Green Destination Travels Employee Attrition Analytics

GREEN
DESTINATIONS

Importing Libraries

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


import warnings
warnings.filterwarnings('ignore')


from sklearn.model_selection import GridSearchCV


from sklearn.model_selection import train_test_split


from sklearn.preprocessing import StandardScaler


from sklearn.metrics  import f1_score, recall_score, precision_score, accuracy_score, classi


from sklearn.ensemble import RandomForestClassifier
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Reading the dataset

```
data=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/greendestination.csv")
data.head()
```

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

5 rows × 35 columns

```
data.columns
```

```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       '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'],
      dtype='object')
```

```
data.shape
```

```
(1470, 35)
```

```
for col in data.columns:
    if data[col].dtypes=='O':
        print(f'{col} : {data[col].unique()}')
    else:
        print(f'{col} : {data[col].min()} - {data[col].max()}')
```

```
Age : 18 - 60
Attrition : ['Yes' 'No']
BusinessTravel : ['Travel_Rarely' 'Travel_Frequently' 'Non-Travel']
DailyRate : 102 - 1499
Department : ['Sales' 'Research & Development' 'Human Resources']
DistanceFromHome : 1 - 29
Education : 1 - 5
EducationField : ['Life Sciences' 'Other' 'Medical' 'Marketing' 'Technical Degree'
 'Human Resources']
EmployeeCount : 1 - 1
EmployeeNumber : 1 - 2068
EnvironmentSatisfaction : 1 - 4
Gender : ['Female' 'Male']
HourlyRate : 30 - 100
JobInvolvement : 1 - 4
JobLevel : 1 - 5
JobRole : ['Sales Executive' 'Research Scientist' 'Laboratory Technician'
 'Manufacturing Director' 'Healthcare Representative' 'Manager'
 'Sales Representative' 'Research Director' 'Human Resources']
JobSatisfaction : 1 - 4
MaritalStatus : ['Single' 'Married' 'Divorced']
MonthlyIncome : 1009 - 19999
MonthlyRate : 2094 - 26999
NumCompaniesWorked : 0 - 9
Over18 : ['Y']
OverTime : ['Yes' 'No']
PercentSalaryHike : 11 - 25
PerformanceRating : 3 - 4
RelationshipSatisfaction : 1 - 4
StandardHours : 80 - 80
StockOptionLevel : 0 - 3
TotalWorkingYears : 0 - 40
TrainingTimesLastYear : 0 - 6
WorkLifeBalance : 1 - 4
YearsAtCompany : 0 - 40
YearsInCurrentRole : 0 - 18
YearsSinceLastPromotion : 0 - 15
YearsWithCurrManager : 0 - 17
```

```
data.drop(['Over18','EmployeeCount'], axis=1, inplace=True)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 33 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   EmployeeNumber            1470 non-null   int64 
 9   EnvironmentSatisfaction   1470 non-null   int64 
 10  Gender                    1470 non-null   object
 11  HourlyRate                1470 non-null   int64 
 12  JobInvolvement            1470 non-null   int64 
 13  JobLevel                  1470 non-null   int64 
 14  JobRole                   1470 non-null   object
 15  JobSatisfaction           1470 non-null   int64 
 16  MaritalStatus             1470 non-null   object
 17  MonthlyIncome             1470 non-null   int64 
 18  MonthlyRate               1470 non-null   int64 
 19  NumCompaniesWorked        1470 non-null   int64 
 20  OverTime                  1470 non-null   object
 21  PercentSalaryHike         1470 non-null   int64 
 22  PerformanceRating         1470 non-null   int64 
 23  RelationshipSatisfaction  1470 non-null   int64 
 24  StandardHours             1470 non-null   int64 
 25  StockOptionLevel          1470 non-null   int64 
 26  TotalWorkingYears         1470 non-null   int64 
 27  TrainingTimesLastYear     1470 non-null   int64 
 28  WorkLifeBalance           1470 non-null   int64 
 29  YearsAtCompany            1470 non-null   int64 
 30  YearsInCurrentRole        1470 non-null   int64 
 31  YearsSinceLastPromotion   1470 non-null   int64 
 32  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(25), object(8)
memory usage: 379.1+ KB
```

```
data.isnull().sum()
```

```
Age                        0
Attrition                  0
BusinessTravel             0
DailyRate                  0
Department                 0
DistanceFromHome           0
Education                  0
EducationField             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
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
```

