Documentation

|  |  |
| --- | --- |
| D:\prjectfiles\logo.jfif | Group no : \_\_\_\_\_\_\_\_\_ |
|  |  |
|  |  |

# 

# Detecting Parkinson’s Disease

# Table of Content

|  |  |
| --- | --- |
| S.no | Content |
| 1 | Table of figures |
| 2 | Abstract |
| 3 | Chapters |
| 4 | Complete Code  (attached .pynb files) |
| 5 | Conclusion |

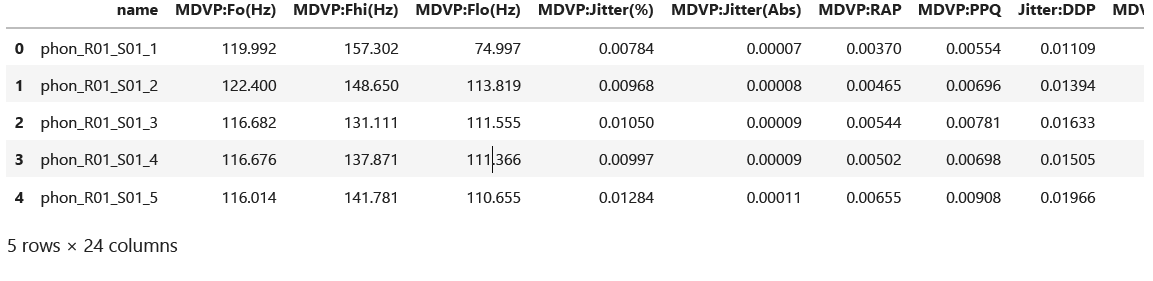
# Exploratory Data Analysis:

* path **=** "C:/Users/Aqueed/Desktop/Data.csv"

symptoms **=** pd**.**read\_csv(path)

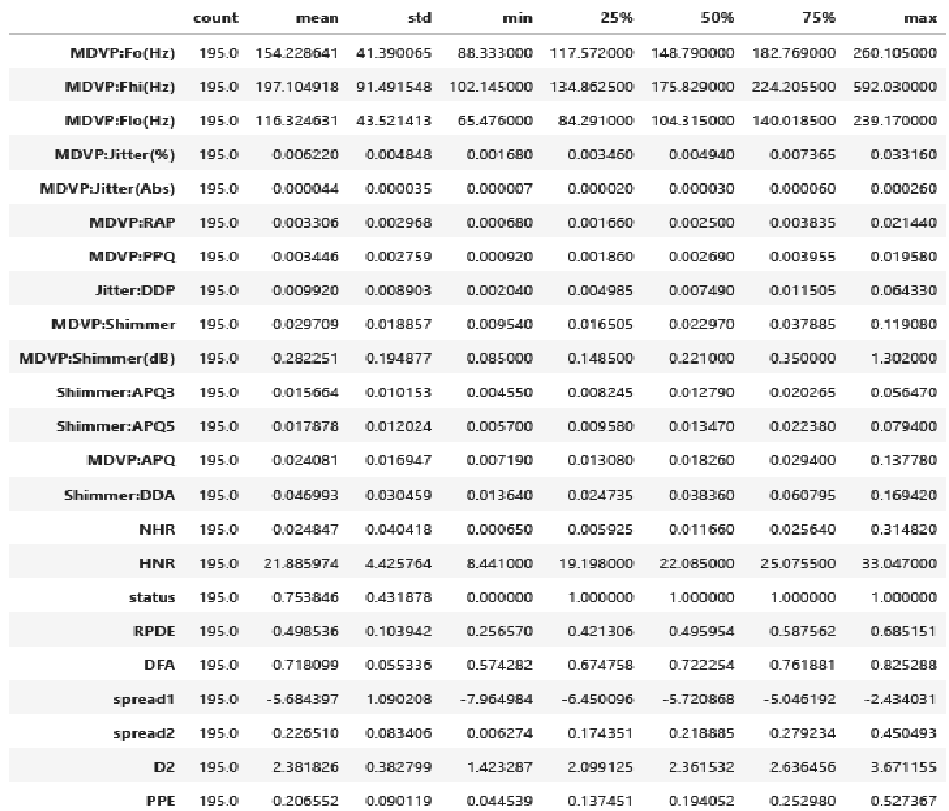
symptoms**.**head()

Output:



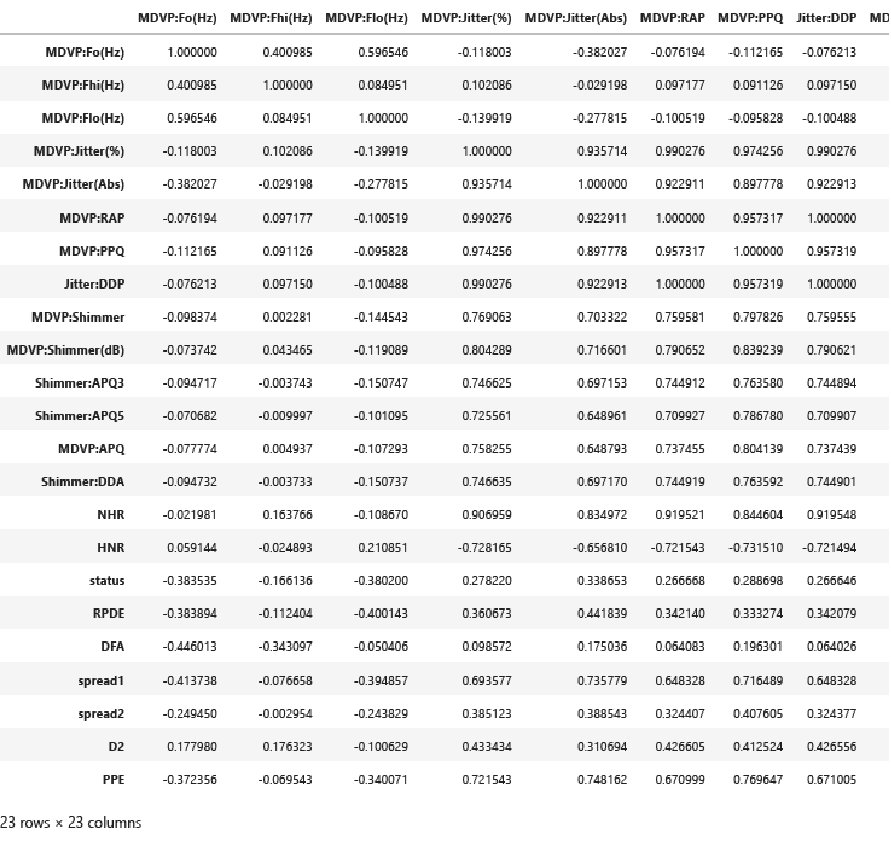
* symptoms**.**describe()**.**transpose()

Output:



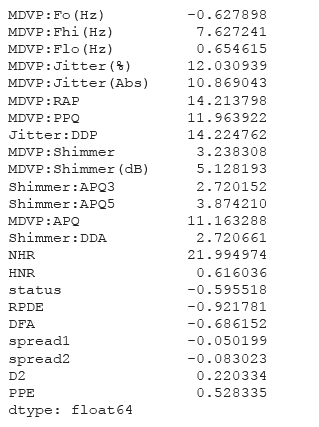
* symptoms**.**corr()

Output:



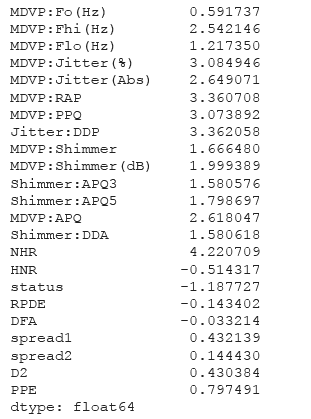
* symptoms**.**kurtosis(numeric\_only **= True**)

