Raisin Dataset

Presented By:- Aniket Wankhede

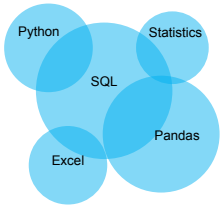
Aniket Wankhede

Aspiring Data Scientist

Education

- **B.B.A HVPM Autonomous College Amravati (2021)**
- **HSC Commerce Maharashtra State Board Pune.(2017)**
- **SSC Maharashtra State board Pune.(2015)**

Skills



Hobbies

Sketch Music Anime Gaming

Contact

Pune, Maharashtra

+7987406411

aniketwankhede2506@gmail.com

Languages

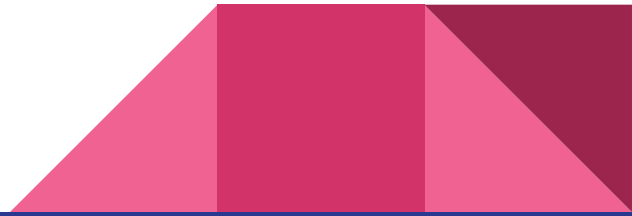
- English
- Marathi
- Hindi

Problem Statement

In this dataset we have to predict the type of raisin by applying ML algorithm

Workflow

- We have collected data from UCI
- We have perform EDA
- Apply ML algorithm
- Compare performance
- Concluded the Best ML algorithm for prediction



Information of data set

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 900 entries, 0 to 899
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Area             900 non-null    int64  
 1   MajorAxisLength  900 non-null    float64
 2   MinorAxisLength  900 non-null    float64
 3   Eccentricity     900 non-null    float64
 4   ConvexArea       900 non-null    int64  
 5   Extent           900 non-null    float64
 6   Perimeter        900 non-null    float64
 7   Class            900 non-null    object 
dtypes: float64(5), int64(2), object(1)
memory usage: 56.4+ KB
```

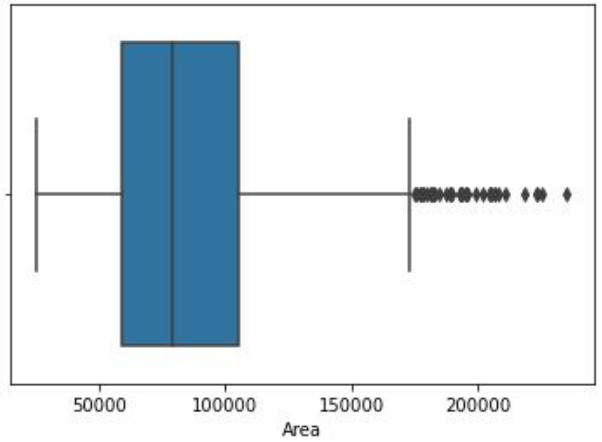
Handling Outlier

```
sns.boxplot(df['Area'])
```

```
<IPython.core.display.Javascript object>
```

```
C:\Users\DW-0622\anaconda3\lib\site-packages\seaborn
and passing other arguments without an explicit keyw
  warnings.warn(
```

