Employee attrition Classification

The issue of keeping one's employees happy and satisfied is a perennial and age-old challenge. If an employee you have invested so much time and money leaves for "greener pastures", then this would mean that you would have to spend even more time and money to hire somebody else. In the spirit of Kaggle, let us therefore turn to our predictive modelling capabilities and see if we can predict employee attrition on this synthetically generated IBM dataset.

In [1]:
```
import warnings
```

In [2]:
```
warnings.filterwarnings('ignore')
```

Let's Dive into it

Import necessary libraries

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Read 'HR-Employee-Attrition.csv' dataset and store it inside a variable

In [4]:
```
df=pd.read_csv('HR-Employee-Attrition.csv')
```

Check head

In [5]:
```
pd.set_option("display.max_columns", None)
```

In [6]:
```
df.head()
```

Out[6]:

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

Check last 5 rows

In [7]:
```
df.tail()
```

Out[7]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	E
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	
1468	49	No	Travel_Frequently	1023	Sales	2	3	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	

Check shape

In [8]:
```
df.shape
```

Out[8]:
```
(1470, 35)
```

View info about the dataset

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

View basic statistical information about the dataset

In [10]:
```
df.describe()
```

Out[10]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNu
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.0C
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.8€
std	9.135373	403.509100	8.106864	1.024165	0.0	602.0
min	18.000000	102.000000	1.000000	1.000000	1.0	1.0C
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.2
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.5C
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.7
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.0C

Check for null values

In [11]:
```
df.isna().sum()
```

Out[11]:
```
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
```

View unique values in all categorical columns

In [12]:
```
pd.set_option("display.max_columns", None)
df.head()
```

Out[12]:

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

In [13]:
```
categorical_columns=[ 'Attrition', 'BusinessTravel',  'Department',
         'EducationField', 'Gender', 'JobRole',
       'MaritalStatus','Over18', 'OverTime']
for i in categorical_columns:
    print("unique values in ",i,"are:" ,df[i].unique())
```

```
unique values in  Attrition are: ['Yes' 'No']
unique values in  BusinessTravel are: ['Travel_Rarely' 'Travel_Frequently' 'N
on-Travel']
unique values in  Department are: ['Sales' 'Research & Development' 'Human Re
sources']
unique values in  EducationField are: ['Life Sciences' 'Other' 'Medical' 'Mar
keting' 'Technical Degree'
 'Human Resources']
unique values in  Gender are: ['Female' 'Male']
unique values in  JobRole are: ['Sales Executive' 'Research Scientist' 'Labor
atory Technician'
 'Manufacturing Director' 'Healthcare Representative' 'Manager'
 'Sales Representative' 'Research Director' 'Human Resources']
unique values in  MaritalStatus are: ['Single' 'Married' 'Divorced']
unique values in  Over18 are: ['Y']
unique values in  OverTime are: ['Yes' 'No']
```

Check the number of unique values in all columns

In [14]:
```
columns_nunique=['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Departmen
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager']
```

In [15]:
```
for i in columns_nunique:
    print("Number of unique elements in ",i,"are:" ,df[i].nunique())
```

```
Number of unique elements in  Age are: 43
Number of unique elements in  Attrition are: 2
Number of unique elements in  BusinessTravel are: 3
Number of unique elements in  DailyRate are: 886
Number of unique elements in  Department are: 3
Number of unique elements in  DistanceFromHome are: 29
Number of unique elements in  Education are: 5
Number of unique elements in  EducationField are: 6
Number of unique elements in  EmployeeCount are: 1
Number of unique elements in  EmployeeNumber are: 1470
Number of unique elements in  EnvironmentSatisfaction are: 4
Number of unique elements in  Gender are: 2
Number of unique elements in  HourlyRate are: 71
Number of unique elements in  JobInvolvement are: 4
Number of unique elements in  JobLevel are: 5
Number of unique elements in  JobRole are: 9
Number of unique elements in  JobSatisfaction are: 4
Number of unique elements in  MaritalStatus are: 3
Number of unique elements in  MonthlyIncome are: 1349
Number of unique elements in  MonthlyRate are: 1427
Number of unique elements in  NumCompaniesWorked are: 10
Number of unique elements in  Over18 are: 1
Number of unique elements in  OverTime are: 2
Number of unique elements in  PercentSalaryHike are: 15
Number of unique elements in  PerformanceRating are: 2
Number of unique elements in  RelationshipSatisfaction are: 4
Number of unique elements in  StandardHours are: 1
Number of unique elements in  StockOptionLevel are: 4
Number of unique elements in  TotalWorkingYears are: 40
Number of unique elements in  TrainingTimesLastYear are: 7
Number of unique elements in  WorkLifeBalance are: 4
Number of unique elements in  YearsAtCompany are: 37
Number of unique elements in  YearsInCurrentRole are: 19
Number of unique elements in  YearsSinceLastPromotion are: 16
Number of unique elements in  YearsWithCurrManager are: 18
```

Print out the names of the columns having only one unique values

In [16]:
```
for i in columns_nunique:
    if df[i].nunique()==1:
        print(i)
```

```
EmployeeCount
Over18
StandardHours
```

Drop these columns as they won't be useful in our predicition

In [17]:
```
df.drop(columns=["EmployeeCount",'Over18',"StandardHours"],inplace=True)
```

In [18]:
```
df.head()
```

Out[18]:

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

Drop EmployeeNumber column aswell

In [19]:
```
df.drop(columns=["EmployeeNumber"],inplace=True)
```

In [20]:
```
df.head()
```

Out[20]:

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

Create following groupby valuecounts

```
In [21]: df.groupby(["Department","EducationField","Gender"],observed=True,dropna=False
```

