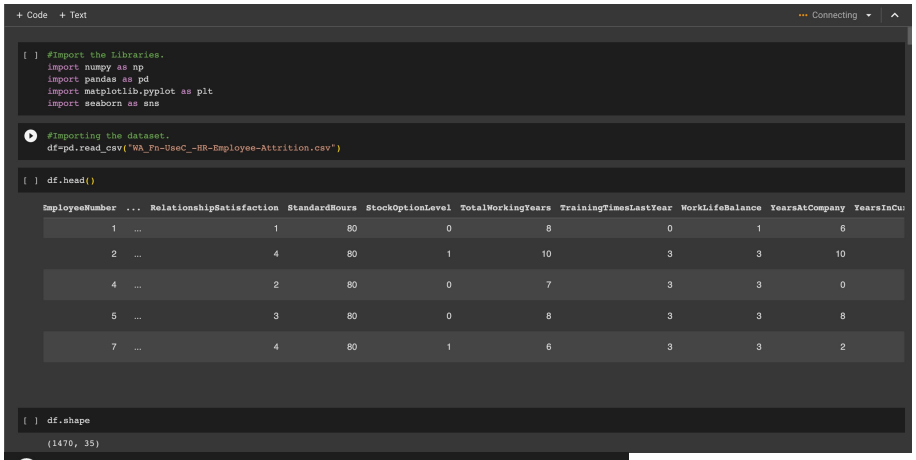
Sourav Samaddar
21MIS1046
Sourav.samaddar2021@vitstudent.ac.in

Assignment-22-09-2023

1.Download the Employee Attrition Dataset
https://www.kaggle.com/datasets/patelprashant/employee-attrition
2.Perfrom Data Preprocessing
3.Model Building using Logistic Regression and Decision Tree and Random Forest
4.Calculate Performance metrics



```
#Import the Libraries.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Importing the dataset.
df=pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")
```

```
df.head()
```

EmployeeNumber	...	RelationshipSatisfaction	StandardHours	StockOptionLevel	TotalWorkingYears	TrainingTimesLastYear	WorkLifeBalance	YearsAtCompany	YearsInCu
1	...	1	80	0	8	0	1	6	
2	...	4	80	1	10	3	3	10	
4	...	2	80	0	7	3	3	0	
5	...	3	80	0	8	3	3	8	
7	...	4	80	1	6	3	3	2	

```
df.shape
```

```
(1470, 35)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
```

```
[6] df.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	...	Rel
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	...	
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.729932	2.063946	...	
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.711561	1.106940	...	
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.000000	1.000000	...	
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.000000	1.000000	...	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000	2.000000	...	
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000	3.000000	...	
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000	5.000000	...	

8 rows × 26 columns

```
#Checking for Null Values.
df.isnull().any()
```

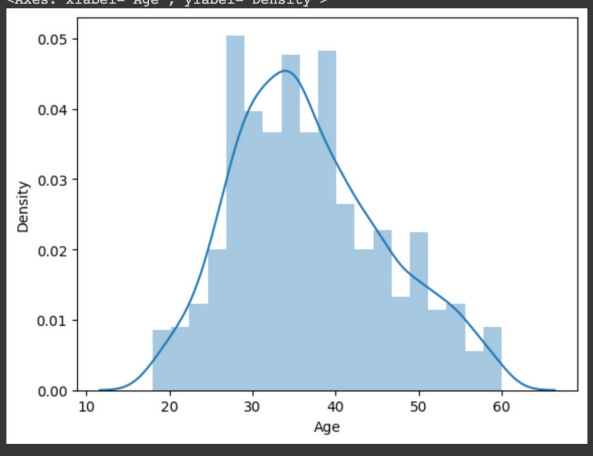
```
Age                         False
Attrition                   False
BusinessTravel              False
DailyRate                   False
Department                  False
DistanceFromHome            False
Education                   False
EducationField              False
EmployeeCount               False
EmployeeNumber              False
EnvironmentSatisfaction     False
Gender                      False
HourlyRate                  False
JobInvolvement              False
JobLevel                    False
JobRole                     False
JobSatisfaction             False
MaritalStatus               False
MonthlyIncome               False
MonthlyRate                 False
NumCompaniesWorked          False
Over18                      False
OverTime                    False
PercentSalaryHike           False
PerformanceRating           False
RelationshipSatisfaction    False
StandardHours               False
StockOptionLevel            False
TotalWorkingYears           False
TrainingTimesLastYear       False
WorkLifeBalance             False
YearsAtCompany              False
YearsInCurrentRole          False
YearsSinceLastPromotion     False
YearsWithCurrManager        False
dtype: bool
```

