```
!pip install --upgrade scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Collecting scikit-learn
  Downloading scikit_learn-1.4.1.post1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.1/12.1 MB 53.9 MB/s eta 0:00:00
Requirement already satisfied: numpy<2.0,>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.25.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.4.0)
Installing collected packages: scikit-learn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.2.2
    Uninstalling scikit-learn-1.2.2:
      Successfully uninstalled scikit-learn-1.2.2
Successfully installed scikit-learn-1.4.1.post1
```

```
import sklearn
print(sklearn.__version__)
```

```
1.4.1.post1
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
#Load the data
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving IBM_HR_Final_cleaned_Data.xlsm to IBM_HR_Final_cleaned_Data.xlsm
```

```
df=pd.read_excel('IBM_HR_Final_cleaned_Data.xlsm')
```

```
df.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Environm
0	49	Travel_Frequently	279	Research & Development	8	1	
1	59	Non-Travel	1420	Human Resources	2	4	
2	59	Non-Travel	1420	Human Resources	2	4	
3	49	Travel_Frequently	279	Research & Development	8	1	
4	49	Travel_Frequently	279	Research & Development	8	1	

5 rows × 30 columns

```
df.tail()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Envi
19473	47	Travel_Rarely	465	Research & Development	1	3	
19474	38	Travel_Rarely	371	Research & Development	2	3	
19475	34	Travel_Rarely	629	Research & Development	27	2	
19476	55	Non-Travel	177	Research & Development	8	1	
19477	27	Travel_Rarely	1134	Research & Development	16	4	

5 rows × 30 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19478 entries, 0 to 19477
Data columns (total 30 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       19478 non-null  int64
 1   BusinessTravel            19478 non-null  object
 2   DailyRate                 19478 non-null  int64
 3   Department                19478 non-null  object
 4   DistanceFromHome          19478 non-null  int64
 5   Education                 19478 non-null  int64
 6   EnvironmentSatisfaction   19478 non-null  int64
 7   Gender                    19478 non-null  object
 8   HourlyRate                19478 non-null  int64
 9   JobInvolvement            19478 non-null  int64
 10  JobLevel                  19478 non-null  int64
 11  JobRole                   19478 non-null  object
 12  JobSatisfaction           19478 non-null  int64
 13  MaritalStatus             19478 non-null  object
 14  MonthlyIncome             19478 non-null  int64
 15  MonthlyRate               19478 non-null  int64
 16  NumCompaniesWorked        19478 non-null  int64
 17  OverTime                  19478 non-null  object
 18  PercentSalaryHike         19478 non-null  int64
 19  RelationshipSatisfaction  19478 non-null  int64
 20  StandardHours             19478 non-null  int64
 21  StockOptionLevel          19478 non-null  int64
 22  TotalWorkingYears         19478 non-null  int64
 23  TrainingTimesLastYear     19478 non-null  int64
 24  WorkLifeBalance           19478 non-null  int64
 25  YearsAtCompany            19478 non-null  int64
 26  YearsInCurrentRole        19478 non-null  int64
 27  YearsSinceLastPromotion   19478 non-null  int64
 28  YearsWithCurrManager      19478 non-null  int64
 29  Employee Source           19478 non-null  object
dtypes: int64(23), object(7)
memory usage: 4.5+ MB
```

in this dataset we have 19478 rows and 30 columns in which 23 are int datatype and 7 are object with no null values present

```
df.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfac
count	19478.000000	19478.000000	19478.000000	19478.000000	19478.00
mean	37.524489	812.367697	8.945836	2.924274	2.74
std	8.860420	402.778087	8.006485	1.026008	1.08
min	18.000000	102.000000	1.000000	1.000000	1.00
25%	31.000000	477.000000	2.000000	2.000000	2.00
50%	36.000000	813.000000	7.000000	3.000000	3.00
75%	43.000000	1176.000000	13.000000	4.000000	4.00
max	60.000000	1499.000000	29.000000	5.000000	4.00

8 rows × 23 columns

Now lets find null values

```
df.isnull().sum()/len(df)*100
```

```
Age                         0.0
BusinessTravel              0.0
DailyRate                   0.0
Department                  0.0
DistanceFromHome            0.0
Education                   0.0
EnvironmentSatisfaction     0.0
Gender                      0.0
HourlyRate                  0.0
JobInvolvement              0.0
JobLevel                    0.0
JobRole                     0.0
JobSatisfaction             0.0
MaritalStatus               0.0
MonthlyIncome               0.0
MonthlyRate                 0.0
NumCompaniesWorked          0.0
OverTime                    0.0
PercentSalaryHike           0.0
RelationshipSatisfaction    0.0
StandardHours               0.0
StockOptionLevel            0.0
TotalWorkingYears           0.0
TrainingTimesLastYear       0.0
WorkLifeBalance             0.0
YearsAtCompany              0.0
YearsInCurrentRole          0.0
YearsSinceLastPromotion     0.0
YearsWithCurrManager        0.0
Employee Source             0.0
dtype: float64
```

There is no null values present in this dataset

```
for i in df:
    print(i)
    print(df[i].unique())
```

```
[3 5 2 4 0 1 6]
WorkLifeBalance
[3 2 4 1]
YearsAtCompany
[10 14  6  0  8  3  5  2  1 37  7  4 20 17  9 15 16 11 12 13 22 18 25 21
 27 40 33 24 19 36 29 31 32 26 30 34 23]
YearsInCurrentRole
[ 7  8  5  0  2  3 10  4 15  1  9 13 11  6 12 16 14 18 17]
YearsSinceLastPromotion
[ 1  6  0  3  2  4  7  5  8 12 13 11 10  9 15 14]
YearsWithCurrManager
[ 7  9  3  8  0  2  4  6 12  5 15  1 10 11 17 13 16 14]
Employee Source
['Seek' 'Indeed' 'Referral' 'Company Website' 'Adzuna' 'GlassDoor' 'Jora'
 'LinkedIn' 'Recruit.net']
```

Separating cat and num columns

```
cat_col=df.select_dtypes(object)
cat_col
```

	BusinessTravel	Department	Gender	JobRole	MaritalStatus	OverTime	Er
0	Travel_Frequently	Research & Development	Male	Research Scientist	Married	No	
1	Non-Travel	Human Resources	Male	Research Scientist	Married	No	
2	Non-Travel	Human Resources	Male	Research Scientist	Married	No	
3	Travel_Frequently	Research & Development	Male	Research Scientist	Married	No	
4	Travel_Frequently	Research & Development	Male	Research Scientist	Married	No	
...	...	...	...	...	...	...	
19473	Travel_Rarely	Research & Development	Female	Laboratory Technician	Single	No	
19474	Travel_Rarely	Research &	Male	Research	Single	Yes	C

```
num_col=df.select_dtypes([int,float])
num_col
```