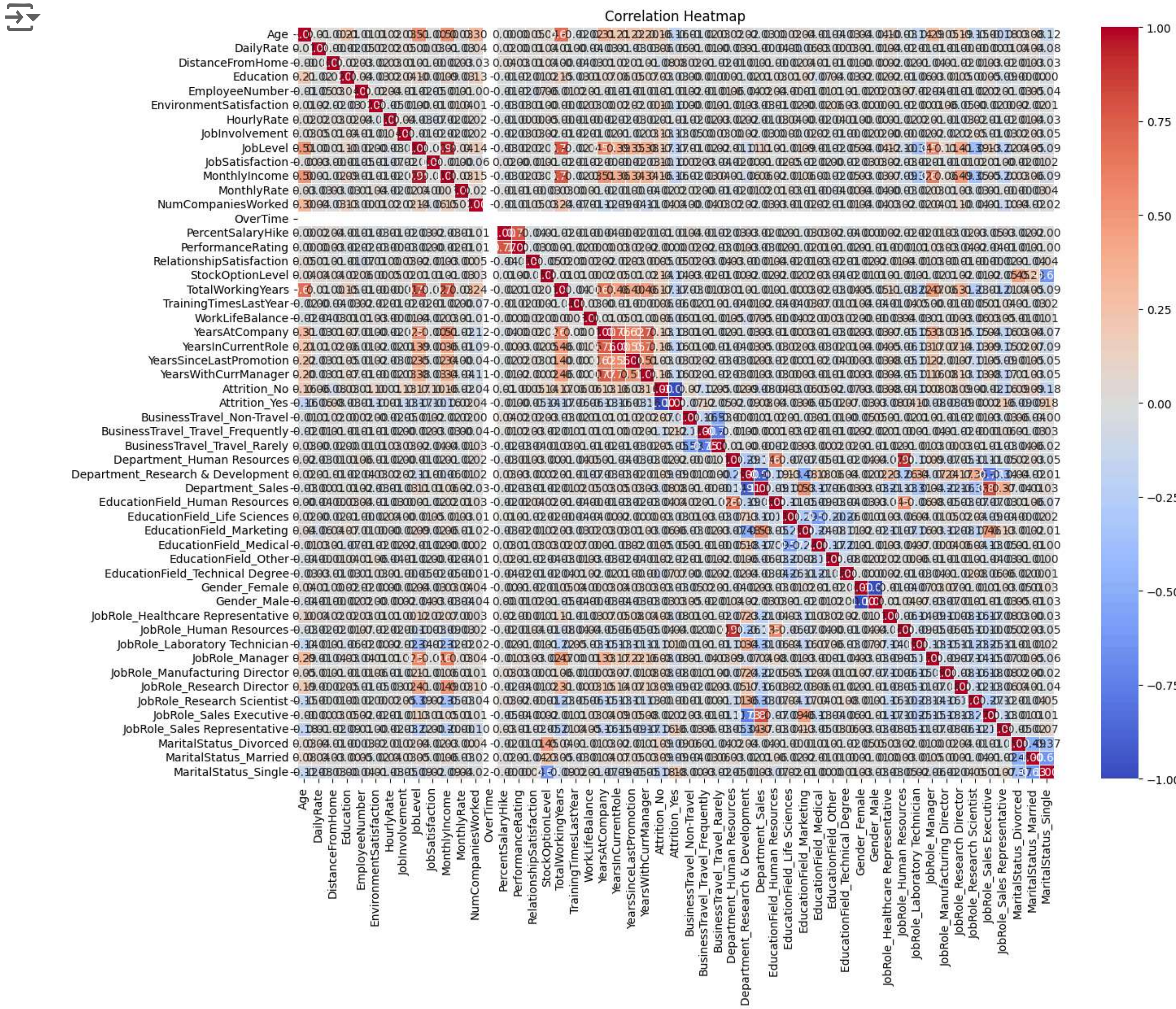
```
data.duplicated().sum()
```

```
categorical_columns = data.select_dtypes(include=['object']).columns
numerical_columns = data.select_dtypes(exclude=['object']).columns
```

