Tequed
Invent Innovate Iterate

Artificial Intelligence And Machine Learning

Parkinson's Disease Detection Python Project

By:

Halaswamy M H

(TLS21A1143)

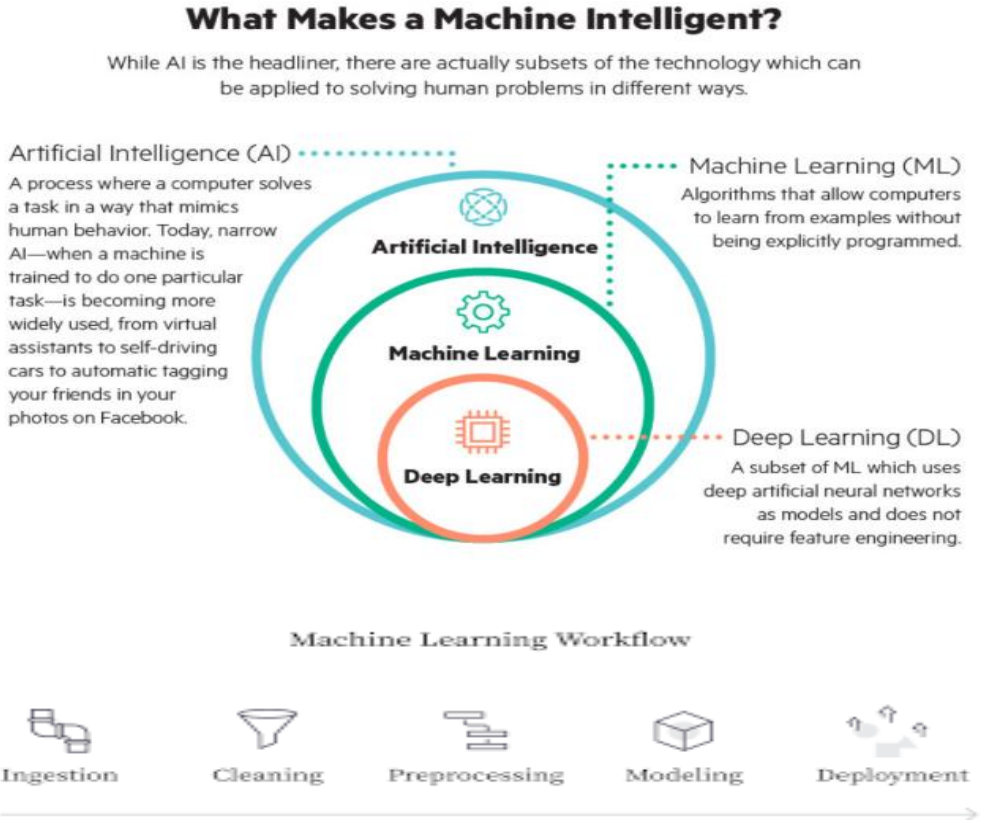
INDEX:

INTRODUCTION:

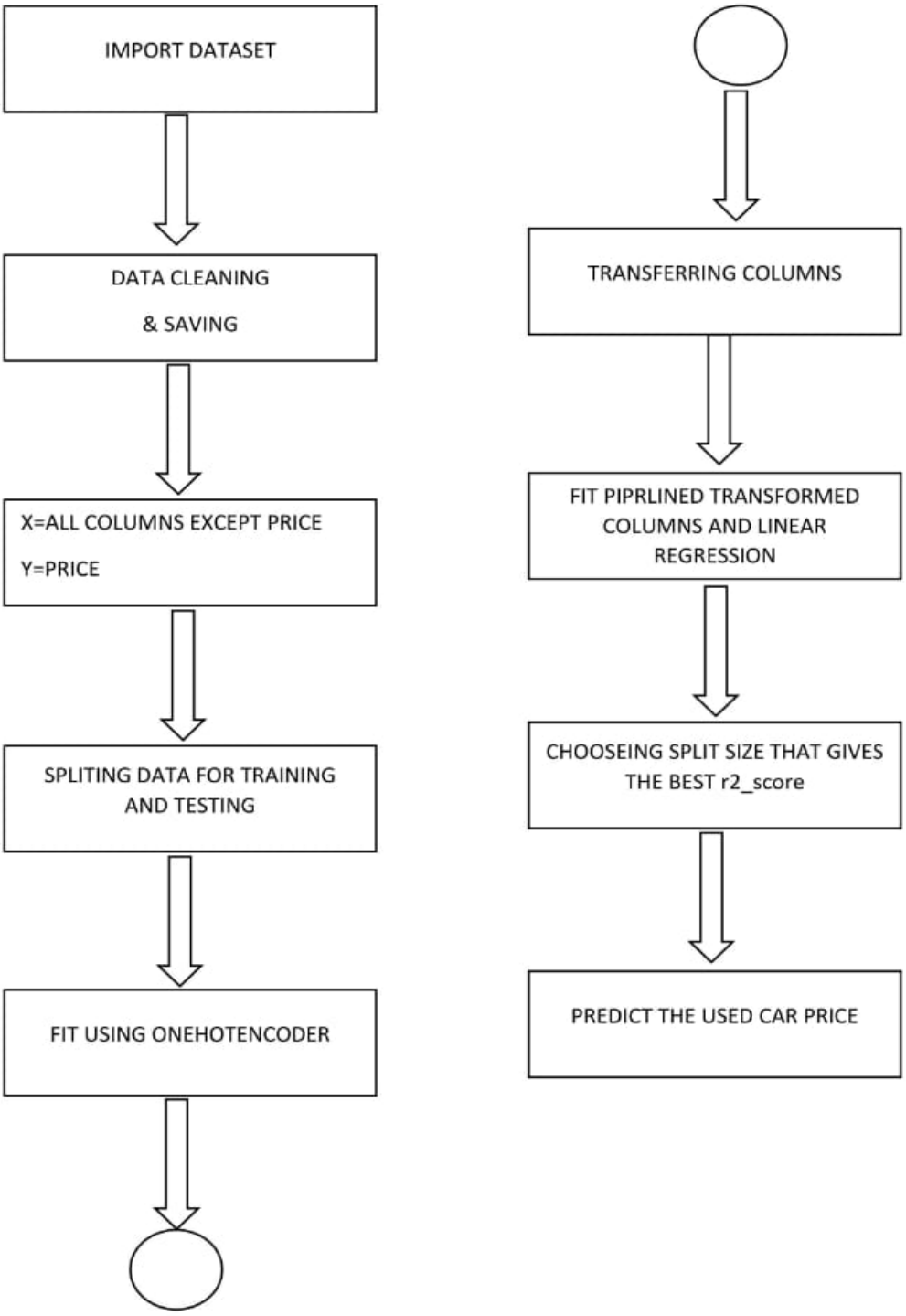
ARTIFICIAL INTELLINGENCE:

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem solving.

The ideal characteristic of artificial intelligence is its ability to rationalize and take actions that have the best chance of achieving a specific goal. A subset of artificial intelligence is machine learning, which refers to the concept that computer program can automatically learn from and adapt to new data without being assisted by humans. Deep learning techniques enable this automatic learning through the absorption of huge amount of unstructured data such as text, images, or video.



ALGORITHIM:


IMPORT DATASET
DATA CLEANING
& SAVING
X=ALL COLUMNS EXCEPT PRICE
Y=PRICE
SPLITING DATA FOR TRAINING
AND TESTING
FIT USING ONEHOTENCODER
TRANSFERRING COLUMNS
FIT PIPRLINED TRANSFORMED
COLUMNS AND LINEAR
REGRESSION
CHOOSEING SPLIT SIZE THAT GIVES
THE BEST r2_score
PREDICT THE USED CAR PRICE

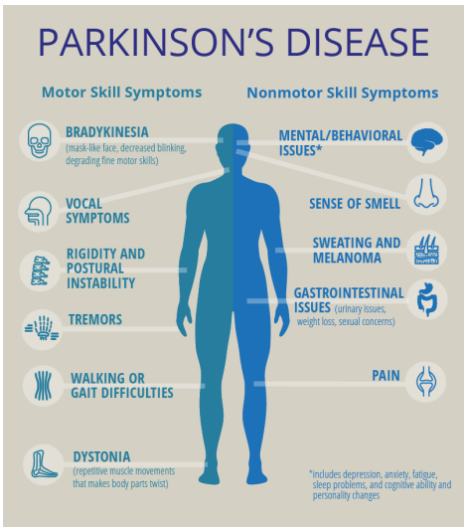
ABSTRACT:

Parkinson is a nervous system disorder that affects movement. The dataset contains 195 records of people with 23 different attributes which contain biomedical measurements. The data is used to separate healthy people from people with Parkinson’s disease.

