

In []:

Assignment 8 th september
1. Take car crashes dataset from seaborn library
2. load the dataset
3. data visualiation
4. Inference is must for each and every graph
5. Submit it by wednesday in html format

Feedback - <https://forms.gle/7vFfvANDVfvDxxW28>

Steps:

- 1.import the necessary libraries
- 2.import the dataset
- 3.Handling null values
- 4.outlier detection---surya
- 5.Seperate Dependent and independent variables
- 6.Encoding
- 7.splitting into training and testing set
- 8.Feature scaling

1.import the necessary libraries

In [16]:

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

2.import the dataset

In [37]:

```
dataset=pd.read_csv("C:\\\\Users\\\\KEERTHI KRISHANA\\\\OneDrive\\\\Documents\\\\employee.csv")
```

In [18]:

```
dataset
```

Out[18]:

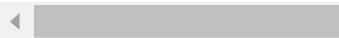
localhost:8888/notebooks/Downloads/assignment 4 ai-ml.ipynb#8.Feature-Scaling

In [18]: dataset

Out[18]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Education
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sci
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sci
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Life Sci
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sci
4	27	No	Travel_Rarely	591	Research & Development	2	1	Life Sci
...
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Life Sci
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Life Sci
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sci
1468	49	No	Travel_Frequently	1023	Sales	2	3	Life Sci
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Life Sci

1470 rows × 35 columns



In [19]: dataset.head(3)

Out[19]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Education
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sci
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sci
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Life Sci

3 rows × 35 columns



In [20]: dataset.tail()

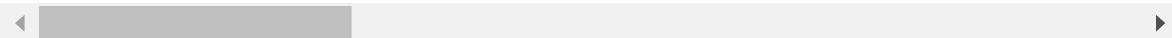
Out[20]:

In [20]: `dataset.tail()`

Out[20]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Ed
1465	36	No	Travel_Frequently	884	Research & Development	23	2	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Lif
1468	49	No	Travel_Frequently	1023	Sales	2	3	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	

5 rows × 35 columns



In [21]: `dataset.shape`

Out[21]: (1470, 35)

In [22]: `dataset.info()`

<class 'pandas.core.frame.DataFrame'>

In [22]: `dataset.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  
 33  YearsSinceLastPromotion 1470 non-null    int64  
 34  YearsWithCurrManager 1470 non-null    int64  
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

In [23]: `dataset.describe()`

Out[23]:

In [23]: `dataset.describe()`

Out[23]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.86530
std	9.135373	403.509100	8.106864	1.024165	0.0	602.02433
min	18.000000	102.000000	1.000000	1.000000	1.0	1.00000
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.25000
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.50000
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.75000
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.00000

8 rows × 26 columns



In []:

Formula



$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

σ = population standard deviation

N = the size of the population

x_i = each value from the population

μ = the population mean

#

In [24]: `dataset.head()`

In [24]: `dataset.head()`

Out[24]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Education
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sci
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sci
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Life Sci
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sci
4	27	No	Travel_Rarely	591	Research & Development	2	1	Med

5 rows × 35 columns



In [25]: `dataset = dataset.drop(columns=['EducationField'])`
`dataset = dataset.drop(columns=['Department'])`

In [26]: `pip install scikit-learn`

```
Requirement already satisfied: scikit-learn in c:\users\keerthi krishana\.conda
\envs\tf\lib\site-packages (1.3.1)
Requirement already satisfied: numpy<2.0,>=1.17.3 in c:\users\keerthi krishana
\.\conda\envs\tf\lib\site-packages (from scikit-learn) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\keerthi krishana\.conda
\envs\tf\lib\site-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\keerthi krishana\.cond
a\envs\tf\lib\site-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\keerthi krishan
a\.conda\envs\tf\lib\site-packages (from scikit-learn) (3.2.0)
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 23.1.2 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [42]: `from sklearn.preprocessing import LabelEncoder`
`le = LabelEncoder()`
`dataset["Gender"] = le.fit_transform(dataset["Gender"])`
`dataset["Attrition"] = le.fit_transform(dataset["Attrition"])`

In [43]: `dataset["MaritalStatus"] = le.fit_transform(dataset["MaritalStatus"])`
`dataset["OverTime"] = le.fit_transform(dataset["OverTime"])`

In [44]: `dataset["Over18"] = le.fit_transform(dataset["Over18"])`

In [45]: `dataset["BusinessTravel"] = le.fit_transform(dataset["BusinessTravel"])`

```
In [45]: dataset["BusinessTravel"] = le.fit_transform(dataset["BusinessTravel"])
```

```
In [47]: dataset["Department"] = le.fit_transform(dataset["Department"])
```

```
In [49]: dataset["EducationField"] = le.fit_transform(dataset["EducationField"])
```

