Parkinsons_Disease_Detection

August 30, 2024

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
pd.set_option('display.max_columns', 30)
```

[2]:
```
# Importing data from a CSV file into a Pandas DataFrame
ps_data = pd.read_csv('parkinsons.csv')
```

[3]:
```
ps_data.head(10)
```

[3]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	\
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	
5	phon_R01_S01_6	120.552	131.162	113.787	0.00968	
6	phon_R01_S02_1	120.267	137.244	114.820	0.00333	
7	phon_R01_S02_2	107.332	113.840	104.315	0.00290	
8	phon_R01_S02_3	95.730	132.068	91.754	0.00551	
9	phon_R01_S02_4	95.056	120.103	91.226	0.00532	

	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	\
0	0.00007	0.00370	0.00554	0.01109	0.04374	
1	0.00008	0.00465	0.00696	0.01394	0.06134	
2	0.00009	0.00544	0.00781	0.01633	0.05233	
3	0.00009	0.00502	0.00698	0.01505	0.05492	
4	0.00011	0.00655	0.00908	0.01966	0.06425	
5	0.00008	0.00463	0.00750	0.01388	0.04701	
6	0.00003	0.00155	0.00202	0.00466	0.01608	
7	0.00003	0.00144	0.00182	0.00431	0.01567	
8	0.00006	0.00293	0.00332	0.00880	0.02093	
9	0.00006	0.00268	0.00332	0.00803	0.02838	

	MDVP:Shimmer(dB)	Shimmer:APQ3	Shimmer:APQ5	MDVP:APQ	Shimmer:DDA	\
0	0.426	0.02182	0.03130	0.02971	0.06545	

```
1              0.626       0.03134       0.04518   0.04368       0.09403
2              0.482       0.02757       0.03858   0.03590       0.08270
3              0.517       0.02924       0.04005   0.03772       0.08771
4              0.584       0.03490       0.04825   0.04465       0.10470
5              0.456       0.02328       0.03526   0.03243       0.06985
6              0.140       0.00779       0.00937   0.01351       0.02337
7              0.134       0.00829       0.00946   0.01256       0.02487
8              0.191       0.01073       0.01277   0.01717       0.03218
9              0.255       0.01441       0.01725   0.02444       0.04324

       NHR     HNR  status      RPDE       DFA   spread1   spread2        D2  \
0  0.02211  21.033       1  0.414783  0.815285 -4.813031  0.266482  2.301442
1  0.01929  19.085       1  0.458359  0.819521 -4.075192  0.335590  2.486855
2  0.01309  20.651       1  0.429895  0.825288 -4.443179  0.311173  2.342259
3  0.01353  20.644       1  0.434969  0.819235 -4.117501  0.334147  2.405554
4  0.01767  19.649       1  0.417356  0.823484 -3.747787  0.234513  2.332180
5  0.01222  21.378       1  0.415564  0.825069 -4.242867  0.299111  2.187560
6  0.00607  24.886       1  0.596040  0.764112 -5.634322  0.257682  1.854785
7  0.00344  26.892       1  0.637420  0.763262 -6.167603  0.183721  2.064693
8  0.01070  21.812       1  0.615551  0.773587 -5.498678  0.327769  2.322511
9  0.01022  21.862       1  0.547037  0.798463 -5.011879  0.325996  2.432792

        PPE
0  0.284654
1  0.368674
2  0.332634
3  0.368975
4  0.410335
5  0.357775
6  0.211756
7  0.163755
8  0.231571
9  0.271362
```

[4]:
```
#Getting number of rows and columns present in the dataset
ps_data.shape
```

[4]:
```
(195, 24)
```

[5]:
```
#Getting information about the dataset
ps_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   name              195 non-null    object
```

```
 1   MDVP:Fo(Hz)        195 non-null    float64
 2   MDVP:Fhi(Hz)       195 non-null    float64
 3   MDVP:Flo(Hz)       195 non-null    float64
 4   MDVP:Jitter(%)     195 non-null    float64
 5   MDVP:Jitter(Abs)   195 non-null    float64
 6   MDVP:RAP           195 non-null    float64
 7   MDVP:PPQ           195 non-null    float64
 8   Jitter:DDP         195 non-null    float64
 9   MDVP:Shimmer       195 non-null    float64
 10  MDVP:Shimmer(dB)   195 non-null    float64
 11  Shimmer:APQ3       195 non-null    float64
 12  Shimmer:APQ5       195 non-null    float64
 13  MDVP:APQ           195 non-null    float64
 14  Shimmer:DDA        195 non-null    float64
 15  NHR                195 non-null    float64
 16  HNR                195 non-null    float64
 17  status             195 non-null    int64
 18  RPDE               195 non-null    float64
 19  DFA                195 non-null    float64
 20  spread1            195 non-null    float64
 21  spread2            195 non-null    float64
 22  D2                 195 non-null    float64
 23  PPE                195 non-null    float64
dtypes: float64(22), int64(1), object(1)
memory usage: 36.7+ KB
```

[6]:
```
#checking for missing values in the column
ps_data.isnull().sum()
```

[6]:
```
name               0
MDVP:Fo(Hz)        0
MDVP:Fhi(Hz)       0
MDVP:Flo(Hz)       0
MDVP:Jitter(%)     0
MDVP:Jitter(Abs)   0
MDVP:RAP           0
MDVP:PPQ           0
Jitter:DDP         0
MDVP:Shimmer       0
MDVP:Shimmer(dB)   0
Shimmer:APQ3       0
Shimmer:APQ5       0
MDVP:APQ           0
Shimmer:DDA        0
NHR                0
HNR                0
status             0
```