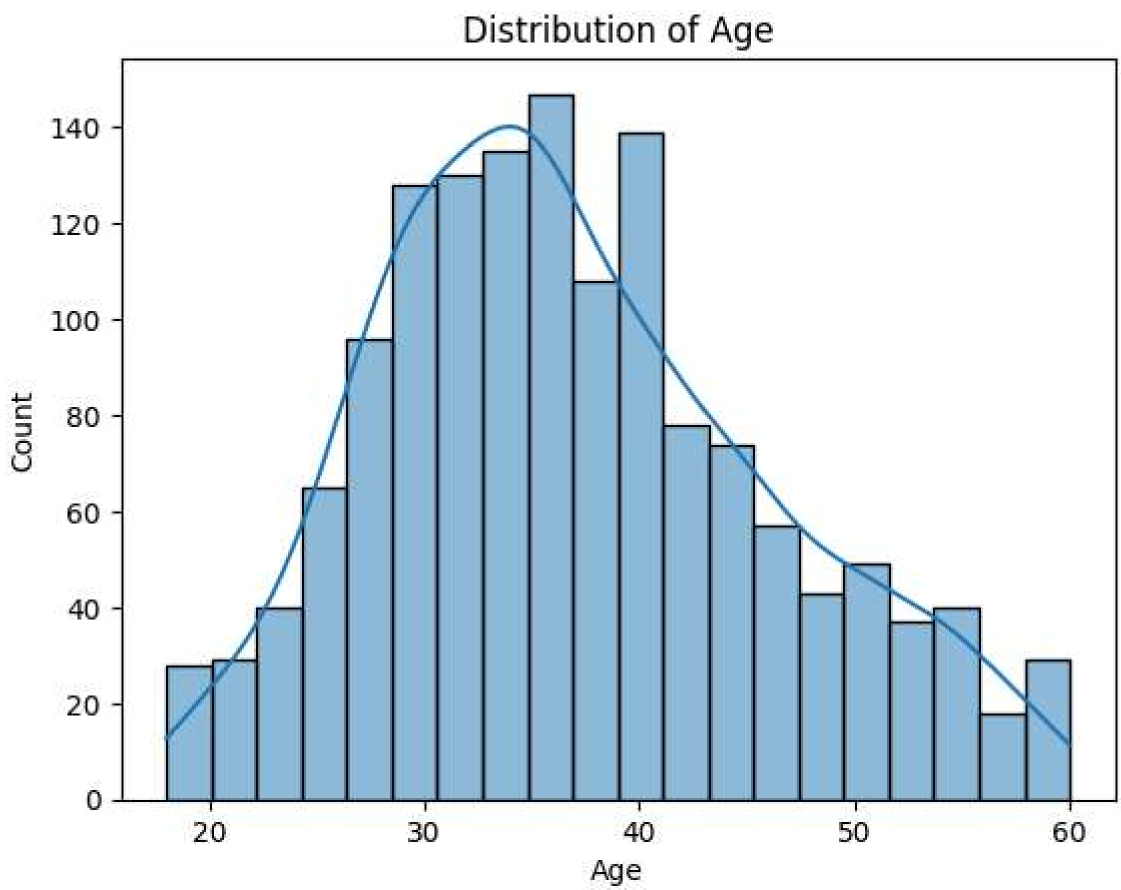
```
ax=sns.countplot(x=data['Attrition'])
for i in ax.containers:
    ax.bar_label(i)
plt.title('Count of each Attrition')
plt.show()
```

```
categorical_columns = categorical_columns[1:]
num_rows = len(categorical_columns)
fig, ax = plt.subplots(nrows=num_rows, ncols=1, figsize=(8, 4 * num_rows))
fig.tight_layout(pad=3.0)
for i, column_name in enumerate(categorical_columns):
    sns.countplot(x=column_name, hue='Attrition', data=data, ax=ax[i])
    ax[i].set_title(f'Attrition by {column_name}')
    ax[i].set_xlabel('')
    ax[i].set_ylabel('Count')
plt.show()
```