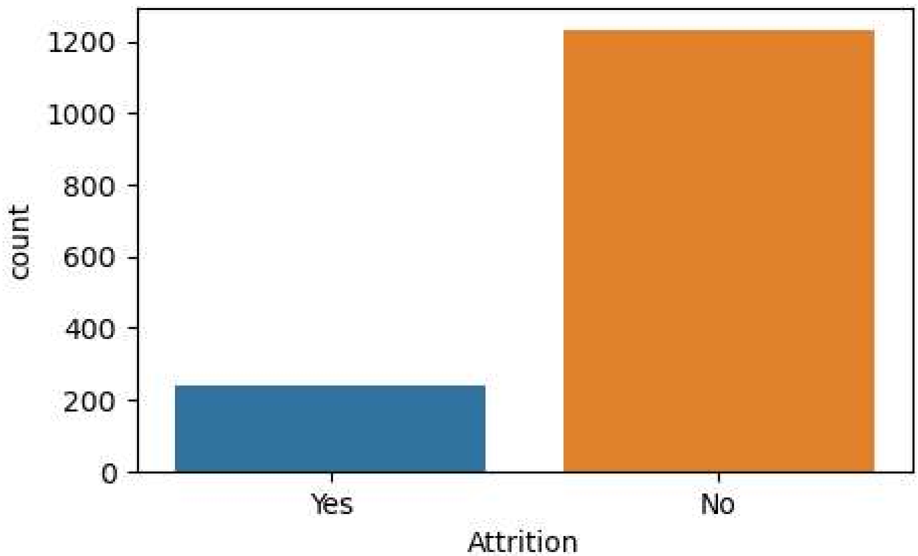
Out[21]:

Department	EducationField	Gender	Age	DailyRate	DistanceFromHome	Education	Environ
Human Resources	Human Resources	Female	310	6084	96	25	
		Male	690	12148	148	59	
	Life Sciences	Female	352	8293	103	28	
		Male	278	3756	36	21	
	Medical	Female	57	2318	28	4	
		Male	461	9065	90	30	
	Other	Male	104	3015	26	9	
	Technical Degree	Female	34	1107	9	4	
		Male	96	1561	12	7	
Research & Development	Life Sciences	Female	6060	125412	1408	494	
		Male	10219	221834	2492	787	
	Medical	Female	5788	125306	1523	442	
		Male	7731	174434	1941	589	
	Other	Female	919	16866	218	71	
		Male	1397	31989	357	127	
	Technical Degree	Female	1395	31396	303	111	
		Male	2089	48147	546	165	
Sales	Life Sciences	Female	2570	57199	583	196	
		Male	3008	70988	805	249	
	Marketing	Female	2646	53745	634	217	
		Male	3384	61981	973	280	
	Medical	Female	1224	30600	332	85	
		Male	1832	40056	426	151	
	Other	Female	105	5300	38	11	
		Male	375	8104	93	34	
	Technical Degree	Female	490	11639	141	37	
		Male	664	17311	152	49	

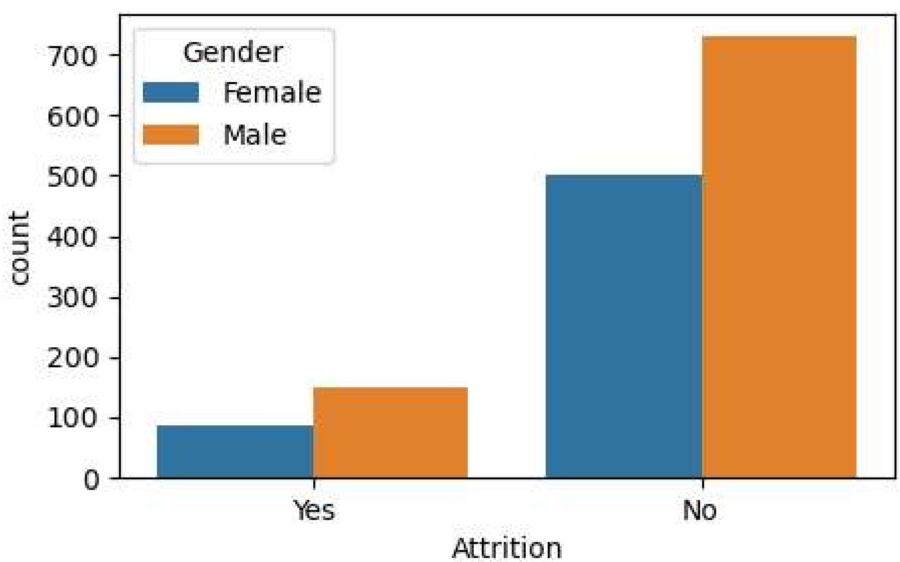
Data Visualization

Plot the following

In [22]:
```
plt.figure(figsize=(5,3))
sns.countplot(x='Attrition',data=df)
plt.show()
```