```
In [51]: dataset["JobRole"] = le.fit_transform(dataset["JobRole"])
```

```
In [52]: corr = dataset.corr()  
corr
```

```
In [52]: corr=dataset.corr()  
corr
```

Out[52]:

Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFrom
-----	-----------	----------------	-----------	------------	--------------

Age	1.000000	-0.150205	0.024751	0.010661	-0.031882	-0.1
-----	----------	-----------	----------	----------	-----------	------

Out[52]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFrom
Age	1.000000	-0.159205	0.024751	0.010661	-0.031882	-0.0
Attrition	-0.159205	1.000000	0.000074	-0.056652	0.063991	0.0
BusinessTravel	0.024751	0.000074	1.000000	-0.004086	-0.009044	-0.0
DailyRate	0.010661	-0.056652	-0.004086	1.000000	0.007109	-0.0
Department	-0.031882	0.063991	-0.009044	0.007109	1.000000	0.0
DistanceFromHome	-0.001686	0.077924	-0.024469	-0.004985	0.017225	1.0
Education	0.208034	-0.031373	0.000757	-0.016806	0.007996	0.0
EducationField	-0.040873	0.026846	0.023724	0.037709	0.013720	0.0
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN
EmployeeNumber	-0.010145	-0.010577	-0.015578	-0.050990	-0.010895	0.0
EnvironmentSatisfaction	0.010146	-0.103369	0.004174	0.018355	-0.019395	-0.0
Gender	-0.036311	0.029453	-0.032981	-0.011716	-0.041583	-0.0
HourlyRate	0.024287	-0.006846	0.026528	0.023381	-0.004144	0.0
JobInvolvement	0.029820	-0.130016	0.039062	0.046135	-0.024586	0.0
JobLevel	0.509604	-0.169105	0.019311	0.002966	0.101963	0.0
JobRole	-0.122427	0.067151	0.002724	-0.009472	0.662431	-0.0
JobSatisfaction	-0.004892	-0.103481	-0.033962	0.030571	0.021001	-0.0
MaritalStatus	-0.095029	0.162070	0.024001	-0.069586	0.056073	-0.0
MonthlyIncome	0.497855	-0.159840	0.034319	0.007707	0.053130	-0.0
MonthlyRate	0.028051	0.015170	-0.014107	-0.032182	0.023642	0.0
NumCompaniesWorked	0.299635	0.043494	0.020875	0.038153	-0.035882	-0.0
Over18	NaN	NaN	NaN	NaN	NaN	NaN
OverTime	0.028062	0.246118	0.016543	0.009135	0.007481	0.0
PercentSalaryHike	0.003634	-0.013478	-0.029377	0.022704	-0.007840	0.0
PerformanceRating	0.001904	0.002889	-0.026341	0.000473	-0.024604	0.0
RelationshipSatisfaction	0.053535	-0.045872	-0.035986	0.007846	-0.022414	0.0
StandardHours	NaN	NaN	NaN	NaN	NaN	NaN
StockOptionLevel	0.037510	-0.137145	-0.016727	0.042143	-0.012193	0.0
TotalWorkingYears	0.680381	-0.171063	0.034226	0.014515	-0.015762	0.0
TrainingTimesLastYear	-0.019621	-0.059478	0.015240	0.002453	0.036875	-0.0
WorkLifeBalance	-0.021490	-0.063939	-0.011256	-0.037848	0.026383	-0.0
YearsAtCompany	0.311309	-0.134392	-0.014575	-0.034055	0.022920	0.0
YearsInCurrentRole	0.212901	-0.160545	-0.011497	0.009932	0.056315	0.0
YearsSinceLastPromotion	0.216513	-0.033019	-0.032591	-0.033229	0.040061	0.0
YearsWithCurrManager	0.202089	-0.156199	-0.022636	-0.026363	0.034282	0.0

35 rows × 35 columns

In []: `plt.subplots(figsize=(20,15))
sns.heatmap(corr, annot=True)`

53 rows × 33 columns

```
In [ ]: plt.subplots(figsize=(20,15))
sns.heatmap(corr, annot=True)
```

```
In [31]: datasetAttrition.value_counts()
```

```
Out[31]: Attrition
0    1233
1    237
Name: count, dtype: int64
```

```
In [32]: datasetGender.value_counts()
```

```
Out[32]: Gender
1    882
0    588
Name: count, dtype: int64
```

```
In [33]: dataset.head()
```

```
Out[33]:
```

	Age	Attrition	BusinessTravel	DailyRate	DistanceFromHome	Education	EmployeeCount	Empl
0	41	1	Travel_Rarely	1102		1	2	1
1	49	0	Travel_Frequently	279		8	1	1
2	37	1	Travel_Rarely	1373		2	2	1
3	33	0	Travel_Frequently	1392		3	4	1
4	27	0	Travel_Rarely	591		2	1	1

5 rows × 33 columns