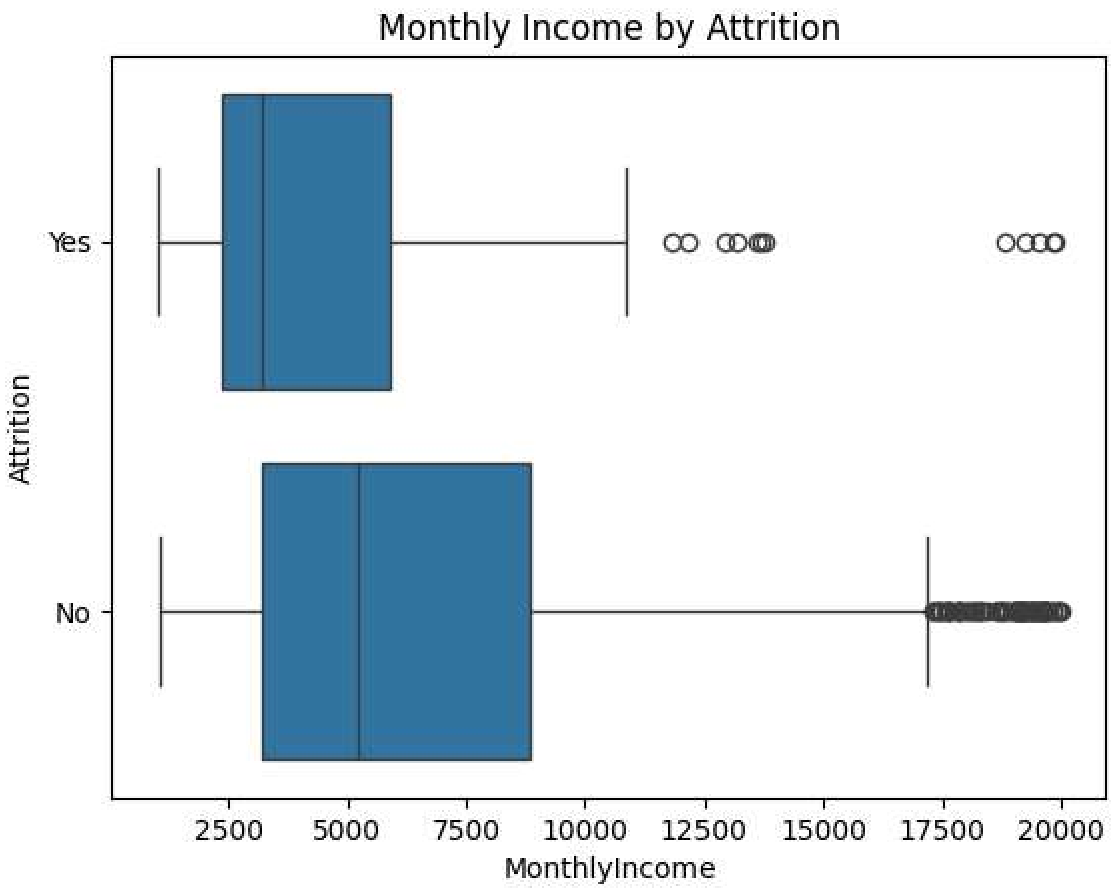
```
# Assuming 'data' is your DataFrame
# Convert categorical variables to numerical using one-hot encoding
data_encoded = pd.get_dummies(data.drop('StandardHours', axis=1))

# Compute correlation matrix
correlation_matrix = data_encoded.corr()

# Plot heatmap
plt.figure(figsize=(15, 12))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```


Correlation Heatmap

