SVM Classification

Import the Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Load the dataset

```
dataset = pd.read_csv('parkinsons_new.csv')
dataset.head()
```

	name	age	sex	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%
0	phon_R01_S01_1	50	M	119.992	157.302	74.997	0.0078
1	phon_R01_S01_2	52	F	122.400	148.650	113.819	0.0096
2	phon_R01_S01_3	54	M	116.682	131.111	111.555	0.0105
3	phon_R01_S01_4	57	F	116.676	137.871	111.366	0.0099
4	phon_R01_S01_5	59	M	116.014	141.781	110.655	0.0128

Next steps: View recommended plots

```
dataset.tail()
```

	name	age	sex	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter
190	phon_R01_S50_2	74	F	174.188	230.978	94.261	0.00
191	phon_R01_S50_3	52	F	209.516	253.017	89.488	0.00
192	phon_R01_S50_4	62	F	174.688	240.005	74.287	0.01
193	phon_R01_S50_5	78	F	198.764	396.961	74.904	0.00
194	phon_R01_S50_6	69	F	214.289	260.277	77.973	0.00

```
dataset.shape
```

```
(195, 20)
```

```
dataset.columns
```

```
Index(['name', 'age', 'sex', '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'],
      dtype='object')
```

```
1  dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   name              195 non-null    object 
 1   age               195 non-null    int64  
 2   sex               195 non-null    object 
 3   MDVP:Fo(Hz)       195 non-null    float64
 4   MDVP:Fhi(Hz)      195 non-null    float64
 5   MDVP:Flo(Hz)      195 non-null    float64
 6   MDVP:Jitter(%)    195 non-null    float64
 7   MDVP:Jitter(Abs)  195 non-null    float64
 8   MDVP:RAP          195 non-null    float64
 9   MDVP:PPQ          195 non-null    float64
 10  Jitter:DDP        195 non-null    float64
 11  MDVP:Shimmer      195 non-null    float64
 12  MDVP:Shimmer(dB)  195 non-null    float64
 13  Shimmer:APQ3      195 non-null    float64
 14  Shimmer:APQ5      195 non-null    float64
 15  MDVP:APQ          195 non-null    float64
 16  Shimmer:DDA       195 non-null    float64
 17  NHR               195 non-null    float64
 18  HNR               195 non-null    float64
 19  status            195 non-null    int64  
dtypes: float64(16), int64(2), object(2)
memory usage: 30.6+ KB
```

```
1 dataset.describe()
```

	age	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVI
count	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.0
mean	64.374359	154.228641	197.104918	116.324631	0.006220	0.000044	0.0
std	9.095051	41.390065	91.491548	43.521413	0.004848	0.000035	0.0
min	50.000000	88.333000	102.145000	65.476000	0.001680	0.000007	0.0
25%	57.000000	117.572000	134.862500	84.291000	0.003460	0.000020	0.0
50%	63.000000	148.790000	175.829000	104.315000	0.004940	0.000030	0.0
75%	72.000000	182.769000	224.205500	140.018500	0.007365	0.000060	0.0
max	79.000000	260.105000	592.030000	239.170000	0.033160	0.000260	0.0

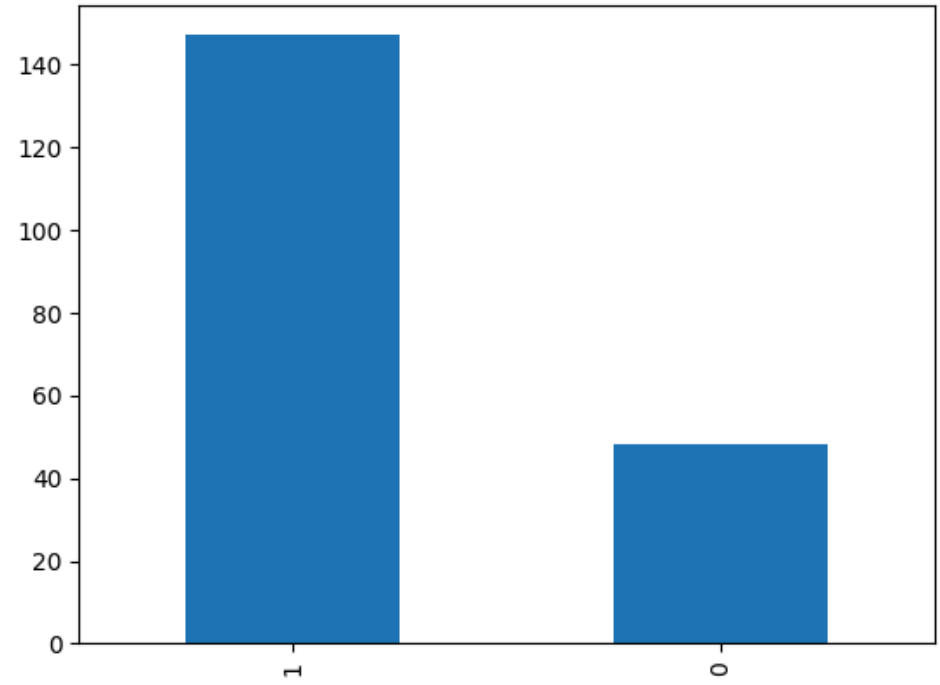
```
1 pd.isnull(dataset).sum()
```

```
name                0
age                 0
sex                 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
dtype: int64
```

```
1 classes = dataset['status'].value_counts()
2 classes
3 classes.plot.bar()
```

```
<Axes: >
```