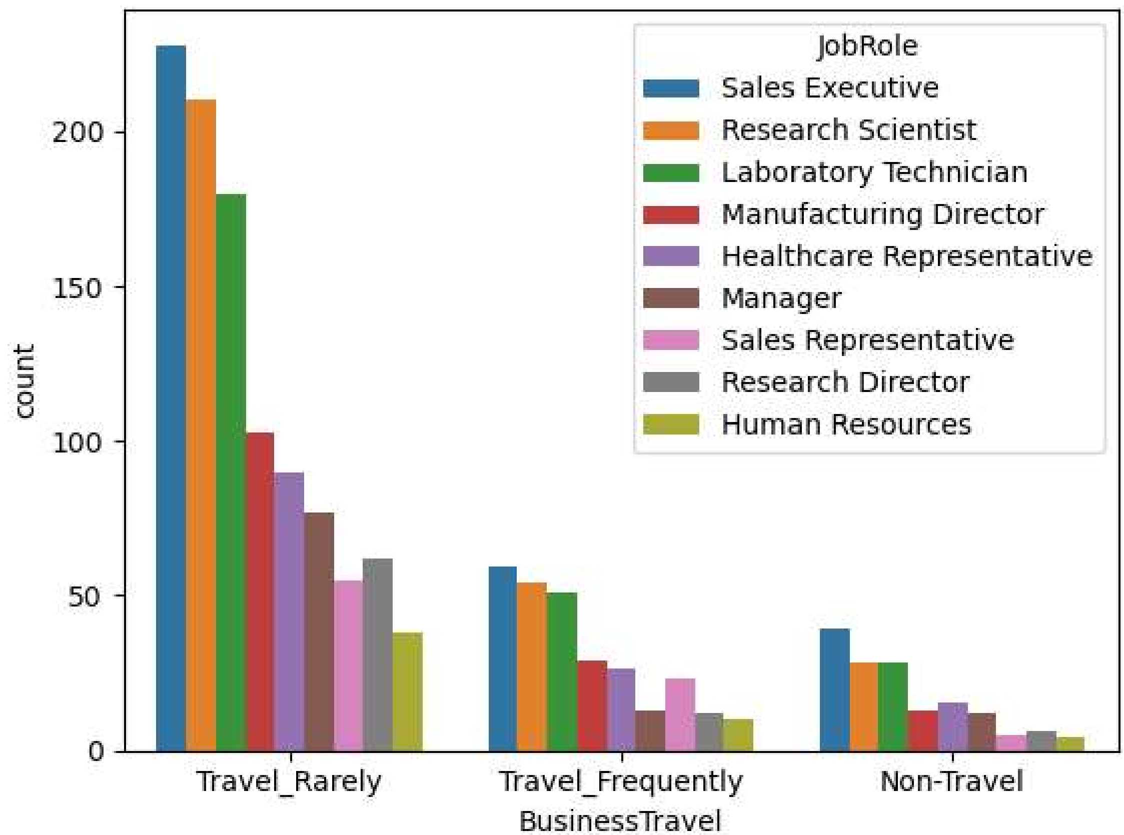


In [23]:
```
plt.figure(figsize=(5,3))
sns.countplot(x='Attrition',hue='Gender',data=df)
plt.show()
```

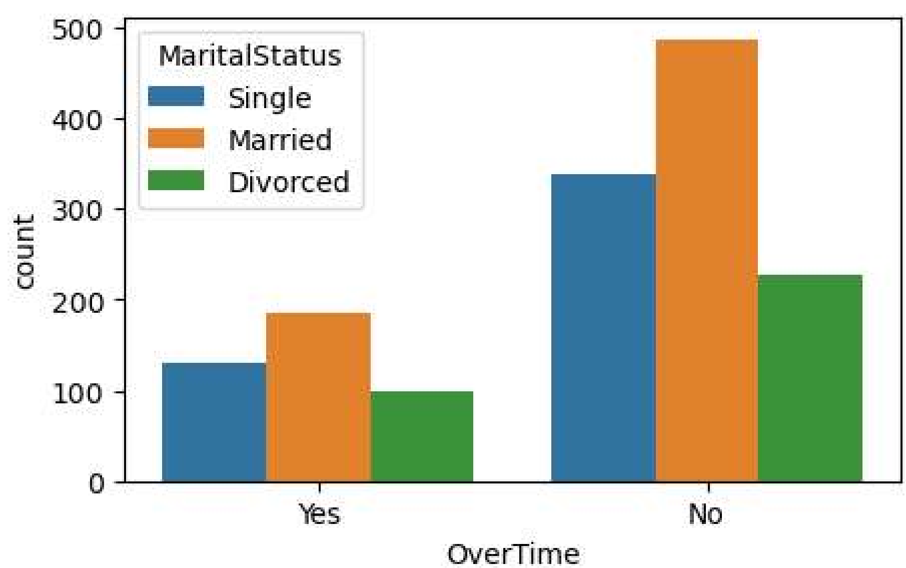


In [24]:
```
sns.countplot(x='BusinessTravel',hue='JobRole',data=df)
```

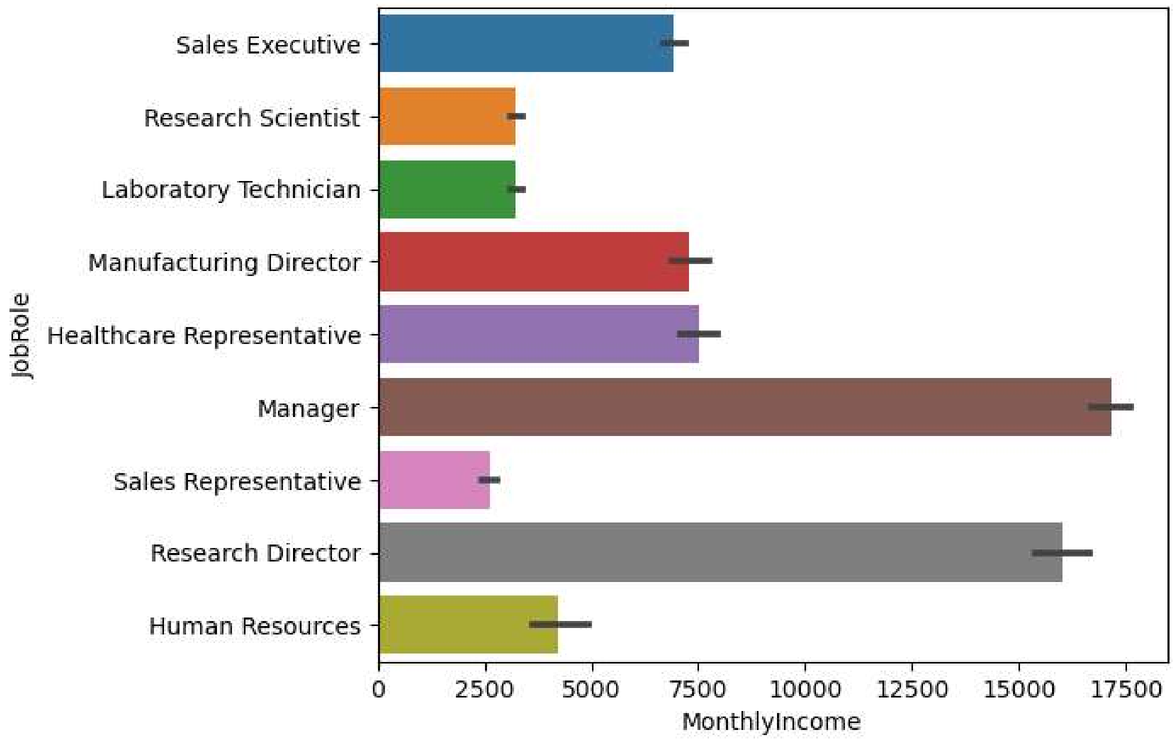
Out[24]: <AxesSubplot:xlabel='BusinessTravel', ylabel='count'>



