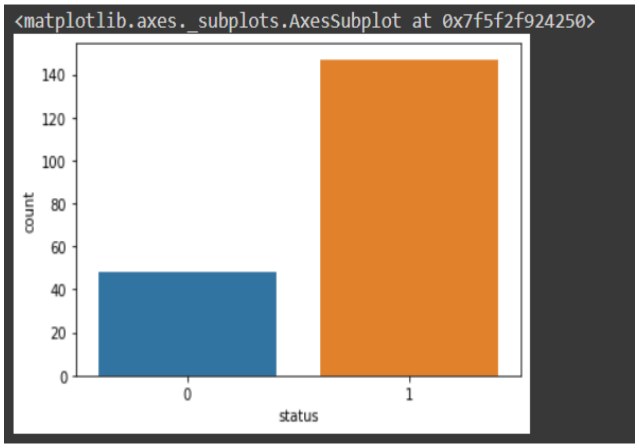
```
    # Reading Data Into Pandas Dataframe
    df = pd.read_csv('data.csv')
```

```
[8] df.head()
```

```
           name  MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  ...  Shimmer:DDA      NHR     HNR  status
0  phon_R01_S01_1      119.992       157.302        74.997         0.00784           0.00007   0.00370   0.00554     0.01109       0.04374  ...      0.06545  0.02211  21.033       1  0.4
1  phon_R01_S01_2      122.400       148.650       113.819         0.00968           0.00008   0.00465   0.00696     0.01394       0.06134  ...      0.09403  0.01929  19.085       1  0.4
2  phon_R01_S01_3      116.682       131.111       111.555         0.01050           0.00009   0.00544   0.00781     0.01633       0.05233  ...      0.08270  0.01309  20.651       1  0.4
3  phon_R01_S01_4      116.676       137.871       111.366         0.00997           0.00009   0.00502   0.00698     0.01505       0.05492  ...      0.08771  0.01353  20.644       1  0.4
4  phon_R01_S01_5      116.014       141.781       110.655         0.01284           0.00011   0.00655   0.00908     0.01966       0.06425  ...      0.10470  0.01767  19.649       1  0.4

5 rows × 24 columns
```

Balance of Data

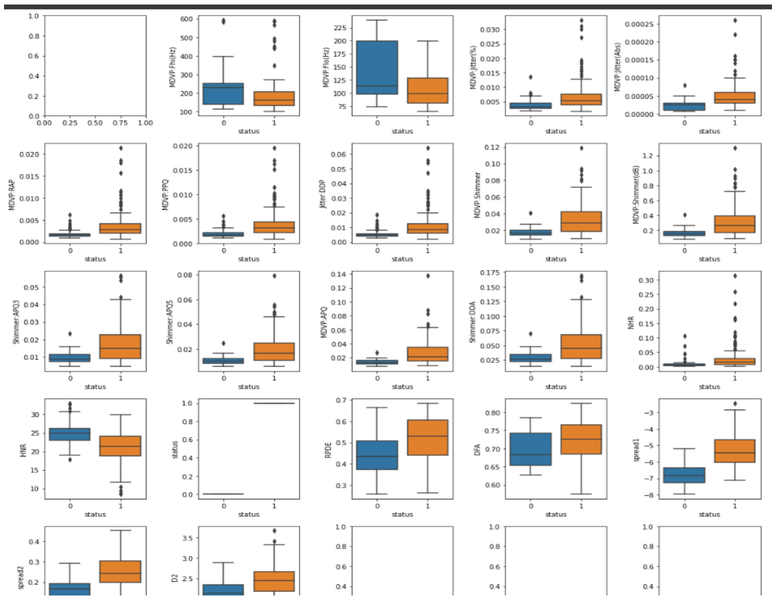
```
sns.countplot(x='status',data=df)
```