```
RPDE                0
DFA                 0
spread1             0
spread2             0
D2                  0
PPE                 0
dtype: int64
```

[7]:
```
#Checking if there is any duplicate entries in the dataset
ps_data.duplicated().sum()
```

[7]:
```
0
```

[8]:
```
# If there are any missing values, we can replace them with the mean or median
  ↪of the data(or any statistical measures)
ps_data.describe()
```

[8]:

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	\
count	195.000000	195.000000	195.000000	195.000000	
mean	154.228641	197.104918	116.324631	0.006220	
std	41.390065	91.491548	43.521413	0.004848	
min	88.333000	102.145000	65.476000	0.001680	
25%	117.572000	134.862500	84.291000	0.003460	
50%	148.790000	175.829000	104.315000	0.004940	
75%	182.769000	224.205500	140.018500	0.007365	
max	260.105000	592.030000	239.170000	0.033160	

	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	\
count	195.000000	195.000000	195.000000	195.000000	195.000000	
mean	0.000044	0.003306	0.003446	0.009920	0.029709	
std	0.000035	0.002968	0.002759	0.008903	0.018857	
min	0.000007	0.000680	0.000920	0.002040	0.009540	
25%	0.000020	0.001660	0.001860	0.004985	0.016505	
50%	0.000030	0.002500	0.002690	0.007490	0.022970	
75%	0.000060	0.003835	0.003955	0.011505	0.037885	
max	0.000260	0.021440	0.019580	0.064330	0.119080	

	MDVP:Shimmer(dB)	Shimmer:APQ3	Shimmer:APQ5	MDVP:APQ	Shimmer:DDA	\
count	195.000000	195.000000	195.000000	195.000000	195.000000	
mean	0.282251	0.015664	0.017878	0.024081	0.046993	
std	0.194877	0.010153	0.012024	0.016947	0.030459	
min	0.085000	0.004550	0.005700	0.007190	0.013640	
25%	0.148500	0.008245	0.009580	0.013080	0.024735	
50%	0.221000	0.012790	0.013470	0.018260	0.038360	
75%	0.350000	0.020265	0.022380	0.029400	0.060795	
max	1.302000	0.056470	0.079400	0.137780	0.169420	

```
                NHR         HNR      status        RPDE         DFA     spread1  \
count   195.000000  195.000000  195.000000  195.000000  195.000000  195.000000
mean      0.024847   21.885974    0.753846    0.498536    0.718099   -5.684397
std       0.040418    4.425764    0.431878    0.103942    0.055336    1.090208
min       0.000650    8.441000    0.000000    0.256570    0.574282   -7.964984
25%       0.005925   19.198000    1.000000    0.421306    0.674758   -6.450096
50%       0.011660   22.085000    1.000000    0.495954    0.722254   -5.720868
75%       0.025640   25.075500    1.000000    0.587562    0.761881   -5.046192
max       0.314820   33.047000    1.000000    0.685151    0.825288   -2.434031

           spread2          D2         PPE
count   195.000000  195.000000  195.000000
mean      0.226510    2.381826    0.206552
std       0.083406    0.382799    0.090119
min       0.006274    1.423287    0.044539
25%       0.174351    2.099125    0.137451
50%       0.218885    2.361532    0.194052
75%       0.279234    2.636456    0.252980
max       0.450493    3.671155    0.527367
```

[9]:
```
# Analyzing the distribution of the 'status' variable to determine how many␣
 ↪people have the disease or not
#(status=1): parkinsons disease
#(status=0): No parkinsons disease
ps_data['status'].value_counts()
```

[9]:
```
status
1    147
0     48
Name: count, dtype: int64
```

[10]:
```
# Calculating the mean of each feature grouped by the 'status' variable to␣
 ↪clearly observe the differences in values
# (useful for machine learning models to make predictions)
# Select only numeric columns for grouping and calculating the mean
numeric_columns = ps_data.select_dtypes(include=['number']).columns
numeric_data = ps_data[numeric_columns]

# Group by 'status' and calculate the mean of numeric columns
mean_values = numeric_data.groupby('status').mean()
mean_values
```

[10]:
```
        MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
status
0        181.937771    223.636750    145.207292        0.003866
1        145.180762    188.441463    106.893558        0.006989
```

```
        MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  \
status
0               0.000023  0.001925  0.002056    0.005776      0.017615
1               0.000051  0.003757  0.003900    0.011273      0.033658

        MDVP:Shimmer(dB)  Shimmer:APQ3  Shimmer:APQ5  MDVP:APQ  Shimmer:DDA  \
status
0               0.162958      0.009504      0.010509  0.013305     0.028511
1               0.321204      0.017676      0.020285  0.027600     0.053027

              NHR        HNR      RPDE       DFA   spread1   spread2        D2  \
status
0        0.011483  24.678750  0.442552  0.695716 -6.759264  0.160292  2.154491
1        0.029211  20.974048  0.516816  0.725408 -5.333420  0.248133  2.456058

              PPE
status
0        0.123017
1        0.233828
```

1 Data Pre Processing

[11]:
```
# Separating all features and the target variable 'status' into distinct␣
 ↪variables
features=ps_data.drop(columns=['name','status'], axis=1)
```

[12]:
```
features
```

[12]:
```
     MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
0        119.992       157.302        74.997         0.00784
1        122.400       148.650       113.819         0.00968
2        116.682       131.111       111.555         0.01050
3        116.676       137.871       111.366         0.00997
4        116.014       141.781       110.655         0.01284
..           …             …             …              …
190      174.188       230.978        94.261         0.00459
191      209.516       253.017        89.488         0.00564
192      174.688       240.005        74.287         0.01360
193      198.764       396.961        74.904         0.00740
194      214.289       260.277        77.973         0.00567

     MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  \
0             0.00007   0.00370   0.00554     0.01109       0.04374
1             0.00008   0.00465   0.00696     0.01394       0.06134
2             0.00009   0.00544   0.00781     0.01633       0.05233
3             0.00009   0.00502   0.00698     0.01505       0.05492
```

```
4             0.00011   0.00655   0.00908     0.01966       0.06425
..                …         …         …           …             …
190           0.00003   0.00263   0.00259     0.00790       0.04087
191           0.00003   0.00331   0.00292     0.00994       0.02751
192           0.00008   0.00624   0.00564     0.01873       0.02308
193           0.00004   0.00370   0.00390     0.01109       0.02296
194           0.00003   0.00295   0.00317     0.00885       0.01884

     MDVP:Shimmer(dB)  Shimmer:APQ3  Shimmer:APQ5  MDVP:APQ  Shimmer:DDA  \
0               0.426       0.02182       0.03130   0.02971      0.06545
1               0.626       0.03134       0.04518   0.04368      0.09403
2               0.482       0.02757       0.03858   0.03590      0.08270
3               0.517       0.02924       0.04005   0.03772      0.08771
4               0.584       0.03490       0.04825   0.04465      0.10470
..                …             …             …         …            …
190             0.405       0.02336       0.02498   0.02745      0.07008
191             0.263       0.01604       0.01657   0.01879      0.04812
192             0.256       0.01268       0.01365   0.01667      0.03804
193             0.241       0.01265       0.01321   0.01588      0.03794
194             0.190       0.01026       0.01161   0.01373      0.03078