Output:



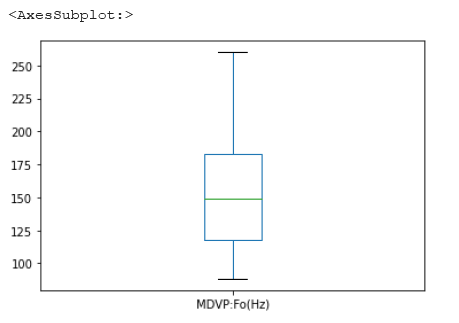
* symptoms**.**skew(numeric\_only **= True**)

Output:



* symptoms['MDVP:Fo(Hz)']**.**plot(kind**=**'box')

Output:



* print('Kurtosis : ',symptoms['MDVP:Fo(Hz)']**.**kurtosis())

print('Skewness : ',symptoms['MDVP:Fo(Hz)']**.**skew())

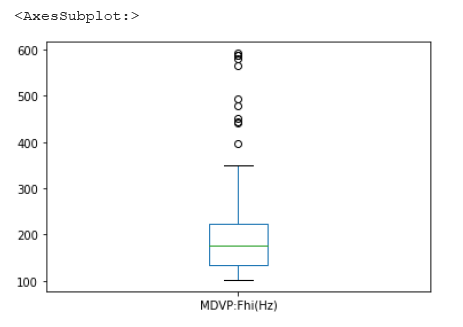
Output:

Skewness : 0.5917374636540785

Kurtosis : -0.6278981066788805

* symptoms['MDVP:Fhi(Hz)']**.**plot(kind**=**'box')

Output:



* print('Skewness : ',symptoms['MDVP:Fhi(Hz)']**.**skew())

print('Kurtosis : ',symptoms['MDVP:Fhi(Hz)']**.**kurtosis())

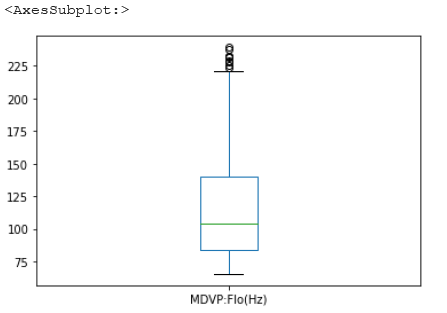
Output:

Skewness : 2.542145997588399

Kurtosis : 7.627241211631892

* symptoms['MDVP:Flo(Hz)']**.**plot(kind**=**'box')

Output:



* print('Skewness : ',symptoms['MDVP:Flo(Hz)']**.**skew())

print('Kurtosis : ',symptoms['MDVP:Flo(Hz)']**.**kurtosis())

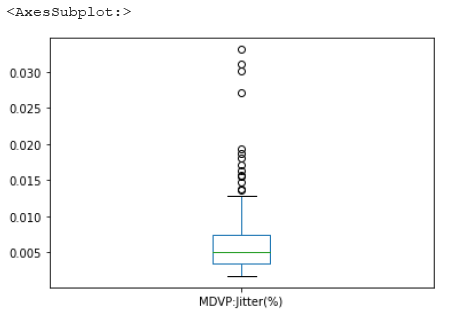
Output:

Skewness : 1.2173504486278077

Kurtosis : 0.6546145211395391

* symptoms['MDVP:Jitter(%)']**.**plot(kind**=**'box')

Output:



* print('Skewness : ',symptoms['MDVP:Jitter(%)']**.**skew())

print('Kurtosis : ',symptoms['MDVP:Jitter(%)']**.**kurtosis())

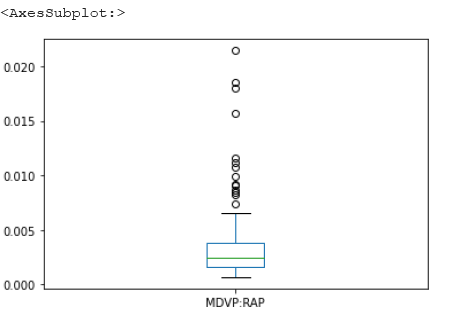
Output:

Skewness : 3.0849462014441817

Kurtosis : 12.030939276179508

* symptoms['MDVP:RAP']**.**plot(kind**=**'box')

Output:



* print('Skewness : ',symptoms['MDVP:RAP']**.**skew())

print('Kurtosis : ',symptoms['MDVP:RAP']**.**kurtosis())

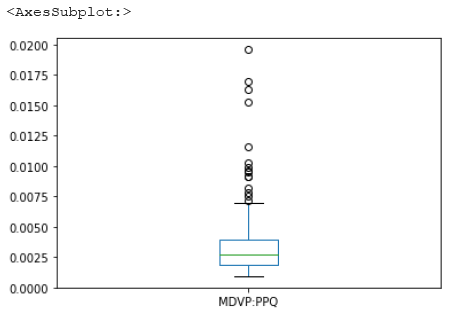
Output:

Skewness : 3.3607084504805536

Kurtosis : 14.213797721522422

* symptoms['MDVP:PPQ']**.**plot(kind**=**'box')

Output:



* print('Skewness : ',symptoms['MDVP:PPQ']**.**skew())

print('Kurtosis : ',symptoms['MDVP:PPQ']**.**kurtosis())

Output:

Skewness : 3.073892457888517

Kurtosis : 11.963922120220282

Data Cleaning

* import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

data\_set = pd.read\_csv("D:\Detecting parkinsons disease\parkinsons.data")

#MDVP:Fo(Hz)

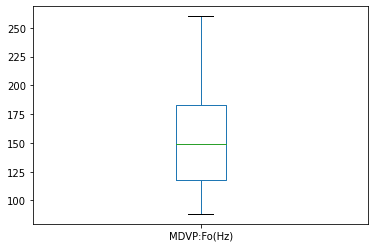
print("The average vocal fundamental frequency of a person is {:.2f} and 99% of the people have a Fo of {:.2f}"

Output:

The average vocal fundamental frequency of a person is 154.23 and 99% of the people have a Fo of 209.89

* data\_set['MDVP:Fo(Hz)'].plot(kind='box')

Output:



* print('Skewness : ',data\_set['MDVP:Fo(Hz)'].skew())

print('Kurtosis : ',data\_set['MDVP:Fo(Hz)'].kurtosis())

print(data\_set['MDVP:Fo(Hz)'].plot.hist())

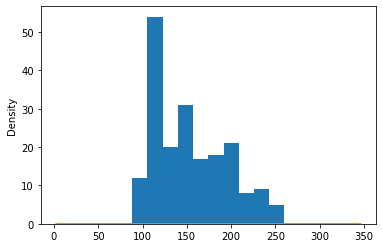
data\_set['MDVP:Fo(Hz)'].plot.kde()

Output:

Skewness : 0.5917374636540784

Kurtosis : -0.6278981066788805

AxesSubplot(0.125,0.125;0.775x0.755)



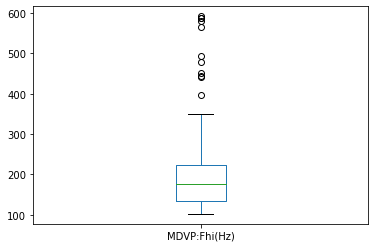
* #MDVP:Fhi (Hz)

print("The maximum vocal fundamental frequency of a person is {:.2f} and 99% of the people have a Fhi of {:.2f}".format(data\_set['MDVP:Fhi(Hz)'].mean(),data\_set['MDVP:Fhi(Hz)'].quantile(0.90)))

data\_set['MDVP:Fhi(Hz)'].plot(kind='box')

Output:

The maximum vocal fundamental frequency of a person is 197.10 and 99% of the people have a Fhi of 261.00



* print('Skewness : ',data\_set['MDVP:Fhi(Hz)'].skew())

print('Kurtosis : ',data\_set['MDVP:Fhi(Hz)'].kurtosis())

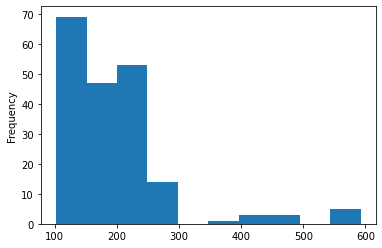
print(data\_set['MDVP:Fhi(Hz)'].plot.hist(bins=10))

Output:

Skewness : 2.542145997588398

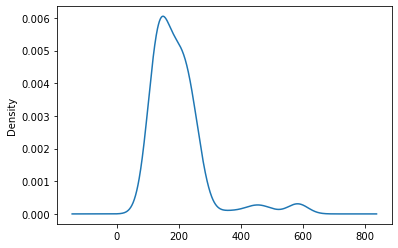
Kurtosis : 7.627241211631889

AxesSubplot(0.125,0.125;0.775x0.755)



* data\_set['MDVP:Fhi(Hz)'].plot.kde()

Output:



* q3 = data\_set['MDVP:Fhi(Hz)'].quantile(0.75)

q1 = data\_set['MDVP:Fhi(Hz)'].quantile(0.25)

t = q3-q1

outliers\_above = q3+t

outliers\_below = q1-t

print("outliers\_above : {}".format(outliers\_above))

print("outliers\_below : {}".format(outliers\_below))

Output:

outliers\_above : 313.5485

outliers\_below : 45.51950000000002

* print(data\_set['MDVP:Fhi(Hz)'].loc[data\_set['MDVP:Fhi(Hz)']>outliers\_above].count())

print(data\_set['MDVP:Fhi(Hz)'].loc[data\_set['MDVP:Fhi(Hz)']<outliers\_below].count())

print(data\_set['MDVP:Fhi(Hz)'].loc[data\_set['MDVP:Fhi(Hz)']>outliers\_above])

mean\_val = data\_set['MDVP:Fhi(Hz)'].loc[data\_set['MDVP:Fhi(Hz)']<=outliers\_above].mean()

data\_set['MDVP:Fhi(Hz)'] = data\_set['MDVP:Fhi(Hz)'].mask(data\_set['MDVP:Fhi(Hz)']>outliers\_above,mean\_val)

print(data\_set['MDVP:Fhi(Hz)'].head(20))

Output:

12

0

16 349.259

73 588.518

102 586.567

115 492.892

116 442.557

117 450.247

118 442.824

120 479.697

149 565.740

186 592.030

187 581.289

193 396.961

Name: MDVP:Fhi(Hz), dtype: float64

0 157.302000

1 148.650000

2 131.111000

3 137.871000

4 141.781000

5 131.162000

6 137.244000

7 113.840000

8 132.068000

9 120.103000

10 112.240000

11 115.871000

12 159.866000

13 179.139000

14 163.305000

15 217.455000

16 177.414634

17 232.181000

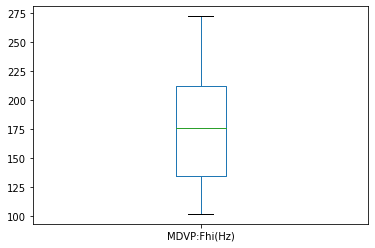
18 175.829000

19 189.398000

Name: MDVP:Fhi(Hz), dtype: float64

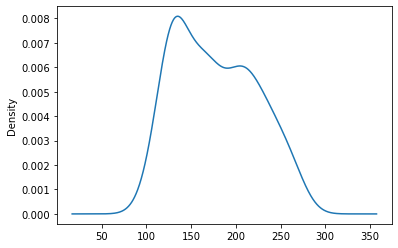
* data\_set['MDVP:Fhi(Hz)'].plot(kind='box')

Output:



* data\_set['MDVP:Fhi(Hz)'].plot.kde()

Output:



* print('Skewness : ',data\_set['MDVP:Fhi(Hz)'].skew())

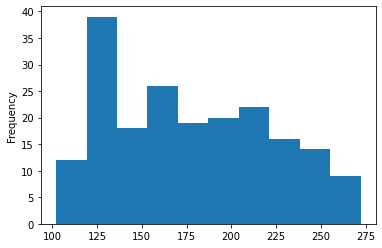
print('Kurtosis : ',data\_set['MDVP:Fhi(Hz)'].kurtosis())

data\_set['MDVP:Fhi(Hz)'].plot.hist()

Output:

Skewness : 0.2984561761401523

Kurtosis : -1.0462818430124097



* #After Treating Outliers

data\_set.kurtosis(numeric\_only = True)

Output:

MDVP:Fo(Hz) -0.627898

MDVP:Fhi(Hz) 7.627241

MDVP:Flo(Hz) -0.338062

MDVP:Jitter(%) -0.457994

MDVP:Jitter(Abs) 0.008361

MDVP:RAP -0.506298

MDVP:PPQ -0.694163

Jitter:DDP -0.508295

MDVP:Shimmer 3.238308

MDVP:Shimmer(dB) 5.128193

Shimmer:APQ3 2.720152

Shimmer:APQ5 3.874210

MDVP:APQ 11.163288

Shimmer:DDA 2.720661

NHR 21.994974

HNR 0.616036

status -0.595518

RPDE -0.921781

DFA -0.686152

spread1 -0.050199

spread2 -0.083023

D2 0.220334

PPE 0.528335

dtype: float64

* data\_set.skew(numeric\_only = True)

Output:

MDVP:Fo(Hz) 0.591737

MDVP:Fhi(Hz) 2.542146

MDVP:Flo(Hz) 0.910501

MDVP:Jitter(%) 0.713532

MDVP:Jitter(Abs) 0.730093

MDVP:RAP 0.737077

MDVP:PPQ 0.640614

Jitter:DDP 0.736060

MDVP:Shimmer 1.666480

MDVP:Shimmer(dB) 1.999389

Shimmer:APQ3 1.580576

Shimmer:APQ5 1.798697

MDVP:APQ 2.618047

Shimmer:DDA 1.580618

NHR 4.220709

HNR -0.514317

status -1.187727

RPDE -0.143402

DFA -0.033214

spread1 0.432139

spread2 0.144430

D2 0.430384

PPE 0.797491

dtype: float64

* data\_set.to\_csv('D:\Detecting parkinsons disease\cleaned-data.csv')

XGBoost

* **import** pandas **as** pd

**import** numpy **as** np

**import** plotly **as** plot

**import** plotly.express **as** px

**import** plotly.graph\_objs **as** go

**import** cufflinks **as** cf

**import** seaborn **as** sns

**import** os

**import** matplotlib.pyplot **as** plt

**from** sklearn.metrics **import** accuracy\_score

**from** plotly.offline **import** init\_notebook\_mode,plot,iplot

**import** plotly.offline **as** pyo

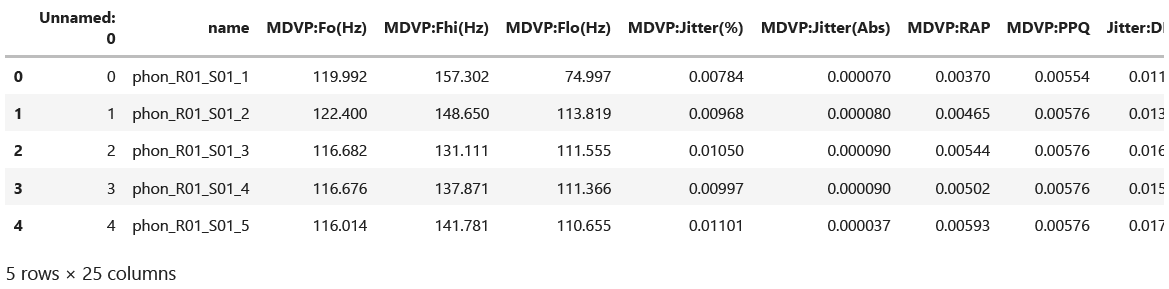
* pyo**.**init\_notebook\_mode(connected**=True**)

cf**.**go\_offline()