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRa
0	49	279	8	1	3	
1	59	1420	2	4	1	
2	59	1420	2	4	1	
3	49	279	8	1	3	
4	49	279	8	1	3	
...	...	...	...	...	...	
19473	47	465	1	3	4	
19474	38	371	2	3	4	
19475	34	629	27	2	4	
19476	55	177	8	1	4	
19477	27	1134	16	4	4	

19478 rows × 23 columns

```
for i in cat_col:
    print(i)
    print(cat_col[i].unique())
```

```
BusinessTravel
['Travel_Frequently' 'Non-Travel' 'Travel_Rarely']
Department
['Research & Development' 'Human Resources' 'Sales']
Gender
['Male' 'Female']
JobRole
```

```
['Research Scientist' 'Manufacturing Director' 'Laboratory Technician'
 'Sales Representative' 'Sales Executive' 'Manager' 'Human Resources'
 'Healthcare Representative' 'Research Director']
MaritalStatus
['Married' 'Single' 'Divorced']
OverTime
['No' 'Yes']
Employee Source
['Seek' 'Indeed' 'Referral' 'Company Website' 'Adzuna' 'GlassDoor' 'Jora'
 'LinkedIn' 'Recruit.net']
```

```
for i in num_col:
    print(i)
    print(num_col[i].unique())
```

```
 1448  601 1221  383 1109  264  918  788 1313 1186 1464  196  796  723
  415  337  937 1492  801  704  301 1120  469 1262 1308  984  174  718
  367 1384  902  669 1457 1421  150  179  363  107 1465 1098  969 1320
 1429  603  968  879  640  266  412 1138  325  634 1253 1202  256 1405
  999  285  404  683 1462  949  652  332  560  359  866 1326  748  990
 1193  271  333 1440  674  441 1342  898  350  992 1288 1108  479 1059
  457  241 1015 1387 1470  365  486 1037  392  567  148  786  370  146
  611  897 1054  181  734 1128 1180  431  572  352 1172 1079 1394 1239
  911 1162  234  468  613 1023  628  590  953  355  835  219 1096 1444
 1382 1378 1266  529]
DistanceFromHome
[ 8  2 26  3  9 10 13 23 18 24  1  5  7 27  6 25 20 16 15 19 21  4 11 29
 22 28 14 12 17]
Education
[1 4 3 2 5]
EnvironmentSatisfaction
[3 1 2 4]
HourlyRate
[ 61  37  95  87  56  79  62  40  33  57  42  81  67  90  44  66  53  31
  94  55  93  84  49  38  32  52  69  86  70  30  50  51  88  80  96  65
  78  45  46  41  82  99  58  39  48  63  72  83  97  75  73  98  36  47
  71  77  43  59  76  60  54 100  35  64  92  91  34  89  68  85  74]
JobInvolvement
[2 3 1 4]
JobLevel
[2 1 5 3 4]
JobSatisfaction
[2 3 1 4]
MonthlyIncome
[5130 6499 4810 ... 9991 5390 4404]
MonthlyRate
[24907 22656 26314 ...  5174 13243 10228]
NumCompaniesWorked
[1 2 3 6 4 9 0 7 8 5]
PercentSalaryHike
[23 13 14 15 11 16 12 17 21 20 22 18 19 24 25]
RelationshipSatisfaction
[4 3 2 1]
StandardHours
[80]
StockOptionLevel
[1 0 2 3]
TotalWorkingYears
[10 16  6 19  7  8  5  2 38  3 20 17  1 12 22 21 13  9 14 11  4 30 23 28
 15 32 24 31 26 37  0 40 29 18 25 34 36 35 33 27]
TrainingTimesLastYear
[3 5 2 4 0 1 6]
WorkLifeBalance
[3 2 4 1]
YearsAtCompany
[10 14  6  0  8  3  5  2  1 37  7  4 20 17  9 15 16 11 12 13 22 18 25 21
 27 40 33 24 19 36 29 31 32 26 30 34 23]
YearsInCurrentRole
[ 7  8  5  0  2  3 10  4 15  1  9 13 11  6 12 16 14 18 17]
YearsSinceLastPromotion
[ 1  6  0  3  2  4  7  5  8 12 13 11 10  9 15 14]
YearsWithCurrManager
[ 7  9  3  8  0  2  4  6 12  5 15  1 10 11 17 13 16 14]
```

Encoding

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
for i in cat_col:
    cat_col[i]=le.fit_transform(cat_col[i])
    print(cat_col[i].unique())
    print(le.classes_)
```

```
[1 0 2]
['Non-Travel' 'Travel_Frequently' 'Travel_Rarely']
[1 0 2]
['Human Resources' 'Research & Development' 'Sales']
[1 0]
['Female' 'Male']
[6 4 2 8 7 3 1 0 5]
['Healthcare Representative' 'Human Resources' 'Laboratory Technician'
 'Manager' 'Manufacturing Director' 'Research Director'
 'Research Scientist' 'Sales Executive' 'Sales Representative']
[1 2 0]
['Divorced' 'Married' 'Single']
[0 1]
['No' 'Yes']
[8 3 7 1 0 2 4 5 6]
['Adzuna' 'Company Website' 'GlassDoor' 'Indeed' 'Jora' 'LinkedIn'
 'Recruit.net' 'Referral' 'Seek']
```

```
cat_col.head()
```

	BusinessTravel	Department	Gender	JobRole	MaritalStatus	OverTime	Employee Source
0	1	1	1	6	1	0	8
1	0	0	1	6	1	0	8
2	0	0	1	6	1	0	8
3	1	1	1	6	1	0	8
4	1	1	1	6	1	0	8