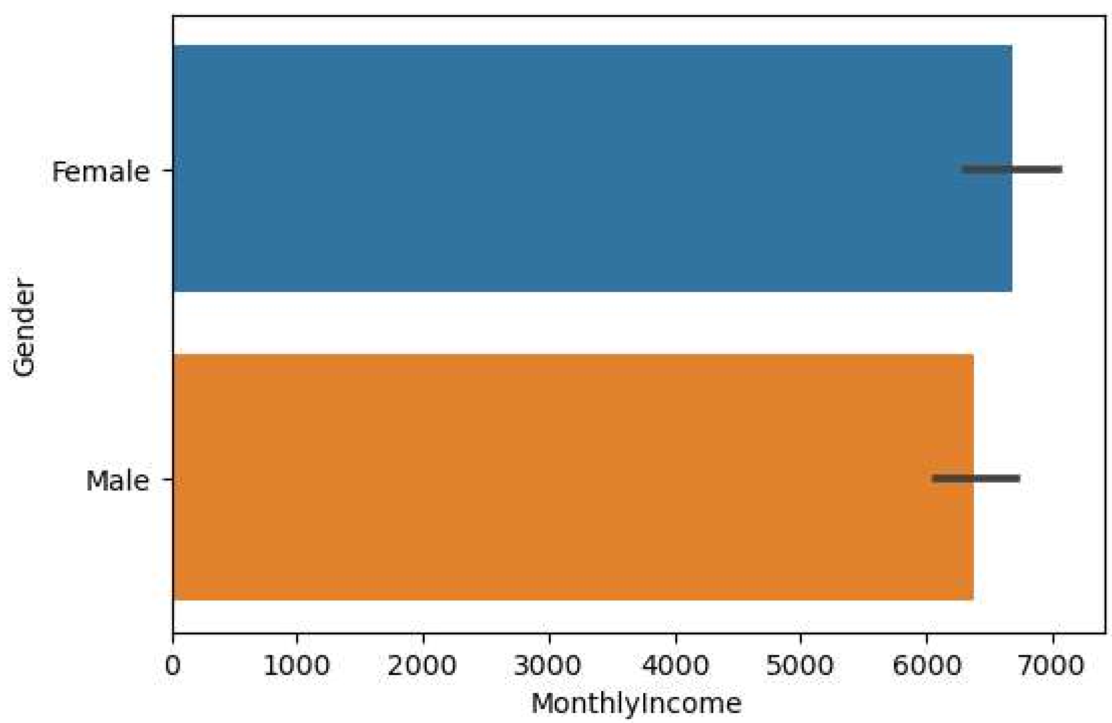
In [25]:
```
plt.figure(figsize=(5,3))
sns.countplot(x='OverTime',hue='MaritalStatus',data=df)
plt.show()
```



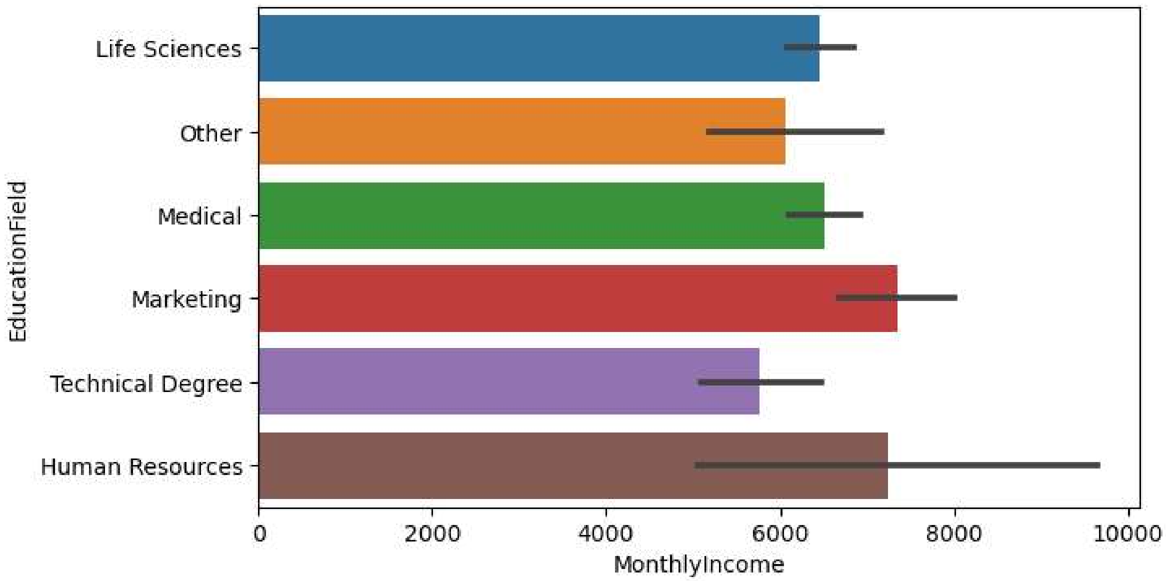
In [26]:
```
plt.figure(figsize=(6,5))
sns.barplot(y='JobRole',x='MonthlyIncome',data=df)
plt.show()
```