         NHR     HNR      RPDE       DFA   spread1   spread2        D2  \
0    0.02211  21.033  0.414783  0.815285 -4.813031  0.266482  2.301442
1    0.01929  19.085  0.458359  0.819521 -4.075192  0.335590  2.486855
2    0.01309  20.651  0.429895  0.825288 -4.443179  0.311173  2.342259
3    0.01353  20.644  0.434969  0.819235 -4.117501  0.334147  2.405554
4    0.01767  19.649  0.417356  0.823484 -3.747787  0.234513  2.332180
..       …     …        …        …        …        …       …
190  0.02764  19.517  0.448439  0.657899 -6.538586  0.121952  2.657476
191  0.01810  19.147  0.431674  0.683244 -6.195325  0.129303  2.784312
192  0.10715  17.883  0.407567  0.655683 -6.787197  0.158453  2.679772
193  0.07223  19.020  0.451221  0.643956 -6.744577  0.207454  2.138608
194  0.04398  21.209  0.462803  0.664357 -5.724056  0.190667  2.555477

          PPE
0    0.284654
1    0.368674
2    0.332634
3    0.368975
4    0.410335
..       …
190  0.133050
191  0.168895
192  0.131728
193  0.123306
194  0.148569
```

```
[195 rows x 22 columns]
```

[13]:
```
status=ps_data['status']
```

[14]:
```
status
```

[14]:
```
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
```

[15]:
```
# Divide the data into training and testing sets for model evaluation
from sklearn.model_selection import train_test_split

feature_train, feature_test, status_train, status_test =␣
 ↪train_test_split(features, status, test_size=0.2, random_state=2)
```

[16]:
```
# Standardizing the data to ensure all values are within the same range
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(feature_train)
feature_train=scaler.transform(feature_train)
feature_test=scaler.transform(feature_test)
```

[17]:
```
feature_train
```

[17]:
```
array([[ 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]])
```

[18]:
```
feature_test
```

[18]:
```
array([[-1.70008583e+00, -9.67968410e-01, -7.70130215e-01,
        -2.75000683e-01,  4.16156683e-01, -2.92615113e-01,
        -9.70869783e-02, -2.91621655e-01, -4.94706656e-01,
        -4.90058396e-01, -5.32488171e-01, -4.26848854e-01,
        -3.60251422e-01, -5.32484688e-01, -3.57189713e-01,
        -1.08840337e-01,  1.06963705e+00,  1.05628304e+00,
         3.72180199e-01,  1.94886208e+00,  3.66935071e-02,
         4.44314482e-01],
       [-1.39044095e+00, -9.29681132e-01, -7.37045677e-01,
         7.42068829e-01,  1.50451280e+00,  8.54349819e-01,
         7.33639862e-01,  8.53234751e-01, -3.12538562e-03,
         3.01660094e-01,  1.16511011e-01, -7.67595149e-02,
        -2.23967413e-01,  1.16829276e-01, -1.19644974e-01,
        -5.22790834e-01,  9.12650090e-01,  1.31721995e+00,
         6.70118138e-01,  4.74318608e-01,  1.42454868e-02,
         7.46859799e-01],
       [-1.35302065e+00, -6.29175292e-01, -7.29027225e-01,
         4.92094897e-01,  1.23242377e+00,  4.52288742e-01,
         3.45291949e-01,  4.53262231e-01, -1.57435662e-01,
        -1.27992014e-01, -6.49095096e-02, -2.59345791e-01,
        -2.60383383e-01, -6.52155416e-02,  2.54927471e-01,
        -6.85306331e-01,  1.63423714e+00, -8.42551171e-01,
         2.43042190e+00,  2.01645645e+00,  4.23263515e-01,
         1.70448737e+00],
       [ 1.04170416e+00,  2.17641374e-01,  6.81254572e-01,
        -5.21158220e-01, -6.72199437e-01, -5.17021295e-01,
        -4.85434892e-01, -5.17060712e-01,  2.00946185e-01,
        -7.30321934e-03,  3.80225376e-01,  1.55839524e-01,
        -8.23037710e-03,  3.80233164e-01, -2.30529235e-01,
        -6.30271202e-01,  1.02404991e+00, -1.08229315e+00,
         7.45935205e-02,  1.00442371e+00,  1.83674893e-01,
         1.66131346e-02],
       [-9.97415782e-01,  4.34584493e+00, -7.21705053e-01,
        -2.92174464e-01, -1.28021377e-01, -2.92615113e-01,
        -3.23341850e-01, -2.92660545e-01, -7.37481456e-01,
        -6.97643122e-01, -7.74693918e-01, -7.20574603e-01,
        -5.81506029e-01, -7.74691576e-01, -3.05128200e-01,
         5.04099919e-01,  6.50570904e-01, -9.71664570e-01,
        -6.75704306e-01, -8.48255200e-01, -5.82834991e-01,
        -9.40724851e-01],
       [-9.42866298e-01, -8.40790856e-01, -1.91454011e-01,
        -6.07027128e-01, -4.00110407e-01, -5.60655830e-01,
        -6.34020180e-01, -5.60694078e-01, -1.04760993e+00,
```

```
 -9.63158469e-01, -1.08329583e+00, -9.99217138e-01,
 -9.05939218e-01, -1.08298323e+00, -5.06387642e-01,
  8.62583639e-01, -1.37906646e-01,  6.45695194e-02,
 -6.90797014e-01, -5.49317204e-01, -9.03760130e-01,
 -7.17451813e-01],
[ 1.11654476e+00,  1.21009469e-01,  1.73728699e+00,
 -8.60817456e-01, -9.44288467e-01, -7.57011240e-01,
 -8.23128729e-01, -7.58083113e-01, -9.71711386e-01,
 -9.24538055e-01, -9.93520731e-01, -8.88871519e-01,
 -8.76144334e-01, -9.93519422e-01, -5.89550838e-01,
  2.18191596e+00, -1.43170413e+00,  4.31667996e-01,
 -1.87014020e+00, -6.49696731e-01,  1.37056937e-01,
 -1.61637865e+00],
[-1.08181685e+00, -7.68280933e-01, -3.97260953e-01,
 -2.80725277e-01, -1.28021377e-01, -2.83264855e-01,
 -1.57871869e-01, -2.82271648e-01,  1.08157030e+00,
  8.56828547e-01,  1.16669268e+00,  1.16482717e+00,
  8.76788050e-01,  1.16701590e+00,  2.11655564e-01,
 -1.05220719e+00,  1.20295881e+00, -2.06049026e-01,
  8.99849281e-01,  9.26297137e-01,  7.05889923e-01,
  7.08666241e-01],
[-1.30006458e+00,  7.04729216e-02, -9.07230027e-01,
 -3.07513440e-02,  4.16156683e-01, -9.93764557e-02,
  5.48752486e-02, -9.94270682e-02,  3.21579625e-01,
  3.98211129e-01,  7.81695609e-02,  2.90000745e-01,
  6.81466028e-01,  7.84876447e-02,  3.06798283e-02,
  9.57608471e-02,  2.41205147e-01,  3.42159590e-01,
  8.64938299e-02,  1.34863999e-01,  1.14522326e+00,
  1.20125076e-01],
[-9.94278751e-01, -7.58966614e-01, -1.52844152e-01,
  7.64967205e-01,  1.23242377e+00,  6.11243122e-01,
  1.42591223e+00,  6.12212349e-01,  1.05844889e+00,
  8.85793858e-01,  1.03670582e+00,  1.57048624e+00,
  5.76632175e-01,  1.03640498e+00, -3.24961157e-01,
 -2.09630161e-01, -6.29027463e-01,  1.97458060e+00,
  1.08764549e+00,  1.00944857e+00, -1.27555004e-01,
  1.33339529e+00],
[-4.90262437e-01, -4.40496653e-01,  2.90102340e-01,
 -6.79538650e-01, -6.72199437e-01, -7.16493457e-01,
 -6.94805071e-01, -7.15488637e-01, -9.39039829e-01,
 -9.00400296e-01, -9.28059719e-01, -9.23007214e-01,
 -7.75172780e-01, -9.28058101e-01, -4.89033804e-01,
  8.80712858e-01, -3.41249278e-01, -1.36057476e+00,
 -7.78624773e-01, -8.54103808e-01, -8.87475153e-01,
 -7.48663060e-01],
[-9.35795530e-01, -7.98594001e-01, -2.05379614e-01,
 -3.99033550e-01, -1.28021377e-01, -4.17285214e-01,
```