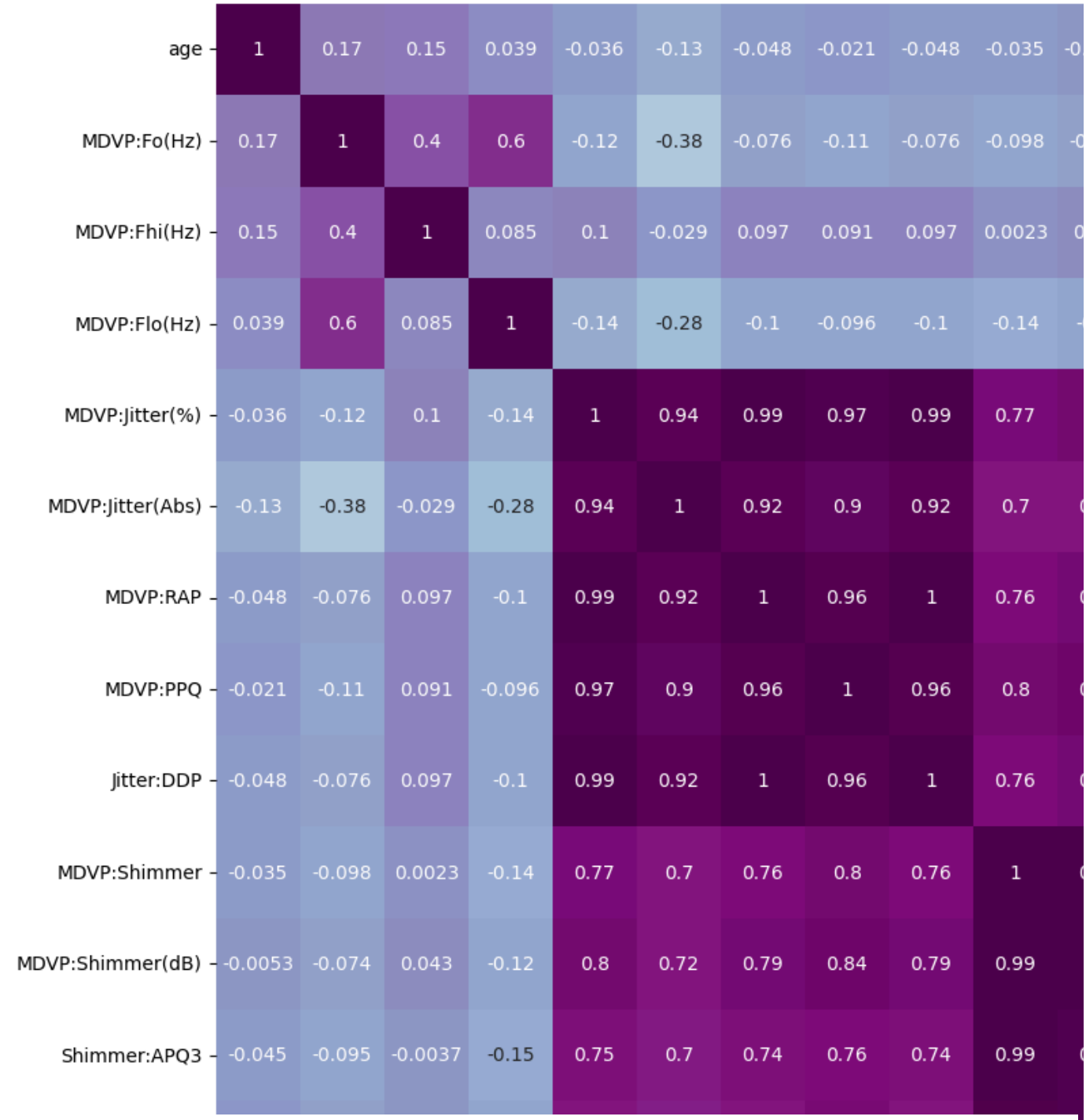

```
1 #droping a particular column, axis =1
2 dataset = dataset.drop(['name'], axis=1)
3 dataset = pd.get_dummies(dataset, prefix_sep='sex')
4 dataset.head()
```

	age	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVF
0	50	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.(
1	52	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.(
2	54	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.(
3	57	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.(
4	59	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.(