```
<AxesSubplot:xlabel='Area'>
```

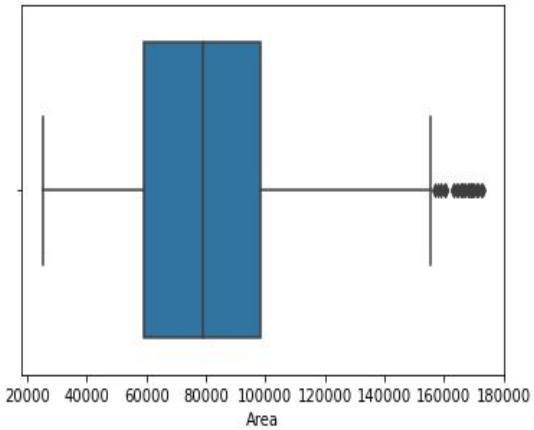



```
sns.boxplot(df['Area'])
```

```
<IPython.core.display.Javascript object>
```

```
C:\Users\DW-0622\anaconda3\lib\site-packages\seaborn\_decorators
and passing other arguments without an explicit keyword will res
  warnings.warn(
```

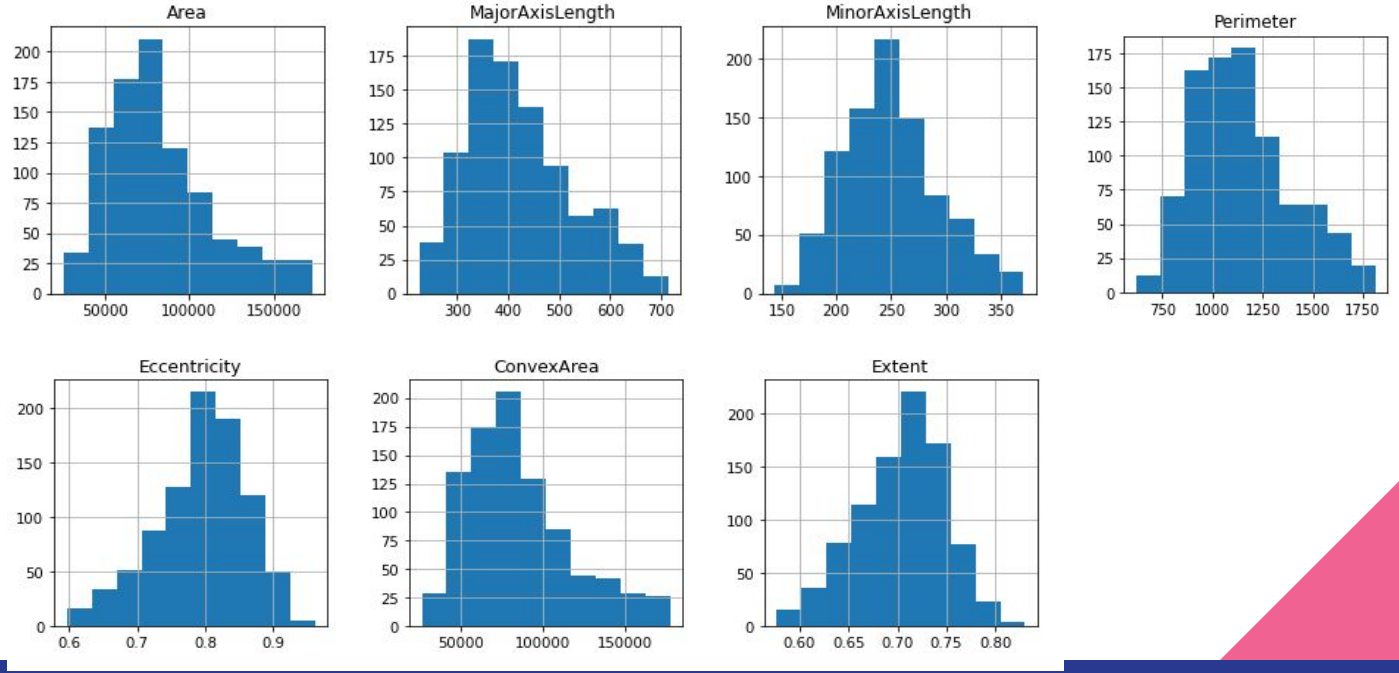
```
<AxesSubplot:xlabel='Area'>
```



We Have check the distribution of data by using histplot

```
df.hist(figsize = (12,12) , layout=(3,3))
plt.show()
```

```
<IPython.core.display.Javascript object>
```

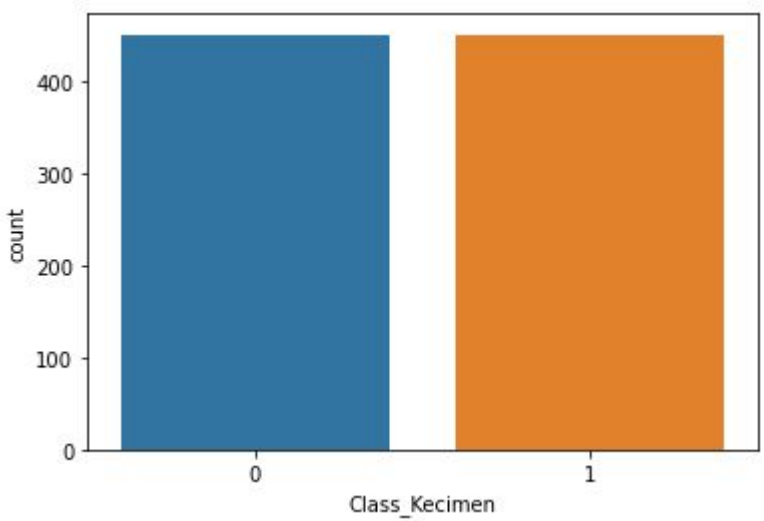