```
 -5.63104474e-01, -4.18366194e-01, -1.05263632e+00,
 -9.67986021e-01, -1.03840828e+00, -1.03970366e+00,
 -9.56700874e-01, -1.03840719e+00, -2.31430733e-01,
  1.12653643e+00, -3.69919314e-01, -3.71085406e-01,
 -1.08276123e+00, -8.52597527e-01, -1.62260941e-01,
 -1.02383686e+00],
[ 1.14343359e+00,  4.61962269e-01,  1.02887806e+00,
 -7.53958371e-01, -9.44288467e-01, -7.72595003e-01,
 -7.62343838e-01, -7.71588679e-01, -2.33836841e-01,
 -3.40404291e-01, -1.62165871e-01, -2.55376525e-01,
 -2.97351110e-01, -1.61848921e-01, -3.83558531e-01,
 -2.03371264e-01,  4.40320486e-03, -1.66778915e+00,
 -4.10966775e-01, -5.61983681e-02,  4.64016354e-01,
 -5.20584992e-01],
[-9.94428133e-01, -6.84097654e-01, -1.57089215e-01,
  6.63832713e-01,  1.23242377e+00,  4.80339515e-01,
  1.14562634e+00,  4.79234473e-01,  1.18863248e+00,
  1.05475817e+00,  1.19287709e+00,  1.68718269e+00,
  6.77051971e-01,  1.19257699e+00, -3.15044679e-01,
 -2.11140929e-01, -5.81472130e-01,  1.86144750e+00,
  1.37652958e+00,  1.27980253e+00,  3.22523792e-02,
  1.72317122e+00],
[-8.57195476e-01, -7.86654839e-01, -5.82848127e-02,
 -2.38744922e-01, -1.28021377e-01, -5.57539078e-01,
 -5.25958152e-01, -5.57577409e-01, -7.61105504e-01,
 -7.50746192e-01, -7.41963411e-01, -8.76169865e-01,
 -6.13507942e-01, -7.41649195e-01, -4.30887179e-01,
  3.32303988e-01,  7.74032948e-01,  2.61996392e-01,
 -6.83047975e-01,  4.82473709e-01, -7.89904155e-01,
 -7.98483076e-01],
[ 1.48148268e-01, -1.43194979e-02, -9.14821727e-01,
 -2.80725277e-01, -4.00110407e-01, -2.11579547e-01,
 -3.57111233e-01, -2.10588262e-01, -6.71635703e-01,
 -6.30057397e-01, -7.99943166e-01, -6.53097066e-01,
 -4.11564835e-01, -7.99629223e-01, -3.56288215e-01,
  8.75317257e-01, -6.48803134e-01,  7.76715016e-02,
 -8.57198216e-01, -1.82714228e-01, -1.09783161e+00,
 -7.83735493e-01],
[-9.39259575e-02, -4.02408731e-01, -9.55116135e-01,
 -6.77630452e-01, -6.72199437e-01, -7.07143200e-01,
 -7.08312824e-01, -7.06138630e-01, -6.53038047e-01,
 -6.78332915e-01, -5.46515531e-01, -7.20574603e-01,
 -7.16686525e-01, -5.46823834e-01, -4.82723318e-01,
  7.12154288e-01, -2.08246176e-01, -1.22635888e+00,
 -3.86515968e-01, -2.55827710e-01, -6.74530773e-01,
 -4.09865504e-01],
[ 1.50623372e+00,  3.74810813e-01, -5.47635010e-01,
```

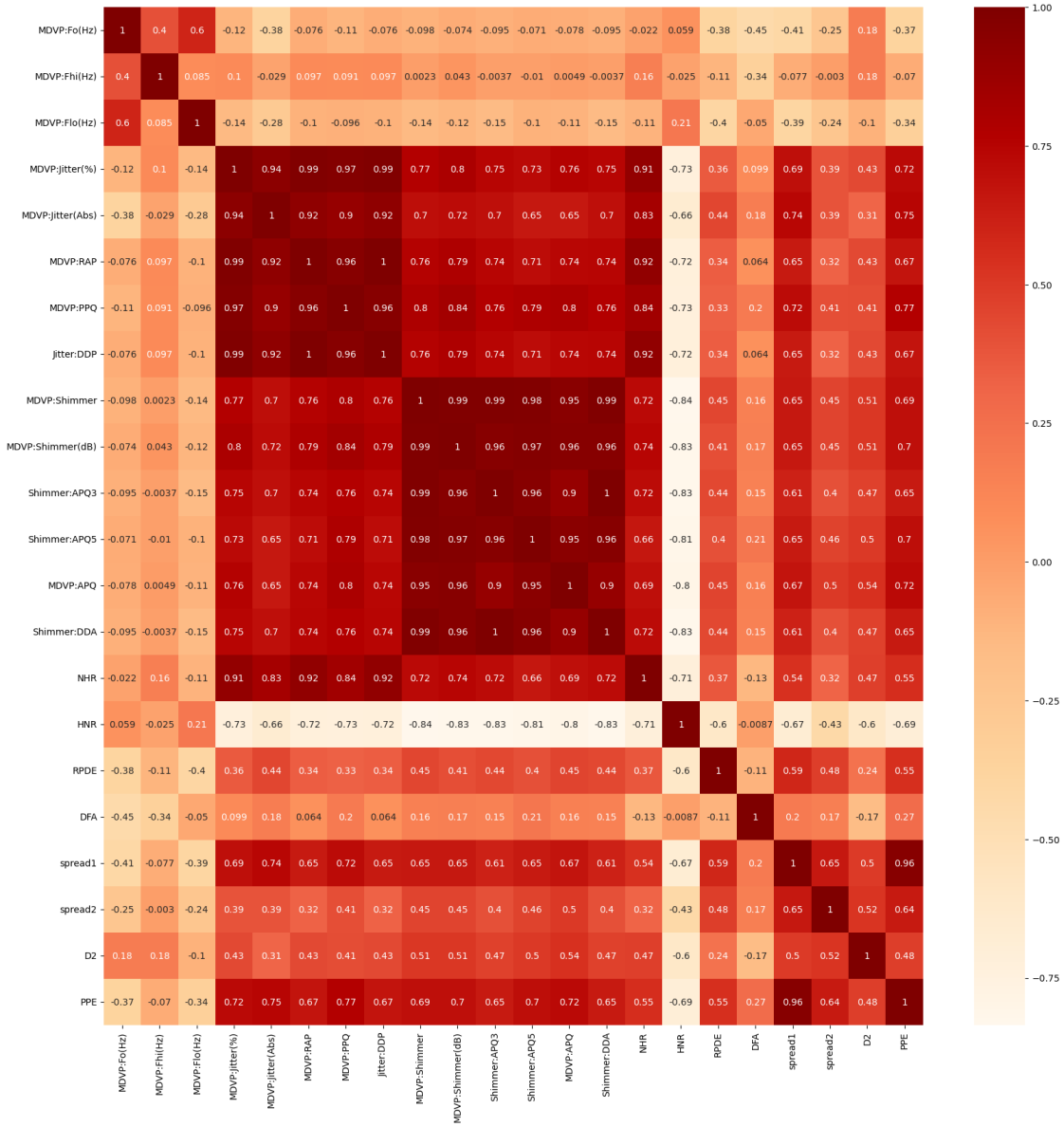
```
 -4.67728677e-01, -6.72199437e-01, -6.88442684e-01,
 -7.79228530e-01, -6.88477506e-01, -9.18934256e-01,
 -8.42469675e-01, -9.06551100e-01, -9.91278604e-01,
 -8.11036993e-01, -9.06237660e-01, -4.66496353e-01,
  6.38558292e-01, -3.23291844e-01, -2.55930334e+00,
  1.34987413e-01,  1.92877440e+00,  1.34592739e+00,
  1.32384340e-01],
[-9.05022750e-01, -6.91041861e-01, -7.95101276e-02,
 -6.03210732e-01, -4.00110407e-01, -6.01173613e-01,
 -5.29335090e-01, -6.00171885e-01, -7.63618701e-01,
 -7.65228847e-01, -8.13035368e-01, -7.48359471e-01,
 -6.58752027e-01, -8.13033207e-01, -4.83174067e-01,
  7.04384622e-01,  9.28142594e-01,  8.31175656e-01,
  3.10734791e-02,  3.79975975e-01, -1.35833065e+00,
  3.69165339e-02],
[ 4.30705131e-01, -7.64629495e-02, -7.22783164e-01,
 -3.30338424e-01, -4.00110407e-01, -3.95467946e-01,
 -3.40226541e-01, -3.95510622e-01,  5.35201343e-01,
  3.16142749e-01,  5.03666143e-01,  4.68617754e-01,
  6.77051971e-01,  5.03986233e-01, -2.05287289e-01,
 -2.59053864e-01,  3.77273009e-01, -6.12060951e-01,
  3.73061900e-01, -1.77854118e-01,  3.90658214e-01,
  4.85243046e-01],
[-1.15237851e-01, -4.68483911e-01,  3.25612629e-01,
 -9.06614207e-01, -9.44288467e-01, -8.50513816e-01,
 -8.87290558e-01, -8.50544293e-01, -1.05716007e+00,
 -9.92123780e-01, -1.11602634e+00, -1.03494054e+00,
 -8.56281077e-01, -1.11633733e+00, -5.66337263e-01,
  1.79256940e+00, -1.73989531e+00, -8.14440784e-01,
 -9.48935976e-01, -7.61455741e-01, -2.42145697e-01,
 -9.99447044e-01],
[ 1.14993173e+00,  3.52959266e-01, -7.21615210e-01,
 -5.32607407e-01, -6.72199437e-01, -4.95204027e-01,
 -4.98942645e-01, -4.94205139e-01, -5.68091999e-01,
 -5.72126776e-01, -4.96952193e-01, -5.57834661e-01,
 -5.73229672e-01, -4.97260262e-01, -1.66973622e-01,
 -6.33508563e-01,  3.66382331e-01, -1.62625904e+00,
 -2.03412110e-01, -1.41679831e-01,  1.96849313e-01,
 -2.06810061e-01],
[-9.09728296e-01, -8.08572837e-01, -2.85092463e-01,
 -7.23427203e-01, -6.72199437e-01, -7.22726962e-01,
 -7.55589962e-01, -7.21721975e-01, -1.05816535e+00,
 -1.00660644e+00, -1.09638804e+00, -1.02541430e+00,
 -9.05939218e-01, -1.09638721e+00, -5.10219009e-01,
  1.02445166e+00, -5.30961981e-02, -1.49955850e-02,
 -1.97547110e-01, -3.46169284e-01, -7.16808594e-01,
 -2.65371371e-01],
```