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5f2f924250>
```

Box Plot

```
fig,axes=plt.subplots(5,5,figsize=(15,15))

axes=axes.flatten()

for i in range(1,len(df.columns)-1):

    sns.boxplot(x='status',y=df.iloc[:,i],data=df,orient='v',ax=axes[i])

plt.tight_layout()

plt.show()
```



# define X_features , Y_labels

X_features = scaler.fit_transform(X)

Y_labels = y

Splitting the dataset into training and testing sets into 80 - 20

from sklearn.model_selection import train_test_split

X_train , X_test , y_train , y_test = train_test_split(X_features, Y_labels , test_size=0.20, random_state=20)

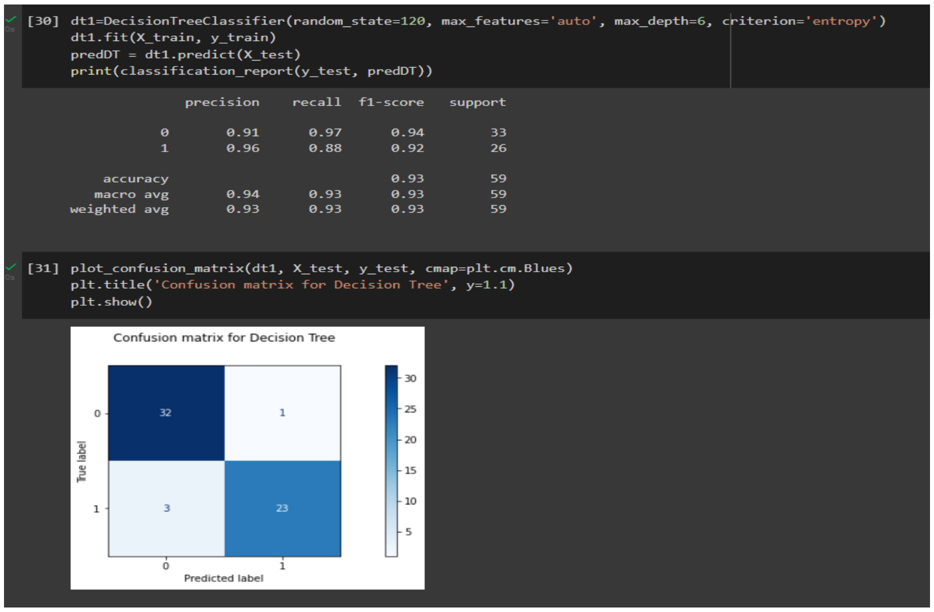
Decision Tree Classifier

clf = DecisionTreeClassifier()

clf.fit(X_train, y_train)

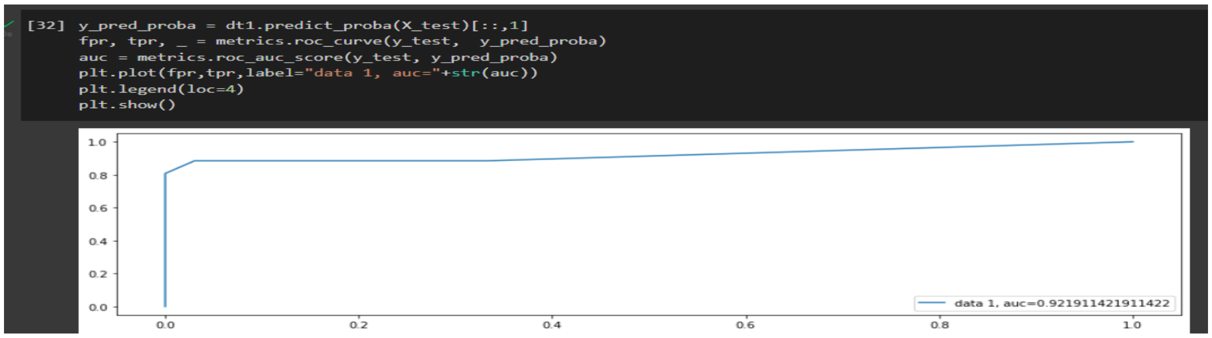
predDT = clf.predict(X_test)

print(classification_report(y_test, predDT))