```
df1=pd.get_dummies(df,drop_first=True)
df1
```

	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	Extent	Perimeter	Class_Kecimen
0	87524	442.246011	253.291155	0.819738	90546	0.758651	1184.040	1
1	75166	406.690687	243.032436	0.801805	78789	0.684130	1121.786	1
2	90856	442.267048	266.328318	0.798354	93717	0.637613	1208.575	1
3	45928	286.540559	208.760042	0.684989	47336	0.699599	844.162	1
4	79408	352.190770	290.827533	0.798846	81463	0.792772	1073.251	1
...	...	...	...	...	...	...	...	...
895	83248	430.077308	247.838695	0.817263	85839	0.668793	1129.072	0
896	87350	440.735698	259.293149	0.808629	90899	0.636476	1214.252	0
897	99657	431.706981	298.837323	0.721684	106264	0.741099	1292.828	0
898	93523	476.344094	254.176054	0.845739	97653	0.658798	1258.548	0
899	85609	512.081774	215.271976	0.907345	89197	0.632020	1272.862	0

900 rows × 8 columns

Hence All EDA Part is done now data is ready for ML Algorithms



Converting Dataset in X and Y variable

- In X variable we have consider all the feature eliminating target
- In Y variable we have consider target column
- We have done standardization for X variable
- We have converted data into train test split
- Apply ML algorithm

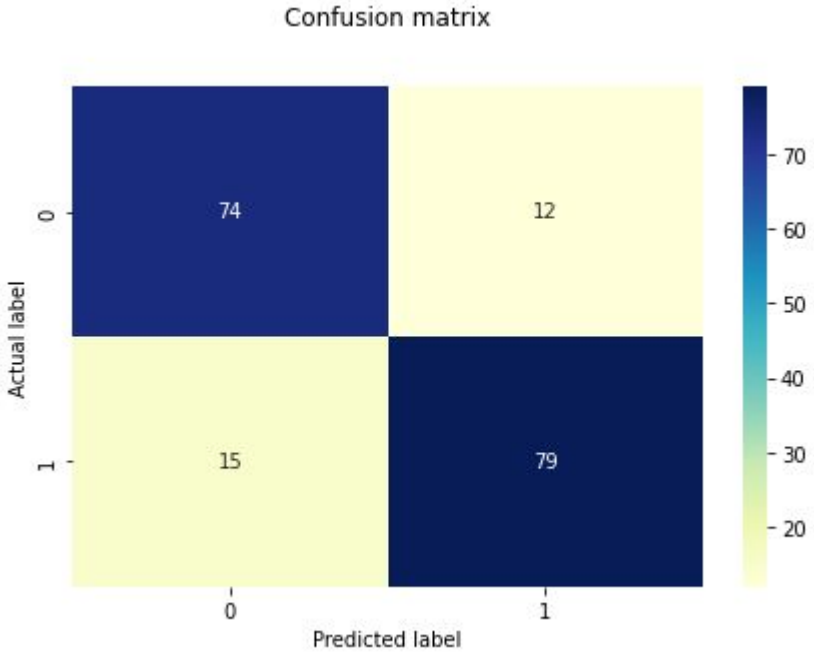
```
y_train.value_counts()
```