THINGS TO IMPLEMENT:

The model can be used to differentiate healthy people from people having Parkinson’s disease.

INTRODUCTION:



Parkinson's disease is a progressive disorder of the central nervous system affecting movement and inducing tremors and stiffness. It has 5 stages to it and affects more than 1 million individuals every year in India. This is chronic and has no cure yet. It is a neurodegenerative disorder affecting dopamine-producing neurons in the brain.

THE DATA:

The data used in this project was downloaded from kaggle. This data includes columns like name, MDVP:Fo(Hz), MDVP: Fhi(Hz), MDVP: Flo(Hz), MDVP:Jitter(%),MDVP:Jitter(Abs),MDVP:RAP,MDVP:PPQ,Jitter:DDP,MDVP:Shimmer,MDVP:Shimmer(dB),Shimmer:APQ3,Shimmer:APQ5,MDVP:APQ,Shimmer:DDA,NHR,HNR,status,RPDE,D2,DFA,spread1,spread2,PPE.

ROLES:

- DATA ACQUISITION AND CLEANING
- DATA VISUALIZATION
- DATA MODELLING
- TESTING
- COMPARISON AND MEASUREMENT

ARTIFICIAL INTELLIGENCE / MACHINE LEARNING CODE TO ANALYZE PARKINSON'S DISEASE:

First, import all the libraries/packages which are necessary to analyze the given dataset about Parkinson's disease.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score
```

Next step is to import the given dataset about Parkinson's disease.

```
parkinson_data=pd.read_csv('Parkinsson disease.csv')
print(parkinson_data)
```

OUTPUT:

```
                name  MDVP:Fo(Hz)  MDVP:Fhi(Hz)  ...    spread2        D2       PP
E
0    phon_R01_S01_1      119.992       157.302  ...  0.266482  2.301442  0.28465
4
1    phon_R01_S01_2      122.400       148.650  ...  0.335590  2.486855  0.36867
4
2    phon_R01_S01_3      116.682       131.111  ...  0.311173  2.342259  0.33263
4
3    phon_R01_S01_4      116.676       137.871  ...  0.334147  2.405554  0.36897
5
4    phon_R01_S01_5      116.014       141.781  ...  0.234513  2.332180  0.41033
5
..              ...          ...           ...  ...       ...       ...       ..
.
190  phon_R01_S50_2      174.188       230.978  ...  0.121952  2.657476  0.13305
0
191  phon_R01_S50_3      209.516       253.017  ...  0.129303  2.784312  0.16889
5
192  phon_R01_S50_4      174.688       240.005  ...  0.158453  2.679772  0.13172
8
193  phon_R01_S50_5      198.764       396.961  ...  0.207454  2.138608  0.12330
6
194  phon_R01_S50_6      214.289       260.277  ...  0.190667  2.555477  0.14856
9

[195 rows x 24 columns]
```

Get the features(x) and labels(y) from the DataFrame (dataset). The features are all the columns except 'names' and 'status', and the labels(y) are those in the 'status' column.

```
x=parkinson_data.drop(columns=['name','status'],axis=1)
y=parkinson_data['status']
print(x)
print(y)
```

OUTPUT:

```
     MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  ...    spread2        D2       PPE
0        119.992       157.302        74.997  ...   0.266482  2.301442  0.284654
1        122.400       148.650       113.819  ...   0.335590  2.486855  0.368674
2        116.682       131.111       111.555  ...   0.311173  2.342259  0.332634
3        116.676       137.871       111.366  ...   0.334147  2.405554  0.368975
4        116.014       141.781       110.655  ...   0.234513  2.332180  0.410335
..           ...           ...           ...  ...        ...       ...       ...
190      174.188       230.978        94.261  ...   0.121952  2.657476  0.133050
191      209.516       253.017        89.488  ...   0.129303  2.784312  0.168895
192      174.688       240.005        74.287  ...   0.158453  2.679772  0.131728
193      198.764       396.961        74.904  ...   0.207454  2.138608  0.123306
194      214.289       260.277        77.973  ...   0.190667  2.555477  0.148569