```
sns.histplot(data['Age'], bins=20, kde=True)
plt.title('Distribution of Age')
plt.show()
```

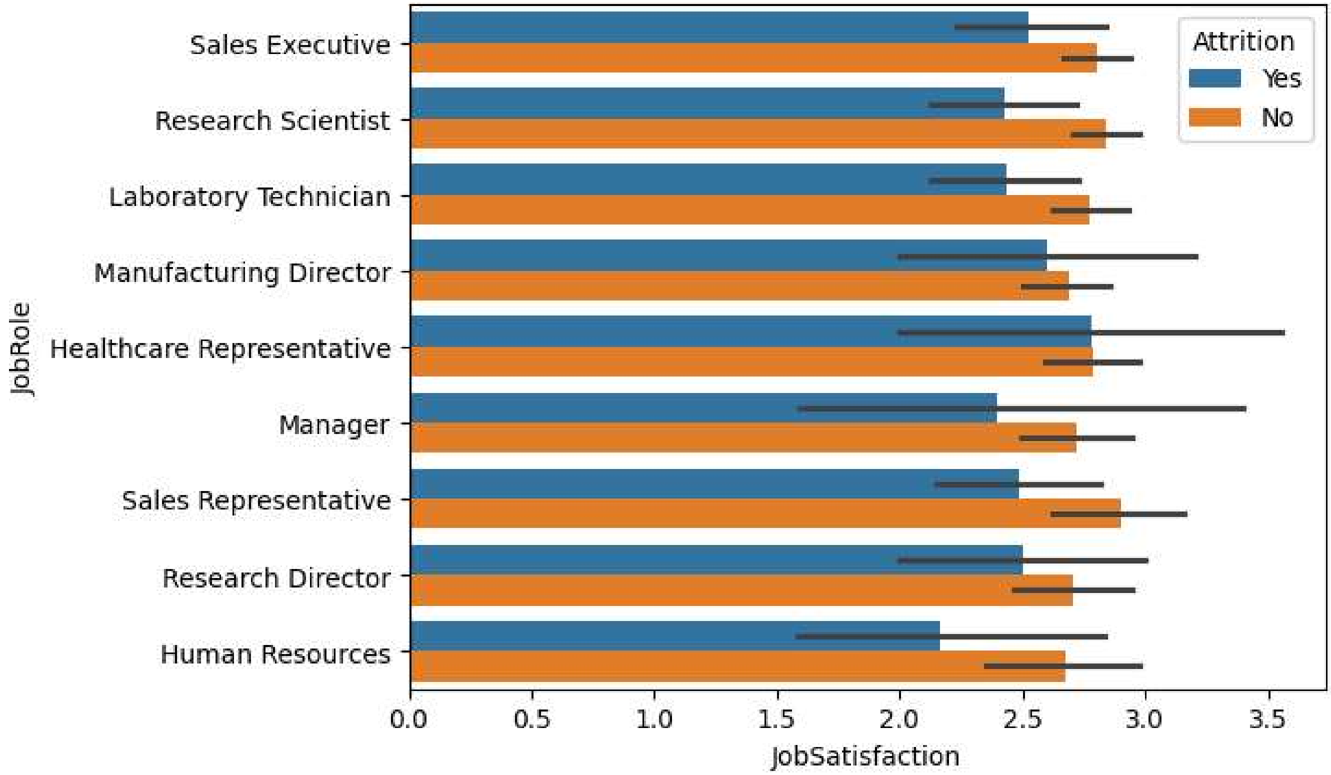