```
df.isnull().sum()
```

```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

```
#Data Visualization.
sns.distplot(df["Age"])
```

```
<ipython-input-9-25fc8198007f>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Age"])
<Axes: xlabel='Age', ylabel='Density'>
```



```
df.corr()
```

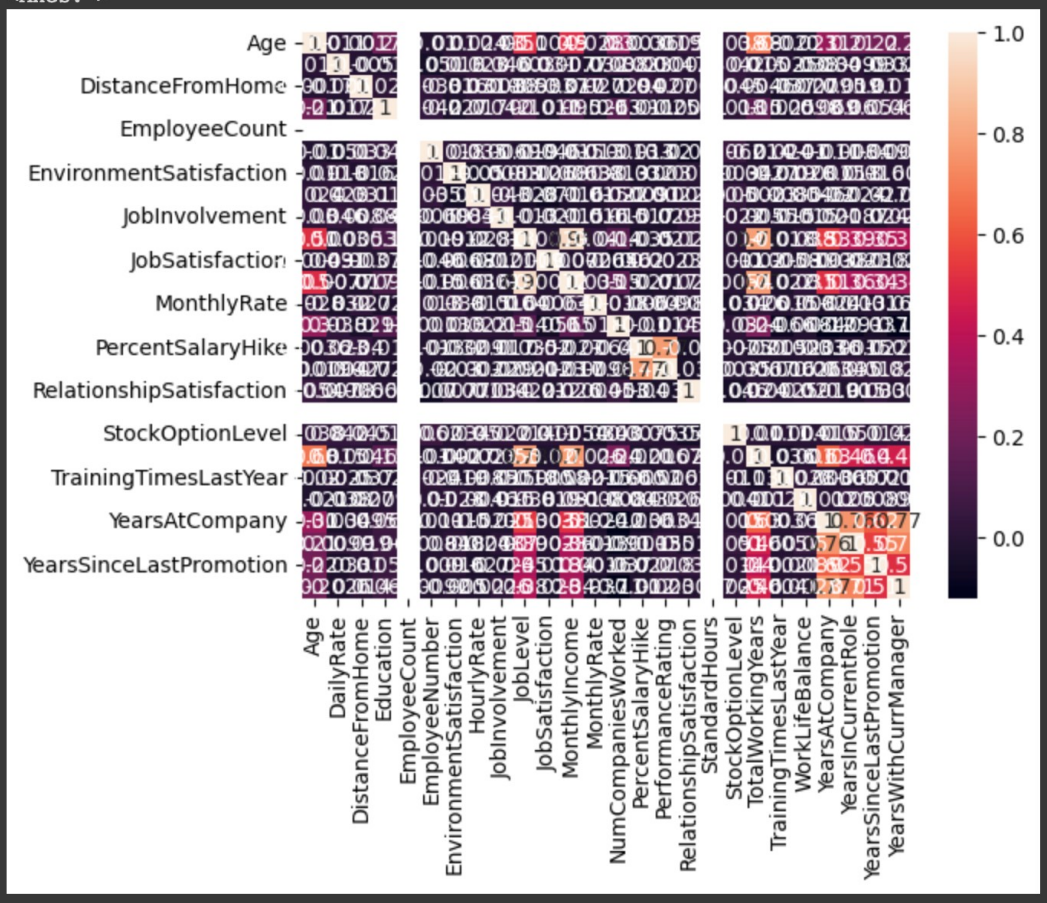
```
<ipython-input-10-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future
  df.corr()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	Hou
Age	1.000000	0.010661	-0.001686	0.208034	NaN	-0.010145	0.010146	
DailyRate	0.010661	1.000000	-0.004985	-0.016806	NaN	-0.050990	0.018355	
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	NaN	0.032916	-0.016075	
Education	0.208034	-0.016806	0.021042	1.000000	NaN	0.042070	-0.027128	
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
EmployeeNumber	-0.010145	-0.050990	0.032916	0.042070	NaN	1.000000	0.017621	
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	NaN	0.017621	1.000000	