```
[30] dt1=DecisionTreeClassifier(random_state=120, max_features='auto', max_depth=6, criterion='entropy')
     dt1.fit(X_train, y_train)
     predDT = dt1.predict(X_test)
     print(classification_report(y_test, predDT))

                   precision    recall  f1-score   support

                0       0.91      0.97      0.94        33
                1       0.96      0.88      0.92        26

         accuracy                           0.93        59
        macro avg       0.94      0.93      0.93        59
     weighted avg       0.93      0.93      0.93        59

[31] plot_confusion_matrix(dt1, X_test, y_test, cmap=plt.cm.Blues)
     plt.title('Confusion matrix for Decision Tree', y=1.1)
     plt.show()
```

Accuracy



```
[32] y_pred_proba = dt1.predict_proba(X_test)[::,1]
     fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
     auc = metrics.roc_auc_score(y_test, y_pred_proba)
     plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
     plt.legend(loc=4)
     plt.show()
```

Logistic Regression

logmodel = LogisticRegression()

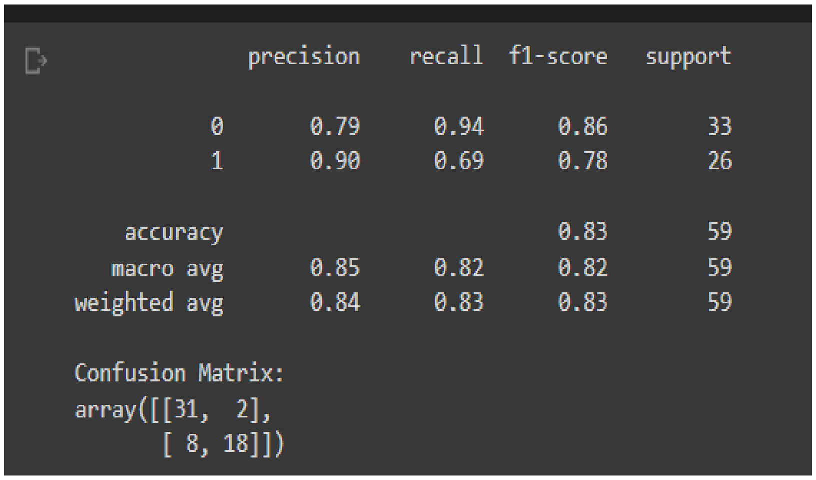
logmodel.fit(X_train, y_train)

predlog = logmodel.predict(X_test)

print(classification_report(y_test, predlog))

print("Confusion Matrix:")

confusion_matrix(y_test, predlog)

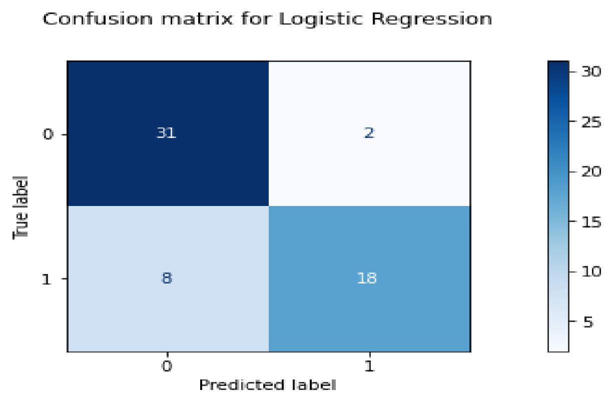


```
              precision    recall  f1-score   support

           0       0.79      0.94      0.86        33
           1       0.90      0.69      0.78        26

    accuracy                           0.83        59
   macro avg       0.85      0.82      0.82        59
weighted avg       0.84      0.83      0.83        59