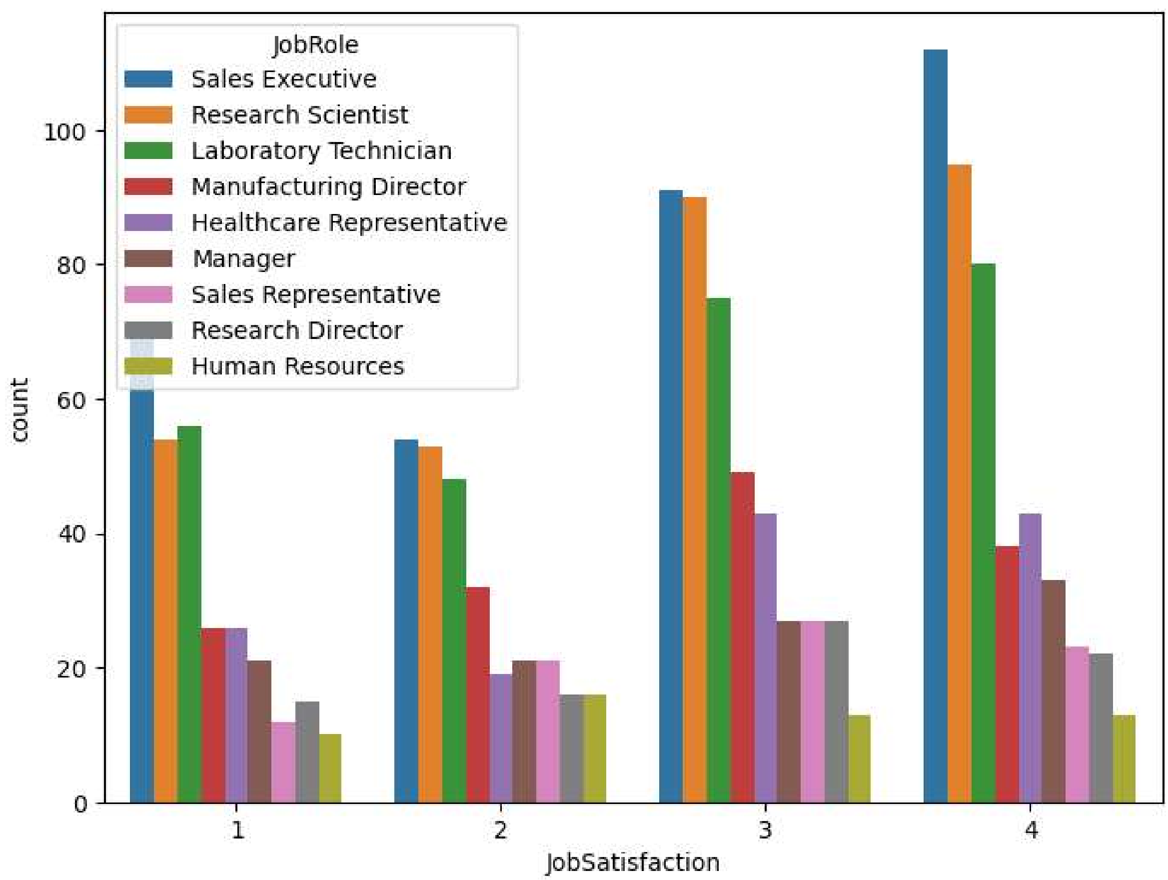
In [27]:
```
plt.figure(figsize=(6,4))
sns.barplot(y='Gender',x='MonthlyIncome',data=df)
plt.show()
```



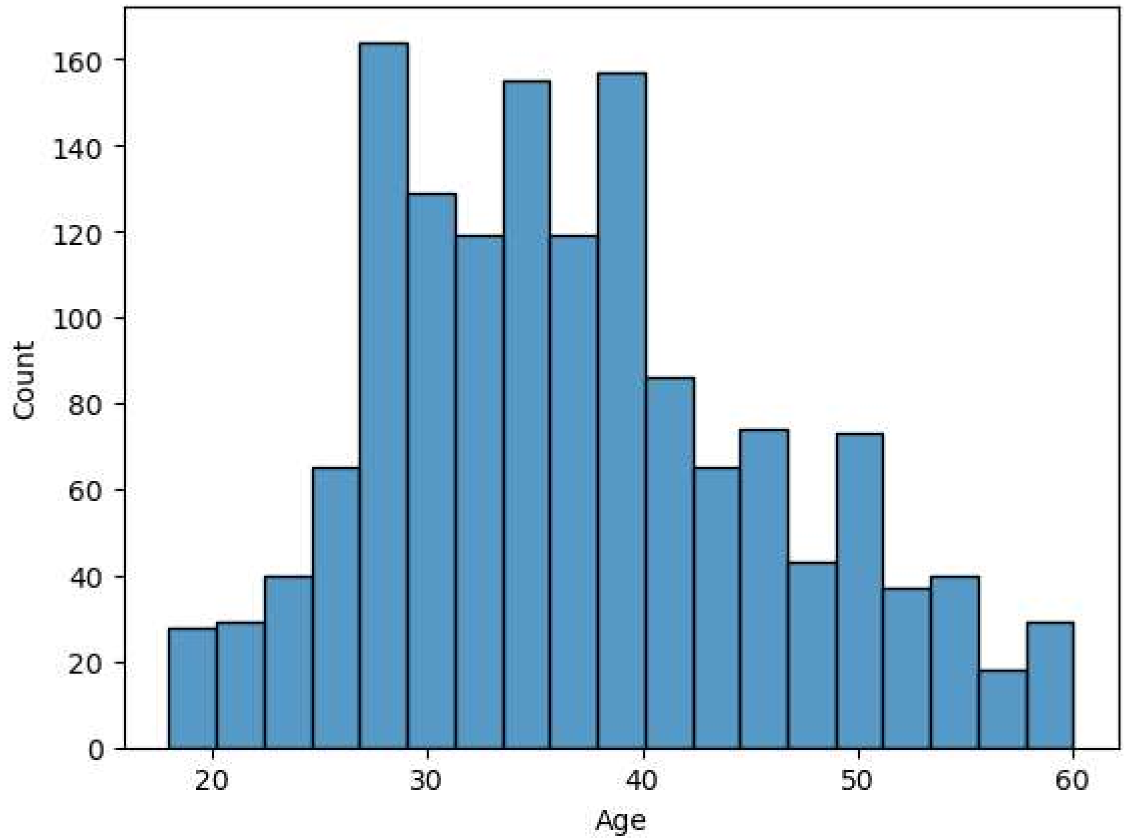
In [28]:
```
plt.figure(figsize=(7,4))
sns.barplot(y='EducationField',x='MonthlyIncome',data=df)
plt.show()
```