```
0    364
1    356
Name: Class_Kecimen, dtype: int64
```

```
y_test.value_counts()
```

```
1    94
0    86
Name: Class_Kecimen, dtype: int64
```

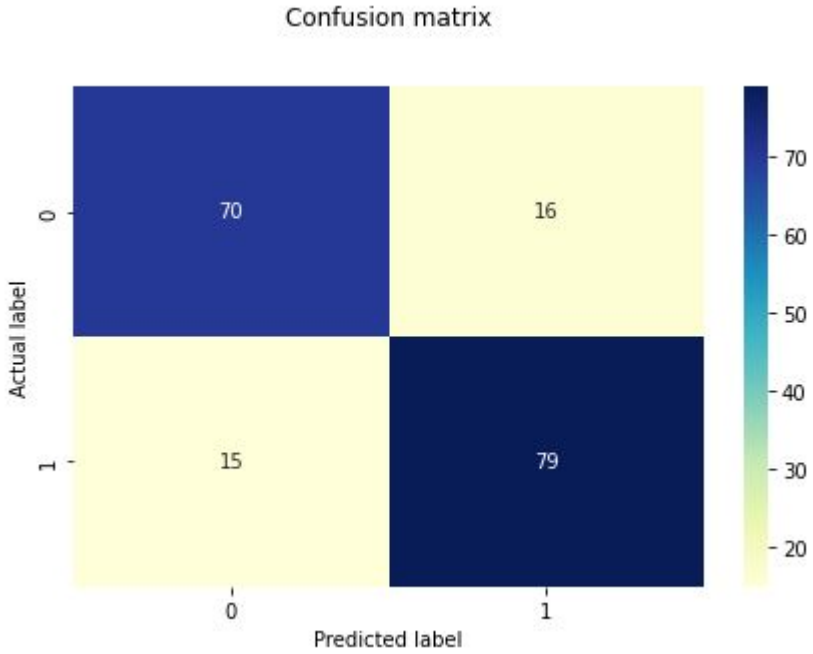
Logistic Regression



	precision	recall	f1-score	support
0	0.83	0.87	0.85	86
1	0.88	0.84	0.86	94
accuracy			0.86	180
macro avg	0.86	0.86	0.86	180
weighted avg	0.86	0.86	0.86	180

train: 0.8666666666666667
test: 0.85
roc curve: 0.8504453240969817

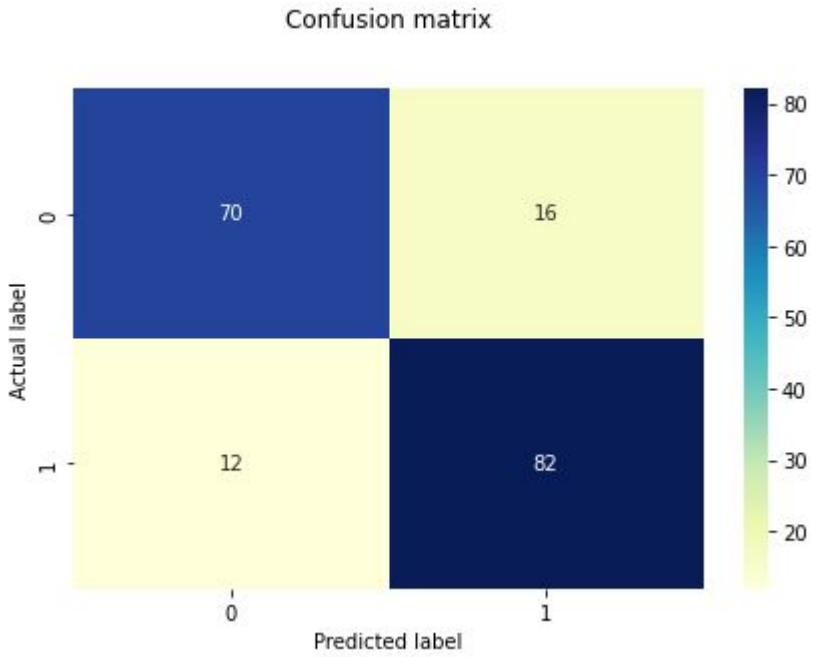
KNN



	precision	recall	f1-score	support
0	0.82	0.83	0.82	86
1	0.84	0.83	0.83	94
accuracy			0.83	180
macro avg	0.83	0.83	0.83	180
weighted avg	0.83	0.83	0.83	180

train: 0.875
test: 0.8444444444444444
roc_curve: 0.8451261751608113

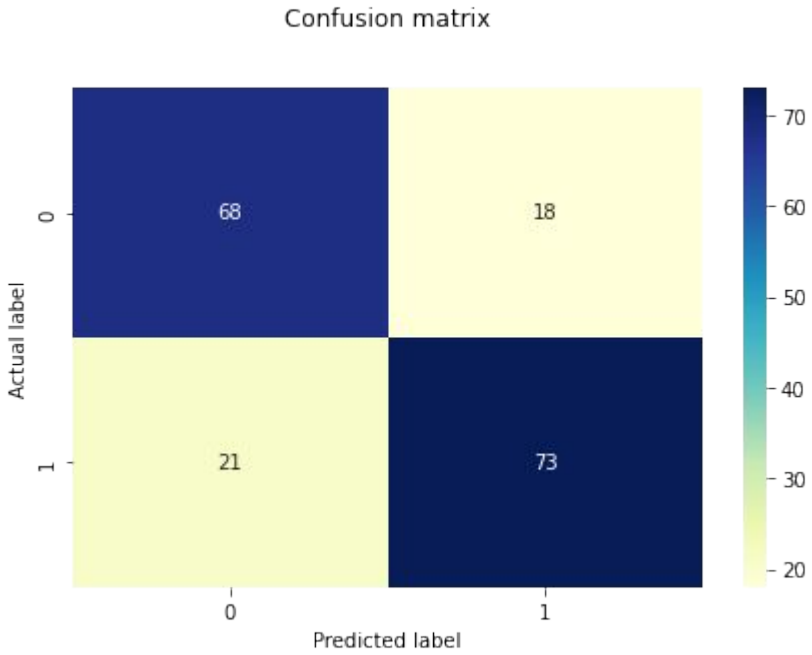
Cross Validation for KNN



	precision	recall	f1-score	support
0	0.85	0.81	0.83	86
1	0.84	0.87	0.85	94
accuracy			0.84	180
macro avg	0.85	0.84	0.84	180
weighted avg	0.84	0.84	0.84	180

train: 0.875
test: 0.8444444444444444
roc_curve: 0.8451261751608113

Decision Tree



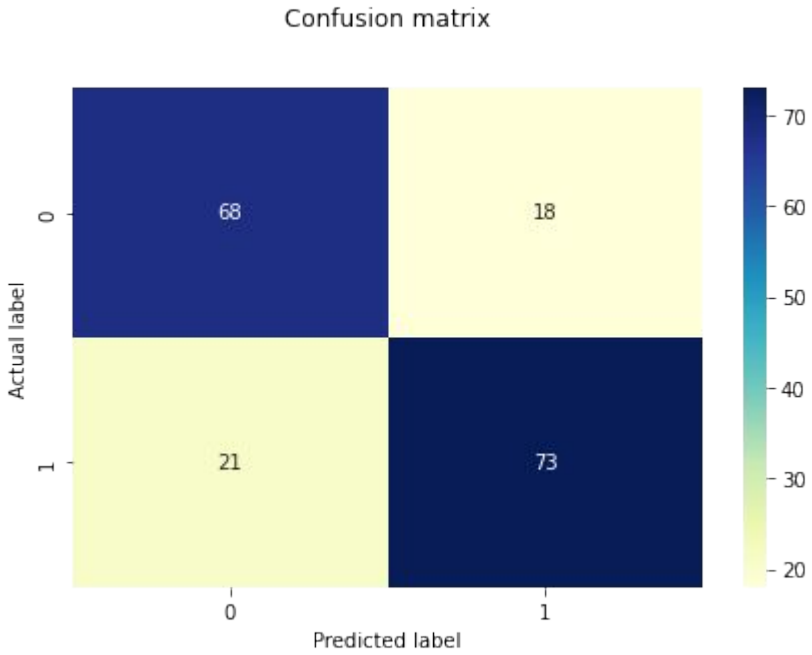
	precision	recall	f1-score	support
0	0.79	0.81	0.80	86
1	0.82	0.80	0.81	94
accuracy			0.81	180
macro avg	0.81	0.81	0.81	180
weighted avg	0.81	0.81	0.81	180

train: 0.875
test: 0.8444444444444444
roc_curve: 0.8451261751608113

Cross Validation for DT