```
In [ ]:
```

```
In [ ]:
```

3. Handling null values

```
In [ ]:
```

```
In [53]: dataset.isnull().any()
```

```
Out[53]: Age           False
```

```
In [53]: dataset.isnull().any()
```

```
Out[53]: 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
```

```
In [54]: dataset.isnull().sum()
```

```
Out[54]: Age           0
```

In [54]: `dataset.isnull().sum()`

Out[54]:

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

In []:

In [56]: `dataset.head()`

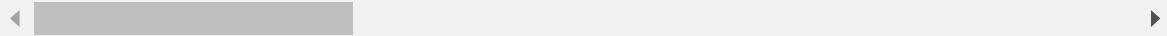
Out[56]:

In [56]: `dataset.head()`

Out[56]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educationl
0	41	1	2	1102	2		1	2
1	49	0	1	279	1		8	1
2	37	1	2	1373	1		2	2
3	33	0	1	1392	1		3	4
4	27	0	2	591	1		2	1

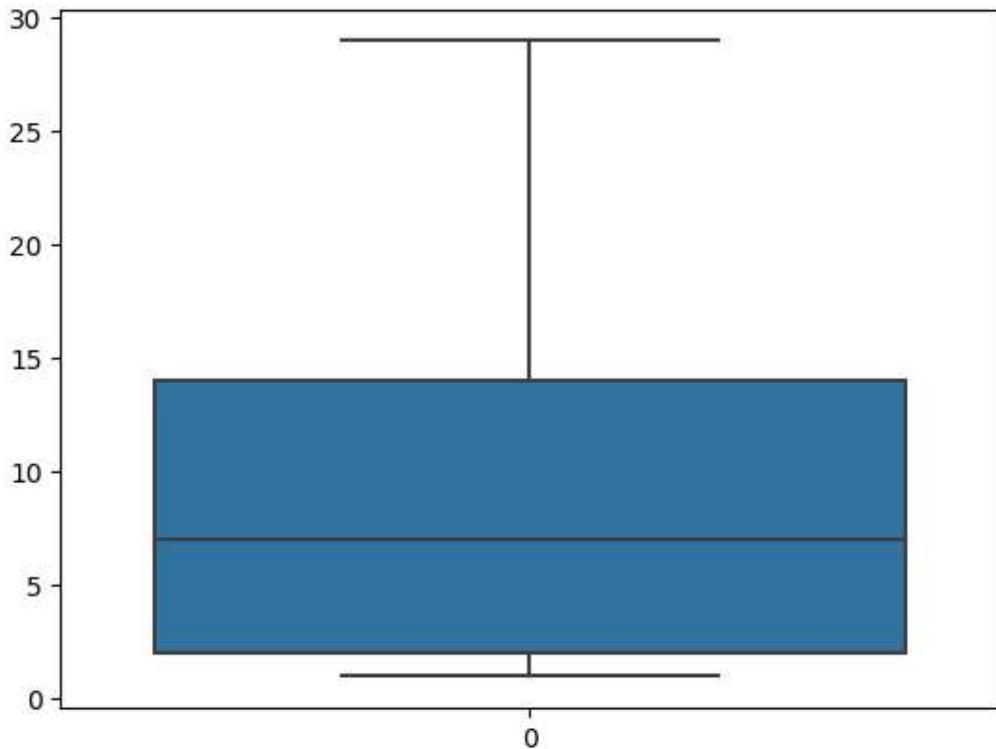
5 rows × 35 columns



4.outliers

In [57]: `sns.boxplot(dataset.DistanceFromHome)`

Out[57]: <Axes: >

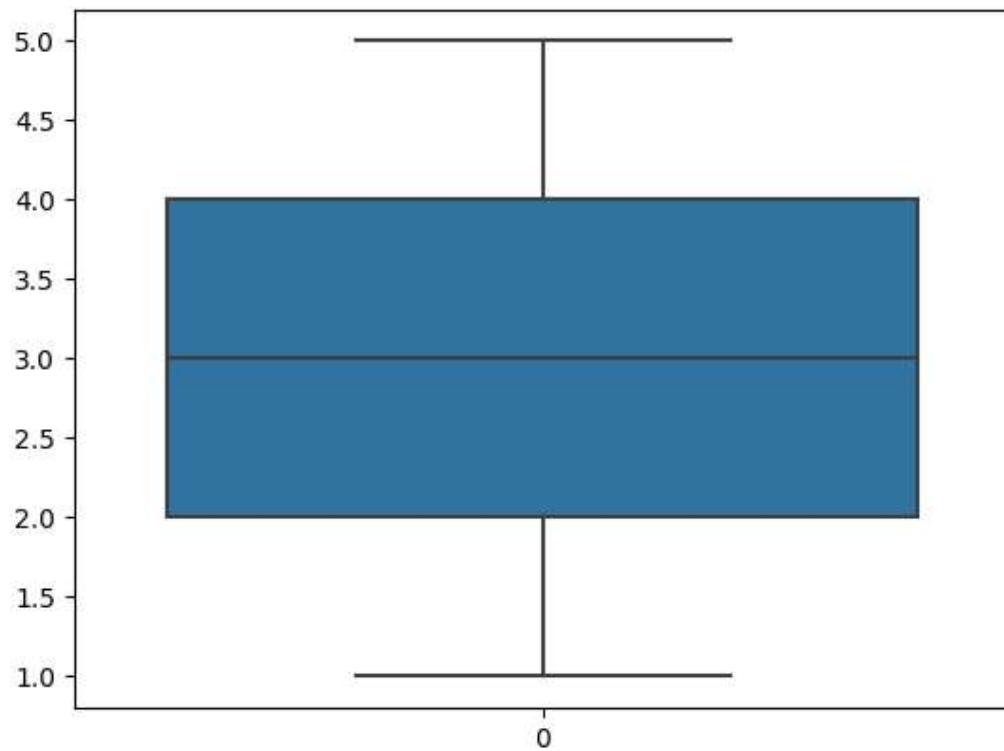


In [58]: `sns.boxplot(dataset.Education)`

Out[58]: <Axes: >

```
In [58]: sns.boxplot(dataset.Education)
```

```
Out[58]: <Axes: >
```



5. Separate dependent and independent variables

```
In [208]: #dataset.iloc[rows, column]
x=dataset.iloc[:,1:4]
y=dataset.iloc[:,22:23]
```