```
[-9.86261894e-01, -6.11056117e-01, -4.14106441e-01,
 -5.65046773e-01, -4.00110407e-01, -5.35721810e-01,
 -4.75304076e-01, -5.35760726e-01, -7.38989374e-01,
 -7.41091088e-01, -7.17649321e-01, -6.49127799e-01,
 -6.76960012e-01, -7.17646711e-01, -4.87005434e-01,
  8.25030257e-01, -1.02653719e+00,  6.88567776e-01,
 -3.04431686e-01,  4.59580994e-02, -1.31247771e+00,
 -4.84751047e-01],
[ 2.14698475e+00,  6.13516536e-01,  2.46058732e+00,
 -8.09296111e-01, -9.71497369e-01, -7.19610210e-01,
 -7.42082208e-01, -7.20683085e-01, -8.20919585e-01,
 -7.94194157e-01, -7.49444670e-01, -7.94402967e-01,
 -8.44694178e-01, -7.49130489e-01, -5.12698129e-01,
  7.35894931e-01, -6.15999889e-01, -1.50856548e+00,
 -1.17660745e+00, -1.45108545e+00, -6.81186250e-02,
 -1.13270084e+00],
[-6.89090449e-01,  3.10172439e+00, -6.76267156e-01,
 -1.29977638e-01,  1.44067653e-01, -3.33132896e-01,
 -1.51117992e-01, -3.33177242e-01, -5.62562967e-01,
 -3.74197154e-01, -7.47574355e-01, -6.05465863e-01,
 -4.47429048e-01, -7.47260166e-01, -9.03462865e-02,
  6.73521785e-01, -8.67335831e-02, -6.63721263e-01,
 -4.58391494e-01,  6.41821794e-01,  7.40929134e-01,
 -5.99406799e-01],
[-1.06814836e+00, -9.21418966e-01, -4.81713000e-01,
 -5.72679564e-01, -4.00110407e-01, -5.51305573e-01,
 -5.25958152e-01, -5.50305181e-01, -8.31977651e-01,
 -7.99021709e-01, -8.42025245e-01, -7.73762779e-01,
 -7.70758723e-01, -8.41711501e-01, -5.12247380e-01,
  1.06286834e+00, -1.09428995e+00,  8.79546455e-01,
 -2.43175607e-01, -3.91192973e-01, -1.36399378e+00,
 -3.07147250e-01],
[-2.84911392e-01, -1.69443416e-02, -8.47282551e-01,
  1.60068452e-01,  1.44067653e-01,  2.06065292e-01,
  2.13591352e-01,  2.06006492e-01, -3.84628642e-01,
 -3.16266532e-01, -3.86603629e-01, -3.05389287e-01,
 -3.31560052e-01, -3.86287737e-01, -1.97624556e-01,
 -5.96818477e-01,  1.31688917e+00,  2.97265209e-01,
  6.65961153e-02, -1.14668718e+00, -1.58260382e+00,
  1.54468174e-01],
[-1.53270282e+00, -8.80883405e-01, -6.09446720e-01,
 -2.23479338e-01,  4.16156683e-01, -2.48980577e-01,
 -9.03331016e-02, -2.50066069e-01, -1.45372318e-01,
 -2.10060394e-01, -1.93961220e-01, -1.22803011e-01,
 -5.56814898e-02, -1.93644420e-01, -3.89643643e-01,
  5.17327442e-02,  4.68869042e-01,  1.47321016e+00,
  5.83195142e-01,  1.18388300e+00,  1.01022954e-01,
```