```
new_df=pd.concat([cat_col,num_col],axis=1)
```

```
new_df
```

	BusinessTravel	Department	Gender	JobRole	MaritalStatus	OverTime	Employee Source
0	1	1	1	6	1	0	8
1	0	0	1	6	1	0	8
2	0	0	1	6	1	0	8
3	1	1	1	6	1	0	8
4	1	1	1	6	1	0	8
...	...	...	...	...	...	...	...
19473	2	1	0	2	2	0	4
19474	2	1	1	6	2	1	1
19475	2	1	0	6	2	0	1
19476	0	1	0	4	1	1	6
19477	2	1	1	1	1	1	1

19478 rows × 30 columns

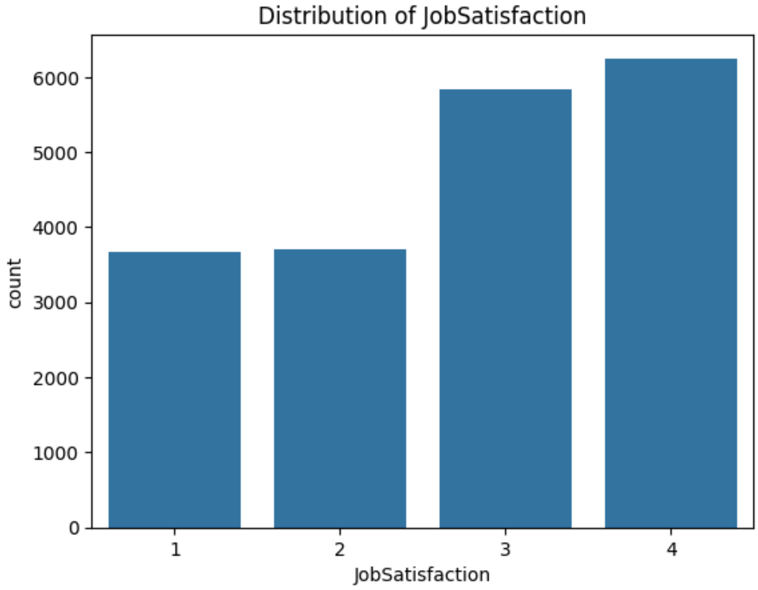
```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19478 entries, 0 to 19477
Data columns (total 30 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   BusinessTravel           19478 non-null  int64
 1   Department               19478 non-null  int64
 2   Gender                   19478 non-null  int64
 3   JobRole                  19478 non-null  int64
 4   MaritalStatus            19478 non-null  int64
 5   OverTime                 19478 non-null  int64
 6   Employee Source          19478 non-null  int64
 7   Age                      19478 non-null  int64
 8   DailyRate                19478 non-null  int64
 9   DistanceFromHome         19478 non-null  int64
```

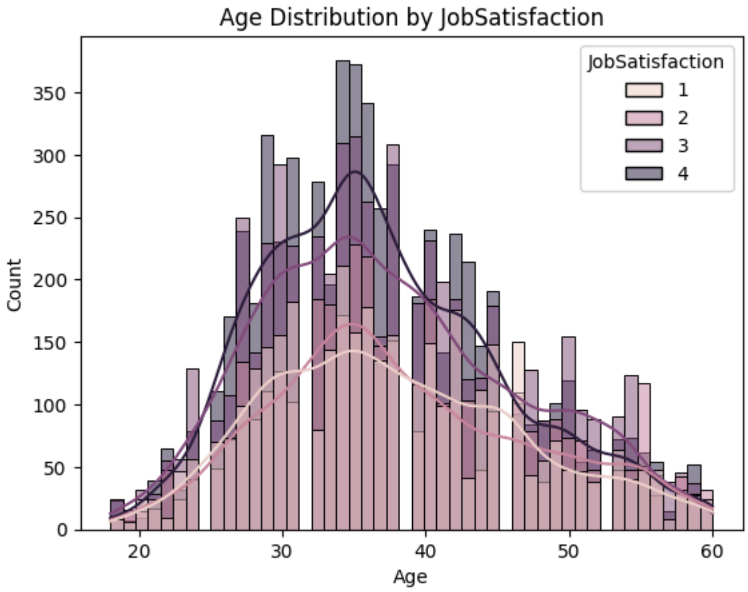
```
 10  Education                 19478 non-null  int64
 11  EnvironmentSatisfaction   19478 non-null  int64
 12  HourlyRate                19478 non-null  int64
 13  JobInvolvement            19478 non-null  int64
 14  JobLevel                  19478 non-null  int64
 15  JobSatisfaction           19478 non-null  int64
 16  MonthlyIncome             19478 non-null  int64
 17  MonthlyRate               19478 non-null  int64
 18  NumCompaniesWorked        19478 non-null  int64
 19  PercentSalaryHike         19478 non-null  int64
 20  RelationshipSatisfaction  19478 non-null  int64
 21  StandardHours             19478 non-null  int64
 22  StockOptionLevel          19478 non-null  int64
 23  TotalWorkingYears         19478 non-null  int64
 24  TrainingTimesLastYear     19478 non-null  int64
 25  WorkLifeBalance           19478 non-null  int64
 26  YearsAtCompany            19478 non-null  int64
 27  YearsInCurrentRole        19478 non-null  int64
 28  YearsSinceLastPromotion   19478 non-null  int64
 29  YearsWithCurrManager      19478 non-null  int64
dtypes: int64(30)
memory usage: 4.5 MB
```

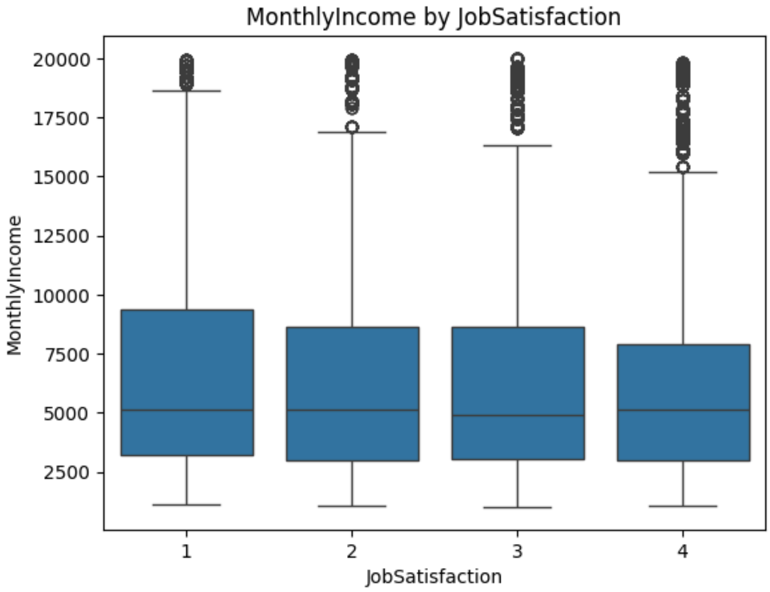
EDA