```
In [209]: x.head()
```

```
Out[209]:
```

	Attrition	BusinessTravel	DailyRate
0	1	2	1102
1	0	1	279
2	1	2	1373
3	0	1	1392
4	0	2	591

```
In [210]: y.head()
```

```
Out[210]:
```

In [210]: `y.head()`

Out[210]:

	OverTime
0	1
1	0
2	1
3	1
4	0

In [211]: `dataset.shape`

Out[211]: (1470, 35)

In [212]: `x.shape`

Out[212]: (1470, 3)

In [213]: `y.shape`

Out[213]: (1470, 1)

Type *Markdown* and *LaTeX*: α^2

In []:

Label encoding on Gender column

In []:

In [214]: `x.head()`

Out[214]:

In [214]: `x.head()`

Out[214]:

	Attrition	BusinessTravel	DailyRate
0	1	2	1102
1	0	1	279
2	1	2	1373
3	0	1	1392
4	0	2	591

In []:

(attachment:image.png)

In []:

```
In [215]: from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()
x_scaled=pd.DataFrame(ms.fit_transform(x),columns=x.columns)
```

In [216]: x_scaled

Out[216]:

	Attrition	BusinessTravel	DailyRate
0	1.0	1.0	0.715820
1	0.0	0.5	0.126700
2	1.0	1.0	0.909807
3	0.0	0.5	0.923407
4	0.0	1.0	0.350036
...
1465	0.0	0.5	0.559771
1466	0.0	1.0	0.365784
1467	0.0	1.0	0.037938
1468	0.0	0.5	0.659270
1469	0.0	1.0	0.376521

1470 rows × 3 columns

```
In [217]: 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_st
```

In [218]: x_train.shape,x_test.shape,y_train.shape,y_test.shape

Out[218]: ((1176, 3), (294, 3), (1176, 1), (294, 1))

In [219]: x_train.head()

Out[219]:

In [219]: `x_train.head()`

Out[219]:

	Attrition	BusinessTravel	DailyRate
1374	0.0	1.0	0.360057
1092	0.0	1.0	0.607015
768	0.0	1.0	0.141732
569	0.0	0.0	0.953472
911	1.0	0.5	0.355762

In [220]: `y_test.head()`

Out[220]:

	OverTime
442	0
1091	0
981	1
785	0
1332	1

In [221]: `x_test.head()`

Out[221]:

	Attrition	BusinessTravel	DailyRate
442	0.0	0.0	0.381532
1091	0.0	1.0	0.338583
981	1.0	0.5	0.400859
785	0.0	1.0	0.994989
1332	1.0	0.5	0.255548

In [222]: `from sklearn.linear_model import LogisticRegression`
`model=LogisticRegression()`

In [223]: `model.fit(x_train,y_train)`

```
C:\Users\KEERTHI KRISHANA\conda\envs\tf\lib\site-packages\sklearn\utils\validation.py:1183: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
y = column_or_1d(y, warn=True)
```

Out[223]:

LogisticRegression
LogisticRegression()

In [224]: `pred=model.predict(x_test)`

```
In [224]: pred=model.predict(x_test)
```

In [225]: pred

In []:

In [1]:

```
In [97]: from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

```
In [226]: accuracy score(y_test, pred)
```

Out[226]: 0.7040816326530612

```
In [227]: confusion_matrix(y_test, pred)
```

```
Out[227]: array([[194,    7],  
                  [ 80, 131]], dtype=int64)
```

```
In [229]: probability=model.predict_proba(x_test)[:,1]  
probability
```

```
In [229]: probability=model.predict_proba(x_test)[:,1]  
probability
```

```
Out[229]: array([0.21381276, 0.22661105, 0.48854887, 0.25902132, 0.47879227,  
0.20579433, 0.50546722, 0.23552353, 0.19838807, 0.48867727,  
0.25508854, 0.48478273, 0.23330162, 0.49892382, 0.24260724,
```