```
grid_model.best_params_
```

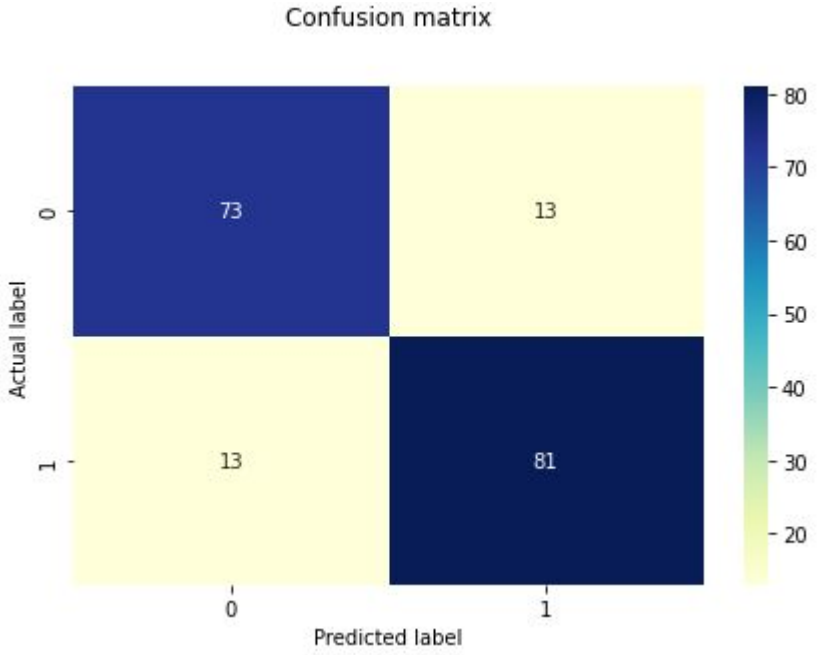
```
{'criterion': 'gini', 'max_depth': 4, 'min_samples_split': 4}
```