```
sns.countplot(data=df, x='JobSatisfaction')
plt.title('Distribution of JobSatisfaction')
plt.show()
```

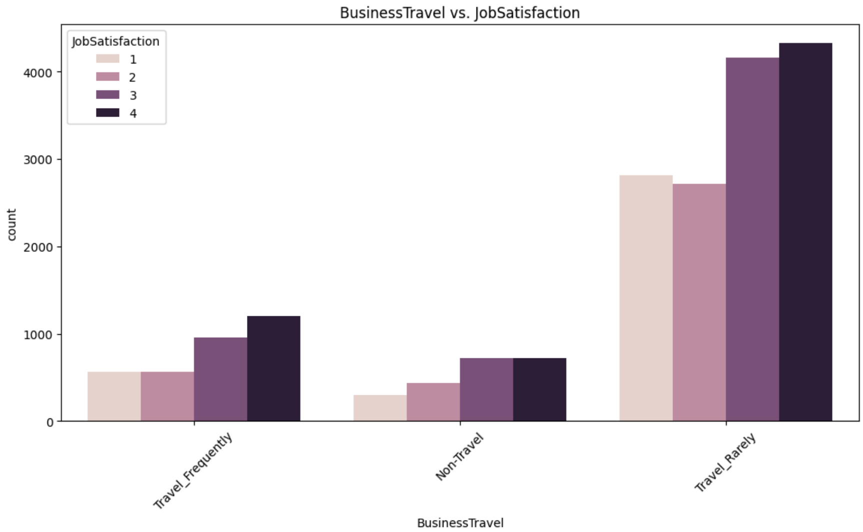


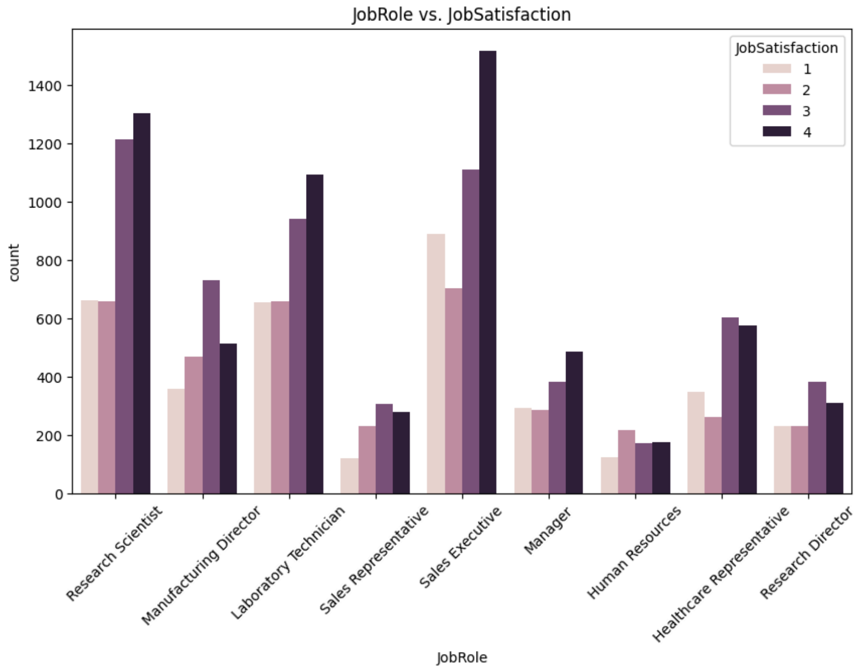

```
sns.histplot(data=df, x='Age', hue='JobSatisfaction', kde=True)
plt.title('Age Distribution by JobSatisfaction')
plt.show()

sns.boxplot(data=df, x='JobSatisfaction', y='MonthlyIncome')
plt.title('MonthlyIncome by JobSatisfaction')
plt.show()
```



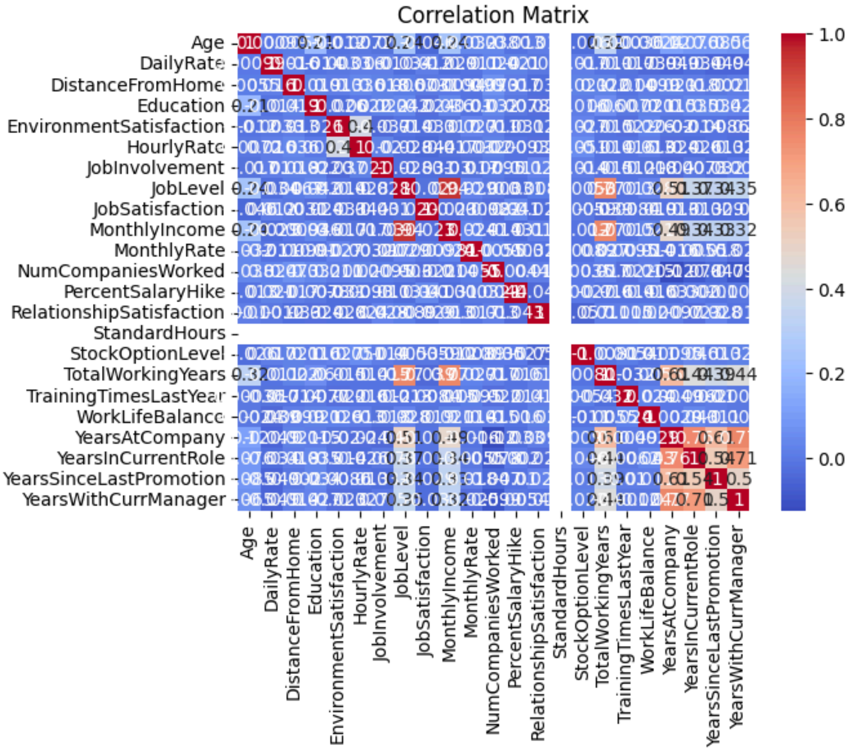