HourlyRate	0.024287	0.023381	0.031131	0.016775	NaN	0.035179	-0.049857	
JobInvolvement	0.029820	0.046135	0.008783	0.042438	NaN	-0.006888	-0.008278	
JobLevel	0.509604	0.002966	0.005303	0.101589	NaN	-0.018519	0.001212	
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	NaN	-0.046247	-0.006784	
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	NaN	-0.014829	-0.006259	
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	NaN	0.012648	0.037600	
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	NaN	-0.001251	0.012594	
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	NaN	-0.012944	-0.031701	
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	NaN	-0.020359	-0.029548	
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	NaN	-0.069861	0.007665	
StandardHours	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	NaN	0.062227	0.003432	

Connected to Python 3 Google Compute Engine backend

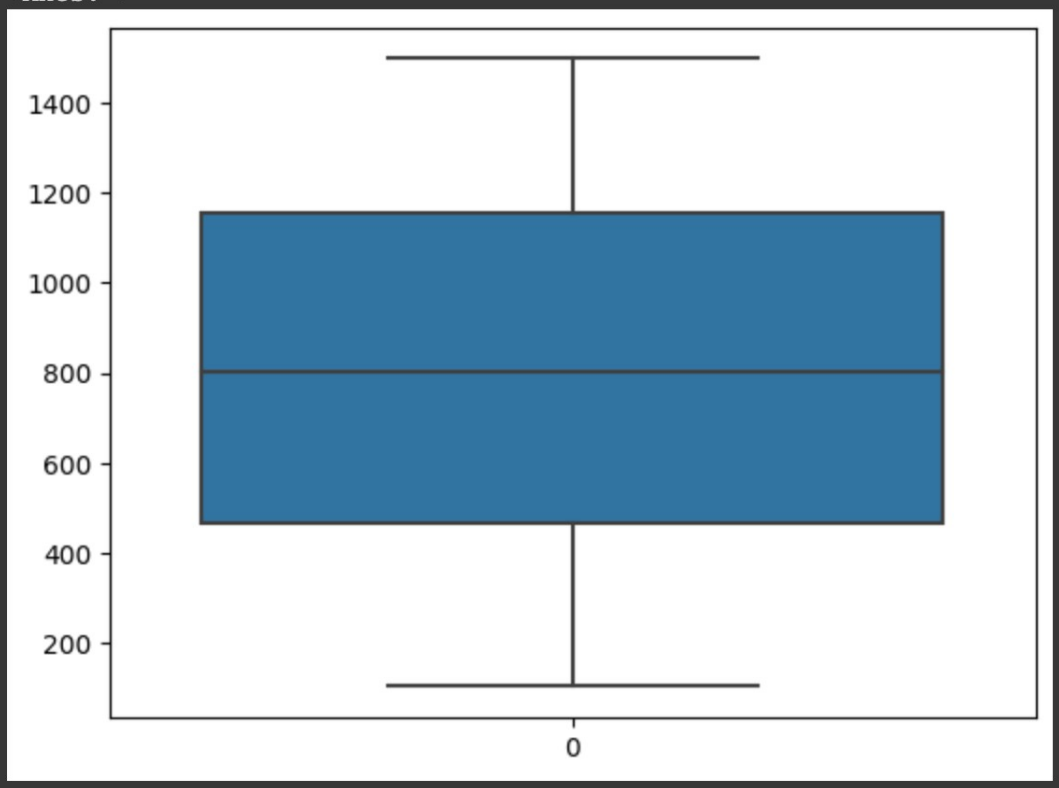
```
sns.heatmap(df.corr(),annot=True)
```