```
sns.boxplot(y='Attrition', x='MonthlyIncome', data=data)
plt.title('Monthly Income by Attrition')
plt.show()
```



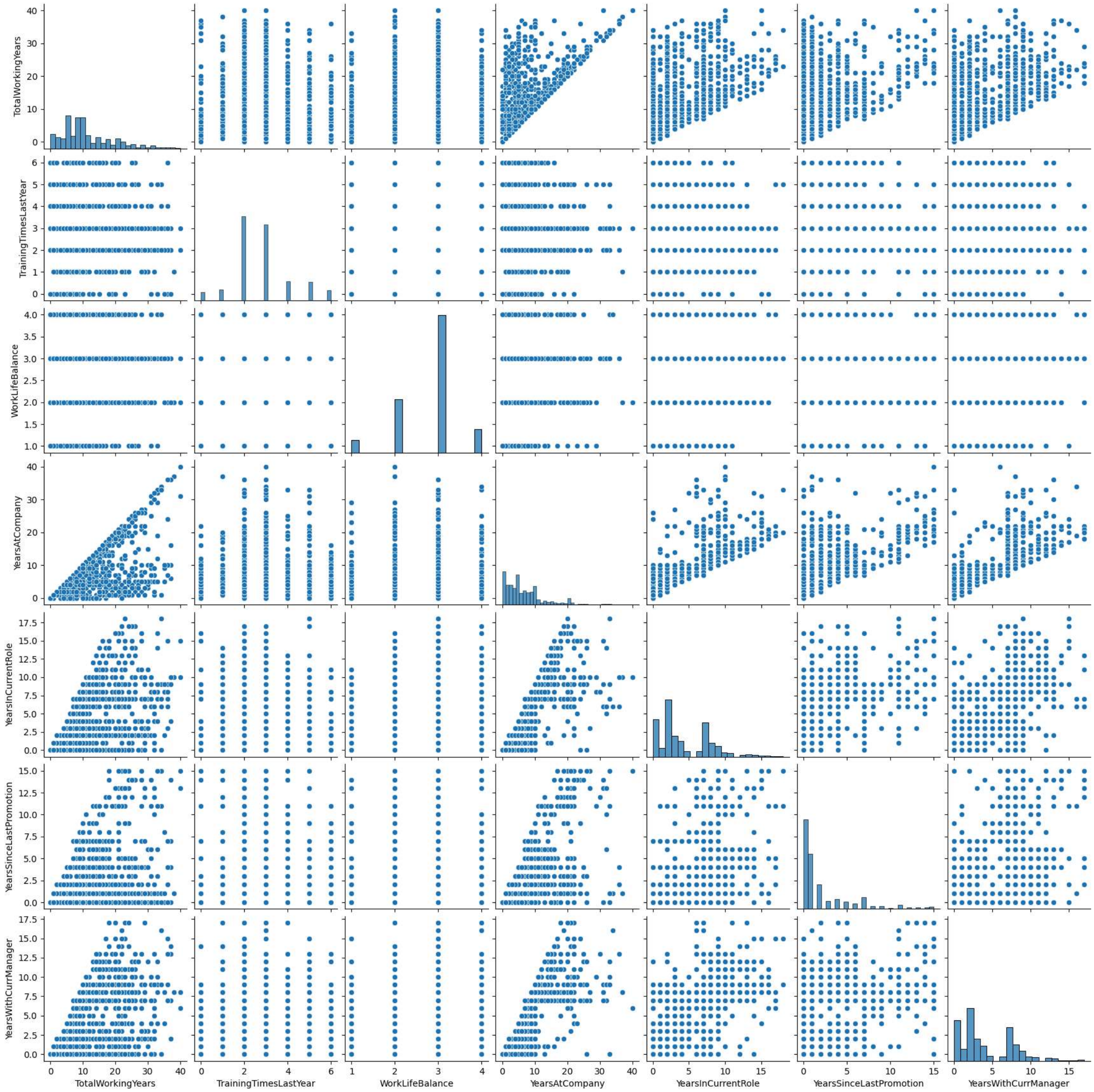