```
train: 0.9013888888888889
test: 0.8388888888888889
roc_curve: 0.8383226125680356
```

	precision	recall	f1-score	support
0	0.76	0.79	0.78	86
1	0.80	0.78	0.79	94
accuracy			0.78	180
macro avg	0.78	0.78	0.78	180
weighted avg	0.78	0.78	0.78	180

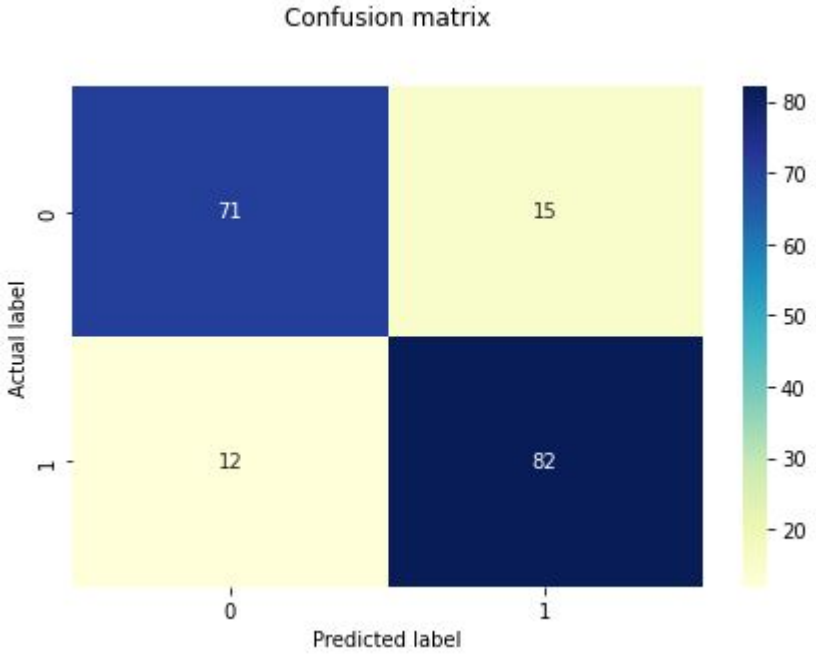
Random Classifier



	precision	recall	f1-score	support
0	0.85	0.85	0.85	86
1	0.86	0.86	0.86	94
accuracy			0.86	180
macro avg	0.86	0.86	0.86	180
weighted avg	0.86	0.86	0.86	180

train: 1.0
test: 0.8555555555555555
roc_curve: 0.85526966848095

Support vector classifier



	precision	recall	f1-score	support
0	0.86	0.83	0.84	86
1	0.85	0.87	0.86	94
accuracy			0.85	180
macro avg	0.85	0.85	0.85	180
weighted avg	0.85	0.85	0.85	180

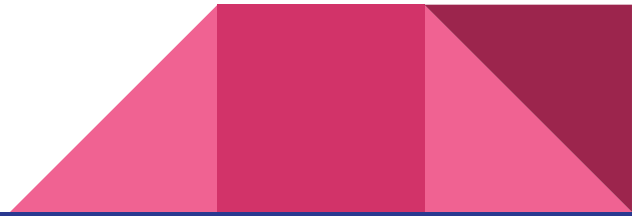
train: 0.875
test: 0.85
roc_curve: 0.8489609104403761

Comparison of Performance measure

	Models	test	traing	roc_curve
0	Logistic	0.85	0.86	0.85
1	KNN	0.84	0.87	0.85
2	Cross_valid_KNN	0.84	0.87	0.84
3	Decision Tree	0.84	0.87	0.84
4	Cross_valid_DT	0.83	0.90	0.83
5	Random Forest	0.85	1.00	0.85
6	SVM	0.85	0.87	0.84

Conclusion

After comparing all the performance measure of all the algorithm we get the best result by logistic regression. Hence for further prediction we will use logistic regression.



THANK YOU