```
<ipython-input-13-8df7bcac526d>:1: FutureWarning: The default value of numeric_only in D
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```



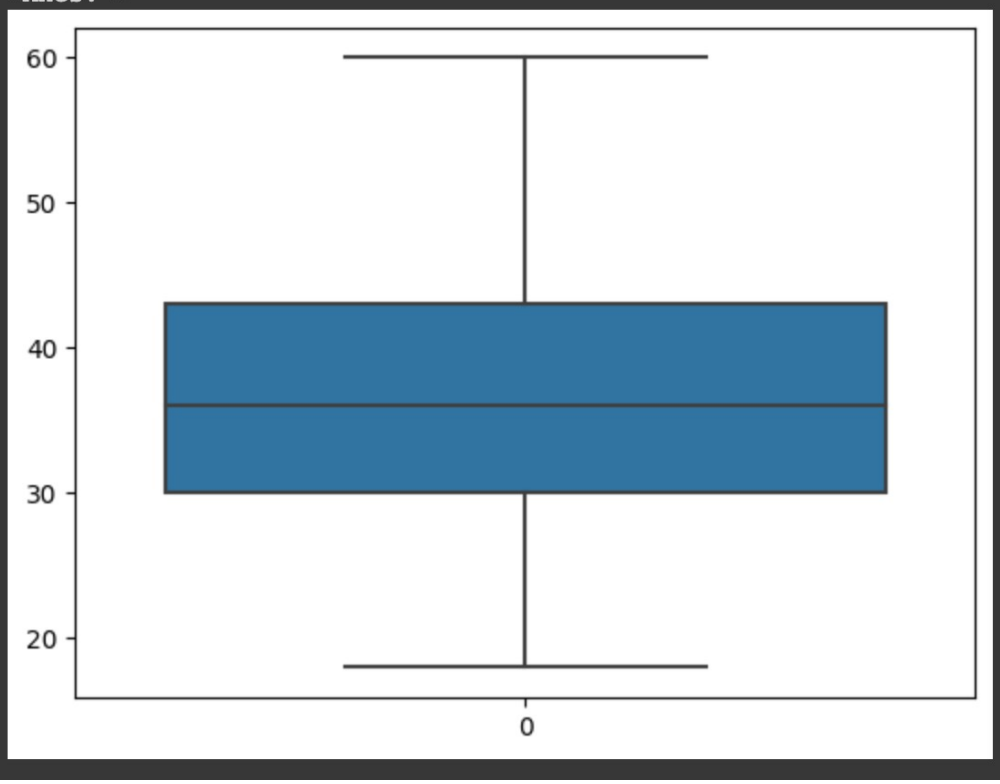
```
[15] sns.boxplot(df.DailyRate)
```

```
<Axes: >
```




```
sns.boxplot(df.Age)
```

```
<Axes: >
```



```
[17] df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSatisfaction
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	1
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	4
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	2
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	3
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	4

5 rows × 35 columns

```
[142] #Splitting Dependent and Independent variables
      x0=df.iloc[:,0:2]
      x1=df.iloc[:,3:4]
      x2=df.iloc[:,5:7]
      x3=df.iloc[:,9:11]
      x4=df.iloc[:,18:21]
      x=pd.concat([x0,x1,x2,x3,x4],axis=1)
      x.head()
```

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfaction	MonthlyIncome	MonthlyRate	NumCompaniesWorked
0	41	Yes	1102	1	2	1	2	5993	19479	8
1	49	No	279	8	1	2	3	5130	24907	1
2	37	Yes	1373	2	2	4	4	2090	2396	6
3	33	No	1392	3	4	5	4	2909	23159	1
4	27	No	591	2	1	7	1	3468	16632	9

```
y=df.PerformanceRating
y.head()
```

```
0                    Sales
1    Research & Development
2    Research & Development
3    Research & Development
4    Research & Development
Name: Department, dtype: object
```

```
y.shape
```

```
(1470,)
```

```
[148] #label encoding
      from sklearn.preprocessing import LabelEncoder
      le=LabelEncoder()
      x.Attrition=le.fit_transform(x.Attrition)
      x.head()
```

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfaction	MonthlyIncome	MonthlyRate	NumCompaniesWorked
0	41	1	1102	1	2	1	2	5993	19479	8
1	49	0	279	8	1	2	3	5130	24907	1
2	37	1	1373	2	2	4	4	2090	2396	6
3	33	0	1392	3	4	5	4	2909	23159	1
4	27	0	591	2	1	7	1	3468	16632	9

```
[149] #feature scaling
      from sklearn.preprocessing import MinMaxScaler
      ms=MinMaxScaler()
      x_scaled=pd.DataFrame(ms.fit_transform(x),columns=x.columns)
```

```
x_scaled
```

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfaction	MonthlyIncome	MonthlyRate	NumCompaniesWorked
0	0.547619	1.0	0.715820	0.000000	0.25	0.000000	0.333333	0.262454	0.698053	0.888889
1	0.738095	0.0	0.126700	0.250000	0.00	0.000484	0.666667	0.217009	0.916001	0.111111
2	0.452381	1.0	0.909807	0.035714	0.25	0.001451	1.000000	0.056925	0.012126	0.666667
3	0.357143	0.0	0.923407	0.071429	0.75	0.001935	1.000000	0.100053	0.845814	0.111111
4	0.214286	0.0	0.350036	0.035714	0.00	0.002903	0.000000	0.129489	0.583738	1.000000
...	...	...	...	...	...	...	...	...	...	...
1465	0.428571	0.0	0.559771	0.785714	0.25	0.996613	0.666667	0.082254	0.409396	0.444444
1466	0.500000	0.0	0.365784	0.178571	0.00	0.997097	1.000000	0.472986	0.777474	0.444444
1467	0.214286	0.0	0.037938	0.107143	0.50	0.998065	0.333333	0.270300	0.123670	0.111111
1468	0.738095	0.0	0.659270	0.035714	0.50	0.998549	1.000000	0.230700	0.447661	0.222222
1469	0.380952	0.0	0.376521	0.250000	0.50	1.000000	0.333333	0.178778	0.326601	0.222222

1470 rows × 10 columns