```
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='BusinessTravel', hue='JobSatisfaction')
plt.title('BusinessTravel vs. JobSatisfaction')
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='JobRole', hue='JobSatisfaction')
plt.title('JobRole vs. JobSatisfaction')
plt.xticks(rotation=45)
plt.show()
```

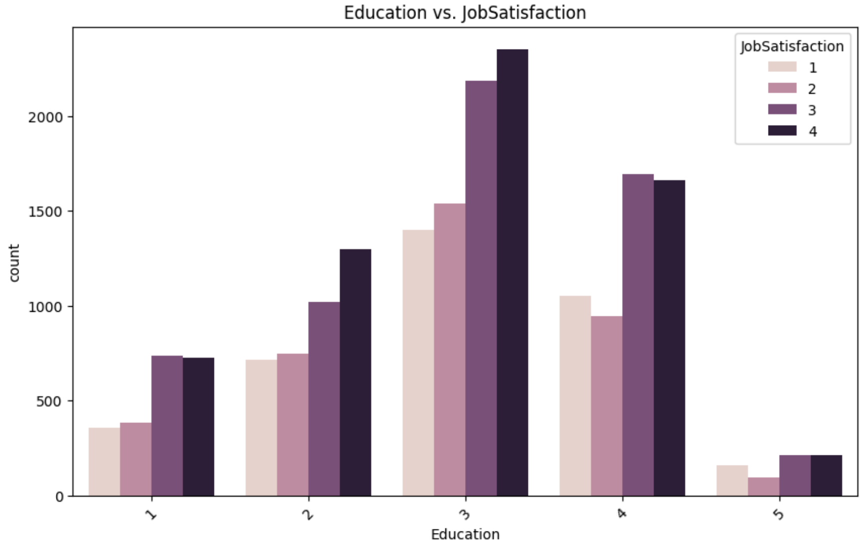




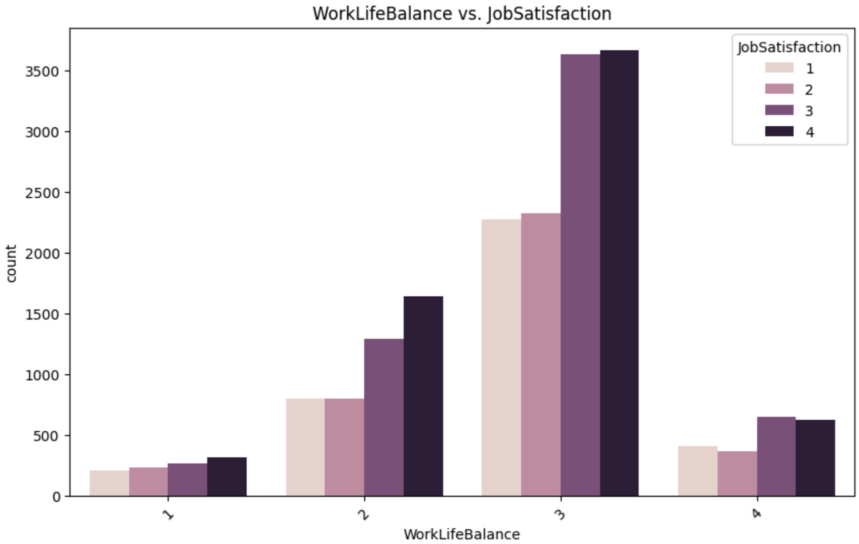

```
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



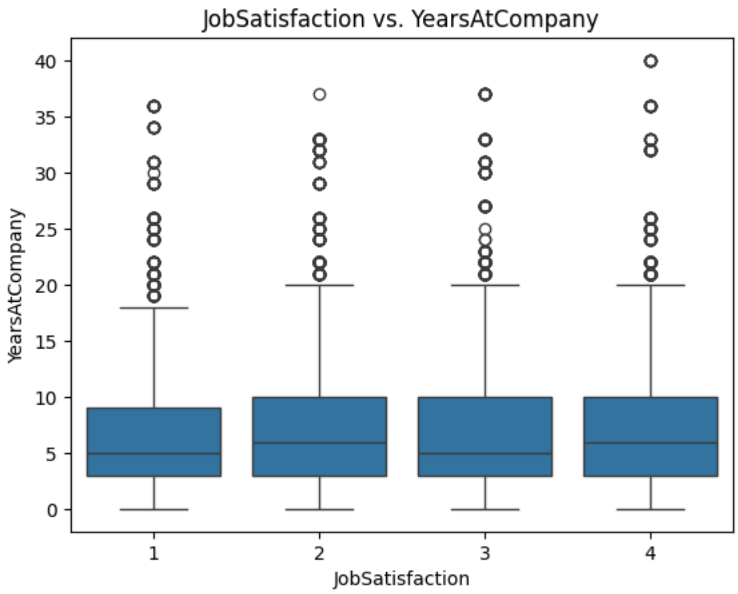

```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Education', hue='JobSatisfaction')
plt.title('Education vs. JobSatisfaction')
plt.xticks(rotation=45)
plt.show()
```



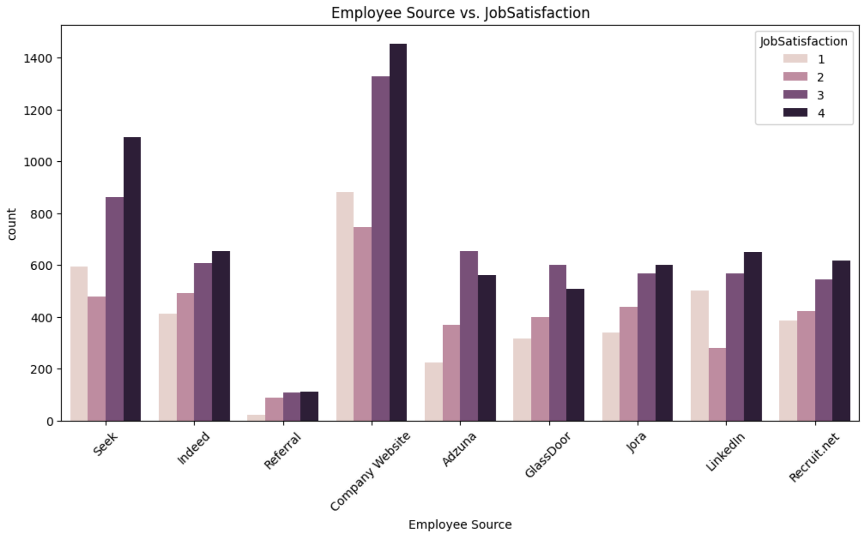

```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='WorkLifeBalance', hue='JobSatisfaction')
plt.title('WorkLifeBalance vs. JobSatisfaction')
plt.xticks(rotation=45)
plt.show()
```