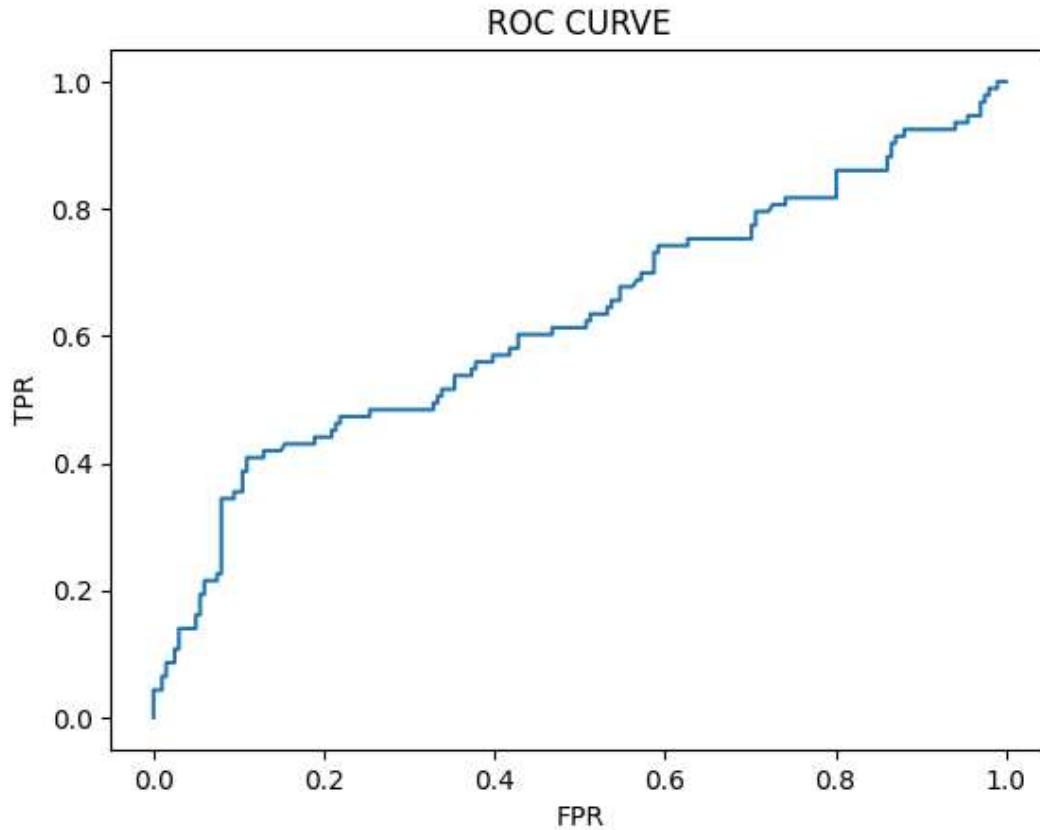
Out[229]: array([0.21381276, 0.22661105, 0.48854887, 0.25902132, 0.47879227, 0.20579433, 0.50546722, 0.23552353, 0.19838807, 0.48867727, 0.25508854, 0.48478273, 0.23330162, 0.49892382, 0.24260724, 0.25210134, 0.22057501, 0.49714359, 0.21828957, 0.24509131, 0.22898086, 0.23508511, 0.22402436, 0.22581365, 0.25721545, 0.22704984, 0.2316077 , 0.25115904, 0.22412474, 0.23487752, 0.22089513, 0.24089021, 0.23464691, 0.49637379, 0.25780422, 0.24077303, 0.2316077 , 0.22647616, 0.49964554, 0.21885924, 0.21421226, 0.24978604, 0.21424465, 0.21159988, 0.23814427, 0.46818917, 0.22715119, 0.22571273, 0.47021815, 0.50657371, 0.24099581, 0.53265394, 0.19829627, 0.2588736 , 0.47443926, 0.21957309, 0.22289988, 0.48374111, 0.24726997, 0.21289766, 0.21564112, 0.25461341, 0.22102765, 0.25037545, 0.2153265 , 0.23922843, 0.51965121, 0.49648589, 0.23891333, 0.52070802, 0.24766422, 0.23054762, 0.21783014, 0.2298655 , 0.23893643, 0.23713312, 0.21275805, 0.23136803, 0.49052072, 0.22222288, 0.21317741, 0.2346355 , 0.22059718, 0.21845382, 0.21912258, 0.24137163, 0.23058176, 0.25036353, 0.24267797, 0.24420217, 0.24069072, 0.25195622, 0.21842097, 0.21783014, 0.21582543, 0.22199011, 0.22536547, 0.21118279, 0.24721015, 0.23364605, 0.5330851 , 0.5242133 , 0.21128957, 0.2475925 , 0.23435911, 0.25206506, 0.23838879, 0.25549103, 0.21219962, 0.2491133 , 0.25843074, 0.23769063, 0.22721877, 0.22173527, 0.23755117, 0.20658185, 0.22015607, 0.2376209 , 0.23418649, 0.24587556, 0.25257337, 0.23225912, 0.21276869, 0.47244046, 0.21134314, 0.20473722, 0.23662275, 0.24519815, 0.24130116, 0.21645605, 0.24063233, 0.49998234, 0.53346832, 0.24038625, 0.2556375 , 0.23880836, 0.47901657, 0.23237342, 0.23780678, 0.23164196, 0.23366914, 0.24734162, 0.2221231 , 0.23609023, 0.25695814, 0.49893969, 0.20984991, 0.22516394, 0.49295803, 0.25392004, 0.22515252, 0.22860731, 0.21866189, 0.53710682, 0.24541193, 0.20956285, 0.24477097, 0.210329 , 0.21639077, 0.24799895, 0.47984899, 0.25275508, 0.21563005, 0.48545574, 0.20599352, 0.23831891, 0.2402457 , 0.24576852, 0.22551098, 0.23845868, 0.21849753, 0.48843682, 0.23129959, 0.25166614, 0.53284557, 0.2398244 , 0.24406011, 0.21718583, 0.24430875, 0.21629289, 0.25007467, 0.23350824, 0.2482862 , 0.52363725, 0.2529005 , 0.21250028, 0.24762836, 0.2269823 , 0.49420881, 0.21776457, 0.22302225, 0.52968242, 0.22412474, 0.21362971, 0.25126764, 0.21580393, 0.21760069, 0.25714191, 0.53882897, 0.22978578, 0.24435601, 0.2202334 , 0.48588842, 0.23123115, 0.22938886, 0.23224745, 0.2234895 , 0.23615965, 0.21479596, 0.2234895 , 0.23849363, 0.25571075, 0.23054762, 0.52864331, 0.50224373, 0.25913216, 0.24477097, 0.22929803, 0.24335066, 0.25648073, 0.23619437, 0.21427705, 0.20811059, 0.19936931, 0.2123071 , 0.2442377 , 0.22570163, 0.24045654, 0.48739478, 0.23679662, 0.22236695, 0.21980403, 0.22844885, 0.21862901, 0.25268239, 0.22462715, 0.22312232, 0.25876284, 0.25206506, 0.21105458, 0.4773516 , 0.22688101, 0.24906542, 0.22901484, 0.22536547, 0.223924 , 0.22913924, 0.24293734, 0.23612494, 0.24953366, 0.22102765, 0.20976472, 0.45877717, 0.22708362, 0.25706839, 0.24913742, 0.25000249, 0.51104697, 0.24601833, 0.51715263, 0.22983143, 0.23836538, 0.22447065, 0.22748924, 0.24190065, 0.23442819, 0.22759072, 0.51861009, 0.53744175, 0.24441537, 0.21405032, 0.22932082, 0.22505181, 0.22382368, 0.22439259, 0.22657732, 0.22844885, 0.25762013, 0.2199691 , 0.24094899, 0.52250079, 0.24770008, 0.23723759, 0.23102594, 0.22248911, 0.23205327, 0.23197302, 0.48334075, 0.25268239, 0.22600448,