```
[151] #Splitting Data into Train and Test.
      from sklearn.model_selection import train_test_split
      x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.2,random_state=0)

[152] x_train.shape,x_test.shape,y_train.shape,y_test.shape

      ((1176, 10), (294, 10), (1176,), (294,))

[153] x_train.head()
```

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfaction	MonthlyIncome	MonthlyRate	NumCompaniesWorked
1374	0.952381	0.0	0.360057	0.714286	0.50	0.937107	1.000000	0.888152	0.388155	0.444444
1092	0.642857	0.0	0.607015	0.964286	0.50	0.747460	1.000000	0.059136	0.100020	0.444444
768	0.523810	0.0	0.141732	0.892857	0.50	0.515239	0.666667	0.388994	0.807990	0.111111
569	0.428571	0.0	0.953472	0.250000	0.75	0.381229	0.000000	0.346393	0.487252	0.111111
911	0.166667	1.0	0.355762	0.821429	0.00	0.615385	0.666667	0.005740	0.238747	0.111111

Modelling building

MODEL BUILDING

Logistic Regression

```
[154] from sklearn.linear_model import LogisticRegression
      model=LogisticRegression()
```

```
[155] model.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
pred=model.predict(x_test)
```

```
[98] pred
```

```
array([7, 2, 0, 2, 0, 2, 0, 2, 0, 0, 7, 0, 2, 0, 7, 2, 2, 0, 2, 2, 2, 7,
       2, 2, 2, 2, 2, 7, 2, 2, 2, 2, 3, 0, 2, 7, 7, 2, 0, 2, 7, 7, 7, 7,
       2, 0, 2, 2, 0, 0, 7, 0, 2, 2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 2, 7, 7,
       0, 0, 2, 0, 7, 2, 2, 2, 2, 2, 2, 7, 0, 2, 7, 2, 2, 2, 2, 2, 7, 2,
       2, 7, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 7, 2, 2, 2, 7, 2, 2, 2,
       2, 7, 7, 2, 2, 2, 2, 7, 9, 2, 2, 2, 2, 0, 0, 7, 7, 2, 2, 7, 7, 0,
       0, 2, 2, 2, 0, 2, 2, 2, 7, 2, 2, 2, 2, 0, 2, 7, 0, 2, 2, 7, 2, 7,
       7, 0, 8, 2, 2, 2, 7, 2, 2, 0, 8, 2, 2, 2, 2, 7, 7, 0, 2, 2, 7, 2,
       2, 7, 2, 7, 2, 2, 7, 0, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 2,
       0, 2, 2, 2, 0, 2, 2, 2, 2, 8, 2, 7, 2, 2, 8, 0, 0, 2, 7, 2, 7, 2,
       2, 2, 2, 7, 2, 0, 2, 2, 0, 2, 9, 2, 2, 2, 2, 2, 2, 2, 2, 7, 0, 2,
       8, 2, 2, 2, 7, 2, 2, 2, 2, 2, 0, 7, 2, 2, 7, 0, 2, 0, 2, 2, 2, 2,
       2, 2, 2, 0, 0, 2, 2, 2, 2, 2, 2, 2, 7, 2, 2, 2, 0, 2, 2, 8, 8, 2,
       2, 0, 7, 2, 2, 0, 2, 2])
```

```
[156] y_test
```

```
442     3
1091    3
981     3
785     4
1332    4
       ..
1439    3
481     3
124     3
198     3
1229    3
Name: PerformanceRating, Length: 294, dtype: int64
```

```
df
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	Relationsh
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	1	2061	...	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	1	2062	...	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	1	2064	...	
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	1	2065	...	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medical	1	2068	...	