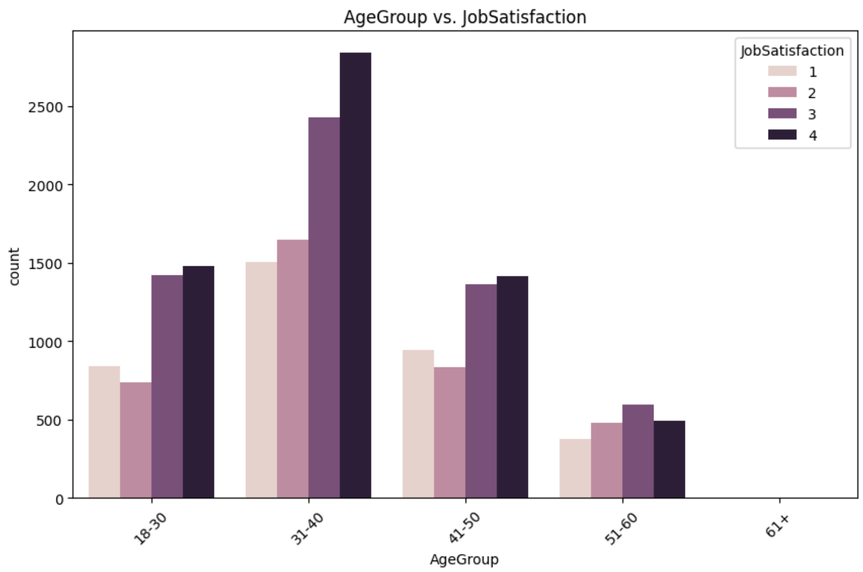

```
sns.boxplot(data=df, x='JobSatisfaction', y='YearsAtCompany')
plt.title('JobSatisfaction vs. YearsAtCompany')
plt.show()
```



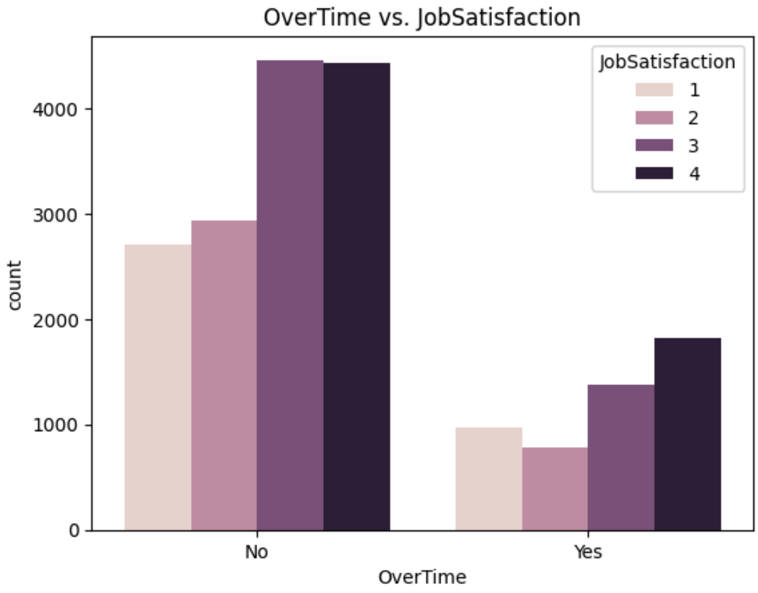

```
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='Employee Source', hue='JobSatisfaction')
plt.title('Employee Source vs. JobSatisfaction')
plt.xticks(rotation=45)
plt.show()
```



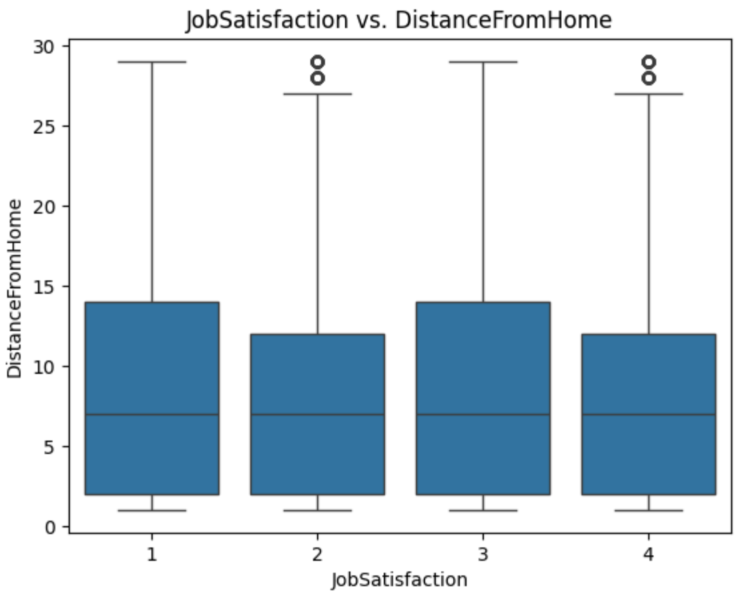

```
df['AgeGroup'] = pd.cut(df['Age'], bins=[18, 30, 40, 50, 60, np.inf], labels=['18-30', '31-40', '41-50', '51-60', '61+'])
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='AgeGroup', hue='JobSatisfaction')
plt.title('AgeGroup vs. JobSatisfaction')
plt.xticks(rotation=45)
plt.show()
```




```
sns.countplot(data=df, x='OverTime', hue='JobSatisfaction')
plt.title('OverTime vs. JobSatisfaction')
plt.show()
```



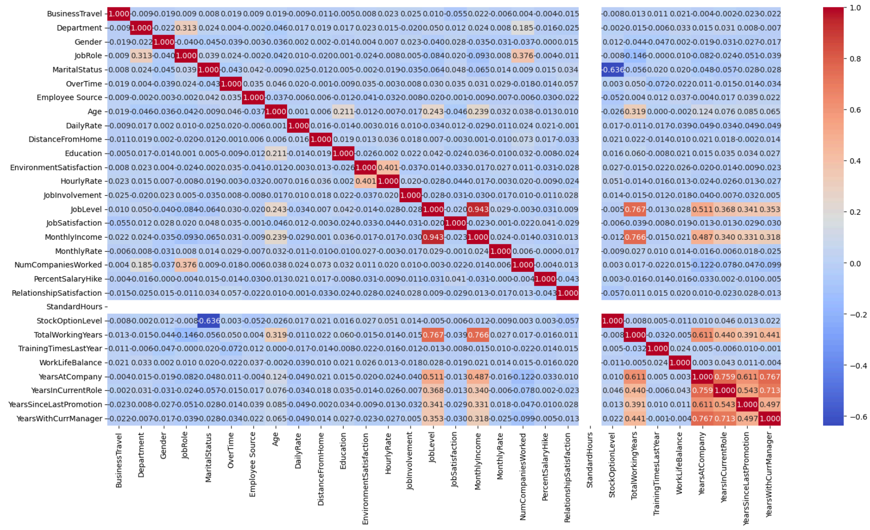

```
sns.boxplot(data=df, x='JobSatisfaction', y='DistanceFromHome')
plt.title('JobSatisfaction vs. DistanceFromHome')
plt.show()
```