* symptoms**=**pd**.**read\_csv("D:\Detecting parkinsons disease\cleaned-data.csv")

symptoms**.**head()

Output:



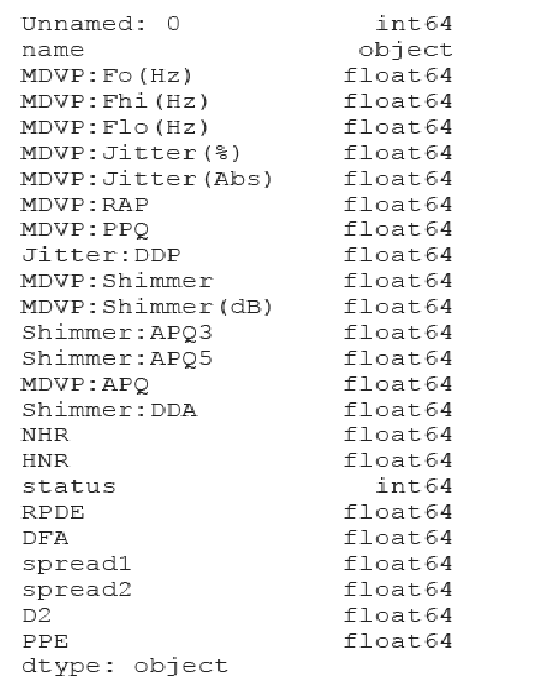
* symptoms**.**shape

Output:

(195, 25)

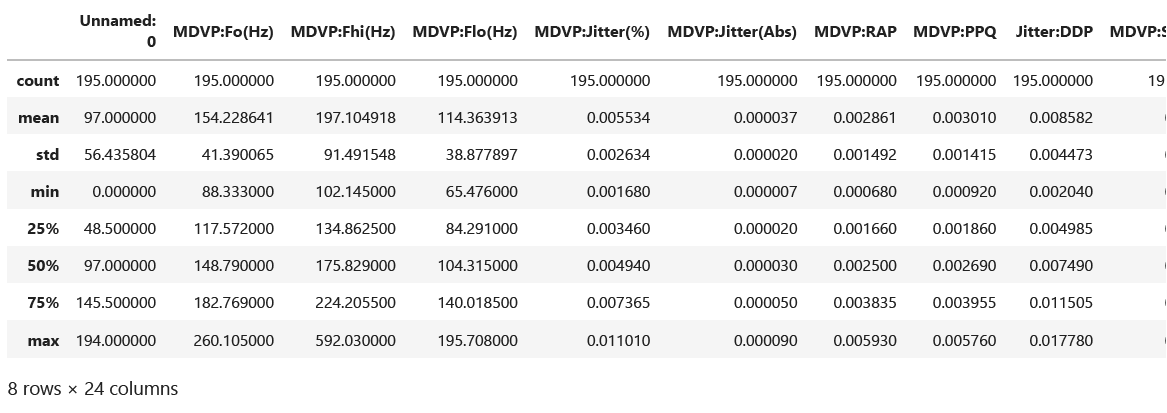
* symptoms**.**dtypes

Output:



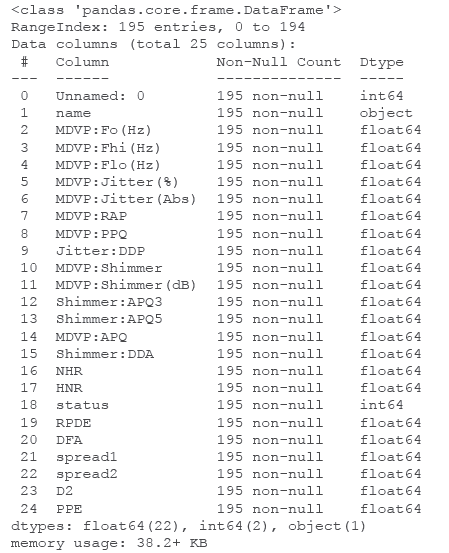
* symptoms**.**describe()

Output:



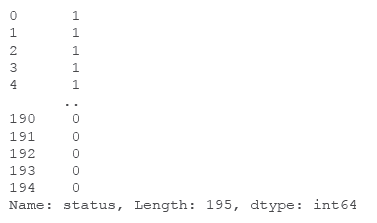
* symptoms**.**info()

Output:



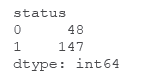
* symptoms['status']

Output:



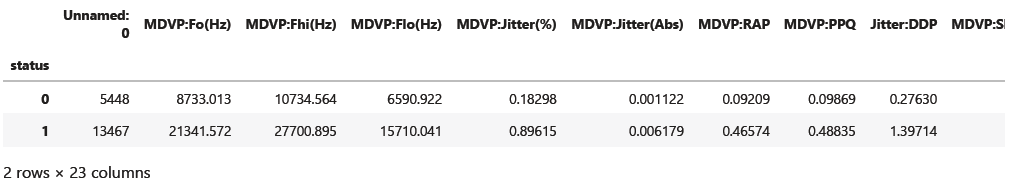
* symptoms**.**groupby('status')**.**size()

Output:



* symptoms**.**groupby('status')**.**sum()

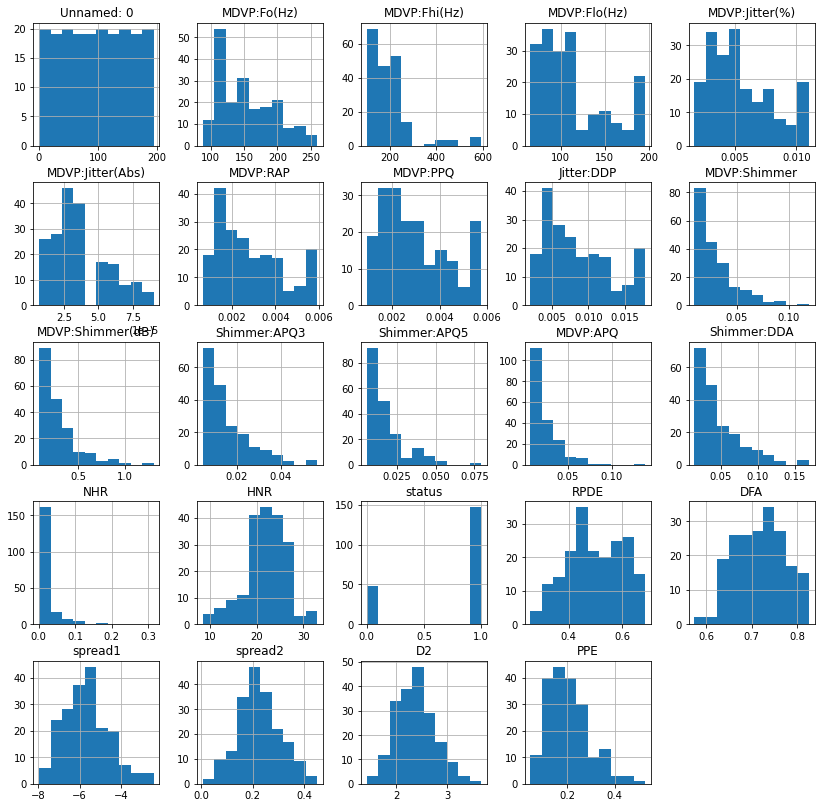
Output:



* symptoms**.**hist(figsize**=**(14,14))

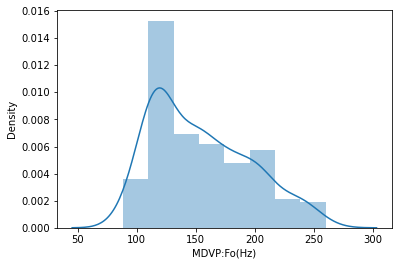
plt**.**show()

Output:



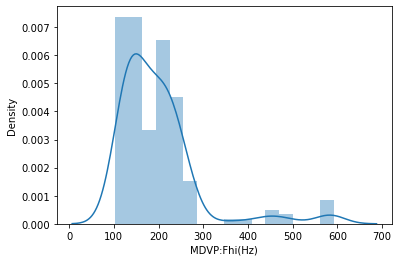
* sns**.**distplot(symptoms['MDVP:Fo(Hz)'])

Output:



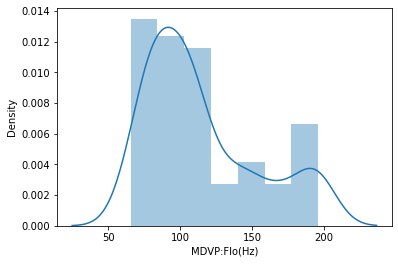
* sns.distplot(symptoms['MDVP:Fhi(Hz)'])

Output:



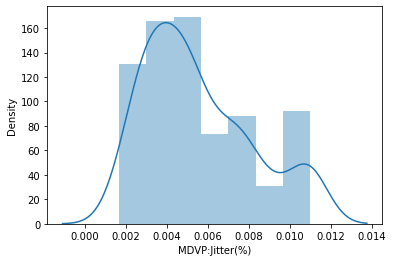
* sns.distplot(symptoms['MDVP:Flo(Hz)'])

Output:



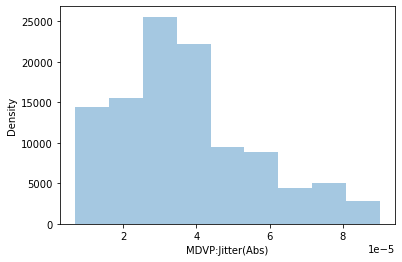
* sns.distplot(symptoms['MDVP:Jitter(%)'])

Output:



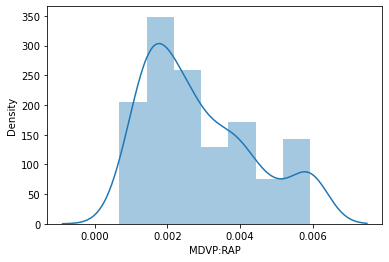
* sns.distplot(symptoms['MDVP:Jitter(Abs)'])

Output:



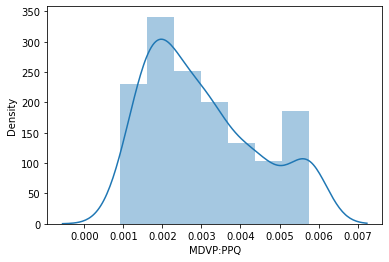
* sns.distplot(symptoms['MDVP:RAP'])

Output:



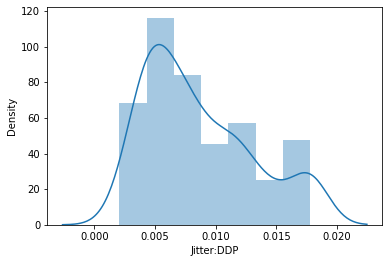
* sns.distplot(symptoms['MDVP:PPQ'])

Output:



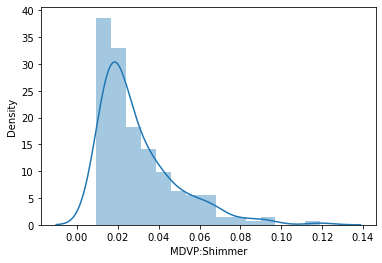
* sns.distplot(symptoms['Jitter:DDP'])

Output:



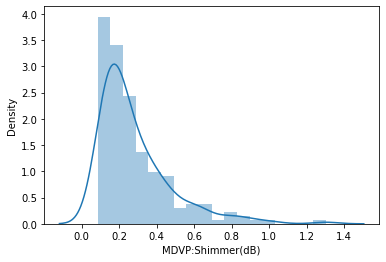
* sns.distplot(symptoms['MDVP:Shimmer'])

Output:



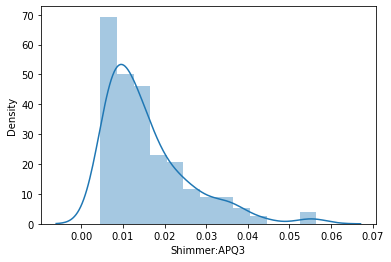
* sns.distplot(symptoms['MDVP:Shimmer(dB)'])

Output:



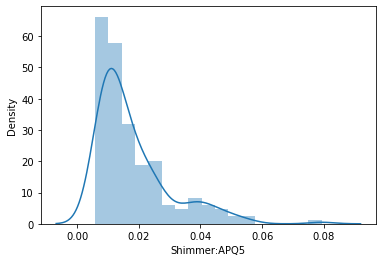
* sns.distplot(symptoms['Shimmer:APQ3'])

Output:

****

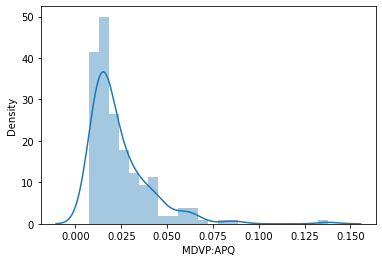
* sns.distplot(symptoms['Shimmer:APQ5'])

Output:



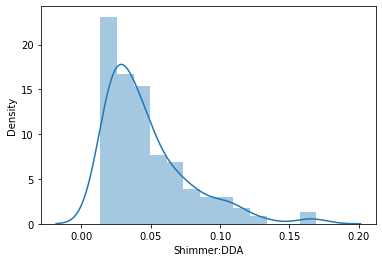
* sns.distplot(symptoms['MDVP:APQ'])

Output:



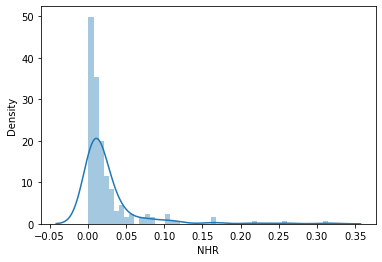
* sns.distplot(symptoms['Shimmer:DDA'])

Output:



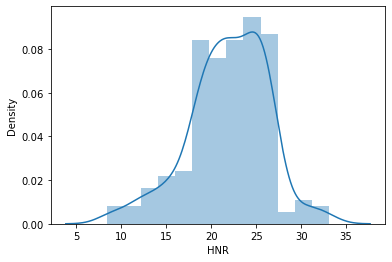
* sns.distplot(symptoms['NHR'])

Output:



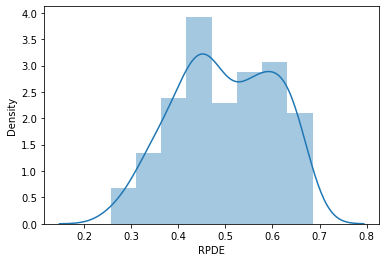
* sns.distplot(symptoms['HNR'])

Output:



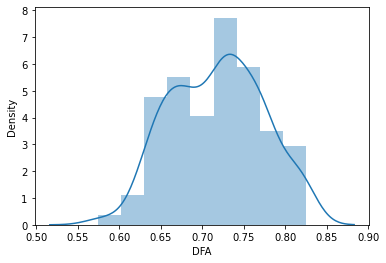
* sns.distplot(symptoms['RPDE'])

Output:



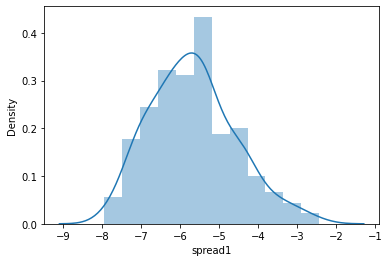
* sns.distplot(symptoms['DFA'])

Output:



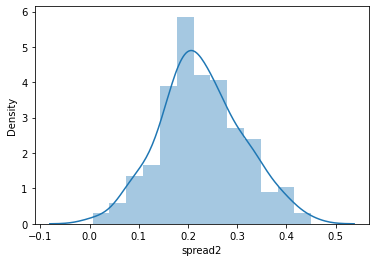
* sns.distplot(symptoms['spread1'])

Output:



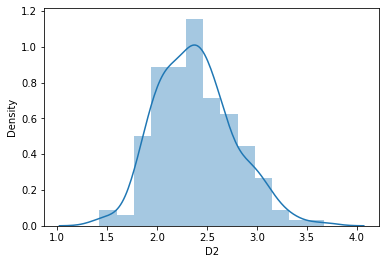
* sns.distplot(symptoms['spread2'])

Output:



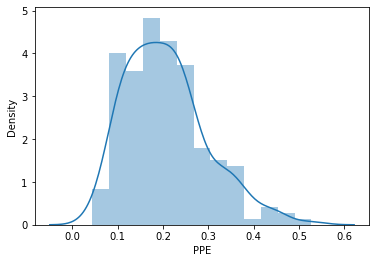
* sns.distplot(symptoms['D2'])

Output:



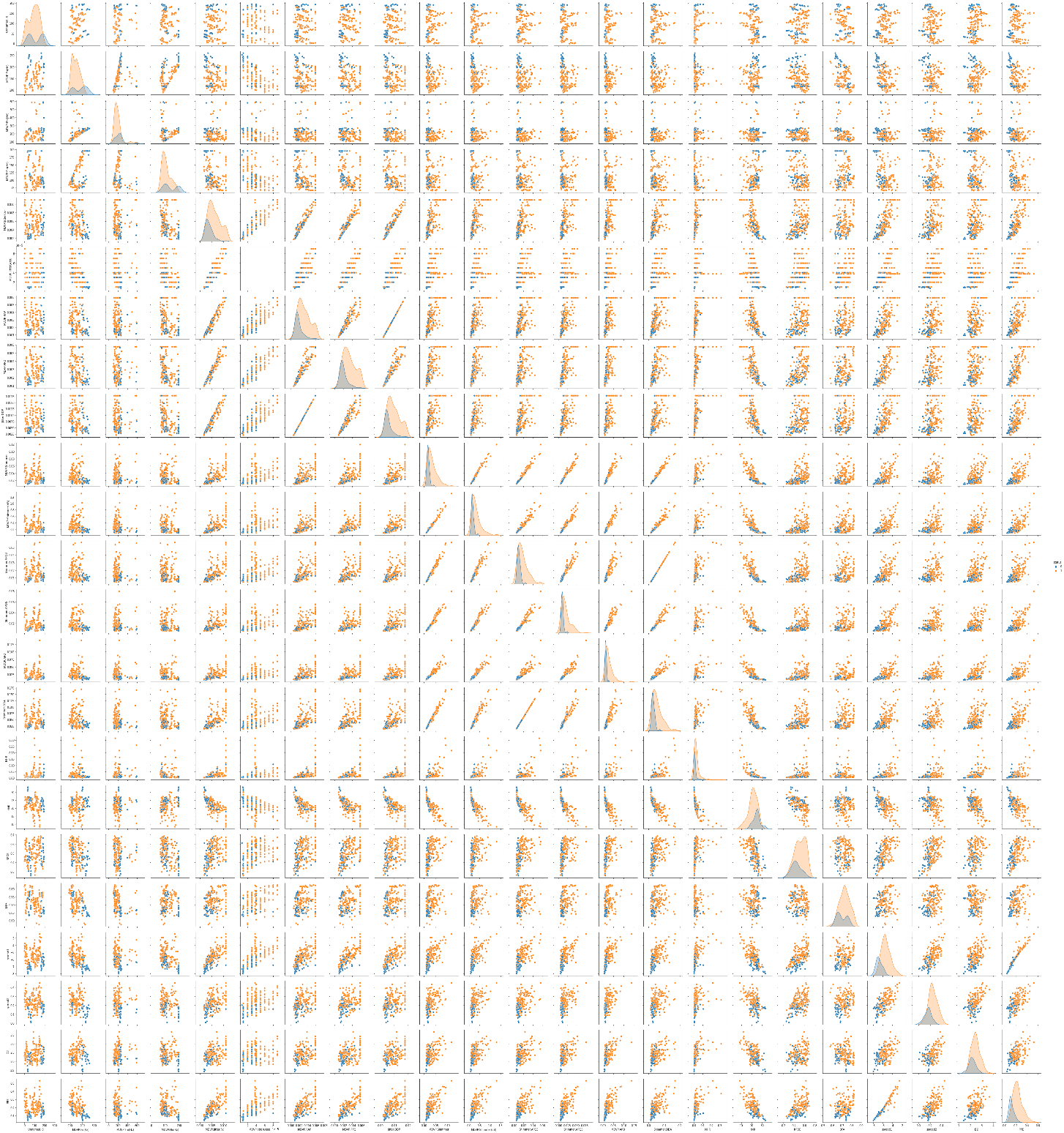
* sns.distplot(symptoms['PPE'])

Output:



* sns.pairplot(symptoms,hue='status')

Output:

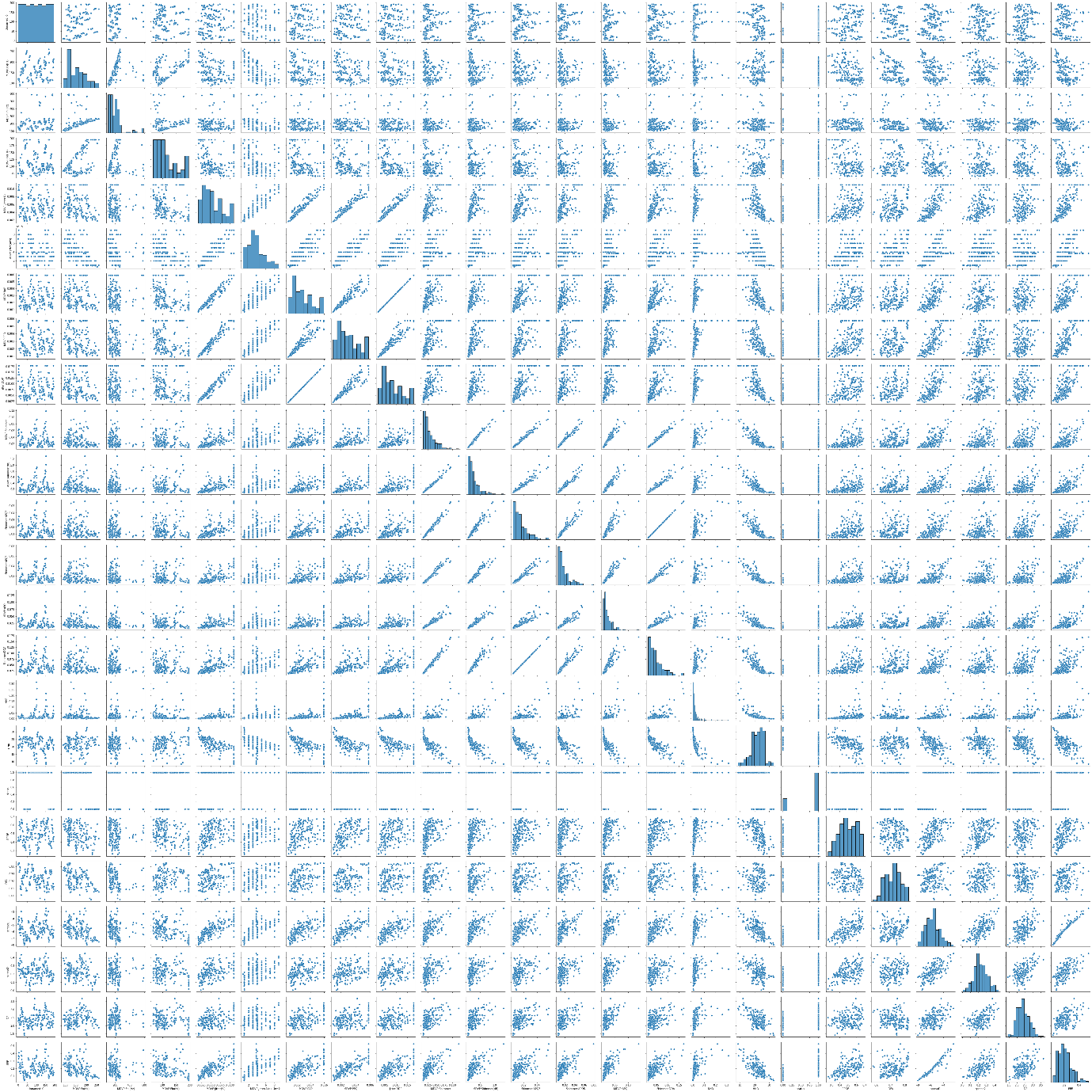


* numeric\_columns=symptoms.columns

numeric\_columns.drop(['name','status'])

sns.pairplot(symptoms[numeric\_columns])

Output:



* y = symptoms["status"]

sns.countplot(y)

status\_temp = symptoms.status.value\_counts()

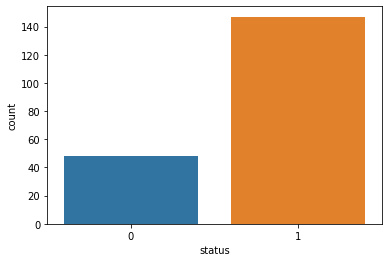
print(status\_temp)

Output:

1 147

0 48

Name: status, dtype: int64

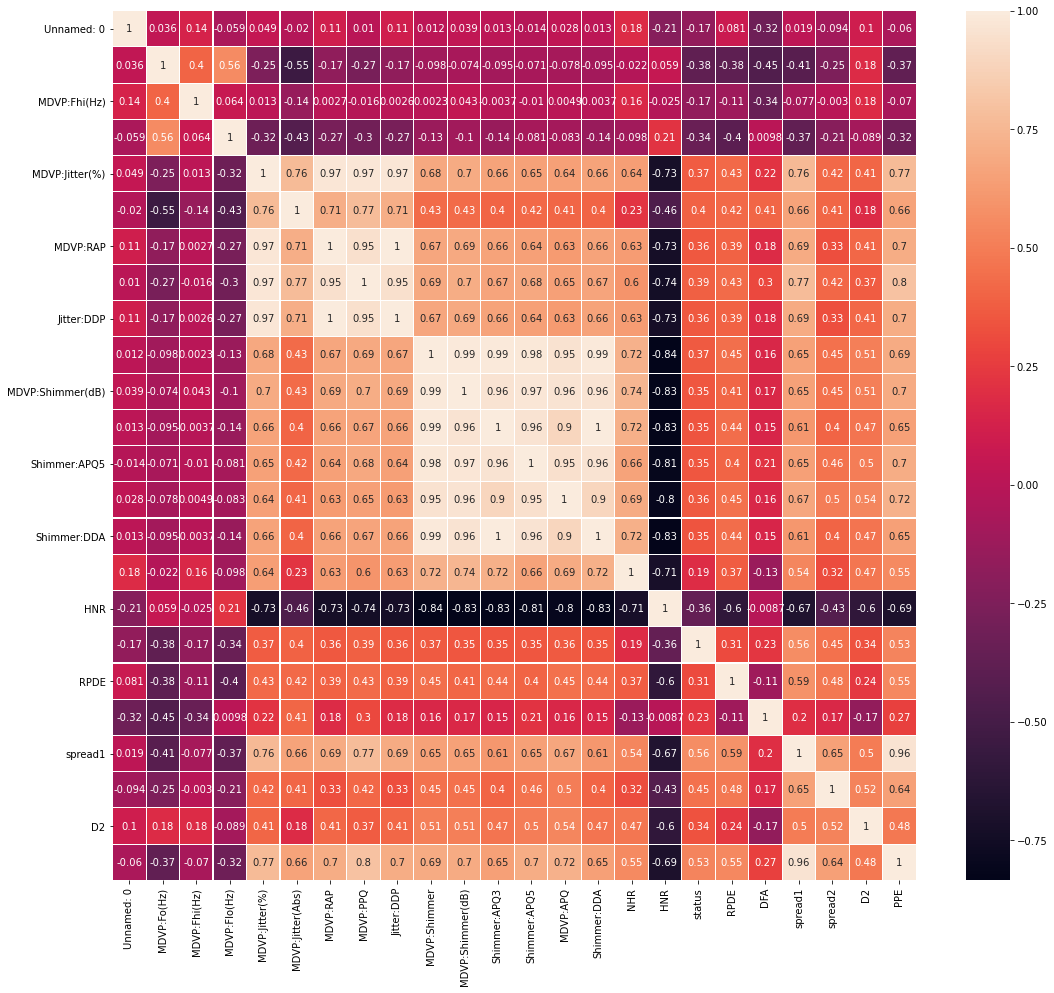


* sns.heatmap(symptoms[numeric\_columns].corr(),annot=True, linewidths=0.1)

fig=plt.gcf()

fig.set\_size\_inches(18,16)

plt.show()



* import xgboost as xgb

from sklearn.metrics import mean\_squared\_error

import pandas as pd

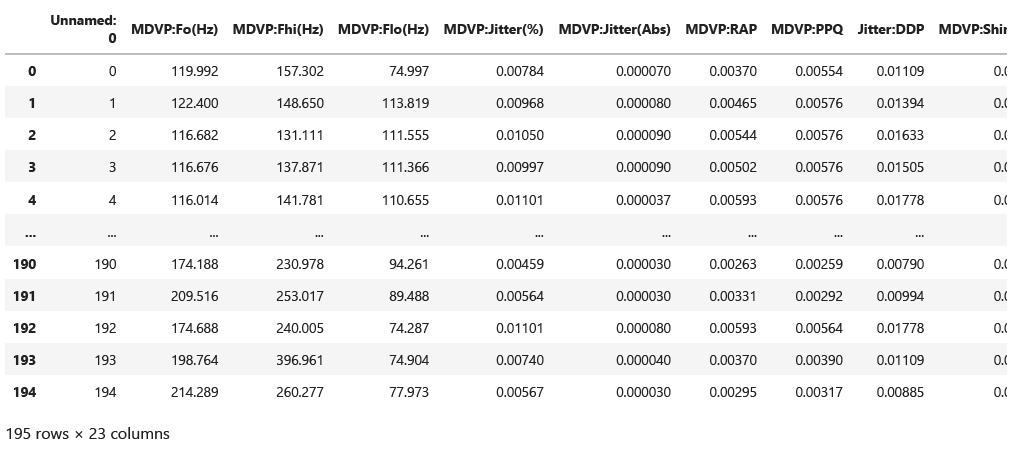
import numpy as np

* X=symptoms.drop(['status','name'],axis=1)

y=symptoms['status']