Confusion Matrix:
array([[31,  2],
       [ 8, 18]])
```

plot_confusion_matrix(logmodel, X_test, y_test, cmap=plt.cm.Blues)

plt.title('Confusion matrix for Logistic Regression', y=1.1)

plt.show()

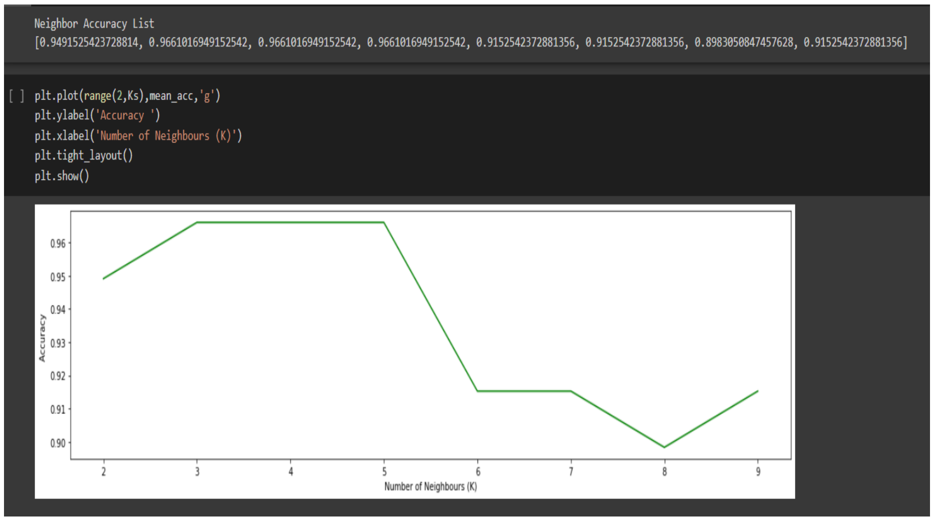


K-NN Classifier

```
import numpy as np

Ks = 10

mean_acc = []

ConfustionMx = [];

for n in range(2,Ks):
```

Train Model and Predict

```
    neigh = KNeighborsClassifier(n_neighbors = n).fit(X_train,y_train)

    yhat=neigh.predict(X_test)

    mean_acc.append(metrics.accuracy_score(y_test, yhat))

print('Neighbor Accuracy List')

print(mean_acc)
```



```
Neighbor Accuracy List
[0.9491525423728814, 0.9661016949152542, 0.9661016949152542, 0.9661016949152542, 0.9152542372881356, 0.9152542372881356, 0.8983050847457628, 0.9152542372881356]
```

```
[ ] plt.plot(range(2,Ks),mean_acc,'g')
    plt.ylabel('Accuracy ')
    plt.xlabel('Number of Neighbours (K)')
    plt.tight_layout()
    plt.show()
```

knn = KNeighborsClassifier(n_neighbors=5)

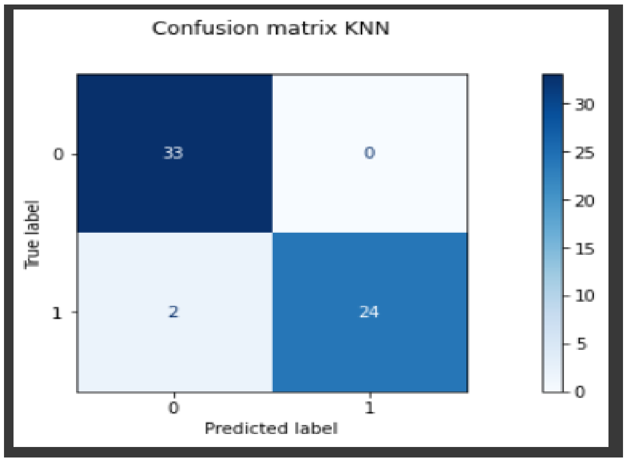
knn.fit(X_train, y_train)

predKNN = knn.predict(X_test)

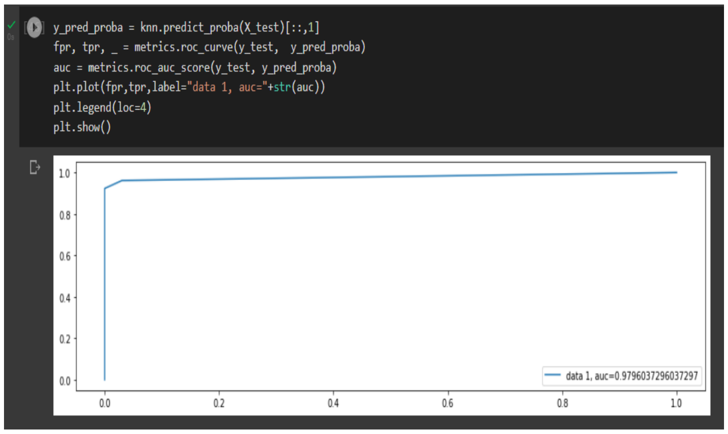
plot_confusion_matrix(knn, X_test, y_test, cmap=plt.cm.Blues)

plt.title('Confusion matrix KNN', y=1.1)

plt.show()



Accuracy