In [29]:
```
plt.figure(figsize=(8,6))
sns.countplot(x='JobSatisfaction',hue='JobRole',data=df)
plt.show()
```



In [30]:
```
sns.histplot(df['Age'],kde=False);
plt.show()
```



Data Preprocessing

Convert Attrition from ('Yes', 'No') to (1,0)

In [31]:
```
def attrition(x):
    if x == "No":
        return 0
    else:
        return 1
```

In [32]:
```
df['Attrition'] = df['Attrition'].apply(attrition)
```

In [33]:
```python
df.head()
```

Out[33]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educ
0	41	1	Travel_Rarely	1102	Sales	1	2	Lif
1	49	0	Travel_Frequently	279	Research & Development	8	1	Lif
2	37	1	Travel_Rarely	1373	Research & Development	2	2	
3	33	0	Travel_Frequently	1392	Research & Development	3	4	Lif
4	27	0	Travel_Rarely	591	Research & Development	2	1	

Convert the rest of the categorical values into numeric using dummy variables and store the results in a new dataframe called 'newdf'

In [34]:
```python
newdf=pd.get_dummies(df,drop_first=True)
newdf.head()
```

Out[34]:

	Age	Attrition	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate
0	41	1	1102	1	2	2	94
1	49	0	279	8	1	3	61
2	37	1	1373	2	2	4	92
3	33	0	1392	3	4	4	56
4	27	0	591	2	1	1	40

Check the shape of our new dataset

In [35]:
```python
newdf.shape
```

Out[35]:
```
(1470, 45)
```

Print unique values in our new dataframe

In [36]:
```
ncategorical_columns=['Age', 'Attrition', 'DailyRate', 'DistanceFromHome', 'Ed
       'EnvironmentSatisfaction', 'HourlyRate', 'JobInvolvement', 'JobLevel',
       'JobSatisfaction', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked'
       'PercentSalaryHike', 'PerformanceRating', 'RelationshipSatisfaction',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager',
       'BusinessTravel_Travel_Frequently', 'BusinessTravel_Travel_Rarely',
       'Department_Research & Development', 'Department_Sales',
       'EducationField_Life Sciences', 'EducationField_Marketing',
       'EducationField_Medical', 'EducationField_Other',
       'EducationField_Technical Degree', 'Gender_Male',
       'JobRole_Human Resources', 'JobRole_Laboratory Technician',
       'JobRole_Manager', 'JobRole_Manufacturing Director',
       'JobRole_Research Director', 'JobRole_Research Scientist',
       'JobRole_Sales Executive', 'JobRole_Sales Representative',
       'MaritalStatus_Married', 'MaritalStatus_Single', 'OverTime_Yes']
```

In [37]:
```
for i in ncategorical_columns:
    print("Unique values in",i,"are",newdf[i].unique())
```

```
2 14 22 15 27 21 17 11 13 37 16 20
 40 24 33 19 36 18 29 31 32 34 26 30 23]
Unique values in YearsInCurrentRole are [ 4  7  0  2  5  9  8  3  6 13  1
15 14 16 11 10 12 18 17]
Unique values in YearsSinceLastPromotion are [ 0  1  3  2  7  4  8  6  5 1
5  9 13 12 10 11 14]
Unique values in YearsWithCurrManager are [ 5  7  0  2  6  8  3 11 17  1
4 12  9 10 15 13 16 14]
Unique values in BusinessTravel_Travel_Frequently are [0 1]
Unique values in BusinessTravel_Travel_Rarely are [1 0]
Unique values in Department_Research & Development are [0 1]
Unique values in Department_Sales are [1 0]
Unique values in EducationField_Life Sciences are [1 0]
Unique values in EducationField_Marketing are [0 1]
Unique values in EducationField_Medical are [0 1]
Unique values in EducationField_Other are [0 1]
Unique values in EducationField_Technical Degree are [0 1]
Unique values in Gender_Male are [0 1]
Unique values in JobRole_Human Resources are [0 1]
```

Split the columns into input and target variables (X and y)

In [38]:
```
X=newdf.drop(columns=['Attrition'])
y=newdf['Attrition']
```

Carry out Feature scaling using StandardScaler

In [39]:
```
from sklearn.preprocessing import StandardScaler
```

In [40]:
```
scalar=StandardScaler()
```

In [41]:
```
X_scalar=scalar.fit_transform(X)
```

In [42]:
```
scaled_X=pd.DataFrame(X_scalar,columns=X.columns)
```

In [43]:
```
scaled_X.head()
```

Out[43]:

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	Jol
0	0.446350	0.742527	-1.010909	-0.891688	-0.660531	1.383138	
1	1.322365	-1.297775	-0.147150	-1.868426	0.254625	-0.240677	
2	0.008343	1.414363	-0.887515	-0.891688	1.169781	1.284725	
3	-0.429664	1.461466	-0.764121	1.061787	1.169781	-0.486709	
4	-1.086676	-0.524295	-0.887515	-1.868426	-1.575686	-1.274014	

Split the dataset into training and testing set

In [44]:
```
from sklearn.model_selection import train_test_split
```

In [45]:
```
X_test,X_train,y_test,y_train=train_test_split(X,y,test_size=0.3)
```