* X

Output:



* X.shape

Output:

(195, 23)

* y**.**shape

Output:

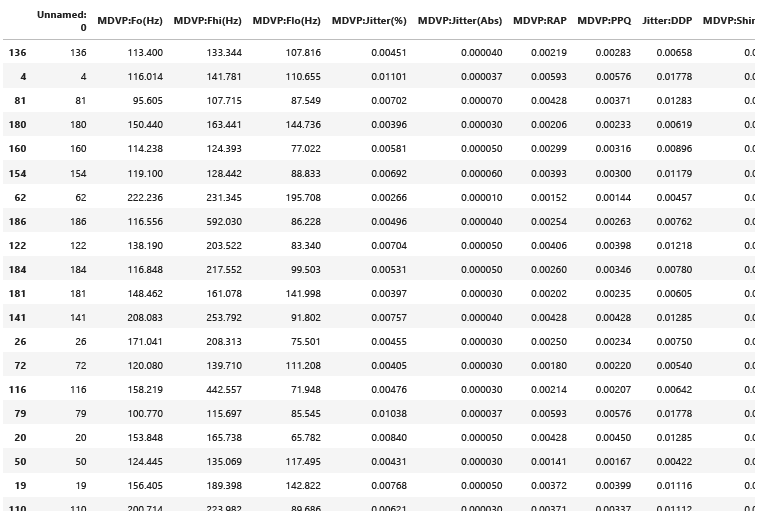
(195,)

* from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=123)

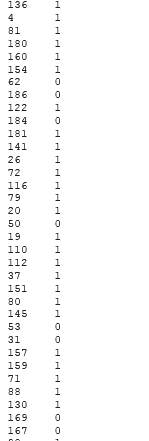
* X\_test

Output:



* y\_test

Output:



* print("train\_set\_x shape"**+**str(X\_train**.**shape))

print("train\_set\_y shape"**+**str(y\_train**.**shape))

print("test\_set\_x shape"**+**str(X\_test**.**shape))

print("test\_set\_y shape"**+**str(y\_test**.**shape))

Output:

train\_set\_x shape(156, 23)

train\_set\_y shape(156,)

test\_set\_x shape(39, 23)

test\_set\_y shape(39,)

* **from** xgboost **import** XGBClassifier

model**=**XGBClassifier()

model**.**fit(X\_train,y\_train)

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)

* **from** sklearn.metrics **import** accuracy\_score

y\_pred**=**model**.**predict(X\_test)

* print("Accuracy on training set: {:.3f}"**.**format(model**.**score(X\_train, y\_train)))

print("Accuracy on test set: {:.3f}"**.**format(model**.**score(X\_test, y\_test)))

Output:

Accuracy on training set: 1.000

Accuracy on test set: 0.974

* from sklearn.metrics import confusion\_matrix

conf\_mat= confusion\_matrix(y\_test, y\_pred)

conf\_mat

Output:

array([[ 9, 1],

[ 0, 29]], dtype=int64)

* import seaborn as sns

#Heatmap for confusionmatrix

class\_names=[0,1] # name of classes

fig, ax = plt.subplots()

tick\_marks = np.arange(len(class\_names))

plt.xticks(tick\_marks, class\_names)

plt.yticks(tick\_marks, class\_names)

# create heatmap

sns.heatmap(pd.DataFrame(conf\_mat), annot=True, cmap="YlGnBu" ,fmt='g')

ax.xaxis.set\_label\_position("top")

plt.tight\_layout()

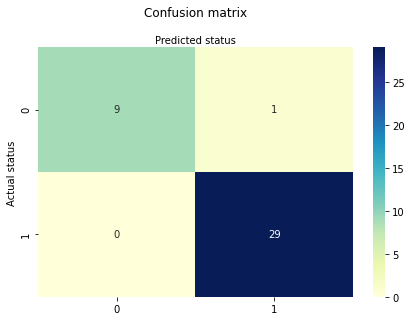
plt.title('Confusion matrix', y=1.1)

plt.ylabel('Actual status')

plt.xlabel('Predicted status')

plt.savefig("HeatMap")

Output:



* from sklearn.metrics import accuracy\_score

from sklearn import metrics

accuracy=accuracy\_score(y\_test,y\_pred)\*100

print("Accuracy\_score is :",accuracy)

print("Precision:",metrics.precision\_score(y\_test, y\_pred)\*100)

print("Recall:",metrics.recall\_score(y\_test, y\_pred)\*100)

Output:

Accuracy\_score is : 97.43589743589743

Precision: 96.66666666666667

Recall: 100.0

* #LogLoss

from sklearn.metrics import log\_loss

logLoss=log\_loss(y\_test,y\_pred)

print("Logloss: %.2f" % (logLoss))

Output:

Logloss: 0.89

* #ROC

from sklearn.metrics import roc\_auc\_score, roc\_curve

# predict probabilities

probs = model.predict\_proba(X\_test)

# keep probabilities for the positive outcome only

probs = probs[:, 1]

auc = roc\_auc\_score(y\_test, probs)

print('AUC - Test Set: %.2f%%' % (auc\*100))

# calculate roc curve

fpr, tpr, thresholds = roc\_curve(y\_test, probs)

# plot no skill

plt.plot([0, 1], [0, 1], linestyle='--')

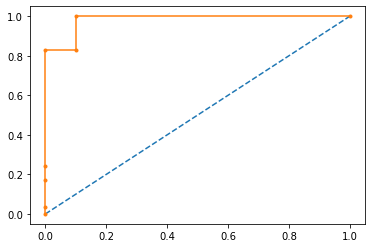
# plot the roc curve for the model

plt.plot(fpr, tpr, marker='.')

# show the plot

plt.show()

Output:



* #F score

from sklearn.metrics import f1\_score

f1 = f1\_score(y\_test, y\_pred)

print('F1 score: %f' % f1)

Output:

F1 score: 0.983051

* columns = list(symptoms.columns)

columns.remove('name')

feature\_columns = columns

feature\_columns.remove('status')

from matplotlib import pyplot

importance = model.feature\_importances\_

for i,v in enumerate(importance):

print('Feature: %s, Score: %.5f' % (feature\_columns[i],v))

pyplot.bar(feature\_columns,importance)

pyplot.xticks(rotation='vertical')

pyplot.show()

Output:

Feature: Unnamed: 0, Score: 0.12557

Feature: MDVP:Fo(Hz), Score: 0.10157

Feature: MDVP:Fhi(Hz), Score: 0.04717

Feature: MDVP:Flo(Hz), Score: 0.06134

Feature: MDVP:Jitter(%), Score: 0.13440

Feature: MDVP:Jitter(Abs), Score: 0.00411

Feature: MDVP:RAP, Score: 0.00329

Feature: MDVP:PPQ, Score: 0.00049

Feature: Jitter:DDP, Score: 0.00104

Feature: MDVP:Shimmer, Score: 0.06775

Feature: MDVP:Shimmer(dB), Score: 0.00049

Feature: Shimmer:APQ3, Score: 0.00916

Feature: Shimmer:APQ5, Score: 0.00000

Feature: MDVP:APQ, Score: 0.00000

Feature: Shimmer:DDA, Score: 0.00000

Feature: NHR, Score: 0.02202

Feature: HNR, Score: 0.00049

Feature: RPDE, Score: 0.09462

Feature: DFA, Score: 0.05433

Feature: spread1, Score: 0.07512

Feature: spread2, Score: 0.04400

Feature: D2, Score: 0.01211

Feature: PPE, Score: 0.14093