```
y_pred_proba = knn.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```

<u>Prediction of Parkinson’s Disease</u>

input_data = (174.688,240.005,74.287,0.0136,0.00008,0.00624,0.00564,0.01873,0.02308,0.256,0.01268,0.01365,0.01667,0.03804,0.10715,17.883,0.407567,0.655683,-6.787197,0.158453,2.679772,0.131728)

# input data to numpy array (tuple to array)

input_data_as_numpy_array = np.asarray(input_data)

# reshape the numpy array

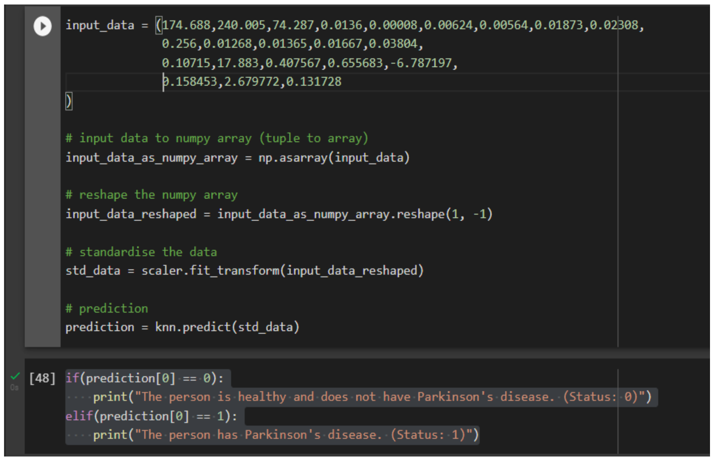
input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)

# standardise the data

std_data = scaler.fit_transform(input_data_reshaped)

# prediction

prediction = knn.predict(std_data)



```
input_data = (174.688,240.005,74.287,0.0136,0.00008,0.00624,0.00564,0.01873,0.02308,
              0.256,0.01268,0.01365,0.01667,0.03804,
              0.10715,17.883,0.407567,0.655683,-6.787197,
              0.158453,2.679772,0.131728
)

# input data to numpy array (tuple to array)
input_data_as_numpy_array = np.asarray(input_data)

# reshape the numpy array
input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)

# standardise the data
std_data = scaler.fit_transform(input_data_reshaped)

# prediction
prediction = knn.predict(std_data)
```

```
[48] if(prediction[0] == 0):
       print("The person is healthy and does not have Parkinson's disease. (Status: 0)")
     elif(prediction[0] == 1):
       print("The person has Parkinson's disease. (Status: 1)")
```

<u>Comparison of the Accuracy</u>

from sklearn.metrics import precision_score,recall_score ,accuracy_score, f1_score, r2_score, log_loss

chart = {

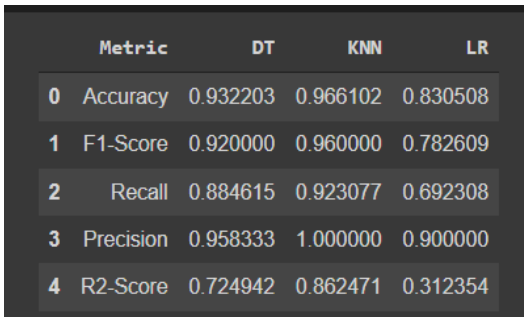
'Metric':["Accuracy", "F1-Score", "Recall", "Precision", "R2-Score"],