In [230]: fpr,tpr@11596688]d@.470448865v@25058030p0ba01971893])

0.23205327, 0.23197302, 0.48334075, 0.25268239, 0.22600448,

In [230]: `fpr, tpr = roc_curve(y_test, y_pred)`In [231]:

```
plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```

In [232]:

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
```

In [233]: `dtc.fit(x_train, y_train)`Out[233]:

```
DecisionTreeClassifier()
DecisionTreeClassifier()
```

In [234]:

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

```
In [234]: pred=dtc.predict(x_test)  
pred
```

```
In [235]: y_test
```

Out[235]:

OverTime	
442	0
1091	0
981	1
785	0
1332	1
...	...
1439	0
481	1
124	1
198	0
1229	1

294 rows × 1 columns

```
In [236]: accuracy_score(y_test, pred)
```

Out[236]: 0.6020408163265306

```
In [237]: confusion_matrix(y_test,pred)
```

```
Out[237]: array([[147,  54],  
                  [ 63,  30]], dtype=int64)
```

```
In [239]: probability = dtc.predict_proba(x_test)[:,1]
probability
```

```
In [239]: probability=dtc.predict_proba(x_test)[:,1]
probability
```

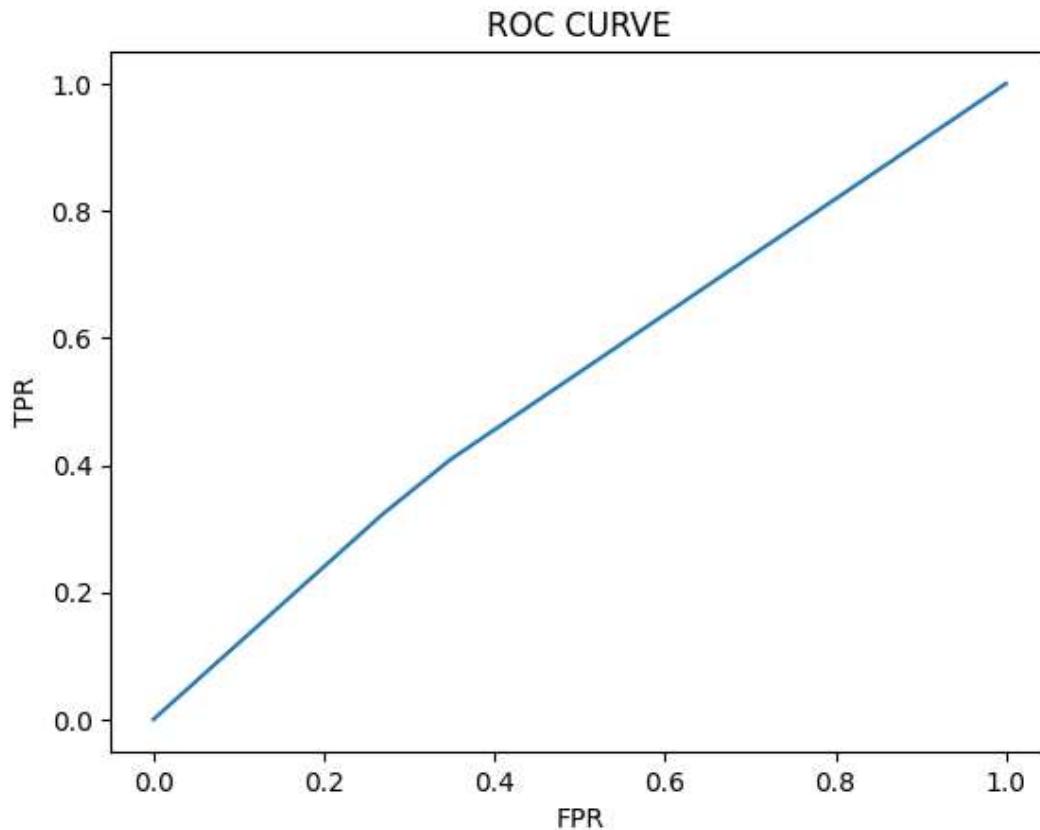
```
Out[239]: array([1.          , 0.          , 1.          , 0.33333333, 1.          ,
       0.          , 1.          , 0.          , 1.          , 0.          ,
       0.          , 0.          , 0.          , 0.          , 1.          ,
```

```
Out[239]: array([1.        , 0.        , 1.        , 0.33333333, 1.        ,
 0.        , 1.        , 0.        , 1.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 1.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 1.        , 0.        , 0.        ,
 0.5       , 0.        , 0.        , 0.        , 0.        ,
 0.        , 1.        , 0.        , 1.        , 1.        ,
 1.        , 0.        , 1.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 1.        ,
 0.        , 0.        , 1.        , 0.        , 1.        ,
 1.        , 0.        , 1.        , 0.33333333, 1.        ,
 1.        , 0.        , 0.        , 0.        , 1.        ,
 0.        , 0.        , 1.        , 0.        , 1.        ,
 1.        , 0.        , 1.        , 0.        , 1.        ,
 0.5       , 0.33333333, 0.        , 0.        , 1.        ,
 0.        , 1.        , 0.        , 1.        , 0.        ,
 1.        , 0.        , 0.33333333, 0.        , 0.        ,
 1.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 1.        , 0.        , 1.        , 0.        ,
 0.        , 0.        , 1.        , 0.5       , 1.        ,
 0.        , 0.        , 0.        , 1.        , 0.        ,
 1.        , 0.        , 0.        , 0.        , 0.        ,
 1.        , 0.        , 0.        , 0.5       , 0.        ,
 1.        , 0.        , 0.        , 1.        , 0.        ,
 0.        , 1.        , 0.        , 1.        , 1.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.5       , 0.        , 0.        , 0.        , 0.        ,
 1.        , 0.        , 0.        , 0.5       , 0.        ,
 1.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 1.        , 1.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 1.        , 0.        , 0.33333333, 0.        , 0.        ,
 0.        , 0.        , 0.        , 1.        , 0.5       ,