Machine Learning Models

Logistic Regression

In [46]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import cross_val_score
```

In [47]:
```
model=LogisticRegression()
```

In [48]:
```
model.fit(X_train,y_train)
```

Out[48]:
```
LogisticRegression()
```

In [49]:
```
y_pred=model.predict(X_test)
```

In [90]:
```
a=metrics.accuracy_score(y_test,y_pred)
a
```

Out[90]:
```
0.8347910592808552
```

In [51]:
```
metrics.confusion_matrix(y_test,y_pred)
```

Out[51]:
```
array([[856,   3],
       [166,   4]], dtype=int64)
```

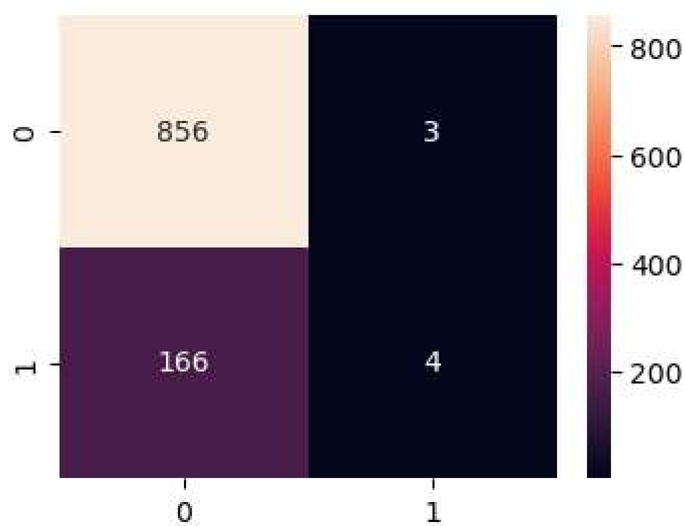
In [52]:
```
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      1.00      0.91       859
           1       0.57      0.02      0.05       170

    accuracy                           0.84      1029
   macro avg       0.70      0.51      0.48      1029
weighted avg       0.79      0.84      0.77      1029
```

In [53]:
```
metrics.confusion_matrix(y_test,y_pred)
```

Out[53]:
```
array([[856,   3],
       [166,   4]], dtype=int64)
```

In [54]:
```
plt.figure(figsize=(4,3))
sns.heatmap(metrics.confusion_matrix(y_test,y_pred),annot=True,fmt='d')
plt.show()
```



Random Forest Classifier

** Choose the best estimator and parameters :GridSearchCV**

In [55]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [56]:
```
model1 = RandomForestClassifier(n_estimators= 100)
```

In [57]:
```
model1.fit(X_train, y_train)
```

Out[57]:
```
RandomForestClassifier()
```

In [58]:
```
forest_params = [{'max_depth': [0.5,1,5,10], 'max_features': list(range(0,14))
```

In [59]:
```
clf = GridSearchCV(model1, forest_params,scoring='accuracy')
```

In [60]:
```
clf.fit(X_train, y_train)
```

Out[60]:
```
GridSearchCV(estimator=RandomForestClassifier(),
             param_grid=[{'max_depth': [0.5, 1, 5, 10],
                          'max_features': [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
11,
                                           12, 13]}],
             scoring='accuracy')
```

In [61]:
```
clf.best_params_
```

Out[61]:
```
{'max_depth': 10, 'max_features': 6}
```

In [62]:
```
clf.best_score_
```

Out[62]:
```
0.8662665985699693
```

In [63]:
```
clf.best_estimator_
```

Out[63]:
```
RandomForestClassifier(max_depth=10, max_features=6)
```

Create Random forest model with the best parameters

In [91]:
```
b=model1.score(X_train, y_train)
b
```

Out[91]:
```
1.0
```

In [65]:
```
y_pred = model1.predict(X_test)
```

In [66]:
```
metrics.accuracy_score(y_test, y_pred)
```

Out[66]:
```
0.8425655976676385
```

In [67]:
```
metrics.confusion_matrix(y_test, y_pred)
```

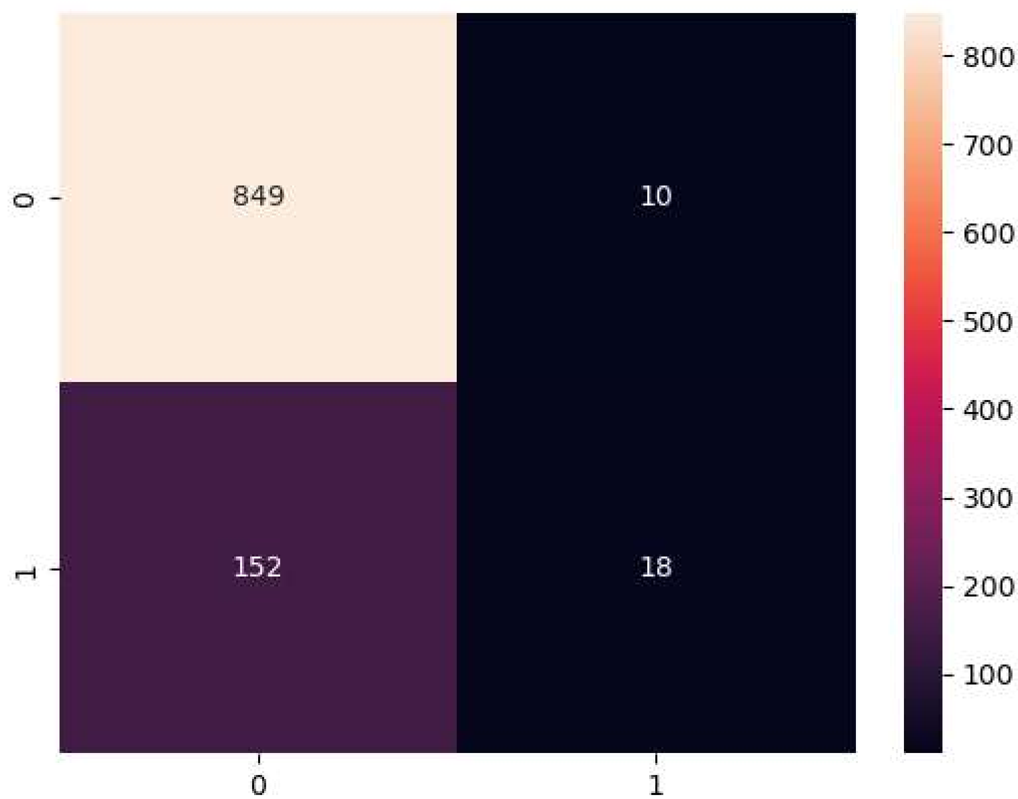
Out[67]:
```
array([[849,  10],
       [152,  18]], dtype=int64)
```

In [68]:
```
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.99      0.91       859
           1       0.64      0.11      0.18       170