Feature selection

```
plt.figure(figsize=(20,10))
sns.heatmap(new_df.corr(),annot=True,cmap='coolwarm',fmt=".3f")
```

```
<Axes: >
```

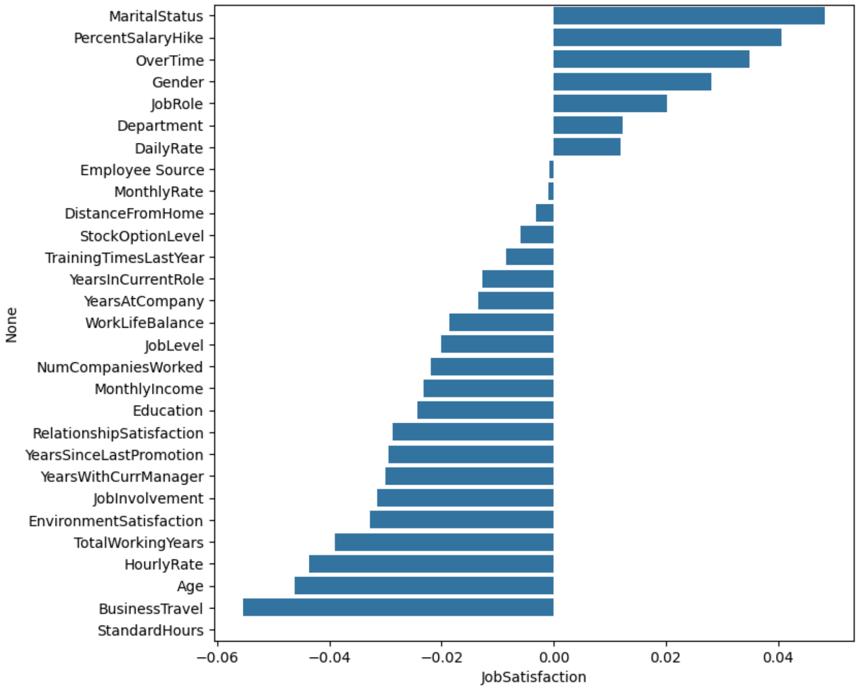


```
corr=new_df.corr()['JobSatisfaction'].reset_index()
corr.sort_values('JobSatisfaction',ascending=False)
```

	index	JobSatisfaction
15	JobSatisfaction	1.000000
4	MaritalStatus	0.048312
19	PercentSalaryHike	0.040541
5	OverTime	0.034882
2	Gender	0.028142
3	JobRole	0.020238
1	Department	0.012380
8	DailyRate	0.011897
6	Employee Source	-0.000816
17	MonthlyRate	-0.000979
9	DistanceFromHome	-0.003112
22	StockOptionLevel	-0.005881
24	TrainingTimesLastYear	-0.008429
27	YearsInCurrentRole	-0.012658
26	YearsAtCompany	-0.013406
25	WorkLifeBalance	-0.018608
14	JobLevel	-0.020063
18	NumCompaniesWorked	-0.021878
16	MonthlyIncome	-0.023214
10	Education	-0.024319
20	RelationshipSatisfaction	-0.028806
28	YearsSinceLastPromotion	-0.029380
29	YearsWithCurrManager	-0.030053
13	JobInvolvement	-0.031465
11	EnvironmentSatisfaction	-0.032697
23	TotalWorkingYears	-0.038994
12	HourlyRate	-0.043590
7	Age	-0.046104
0	BusinessTravel	-0.055352
21	StandardHours	NaN

```
corelation = pd.DataFrame(new_df.corr())
corelation = pd.DataFrame(corelation['JobSatisfaction'])
corelation=corelation.sort_values('JobSatisfaction',ascending=False)
indices_to_remove = ['JobSatisfaction']
corelation = corelation.drop(indices_to_remove)
plt.figure(figsize=(8,8))
sns.barplot(x=corelation['JobSatisfaction'],y=corelation.index)
```

```
<Axes: xlabel='JobSatisfaction', ylabel='None'>
```




```
new_df.columns
```

```
Index(['BusinessTravel', 'Department', 'Gender', 'JobRole', 'MaritalStatus',
       'OverTime', 'Employee Source', 'Age', 'DailyRate', 'DistanceFromHome',
       'Education', 'EnvironmentSatisfaction', 'HourlyRate', 'JobInvolvement',
       'JobLevel', 'JobSatisfaction', 'MonthlyIncome', 'MonthlyRate',
       'NumCompaniesWorked', 'PercentSalaryHike', 'RelationshipSatisfaction',
       'StandardHours', 'StockOptionLevel', 'TotalWorkingYears',
       'TrainingTimesLastYear', 'WorkLifeBalance', 'YearsAtCompany',
       'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

```
selected_columns = ['MaritalStatus', 'PercentSalaryHike', 'OverTime', 'Gender', 'JobRole', 'EnvironmentSatisfaction', 'TotalWorkingYear:
sat= new_df[selected_columns]
```

```
sat
```

	MaritalStatus	PercentSalaryHike	OverTime	Gender	JobRole	EnvironmentSatisf
0	1	23	0	1	6	
1	1	23	0	1	6	
2	1	23	0	1	6	
3	1	23	0	1	6	
4	1	23	0	1	6	
...	...	...	...	...	...	
19473	2	13	0	0	2	
19474	2	11	1	1	6	
19475	2	15	0	0	6	
19476	1	14	1	0	4	
19477	1	11	1	1	1	

19478 rows × 12 columns

Feature Selection