'DT':[accuracy_score(y_test, predDT), f1_score(y_test, predDT), recall_score(y_test, predDT), precision_score(y_test, predDT), r2_score(y_test, predDT)],

'KNN':[accuracy_score(y_test, predKNN), f1_score(y_test, predKNN), recall_score(y_test, predKNN), precision_score(y_test, predKNN), r2_score(y_test, predKNN)],

'LR':[accuracy_score(y_test, predlog), f1_score(y_test, predlog), recall_score(y_test, predlog), precision_score(y_test, predlog), r2_score(y_test, predlog)],}

chart = pd.DataFrame(chart)



	Metric	DT	KNN	LR
0	Accuracy	0.932203	0.966102	0.830508
1	F1-Score	0.920000	0.960000	0.782609
2	Recall	0.884615	0.923077	0.692308
3	Precision	0.958333	1.000000	0.900000
4	R2-Score	0.724942	0.862471	0.312354

Analysis -: We have tried to import the dataset from the url mentioned above in the report and then found out the accuracy and prediction of the classifications performed for the status of a patient whether he/she has a parkinson's disease or not if yes then the status shows 1 else 0.The balance data found for the status is approximately 50 for 0 and 145 for yes. In addition we performed Correlation analysis which can reveal meaningful relationships between different metrics or groups of metrics. Information about those connections can provide new insights and reveal interdependencies.For evaluating performances we did train and test on the dataset.

Unhealthy status (The patient has the Parkinson's Disease)

Healthy(The patient doesn't has Parkinson's disease)

We performed Decision tree classifier algorithm of data mining for accuracy of detecting the disease we found that the model gave the accuracy of approximately 92% with confusion matrix details as following -:

- When the true label and predicted label was showing unhealthy status the result which it gave was 32.This shows that our prediction was correct for the unhealthy status
- When the true label was unhealthy status and the predicted label was showing healthy status the result which it gave was 1.It means that here the patient was unhealthy but the model predicted incorrectly that the patient was healthy.
- When the true label and predicted label was showing healthy status the result which it gave was 23 which implies that it correctly predicted for 23 and incorrectly predicted when predicted label was unhealthy and true label was healthy which predicted 3 .

We performed K-NN algorithm of data mining for accuracy of detecting the disease we found that the model gave the accuracy of approximately 97% with confusion matrix as follows -:

- When the true label and predicted label was showing unhealthy status the result which it gave was 33.This shows that our prediction was correct for the unhealthy status
- When the true label was unhealthy status and the predicted label was showing healthy status the result which it gave was 0.It means that here the patient was unhealthy but the model predicted incorrectly that the patient was healthy.
- When the true label and predicted label was showing healthy status the result which it gave was 24 which implies that it correctly predicted for 24 and incorrectly predicted when predicted label was unhealthy and true label was healthy which predicted 2.

We performed Logistic Regression algorithm of data mining for accuracy of detecting the disease we found that the model gave the accuracy of approximately 83% with confusion matrix as follows -:

- For the true label and predicted label unhealthy status the result which it gave was 31.This shows that our prediction was correct for the unhealthy status.

- When the true label was unhealthy status and the predicted label was showing healthy status the result which it gave was 2. It means that here the patient was unhealthy but the model predicted incorrectly that the patient was healthy.
- When the true label and predicted label was showing healthy status the result which it gave was 18 which implies that it correctly predicted for 24 and incorrectly predicted when predicted label was unhealthy and true label was healthy which predicted 8.

Conclusion - After the analysis of the problem statement and compiling the code we found out the accuracy for each of the models and found out that the model which gave the highest accuracy was KNN with a percentage of 97.After the compilation of the code we can conclude the model is showing the correct status for the prediction of the Parkinson's disease for each patient .