    accuracy                           0.84      1029
   macro avg       0.75      0.55      0.55      1029
weighted avg       0.81      0.84      0.79      1029
```

Visualize confusion matrix

In [70]:
```
sns.heatmap(metrics.confusion_matrix(y_test,y_pred), annot = True, fmt = 'd')
plt.show()
```



Support Vector Machine

In [71]:
```
from sklearn.svm import SVC
```

In [72]:
```
model = SVC()
```

In [73]:
```
model.fit(X_train, y_train)
```

Out[73]:
```
SVC()
```

In [92]:
```
c=model.score(X_train,y_train)
c
```

Out[92]:
```
0.8480725623582767
```

In [75]:
```
y_pred = model.predict(X_test)
```

In [76]:
```
from sklearn import metrics
```

In [77]:
```
metrics.accuracy_score(y_test, y_pred)
```

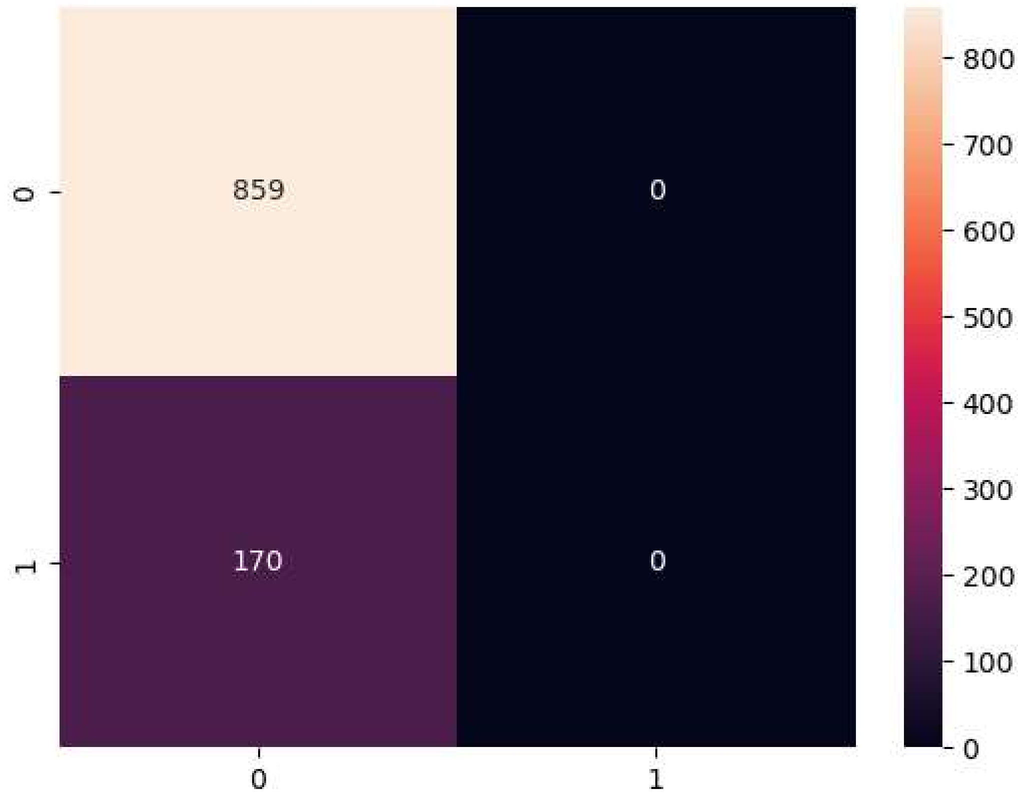
Out[77]:
```
0.8347910592808552
```

In [78]:
```
metrics.confusion_matrix(y_test,y_pred)
```

Out[78]:
```
array([[859,   0],
       [170,   0]], dtype=int64)
```

Visualize confusion matrix

In [79]:
```
sns.heatmap(metrics.confusion_matrix(y_test,y_pred), annot = True, fmt = 'd')
plt.show()
```



View score of different models in one dataframe

In [102]:
```
asd=([[a],[b],[c]])
pd.DataFrame(asd)
```

Out[102]:

	0
0	0.834791
1	1.000000
2	0.848073

Use PCA to reduce dimensionality of the data

Import PCA and fit our X_train

In [ ]:
```
from sklearn.decomposition import PCA
```

In [ ]:
```
PCA(n_components = 0.95)
```

Apply the mapping (transform) to both the training set and the test set.

In [ ]:
```
train_X = PCA.transform(X_train)
test_X = PCA.transform(X_test)
```

Import machine learning model of our choice, we are going with RandomForest for this problem

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
```

Create RandomForest model with the best parameter we got earlier and train it

In [112]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
model1 = RandomForestClassifier(n_estimators= 100)
model1.fit(X_train, y_train)
```

Out[112]:
```
RandomForestClassifier()
```

Check the score of our model

In [113]:
```
model1.score(X_train, y_train)
```

Out[113]:
```
1.0
```

Make predictions with X_test and check the accuracy score

In [114]:
```
metrics.accuracy_score(y_test, y_pred)
```

Out[114]:
```
0.8347910592808552
```

Print Confusion matrix and Classification report

In [115]:
```
metrics.confusion_matrix(y_test, y_pred)
```

Out[115]:
```
array([[859,   0],
       [170,   0]], dtype=int64)
```

In [116]:
```
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      1.00      0.91       859
           1       0.00      0.00      0.00       170

    accuracy                           0.83      1029
   macro avg       0.42      0.50      0.45      1029
weighted avg       0.70      0.83      0.76      1029
```