Next steps: View recommended plots

```
1 #finding correlation between the features
2
3 plt.figure(figsize=(20,17.5))
4 sns.heatmap(corr_var, annot=True, cmap='BuPu')
```

<Axes: >




```
1 X = dataset.loc[:, dataset.columns !="status"]
2 y = dataset["status"]
```

```
1 from sklearn.model_selection import train_test_split
2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state =0)
3 print ("Dimention for X train:", X_train.shape)
4 print ("Dimention for X test:", X_test.shape)
5 print ("Dimention for y train:", y_train.shape)
6 print ("Dimention for y test:", y_test.shape)
```

```
Dimention for X train: (156, 19)
Dimention for X test: (39, 19)
```

```
Dimention for y train: (156,)
Dimention for y test: (39,)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler().fit(X_train)
X_train = sc.transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn import svm
cl = svm.SVC(kernel='linear', C=0.01)
cl.fit(X_train, y_train)
```

```
▾ SVC
SVC(C=0.01, kernel='linear')
```

```
y_pred = cl.predict(X_train)
```

```
y_pred_1 = cl.predict(X_test)
y_pred_1
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

```
from sklearn.metrics import accuracy_score, confusion_matrix
cm = confusion_matrix(y_test, y_pred_1)
cm
```

```
array([[ 0, 10],
       [ 0, 29]])
```

```
acc = accuracy_score(y_test, y_pred_1)
acc
```

```
0.7435897435897436
```

```
from sklearn.model_selection import GridSearchCV
parameters = {'C': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 5000],
              'degree': [2, 3, 4, 5],
              'gamma':[0.001, 0.01, 0.1, 0.5, 1],
              'kernel': ['rbf','poly']
              }
cl = svm.SVC()
grid = GridSearchCV(cl, parameters, cv=10)
grid.fit(X_train, y_train)
print(grid.best_params_)
print(grid.best_estimator_)
```

```
{'C': 5, 'degree': 4, 'gamma': 0.1, 'kernel': 'poly'}
SVC(C=5, degree=4, gamma=0.1, kernel='poly')
```

```
from sklearn.metrics import classification_report
grid_prediction = grid.predict(X_test)
print(classification_report(y_test, grid_prediction))
```

```
                precision    recall  f1-score   support

             0       1.00      0.80      0.89        10
             1       0.94      1.00      0.97        29

      accuracy                           0.95        39
     macro avg       0.97      0.90      0.93        39
  weighted avg       0.95      0.95      0.95        39
```

SVM Regression

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import pandas as pd
```

```
1 dataset = pd.read_csv('Position_Salaries.csv')
```

```
1 dataset.head()
```

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000

Next steps: View recommended plots

```
1 dataset.tail()
```

	Position	Level	Salary
5	Region Manager	6	150000
6	Partner	7	200000
7	Senior Partner	8	300000
8	C-level	9	500000
9	CEO	10	1000000

```
1 dataset.columns
```

```
Index(['Position', 'Level', 'Salary'], dtype='object')
```

```
1 dataset.shape
```

```
(10, 3)
```

```
1 dataset.describe()
```

	Level	Salary
count	10.00000	10.000000
mean	5.50000	249500.000000
std	3.02765	299373.883668
min	1.00000	45000.000000
25%	3.25000	65000.000000
50%	5.50000	130000.000000
75%	7.75000	275000.000000
max	10.00000	1000000.000000

```
1 print(str('Any missing data or NaN in the dataset:'),dataset.isnull().values.any())
```

```
Any missing data or NaN in the dataset: False
```

```
1 dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Position  10 non-null     object
 1   Level     10 non-null     int64 
 2   Salary    10 non-null     int64 
dtypes: int64(2), object(1)
memory usage: 368.0+ bytes
```

```
1 X = dataset.iloc[:, 1:2].values
2 y = dataset.iloc[:, 2].values
```

```
1 y = y.reshape(-1,1)
```

```
1 from sklearn.preprocessing import StandardScaler
2 sc_X = StandardScaler()
3 sc_y = StandardScaler()
4 X = sc_X.fit_transform(X)
5 y = sc_y.fit_transform(y)
```

```
1 from sklearn.svm import SVR
2 regressor = SVR(kernel = 'rbf')
3 regressor.fit(X, y)
```