```
sns.barplot(y='JobRole', x='JobSatisfaction', hue='Attrition',data=data)
```

```
<Axes: xlabel='JobSatisfaction', ylabel='JobRole'>
```




```
sns.pairplot(data=data[['TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance', 'Ye
```

<seaborn.axisgrid.PairGrid at 0x7cef83939840>



Data Preprocessing

```
for col in data.columns:
    if data[col].dtypes=='O':
        print(f'{col} : {data[col].unique()}')
```

```
Attrition : ['Yes' 'No']
BusinessTravel : ['Travel_Rarely' 'Travel_Frequently' 'Non-Travel']
Department : ['Sales' 'Research & Development' 'Human Resources']
EducationField : ['Life Sciences' 'Other' 'Medical' 'Marketing' 'Technical Degree'
 'Human Resources']
Gender : ['Female' 'Male']
JobRole : ['Sales Executive' 'Research Scientist' 'Laboratory Technician'
 'Manufacturing Director' 'Healthcare Representative' 'Manager'
 'Sales Representative' 'Research Director' 'Human Resources']
MaritalStatus : ['Single' 'Married' 'Divorced']
```

```
data.Attrition=[1 if val=='Yes' else 0 for val in data.Attrition]
```

```
data.Gender=[1 if val=='Male' else 0 for val in data.Gender]
```

```
data.OverTime=[1 if val=='Yes' else 0 for val in data.OverTime]
```

```
BusinessTravel_dummies=pd.get_dummies(data.BusinessTravel)
Department_dummies=pd.get_dummies(data.Department)
EducationField_dummies=pd.get_dummies(data.EducationField)
JobRole_dummies=pd.get_dummies(data.JobRole)
MaritalStatus_dummies=pd.get_dummies(data.MaritalStatus)


data=pd.concat([data,
               BusinessTravel_dummies.iloc[:,:2],
               Department_dummies.iloc[:,:2],
               EducationField_dummies.iloc[:,:5],
               JobRole_dummies.drop(columns='Human Resources'),
               MaritalStatus_dummies.iloc[:,:2]
               ],axis=1)

data.drop(['BusinessTravel','Department','EducationField','JobRole','MaritalStatus'],axis=1,
data
```

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeNumber	Environmen
0	41	1	1102	1	2	1	
1	49	0	279	8	1	2	
2	37	1	1373	2	2	4	
3	33	0	1392	3	4	5	
4	27	0	591	2	1	7	
...	...	...	...	...	...	...	
1465	36	0	884	23	2	2061	
1466	39	0	613	6	1	2062	
1467	27	0	155	4	3	2064	
1468	49	0	1023	2	3	2065	
1469	34	0	628	8	3	2068	

1470 rows × 47 columns

Train Test Split

```
X=data.drop('Attrition',axis=1)
y=data.Attrition
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=11,train_size=0.80,shuffle=1
```

```
X_train.shape , X_test.shape , y_train.shape , y_test.shape
```

```
((1176, 46), (294, 46), (1176,), (294,))
```

```
X_train.head()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfactic
38	36	852	5	4	51	
907	44	1099	5	3	1267	
311	45	1249	7	3	425	
77	45	193	6	4	101	
287	38	688	23	4	393	