[195 rows x 22 columns]
```

```
0      1
1      1
2      1
3      1
4      1
      ..
190    0
191    0
192    0
193    0
194    0
Name: status, Length: 195, dtype: int64
```

Now, split the dataset into training and testing sets keeping 20% of the data for testing and printing.

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=2)
print(x.shape,x_train.shape,x_test.shape)
```

OUTPUT:

```
(195, 22) (156, 22) (39, 22)
```

Initialize an XGBClassifier and train the model. This classifies using eXtreme Gradient Boosting- using gradient boosting algorithms for modern data science problems. It falls under the

category of Ensemble Learning in ML, where we train and predict using many models to produce one superior output.

What is XGBoost?

XGBoost is a new Machine Learning algorithm designed with speed and performance in mind. XGBoost stands for eXtreme Gradient Boosting and is based on decision trees. In this project, we will import the XGBClassifier from the xgboost library; this is an implementation of the scikit-learn API for XGBoost classification.

```
from xgboost import XGBClassifier
model = XGBClassifier()
model.fit(x_train, y_train,)
print(model)
```

OUTPUT:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=4, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

To standardize and fit the Data and transfer the data in similar range.

```
#Data Standardization
scaler=StandardScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
print(x_train)
```

OUTPUT:

```
[[ 0.63239631 -0.02731081 -0.87985049 ... -0.97586547 -0.55160318
   0.07769494]
 [-1.05512719 -0.83337041 -0.9284778  ...  0.3981808  -0.61014073
   0.39291782]
 [ 0.02996187 -0.29531068 -1.12211107 ... -0.43937044 -0.62849605
  -0.50948408]
 ...
 [-0.9096785  -0.6637302  -0.160638   ...  1.22001022 -0.47404629
  -0.2159482 ]
 [-0.35977689  0.19731822 -0.79063679 ... -0.17896029 -0.47272835
   0.28181221]
 [ 1.01957066  0.19922317 -0.61914972 ... -0.716232    1.23632066
  -0.05829386]]
```

Support Vector Machine Model(SVM) and Accuracy Score

```
#Support Vector Machine Model(SVM)
model=svm.SVC(kernel='linear')

#Training the svm model with training data
model.fit(x_train,y_train)

#Accuracy Score
#Accuracy Score of training data
x_train_prediction=model.predict(x_train)
training_data_accuracy=accuracy_score(y_train,x_train_prediction)
print("Accuracy score of training data: ",training_data_accuracy)

#Accuracy Score of testing data
x_test_prediction=model.predict(x_test)
test_data_accuracy=accuracy_score(y_test,x_test_prediction)
print("Accuracy score of test data: ",test_data_accuracy)
```

OUTPUT:

```
[20:40:20] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0
/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric u
sed with the objective 'binary:logistic' was changed from 'error' to 'logloss'.
Explicitly set eval_metric if you'd like to restore the old behavior.
Accuracy score of training data:  0.8846153846153846
Accuracy score of test data:  0.8717948717948718
```

Predicting the output that the person is either having Parkinson's disease or not.

```
#Predicting
input_data=(119.992,157.302,74.997,0.00784,0.00007,0.0037,0.00554,0.01109,0.04374,0.426,0.02182,
#changing input data to a numpy 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)
#standarize the data
std_data= scaler.transform(input_data_reshaped)

prediction= model.predict(std_data)
print(prediction)

if(prediction[0]==0):
    print("Person is not having Parkinson Disease")
else:
    print("Person is having Parkinson Disease")
```

OUTPUT:

```
[20:42:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0
/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric u
sed with the objective 'binary:logistic' was changed from 'error' to 'logloss'.
Explicitly set eval_metric if you'd like to restore the old behavior.
[1]
Person is having Parkinson Disease
```

```
[0]
Person is  not having Parkinson Disease
```

CONCLUSION:

Parkinson’s disease affects the CNS of the brain and has yet no treatment unless it’s detected early. Late detection leads to no treatment and loss of life. Thus its early detection is significant. For early detection of the disease, we utilized machine learning algorithms such as XGBoost . We checked our Parkinson disease data and find out XGBoost is the best Algorithm to predict the onset of the disease which will enable early treatment and save a life. Using support vector and Standard scaler we conclude whether the person has got Parkinson’s disease or not.