 0.        , 0.        , 0.        , 0.        , 1.        ,
 0.        , 0.        , 0.        , 0.        , 0.5       ,
 0.        , 0.        , 0.        , 0.5       , 0.        ,
 0.5       , 0.5       , 0.        , 1.        , 1.        ,
 0.        , 1.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 1.        , 1.        ,
 1.        , 0.        , 1.        , 0.        , 0.        ,
 0.        , 1.        , 0.        , 0.        , 1.        ,
 0.5       , 0.        , 0.33333333, 1.        , 1.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 1.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 1.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.5       , 0.        ,
 0.        , 0.        , 0.        , 1.        , 0.        ,
 1.        , 1.        , 0.        , 0.        , 1.        ,
 1.        , 0.        , 1.        , 0.5       , 0.        ,
 1.        , 0.        , 0.        , 0.5       , 1.        ,
 1.        , 0.        , 0.        , 0.        , 1.        ,
 0.        , 1.        , 1.        , 1.        , 0.        ,
 1.        , 0.        , 1.        , 1.        , 1.        ,
 0.        , 0.        , 1.        , 0.        , 0.        ,
 0.        , 0.5       , 0.        , 0.        , 0.5       ,
 0.        , 0.        , 0.        , 0.        , 1.        ,
 1.        , 0.5       , 1.        , 0.        , 1.        ])
```

```
In [240]: fpr,tpr,thresholds = roc_curve(y_test,probability) ]
```

```
In [240]: fpr, tpr, thresholds = roc_curve(y_test, probability)]
```

```
In [241]: plt.plot(fpr, tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```



```
In [242]: from sklearn import tree
plt.figure(figsize=(25,15))
tree.plot_tree(dtc,filled=True)
```

```
In [242]: from sklearn import tree
plt.figure(figsize=(25,15))
tree.plot_tree(dtc,filled=True)

  [ 0.5349487418452936,  0.4891304347826087, 'x[2] <= 0.685\ngini = 0.278\nsamples = 6\nvalue = [5, 1]'),
   Text(0.5312208760484622,  0.4673913043478261, 'gini = 0.5\nsamples = 2\nvalue = [1, 1']),
   Text(0.5386766076421249,  0.4673913043478261, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]'),
   Text(0.5461323392357875,  0.5108695652173914, 'x[2] <= 0.699\ngini = 0.444\nsamples = 9\nvalue = [6, 3]'),
   Text(0.5424044734389561,  0.4891304347826087, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
   Text(0.5498602050326188,  0.4891304347826087, 'x[2] <= 0.705\ngini = 0.245\nsamples = 7\nvalue = [6, 1]'),
   Text(0.5461323392357875,  0.4673913043478261, 'gini = 0.0\nsamples = 5\nvalue = [5, 0]'),
   Text(0.5535880708294502,  0.4673913043478261, 'x[2] <= 0.707\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
   Text(0.5498602050326188,  0.44565217391304346, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
   Text(0.5573159366262814,  0.44565217391304346, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]')
```

```
In [244]: 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']
}
```

```
In [245]: grid_search=GridSearchCV(estimator=dtc,param_grid=parameter, cv=5, scoring="accuracy")
```

```
In [246]: grid_search.fit(x_train,y_train)
```

In [246]: `grid_search.fit(x_train,y_train)`