1470 rows × 35 columns

```
[158] model.predict(ms.transform([[41,  1 ,1102,  1,  2,  1,  2,  5993, 19479,  8]]))

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
  warnings.warn(
array([3])
```

Evaluation of classification model

```
[159] #Accuracy score
      from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
```

```
[160] accuracy_score(y_test,pred)

      0.09523809523809523
```

```
confusion_matrix(y_test,pred)

array([[ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0],
       [37, 12, 54, 24, 18,  8,  4, 35, 24, 11,  7,  3,  1,  3,  1,  1,
         2],
       [ 8,  5, 11,  2,  4,  2,  1,  6,  5,  1,  0,  1,  1,  1,  1,  0,
         0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
```

```
[162] pd.crosstab(y_test,pred)
```

col_0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	16	17
PerformanceRating																	
3	37	12	54	24	18	8	4	35	24	11	7	3	1	3	1	1	2
4	8	5	11	2	4	2	1	6	5	1	0	1	1	1	1	0	0

```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00         0
           3       0.92      0.10      0.18       245
           4       0.18      0.08      0.11        49
           5       0.00      0.00      0.00         0
           6       0.00      0.00      0.00         0
           7       0.00      0.00      0.00         0
           8       0.00      0.00      0.00         0
           9       0.00      0.00      0.00         0
          10       0.00      0.00      0.00         0
          11       0.00      0.00      0.00         0
          12       0.00      0.00      0.00         0
          13       0.00      0.00      0.00         0
          14       0.00      0.00      0.00         0
          16       0.00      0.00      0.00         0
          17       0.00      0.00      0.00         0

    accuracy                           0.10       294
   macro avg       0.06      0.01      0.02       294
weighted avg       0.80      0.10      0.17       294
```

```
[164] probability=model.predict_proba(x_test)[:,1]
```

```
probability
```

```
0.1433304 , 0.15496154, 0.12800481, 0.13912284, 0.17382931,
0.18160542, 0.15095983, 0.15281618, 0.17614903, 0.1480909 ,
0.1337793 , 0.17096722, 0.17520951, 0.13977586, 0.13745433,
0.16531321, 0.12306327, 0.13069866, 0.14276321, 0.14809776,
0.15561902, 0.14428206, 0.14207862, 0.14735286, 0.18089769,
0.13644711, 0.18201748, 0.20983764, 0.13532801, 0.13358169,
0.11872259, 0.19006698, 0.14937338, 0.17242021, 0.16046445,
0.16471189, 0.10936733, 0.15378844, 0.168894  , 0.15266309,
0.15928175, 0.1871727 , 0.16706366, 0.15816173, 0.16218187,
0.13007243, 0.17397223, 0.1376382 , 0.13884367, 0.14530225,
0.13986828, 0.14944272, 0.1631796 , 0.14938843, 0.1323239 ,
0.16505872, 0.18242115, 0.12709696, 0.12237972, 0.12755602,
0.13424365, 0.12928038, 0.16833782, 0.15838749, 0.14795144,
```

Decision Tree

```
[110] from sklearn.tree import DecisionTreeClassifier
      dtc=DecisionTreeClassifier()
```

```
[111] dtc.fit(x_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
pred=dtc.predict(x_test)
```

```
[113] pred
```

```
[113] pred
```

```
array([11,  2,  7, 12,  0,  1,  7,  7,  9,  2, 10, 14,  6,  2, 13,  2,  8,
        5, 13,  2,  1,  2,  9,  2,  2,  6,  8,  3,  2,  5,  0,  0,  4,  3,
        2,  8,  7,  2,  9,  2,  2,  8,  3,  7,  3,  0,  0,  3,  0,  2,  2,
        7,  8,  2,  2,  1,  2,  3,  2,  3,  0,  5,  8,  4,  0,  2,  2,  4,
        2,  2,  7, 10,  8,  4,  0,  7,  5,  5,  2,  3,  3,  8,  7,  1,  2,
        3,  2,  0,  2,  7,  0,  7,  0,  4,  3,  0,  2,  4,  3,  2,  7,  7,
       10,  2,  8,  0,  8,  7,  2,  8,  7, 13,  0,  9,  7,  4,  4,  7,  7,
        8,  3,  8,  9,  1, 17,  2,  2,  3,  0,  7,  8,  3,  3,  2,  7,  7,
        4,  8,  0,  7, 11,  2,  5,  7,  0,  2,  3, 10,  8,  5,  4, 11,  2,
        1,  4,  3, 16,  7,  1, 10,  8,  4,  2, 14,  1,  8,  0,  5,  7,  4,
        0,  2,  2,  2,  9,  2,  3, 11,  2, 10,  4,  0,  7,  1,  2,  1,  7,
        7,  8,  7,  1,  0,  2,  1,  4,  8,  9,  7,  2,  7,  2,  0,  2,  0,
        0,  4,  3,  3,  9,  3,  2,  8,  2,  2,  3,  2,  2,  6,  1,  0,  0,
        2,  0,  8,  3,  2,  2,  7,  0,  1, 12,  1,  7,  7,  4,  0,  4,  2,
        2, 17,  0,  4,  8, 13,  8,  5,  0,  0,  7,  4,  0,  5,  0,  8,  0,
        2,  9,  2,  2,  2,  3,  4,  9,  1,  9,  2,  0,  0,  0,  0,  2,  0,
        8,  3,  7,  8,  6,  8,  7,  0,  6,  4,  0,  8, 10,  2,  0,  7,  7,
        7,  0,  1,  9,  7])
```

```
y_test
```

```
442     7
1091    2
981     2
785     1
1332    0
       ..
1439    4
481     3
124     7
198     0
1229    0
Name: YearsWithCurrManager, Length: 294, dtype: int64
```

```
df
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSatisfacti
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	1	2061	...	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	1	2062	...	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	1	2064	...	
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	1	2065	...	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medical	1	2068	...	

1470 rows × 35 columns