5 rows × 46 columns

```
y_train[:5]
```

Scaling

```
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
X_train
```

```
array([[-0.10626748,  0.12321901, -0.51540633, ..., -0.2515773 ,
        -0.53276899,  1.09271673],
       [ 0.76765248,  0.72676817, -0.51540633, ..., -0.2515773 ,
        -0.53276899,  1.09271673],
       [ 0.87689248,  1.093296  , -0.26755606, ..., -0.2515773 ,
         1.87698613, -0.91515026],
       ...,
       [-1.19866743,  0.71943761, -0.39148119, ..., -0.2515773 ,
        -0.53276899,  1.09271673],
       [-0.32474747, -0.78821353, -0.26755606, ..., -0.2515773 ,
        -0.53276899, -0.91515026],
       [-0.87094744, -0.49743478, -0.01970578, ..., -0.2515773 ,
        -0.53276899,  1.09271673]])
```

```
X_test
```

```
array([[-1.30790742, -0.27751808, -0.76325661, ..., -0.2515773 ,
        -0.53276899,  1.09271673],
       [-0.98018744,  0.87093578, -0.01970578, ..., -0.2515773 ,
        -0.53276899,  1.09271673],
       [-0.65246745,  0.42621535, -0.76325661, ..., -0.2515773 ,
         1.87698613, -0.91515026],
       ...,
       [ 0.11221251,  1.26922936, -1.01110688, ..., -0.2515773 ,
        -0.53276899,  1.09271673],
       [ 0.11221251, -1.17184599, -0.26755606, ..., -0.2515773 ,
         1.87698613, -0.91515026],
       [ 2.5154924 ,  1.70417572,  2.33487182, ..., -0.2515773 ,
        -0.53276899,  1.09271673]])
```

Model and Feature Importance

```
from sklearn.ensemble import RandomForestClassifier
```

```
max_accuracy = 0

for x in range(120):
    RF = RandomForestClassifier(random_state=x)
    RF.fit(X_train,y_train)
    y_pred = RF.predict(X_test)
    current_accuracy = round(accuracy_score(y_pred,y_test)*100,2)
    if(current_accuracy>max_accuracy):
        max_accuracy = current_accuracy
        best_x = x

print(max_accuracy)
print(best_x)
```

```
86.39
106
```

```
RF=RandomForestClassifier(criterion='entropy', random_state=best_x, n_estimators=19)
```

```
RF.fit(X_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=19, random_state=106)
```

```
y_pred = RF.predict(X_test)
y_pred
```

```
array([0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 0, 0,
       0, 0, 0, 0, 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, 1, 0,
       0, 0, 0, 0, 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, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0])
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.99      0.92       246
           1       0.82      0.19      0.31        48

    accuracy                           0.86       294
   macro avg       0.84      0.59      0.61       294
weighted avg       0.86      0.86      0.82       294
```

```
train_acc=RF.score(X_train,y_train)
test_acc=accuracy_score(y_test,y_pred)
recal=recall_score(y_test,y_pred)
prec=precision_score(y_test,y_pred)
f1=f1_score(y_test,y_pred)
print("Training Accuracy :", train_acc)
print("Testing Accuracy :", test_acc)
print("F1 Score :", f1)
print("Recall :", recal)
print("Precision :", prec)
```

```
Training Accuracy : 0.9931972789115646
Testing Accuracy : 0.8605442176870748
F1 Score : 0.30508474576271183
Recall : 0.1875
Precision : 0.8181818181818182
```

```
conf_matrix = pd.DataFrame(data = confusion_matrix(y_test,y_pred),
                           columns = ['Predicted:0', 'Predicted:1'],
                           index =['Actual:0', 'Actual:1'])
plt.figure(figsize = (5, 3))
sns.heatmap(conf_matrix, annot = True, fmt = 'd')
plt.show()
```


```
importances = RF.feature_importances_

feature_importance = pd.DataFrame({'Feature': X.columns, 'Importance': importances, 'Percent
feature_importance = feature_importance.sort_values('Importance', ascending=False)
feature_importance.head()
```

	Feature	Importance	Percentage
11	MonthlyIncome	0.079355	7.94
0	Age	0.064235	6.42
4	EmployeeNumber	0.056847	5.68
1	DailyRate	0.054839	5.48
2	DistanceFromHome	0.054806	5.48

```
feature_importance.plot(x='Feature', y='Importance', kind='barh', figsize=(15,12))
```

<Axes: ylabel='Feature'>