```
  6.76221560e-01],
[-1.10711228e+00, -7.52786602e-01, -4.67922160e-01,
 -2.48285911e-01,  1.44067653e-01, -1.77295269e-01,
 -2.52426144e-01, -1.77343793e-01, -6.89228079e-01,
 -6.92815570e-01, -6.37225791e-01, -6.72149547e-01,
 -6.52130941e-01, -6.36911081e-01, -2.96563968e-01,
 -8.70421095e-02,  5.62667579e-01, -8.70138226e-01,
  4.63285119e-02, -4.76556757e-01, -1.31519187e+00,
  4.36736081e-02],
[-1.19034319e+00, -7.19704711e-01, -3.47825061e-01,
  2.13497995e-01,  6.88245713e-01,  3.47637782e-03,
  4.29715408e-01,  2.38411858e-03, -2.05186399e-01,
 -2.10060394e-01, -1.54684613e-01, -2.14890002e-01,
 -2.63693926e-01, -1.54367627e-01, -3.86488400e-01,
 -1.28264500e-01,  3.65510702e-01,  1.85765335e+00,
  9.04569134e-01, -2.33410008e-01, -1.02476887e+00,
  1.16249494e+00],
[-4.34318718e-01, -2.27042591e-01, -9.38944466e-01,
 -4.94443448e-01, -4.00110407e-01, -5.70006088e-01,
 -5.09073460e-01, -5.69005195e-01, -7.46528964e-01,
 -6.97643122e-01, -8.60728392e-01, -7.52328738e-01,
 -4.12668349e-01, -8.60726456e-01, -3.85361527e-01,
  7.05032094e-01, -6.26487555e-01, -1.00569982e+00,
  8.10400161e-03,  3.48285229e-01,  3.19746607e-01,
 -9.03631484e-02],
[ 9.06562857e-01,  1.02569112e-01, -8.54919172e-01,
  2.23038985e-01, -1.28021377e-01,  3.18268384e-01,
  1.02152386e-01,  3.19245465e-01, -4.18286145e-02,
 -1.13509358e-01,  1.14640696e-01, -5.13562069e-02,
 -2.54314055e-01,  1.14335511e-01, -4.06546732e-01,
  6.72720746e-02,  5.09675919e-03,  3.90586460e-01,
 -1.55890705e-01,  6.27064945e-01,  5.45279652e-01,
 -2.42890692e-01],
[ 1.43584970e+00,  6.71584281e-01, -9.07117724e-01,
 -1.56692409e-01, -4.00110407e-01, -1.64828259e-01,
 -1.40987177e-01, -1.64877117e-01, -6.24890244e-01,
 -5.23851258e-01, -5.82051509e-01, -5.70536315e-01,
 -6.46613370e-01, -5.82048259e-01,  3.71220721e-01,
 -8.92003499e-02, -3.20601978e-01, -1.03328687e+00,
 -4.85227013e-02, -4.08644653e-01,  4.10778366e-01,
 -6.40796560e-01],
[-1.52451168e+00, -1.07977346e+00, -6.31053866e-01,
 -7.84562930e-02,  4.16156683e-01, -5.26251678e-02,
 -9.03331016e-02, -5.16381438e-02,  3.75884006e-02,
 -1.71439979e-01,  1.44565731e-01,  5.00738283e-03,
 -4.96121614e-02,  1.44260687e-01, -3.80628663e-01,
  5.45384566e-02,  1.38658201e+00,  1.12087712e+00,
```