```
[116] dtc.predict(ms.transform([[41,  1 ,1102,  1,  2,  1,  2,  5993, 19479,  8]]))

    /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
      warnings.warn(
    array([5])
```

```
[171] #Accuracy score
      from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve

[172] accuracy_score(y_test,pred)

      0.09523809523809523

      confusion_matrix(y_test,pred)

      array([[ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
               0],
             [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
               0],
             [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
               0],
             [37, 12, 54, 24, 18,  8,  4, 35, 24, 11,  7,  3,  1,  3,  1,  1,
               2],
             [ 8,  5, 11,  2,  4,  2,  1,  6,  5,  1,  0,  1,  1,  1,  1,  0,
               0],
             [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
               0],
             [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
               0],
             [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
               0],
             [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
               0],
             [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
               0],
             [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
```

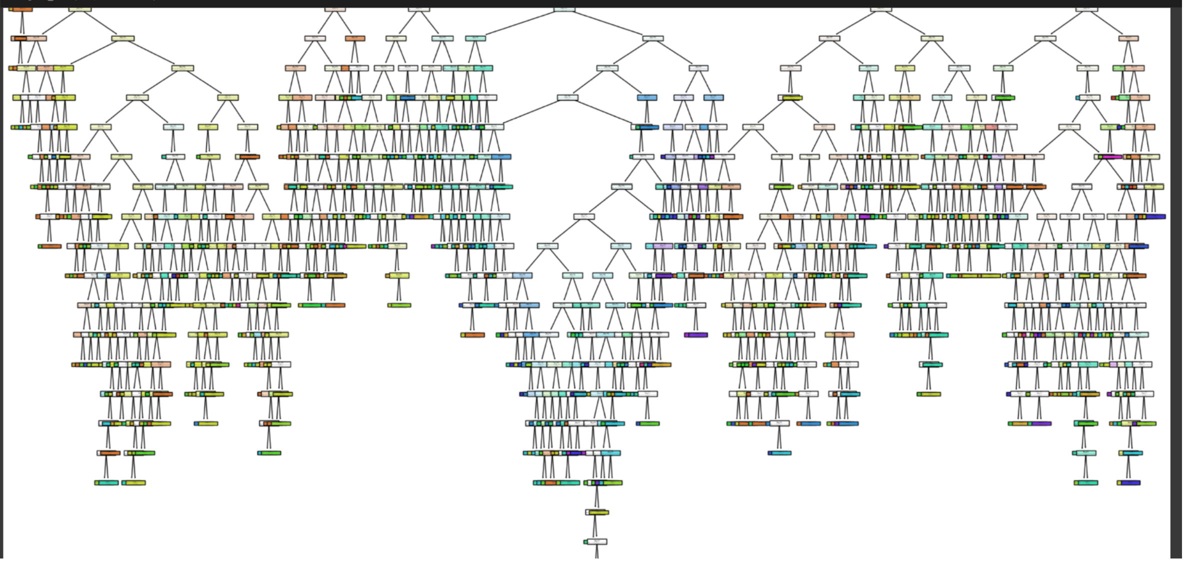
```
[174] pd.crosstab(y_test,pred)
```

col_0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	16	17
PerformanceRating																	
3	37	12	54	24	18	8	4	35	24	11	7	3	1	3	1	1	2
4	8	5	11	2	4	2	1	6	5	1	0	1	1	1	1	0	0

```
print(classification_report(y_test,pred))

              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00         0
           3       0.92      0.10      0.18       245
           4       0.18      0.08      0.11        49
           5       0.00      0.00      0.00         0
           6       0.00      0.00      0.00         0
           7       0.00      0.00      0.00         0
           8       0.00      0.00      0.00         0
           9       0.00      0.00      0.00         0
          10       0.00      0.00      0.00         0
          11       0.00      0.00      0.00         0
          12       0.00      0.00      0.00         0
          13       0.00      0.00      0.00         0
          14       0.00      0.00      0.00         0
          16       0.00      0.00      0.00         0
          17       0.00      0.00      0.00         0

    accuracy                           0.10       294
   macro avg       0.06      0.01      0.02       294
weighted avg       0.80      0.10      0.17       294