```
X=sat.drop('JobSatisfaction',axis=1)
X
```

	MaritalStatus	PercentSalaryHike	OverTime	Gender	JobRole	EnvironmentSatisf
0	1	23	0	1	6	
1	1	23	0	1	6	
2	1	23	0	1	6	
3	1	23	0	1	6	
4	1	23	0	1	6	
...	...	...	...	...	...	
19473	2	13	0	0	2	
19474	2	11	1	1	6	
19475	2	15	0	0	6	
19476	1	14	1	0	4	
19477	1	11	1	1	1	

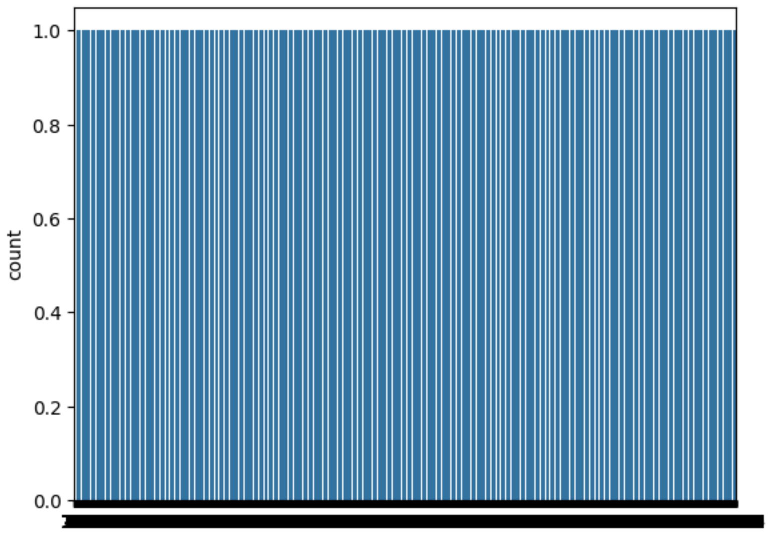
19478 rows × 11 columns

```
y=sat['JobSatisfaction']
y
```

```
0        2
1        2
2        2
3        2
4        2
        ..
19473    2
19474    4
19475    2
19476    1
19477    2
Name: JobSatisfaction, Length: 19478, dtype: int64
```

```
sns.countplot(df['JobSatisfaction'])
```

```
<Axes: ylabel='count'>
```



Train test split

```
from sklearn.model_selection import train_test_split
```

```
Xtrain,Xtest,ytrain,ytest=train_test_split(X,y,stratify=y,test_size=0.2,random_state=42)
```

```
Xtrain.shape , Xtest.shape
```

```
((15582, 11), (3896, 11))
```

```
ytrain.shape , ytest.shape
```

```
((15582,), (3896,))
```

Scaling

```
from sklearn.preprocessing import StandardScaler
se=StandardScaler()
```

```
Xtrain=se.fit_transform(Xtrain)
Xtest=se.fit_transform(Xtest)
```

Training models

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier

from sklearn.metrics import accuracy_score,classification_report
from sklearn.metrics import roc_auc_score,roc_curve
```

```
knn=KNeighborsClassifier(n_neighbors=3)
lr=LogisticRegression()
dt=DecisionTreeClassifier()
ra=RandomForestClassifier()
ad=AdaBoostClassifier()
svm=SVC(probability=True)
gau=GaussianNB()
bag=BaggingClassifier()
Gr=GradientBoostingClassifier()
```

```
Training_score= []
Testing_score= []
def model_building(model):
    model.fit(Xtrain, ytrain)
    ytrain_pred= model.predict(Xtrain)
    ytest_pred= model.predict(Xtest)
    a= accuracy_score(ytrain, ytrain_pred)
    b= accuracy_score(ytest, ytest_pred)
    Training_score.append(a)
    Testing_score.append(b)
    print(model)
    print("Train Data\n", accuracy_score(ytrain,ytrain_pred))
    print("Test Data\n", accuracy_score(ytest,ytest_pred))
```

```
model_building(knn)
```

```
KNeighborsClassifier(n_neighbors=3)
Train Data
 0.9896675651392632
Test Data
 0.981776180698152
```

```
model_building(lr)
```

```
LogisticRegression()
Train Data
 0.34539853677319987
Test Data
 0.3408624229979466
```

```
model_building(dt)
```

```
DecisionTreeClassifier()
Train Data
 1.0
Test Data
 0.8986139630390144
```

```
model_building(ra)
```

```
RandomForestClassifier()
Train Data
 1.0
Test Data
 0.9961498973305954
```

```
model_building(ad)
```

```
AdaBoostClassifier()
Train Data
 0.3762674881273264
Test Data
 0.36113963039014374
```

```
model_building(svm)
```

```
SVC(probability=True)
Train Data
 0.6419586702605571
Test Data
 0.6054928131416838
```

```
model_building(gau)
```

```
GaussianNB()
Train Data
 0.340007701193685
Test Data
 0.32931211498973306
```

```
model_building(bag)
```

```
BaggingClassifier()
Train Data
 0.9998074701578745
Test Data
 0.9840862422997947
```

```
model_building(Gr)
```

```
GradientBoostingClassifier()
Train Data
 0.6336798870491593
Test Data
 0.48716632443531827
```

```
Models= ["k-Nearest Neighbors","Logistic Regression" ,"Decision Tree Classifier", "Random forest Classifier" ,
        "Ada-Boosting Classifier","svm","GaussianNB","Bagging Classifier", "Gradiant- Bossting Classifier"]
```

```
new_df1 = pd.DataFrame({"Algorithms":Models,
                   "Training Score":Training_score,
                   "Testing Score":Testing_score,})
```

```
new_df1
```

	Algorithms	Training Score	Testing Score
5	svm	0.641959	0.605493

Hypertunning

Random forest

```
from sklearn.model_selection import RandomizedSearchCV
```

```
ra=RandomForestClassifier()
```

```
random_forest_params = {
    'n_estimators': [25,50,75,100],
    'max_depth': [2, 3, 5, 10, 20],
    'min_samples_leaf': [5, 10, 20, 50, 100],
    'min_samples_split': [2, 5, 10],
    'criterion': ["gini", "entropy"],
    'max_features': ['auto', 'sqrt'],
    'bootstrap': [True, False],
    'class_weight' : ["balanced", "balanced_subsample"]

}
```

```
ra_reg=RandomizedSearchCV(ra,param_distributions=random_forest_params,random_state=42,scoring='accuracy',cv=5,n_jobs=-1)
```

```
model_building(ra_reg)
```

```
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(), n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'class_weight': ['balanced',
                                                         'balanced_subsample'],
                                        'criterion': ['gini', 'entropy'],
                                        'max_depth': [2, 3, 5, 10, 20],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [5, 10, 20, 50,
                                                             100],
                                        'min_samples_split': [2, 5, 10],
                                        'n_estimators': [25, 50, 75, 100]},
                   random_state=42, scoring='accuracy')
Train Data
 0.835515338210756
Test Data
 0.7517967145790554
```

gaussianNB

```
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import GaussianNB
```