```
        4.91536327e-01,  2.83621044e-01, -9.46868610e-01,
        5.54519145e-01],
       [ 1.23761922e+00,  2.95888297e-01, -5.91612964e-01,
       -2.94082662e-01, -6.72199437e-01, -1.43010991e-01,
       -3.80749802e-01, -1.44099324e-01, -5.13806951e-01,
       -4.32127775e-01, -5.06303766e-01, -4.72892350e-01,
       -5.19157474e-01, -5.06611880e-01, -3.83783906e-01,
        5.21643923e-02, -1.55114544e+00, -8.44865548e-02,
       -9.40564247e-01, -5.14154950e-01,  1.37686127e+00,
       -9.95317721e-01],
       [ 2.16159933e+00,  5.67864191e-01,  2.56219930e+00,
       -8.37919081e-01, -9.71497369e-01, -7.44544230e-01,
       -7.48836085e-01, -7.44577548e-01, -8.58617536e-01,
       -8.32814572e-01, -8.14905683e-01, -7.79319753e-01,
       -8.34210792e-01, -8.15215252e-01, -5.17656368e-01,
        8.08411807e-01, -5.50290302e-01, -1.57665467e+00,
       -1.23164638e+00, -1.57435339e+00, -1.56698801e-01,
       -1.25321272e+00],
       [-1.38571051e+00, -9.92566629e-01, -5.10574936e-01,
       -8.22726889e-02,  4.16156683e-01,  9.70988289e-03,
       -3.63020875e-02,  9.65634621e-03, -3.51957085e-01,
       -3.98334912e-01, -2.59422233e-01, -3.64134437e-01,
       -4.38049177e-01, -2.59105741e-01, -3.41188122e-01,
       -2.34449925e-01,  5.39845893e-01,  1.27570853e+00,
        5.73962113e-01, -9.23122088e-01,  8.96966890e-02,
        6.04403514e-01],
       [-7.58702661e-01, -5.02318921e-01, -4.67562790e-01,
        6.21852358e-01,  9.60334743e-01,  7.63963996e-01,
        3.21653380e-01,  7.62851350e-01, -1.38335367e-01,
       -1.56957324e-01, -1.38786938e-01, -2.37117897e-01,
       -2.14035784e-01, -1.39093319e-01,  2.42757247e-01,
       -1.90575389e-02,  1.29343016e+00, -1.72520611e+00,
        5.31555286e-01,  6.43681110e-01,  1.26118997e+00,
        4.43917638e-01]])
```

[19]:
```
# Correlation Matrix
correl=features.corr()
plt.figure(figsize=(20,20))
sns.heatmap(correl,annot=True,cmap='OrRd')
plt.show()
```



2 SVM Model Training

[20]:
```
# Importing the SVM model from scikit-learn
from sklearn import svm

model = svm.SVC(kernel='linear')
```

[21]:
```
model.fit(feature_train, status_train)
```

[21]: SVC(kernel='linear')

[22]:
```
#Evaluating the performance of the SVM model
# Accuracy of the model on the training data
from sklearn.metrics import accuracy_score, confusion_matrix

feature_train_prediction = model.predict(feature_train)
training_data_accuracy = accuracy_score(status_train, feature_train_prediction)
print("Accuracy of the model on the training data is␣
 ↪",training_data_accuracy*100)
```

```
Accuracy of the model on the training data is  88.46153846153845
```

[23]:
```
# Accuracy of the model on the test data
feature_test_prediction = model.predict(feature_test)
test_data_accuracy = accuracy_score(status_test, feature_test_prediction)
print("Accuracy of the model on the test data is ",test_data_accuracy*100)
```

```
Accuracy of the model on the test data is  87.17948717948718
```

[24]:
```
# The accuracy scores for training and test data should be close to each other;␣
 ↪otherwise, the model may be overfitting or underfitting
```

3 XGboost Model Training

[25]:
```
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

model=XGBClassifier()
model.fit(feature_train,status_train)
XGboost_predict=model.predict(feature_test)
```

[26]:
```
# Getting accuracy score
score_XGboost = accuracy_score(status_test,XGboost_predict)
print('Accuracy Score for Linear Regression is ', score_XGboost*100)
```

```
Accuracy Score for Linear Regression is  87.17948717948718
```

4 Logistic Regression Model Training

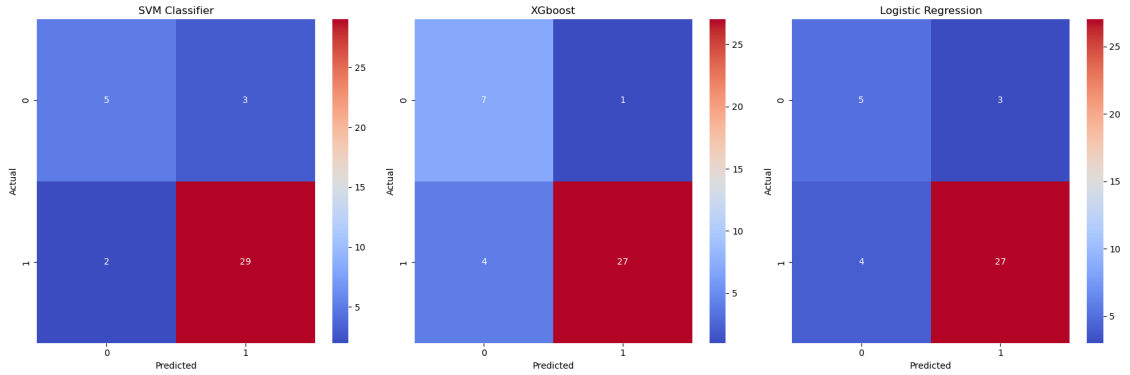
[27]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

log_reg = LogisticRegression()
log_reg.fit(feature_train,status_train)
```

[27]:
```
LogisticRegression()
```

[28]:
```
predcted_log_reg=log_reg.predict(feature_test)
# Getting accuracy score
score_log_reg = accuracy_score(status_test, predcted_log_reg)
print('Accuracy Score for Logistic Regression is ', score_log_reg*100)
```

```
Accuracy Score for Logistic Regression is  82.05128205128204
```