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWa
  _warn_prf(average, modifier, msg_start, len(result))
```

Hyper parameter tuning

```
from sklearn import tree
plt.figure(figsize=(25,15))
tree.plot_tree(dtc,filled=True)
```



```
from sklearn.model_selection import GridSearchCV
parameter={
 'criterion':['gini','entropy'],
  'splitter':['best','random'],
  'max_depth':[1,2,3,4,5],
  'max_features':['auto', 'sqrt', 'log2']

}
```

```
[182] grid_search=GridSearchCV(estimator=dtc,param_grid=parameter,cv=5,scoring="accuracy")
```

```
[183] grid_search.fit(x_train,y_train)

     /usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3
       warnings.warn(
     /usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3
       warnings.warn(
     /usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3
       warnings.warn(
     /usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3
       warnings.warn(
```

```
[184] grid_search.best_params_

     {'criterion': 'gini',
      'max_depth': 4,
      'max_features': 'sqrt',
      'splitter': 'random'}
```

```
dtc_cv=DecisionTreeClassifier(criterion= 'entropy',
 max_depth=3,
 max_features='sqrt',
 splitter='best')
dtc_cv.fit(x_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=3, max_features='sqrt')
```

```
[186] pred=dtc_cv.predict(x_test)
```

```
[187] print(classification_report(y_test,pred))

                   precision    recall  f1-score   support

               3       0.83      1.00      0.91       245
               4       0.00      0.00      0.00        49

        accuracy                           0.83       294
       macro avg       0.42      0.50      0.45       294
    weighted avg       0.69      0.83      0.76       294
```

Random Forest

```
[177] from sklearn.ensemble import RandomForestClassifier
      rfc=RandomForestClassifier()
```

```
[178] forest_params = [{'max_depth': list(range(10, 15)), 'max_features': list(range(0,14))}]
```

```
[188] rfc_cv= GridSearchCV(rfc,param_grid=forest_params,cv=10,scoring="accuracy")
```

```
rfc_cv.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:378: FitFailedWarning:
50 fits failed out of a total of 700.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
50 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py", line 340, in fit
    self._validate_params()
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 600, in _validate_params
    validate_parameter_constraints(
  File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/_param_validation.py", line 97, in validate_parameter_constraints
```

```
rfc_cv.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:378: FitFa
50 fits failed out of a total of 700.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='rai

Below are more details about the failures:
--------------------------------------------------------------------------------
50 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py",
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py", line 340, i
    self._validate_params()
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 600, in _validate_
    validate_parameter_constraints(
  File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/_param_validation.py", line
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of Ra

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:952: UserWarni
 0.84608866 0.8443865  0.84608141 0.84353904 0.84183688 0.84523396
 0.84266985 0.84608141        nan 0.84950022 0.84950022 0.8478053
 0.8478053  0.8460959  0.84608866 0.84268434 0.84693611 0.8452412
 0.8452412  0.8435318  0.8435318  0.84608866        nan 0.84950022
 0.84865276 0.84779806 0.84694336 0.8452412  0.84523396 0.84437926
 0.84182964 0.84353904 0.8443865  0.84014197 0.84097494 0.84269158
        nan 0.84865276 0.84865276 0.84865276 0.84523396 0.8452412
 0.84523396 0.8426771  0.8435318  0.84269883 0.84693611 0.84098218
 0.8418224  0.84439374        nan 0.84950022 0.84865276 0.8469506
 0.84779806 0.8435318  0.84694336 0.84354628 0.84183688 0.84269158
 0.8452412  0.84182964 0.8384543  0.83758511]
  warnings.warn(
```

▸ GridSearchCV

▸ estimator: RandomForestClassifier

▸ RandomForestClassifier

```
[191] pred=rfc_cv.predict(x_test)
```

```
[192] print(classification_report(y_test,pred))
```

```
                  precision    recall  f1-score   support

               3       0.83      1.00      0.91       245
               4       0.00      0.00      0.00        49

        accuracy                           0.83       294
       macro avg       0.42      0.50      0.45       294
    weighted avg       0.69      0.83      0.76       294
```

```
rfc_cv.best_params_
```

```
{'max_depth': 10, 'max_features': 2}
```