```
C:\Users\KEERTHI KRISHANA\.conda\envs\tf\lib\site-packages\sklearn\model_selection\_validation.py:425: FitFailedWarning:
100 fits failed out of a total of 300.
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:
-----
-
100 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\KEERTHI KRISHANA\.conda\envs\tf\lib\site-packages\sklearn\model_selection\_validation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\KEERTHI KRISHANA\.conda\envs\tf\lib\site-packages\sklearn\base.py", line 1145, in wrapper
    estimator._validate_params()
  File "C:\Users\KEERTHI KRISHANA\.conda\envs\tf\lib\site-packages\sklearn\base.py", line 638, in _validate_params
    validate_parameter_constraints()
  File "C:\Users\KEERTHI KRISHANA\.conda\envs\tf\lib\site-packages\sklearn\utils\_param_validation.py", line 95, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of DecisionTreeClassifier must be an int in the range [1, inf), a float in the range (0.0, 1.0], a str among {'log2', 'sqrt'} or None. Got 'auto' instead.

    warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\KEERTHI KRISHANA\.conda\envs\tf\lib\site-packages\sklearn\model_selection\_search.py:979: UserWarning: One or more of the test scores are non-finite:
[      nan      nan 0.72534079 0.72025604 0.72534079 0.72534079
      nan      nan 0.71938334 0.71259647 0.72448972 0.73131266
      nan      nan 0.72449693 0.72364587 0.72706094 0.72109989
      nan      nan 0.73640822 0.71769924 0.71768842 0.72619906
      nan      nan 0.71004327 0.71684097 0.72024522 0.72024522
      nan      nan 0.72025604 0.72534079 0.72025604 0.72534079
      nan      nan 0.72278759 0.71429859 0.72363866 0.71429499
      nan      nan 0.72109989 0.72534439 0.72960332 0.72194374
      nan      nan 0.71768842 0.73046159 0.72280923 0.72706455
      nan      nan 0.71600433 0.71515687 0.70833393 0.7211071 ]
```

`warnings.warn(`

Out[246]:

```
▶      GridSearchCV
  ▶ estimator: DecisionTreeClassifier
    ▶ DecisionTreeClassifier
```

In [247]: `grid_search.best_params_`

Out[247]: `{'criterion': 'gini',`

`'max_depth': 4,`

In [248]: `dtc_cv=DecisionTreeClassifier(criterion= 'entropy',`
`'Splitter': 'best'}`
`max_depth=3,`
`max_features='sart'.`

```
In [248]: dtc_cv = DecisionTreeClassifier(criterion= 'entropy',
                                     max_depth=3,
                                     max_features='sqrt',
                                     splitter='best')
dtc_cv.fit(x_train,y_train)
```

```
Out[248]: DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=3, max_features='sqrt')
```

```
In [249]: pred=dtc_cv.predict(x_test)
```

```
In [250]: print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	0.69	0.99	0.81	201
1	0.60	0.03	0.06	93
accuracy			0.69	294
macro avg	0.64	0.51	0.44	294
weighted avg	0.66	0.69	0.57	294

In []: RANDOM FOREST

```
In [251]: from sklearn.ensemble import RandomForestClassifier  
rfc=RandomForestClassifier()
```

```
In [252]: forest_params = [ {'max_depth': list(range(10, 15)), 'max_features': list(range(0, 10))}]
```

```
In [253]: rfc_cv = GridSearchCV(rfc, param_grid=forest_params, cv=10, scoring="accuracy")
```

```
In [254]: rfc_cv.fit(x_train,y_train)
```

```
    return fit_method(estimator, *args, **kwargs)
C:\Users\KEERTHI KRISHANA\.conda\envs\tf\lib\site-packages\sklearn\base.py:11
52: DataConversionWarning: A column-vector y was passed when a 1d array was e
xpected. Please change the shape of y to (n_samples,), for example using ravel
[()]

```

```
    return fit_method(estimator, *args, **kwargs)
C:\Users\KEERTHI KRISHANA\.conda\envs\tf\lib\site-packages\sklearn\base.py:11
52: DataConversionWarning: A column-vector y was passed when a 1d array was e
xpected. Please change the shape of y to (n_samples,), for example using ravel
1().
```

```
    return fit_method(estimator, *args, **kwargs)
C:\Users\KEERTHI KRISHANA\.conda\envs\tf\lib\site-packages\sklearn\base.py:11
52: DataConversionWarning: A column-vector y was passed when a 1d array was e
xpected. Please change the shape of y to (n_samples,), for example using ravel
1()
```

```
    return fit_method(estimator, *args, **kwargs)
C:\Users\KEERTHI KRISHANA\.conda\envs\tf\lib\site-packages\sklearn\base.py:11
```

In [255]: ~~ValueError: dataconversionwarning: A column vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().~~

```
In [255]: expected_1D Please change the shape of y to (n_samples,), for example using ravel().
```

```
In [256]: print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	0.71	0.90	0.79	201
1	0.47	0.20	0.29	93
accuracy			0.68	294
macro avg	0.59	0.55	0.54	294
weighted avg	0.63	0.68	0.63	294

```
In [257]: rfc_cv.best_params_
```

```
Out[257]: {'max_depth': 10, 'max_features': 2}
```

```
In [ ]:
```