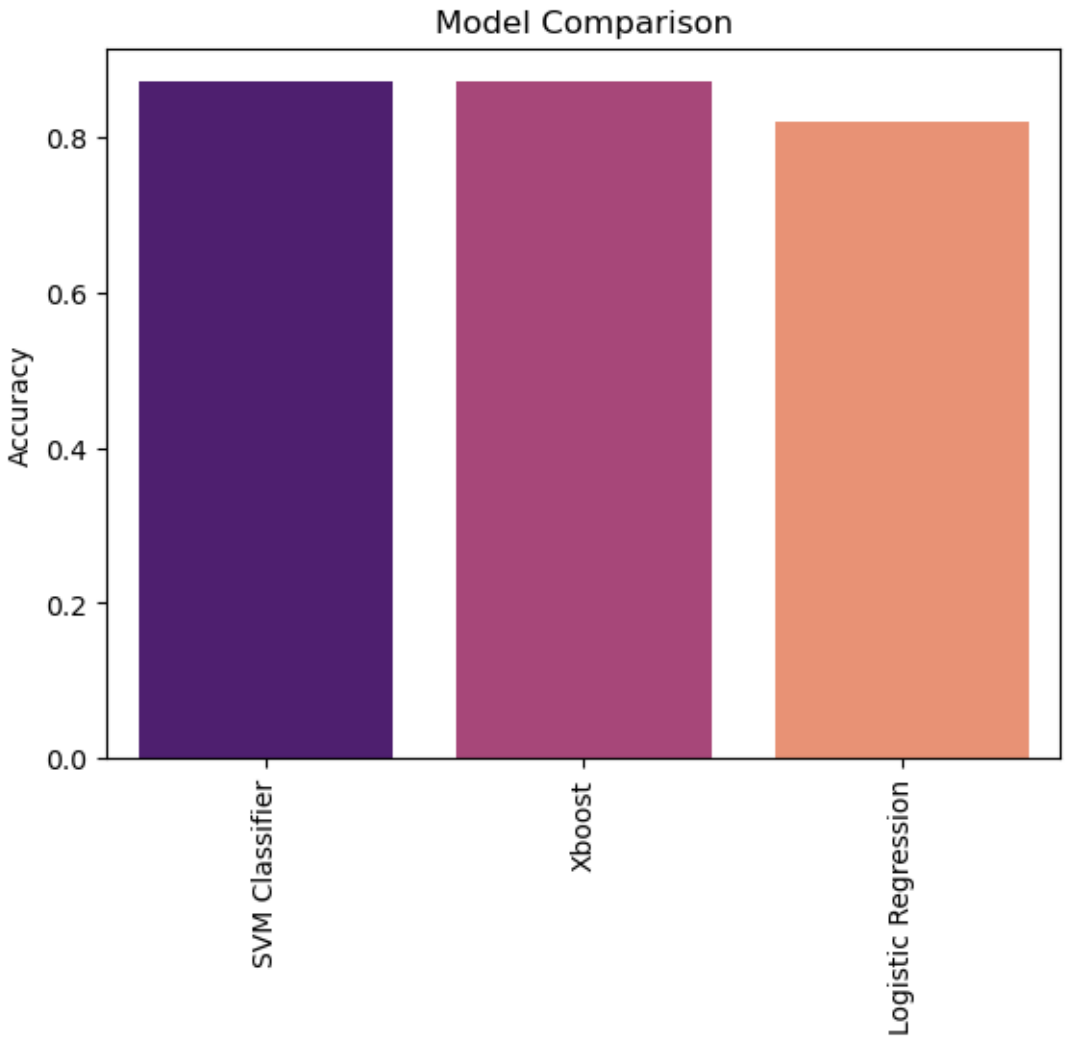
[29]:
```
# Confusion Matrix for comparing actual vs predicted values across different␣
 ↪models

# Create a 1x3 grid of subplots
fig, ax = plt.subplots(1, 3, figsize=(18, 6))

# Plot heatmaps in the 1x3 grid
sns.heatmap(confusion_matrix(status_test, feature_test_prediction), annot=True,␣
 ↪cmap='coolwarm', ax=ax[0])
ax[0].set_title('SVM Classifier')
ax[0].set_xlabel('Predicted')
ax[0].set_ylabel('Actual')

sns.heatmap(confusion_matrix(status_test, XGboost_predict), annot=True,␣
 ↪cmap='coolwarm', ax=ax[1])
ax[1].set_title('XGboost')
ax[1].set_xlabel('Predicted')
ax[1].set_ylabel('Actual')

sns.heatmap(confusion_matrix(status_test, predcted_log_reg), annot=True,␣
 ↪cmap='coolwarm', ax=ax[2])
ax[2].set_title('Logistic Regression')
ax[2].set_xlabel('Predicted')
ax[2].set_ylabel('Actual')

# Adjust layout and display the plot
plt.tight_layout()
plt.show()
```



5 Model Comparision

[30]:
```
models = ['SVM Classifier', 'Xboost', 'Logistic Regression']
accuracy = [
    accuracy_score(status_test, feature_test_prediction),
    accuracy_score(status_test, XGboost_predict),
    accuracy_score(status_test, predcted_log_reg)
]
data = pd.DataFrame({
    'Model': models,
    'Accuracy': accuracy
})

sns.barplot(x=models, y=accuracy, palette='magma',data=data, hue='Model',␣
 ↪dodge=False, legend=False).set_title('Model Comparison')
plt.xticks(rotation=90)
plt.ylabel('Accuracy')
```

[30]:
```
Text(0, 0.5, 'Accuracy')
```



[31]:
```
# use different models predicted values to compare acutal vs predicted value
pd.DataFrame({'actual':status_test,'predict':predcted_log_reg})
```

[31]:

	actual	predict
10	1	1
79	1	1
164	1	1
142	1	1
186	0	0
133	1	0
35	0	0
137	1	1
25	1	1
2	1	1

12	1	0
128	1	0
144	1	1
3	1	1
48	0	0
29	1	1
14	1	1
119	1	1
6	1	1
23	1	1
108	1	0
143	1	1
129	1	1
174	0	1
45	0	0
120	1	1
173	0	1
125	1	1
9	1	1
163	1	1
54	1	1
13	1	1
109	1	1
194	0	1
78	1	1
114	1	1
44	0	0
82	1	1
158	1	1

6 Developing a predictive system(Test the model using differen input)

[32]:
```
feature_names = [
    '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', 'RPDE', 'DFA',␣
 ↪'spread1',
    'spread2', 'D2', 'PPE'
]

# Convert input data to a pandas DataFrame with feature names
```

```
input1 = (119.99200, 157.30200, 74.99700, 0.00784, 0.00007, 0.00370, 0.00554, 0.
 ↪01109, 0.04374, 0.42600, 0.02182, 0.03130, 0.02971, 0.06545, 0.02211, 21.
 ↪03300,0.414783, 0.815285, -4.813031, 0.266482, 2.301442, 0.284654)
# input1 = (214.28900, 260.27700, 77.97300, 0.00567, 0.00003, 0.00295, 0.00317,␣
 ↪0.00885, 0.01884, 0.19000, 0.01026, 0.01161, 0.01373, 0.03078, 0.04398, 21.
 ↪20900,0.462803, 0.664357, -5.724056, 0.190667, 2.555477, 0.148569)

input1_df = pd.DataFrame([input1], columns=feature_names)

#or you can use reshape instead of dataframe and feature names
# input1_array = np.asarray(input1)
# input1_reshaped = input1_array.reshape(1,-1)

# Standardize the data using the scaler
data1 = scaler.transform(input1_df)

# Make a prediction using the model
prediction = model.predict(data1)

# Output the prediction result
print(prediction)

if prediction[0] == 0:
    print("The Person does not have Parkinson's Disease")
else:
    print("The Person has Parkinson's")
```

```
[1]
The Person has Parkinson's
```