

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

IMPORT DATASET

```
In [2]: df=pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")
```

```
In [3]: df
```

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7
...	...	...	...	...	...	...	...	...	...	...
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	1	2061
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	1	2062
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	1	2064
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	1	2065
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medical	1	2066

1470 rows × 35 columns

```
In [4]: df.head()
```

Out[4]:

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

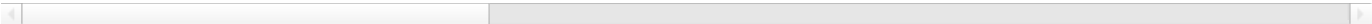
5 rows × 35 columns

```
In [5]: df.tail()
```

Out[5]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	1	2061
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	1	2062
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	1	2064
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	1	2065

5 rows × 35 columns



```
In [6]: df.shape
```

Out[6]: (1470, 35)

```
In [7]: 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
```

```
In [8]: df.describe()
```

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

8 rows × 10 columns



```
In [9]: df.Attrition.value_counts()
```

```
Out[9]: No      1233
        Yes     237
        Name: Attrition, dtype: int64
```

# Checking for NULL Values

```
In [10]: df.isnull().any()
```

```
Out[10]: 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
```

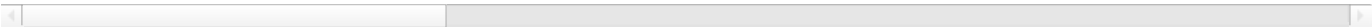
## Data Visualization

```
In [11]: corr=df.corr()
corr
```

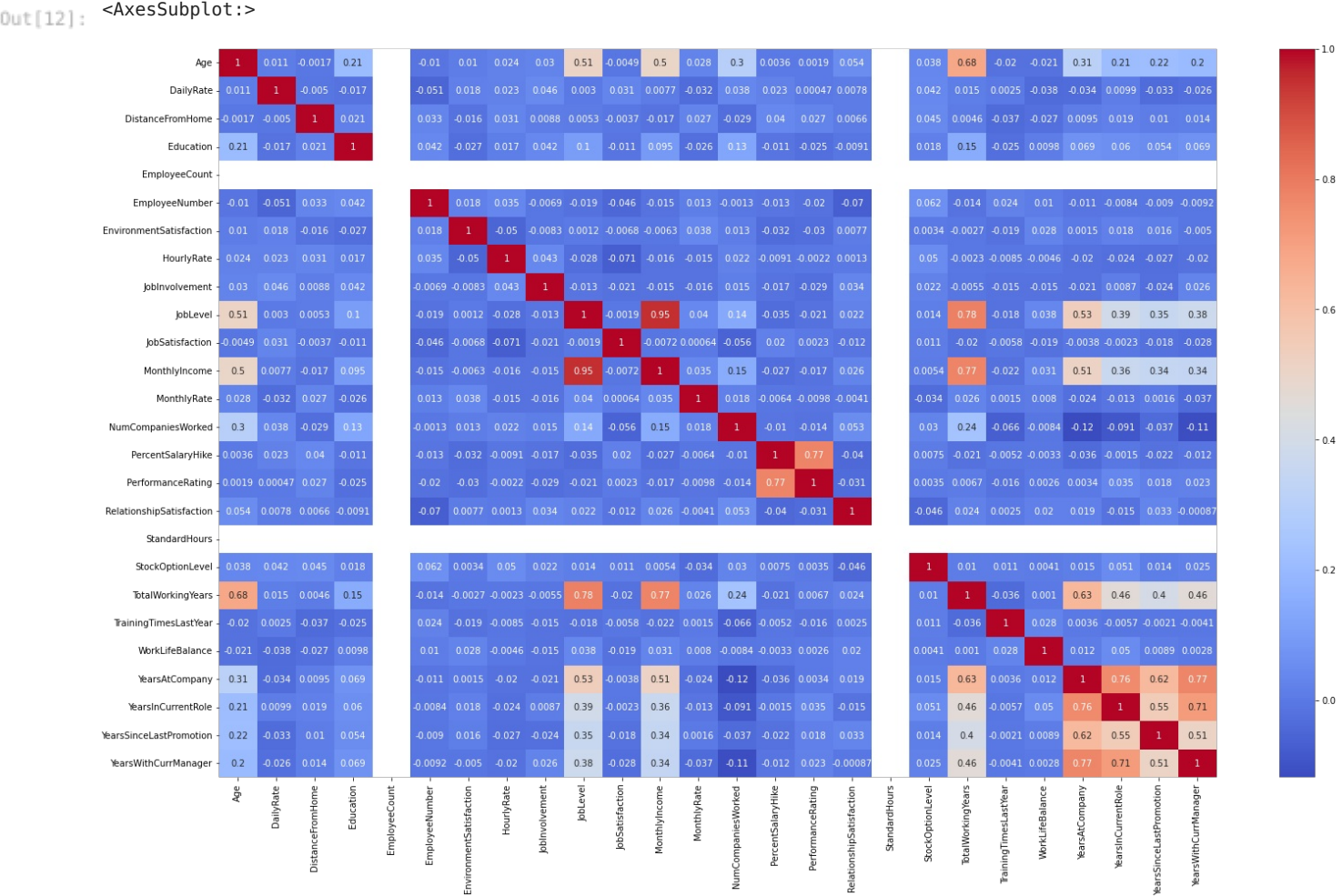
	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate
Age	1.000000	0.010661	-0.001686	0.208034	NaN	-0.010145	0.010146	0.024287
DailyRate	0.010661	1.000000	-0.004985	-0.016806	NaN	-0.050990	0.018355	0.046135
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	NaN	0.032916	-0.016075	0.031131
Education	0.208034	-0.016806	0.021042	1.000000	NaN	0.042070	-0.027128	0.042438
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
EmployeeNumber	-0.010145	-0.050990	0.032916	0.042070	NaN	1.000000	0.017621	-0.006888
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	NaN	0.017621	1.000000	-0.049857
HourlyRate	0.024287	0.023381	0.031131	0.016775	NaN	0.035179	-0.049857	1.000000
JobInvolvement	0.029820	0.046135	0.008783	0.042438	NaN	-0.006888	-0.008278	0.001904
JobLevel	0.509604	0.002966	0.005303	0.101589	NaN	-0.018519	0.001212	-0.004892
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	NaN	-0.046247	-0.006784	-0.000473
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	NaN	-0.014829	-0.006259	-0.027707
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	NaN	0.012648	0.037600	-0.027473
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	NaN	-0.001251	0.012594	0.000473
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	NaN	-0.012944	-0.031701	-0.011111
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	NaN	-0.020359	-0.029548	-0.024539
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	NaN	-0.069861	0.007665	0.006557

	StandardHours	NaN	NaN	NaN	NaN	NaN	NaN	NaN
StockOptionLevel	0.037510	0.042143		0.044872	0.018422		0.062227	0.003432
TotalWorkingYears	0.680381	0.014515		0.004628	0.148280		-0.014365	-0.002693
TrainingTimesLastYear	-0.019621	0.002453		-0.036942	-0.025100		0.023603	-0.019359
WorkLifeBalance	-0.021490	-0.037848		-0.026556	0.009819		0.010309	0.027627
YearsAtCompany	0.311309	-0.034055		0.009508	0.069114		-0.011240	0.001458
YearsInCurrentRole	0.212901	0.009932		0.018845	0.060236		-0.008416	0.018007
YearsSinceLastPromotion	0.216513	-0.033229		0.010029	0.054254		-0.009019	0.016194
YearsWithCurrManager	0.202089	-0.026363		0.014406	0.069065		-0.009197	-0.004999

26 rows × 26 columns

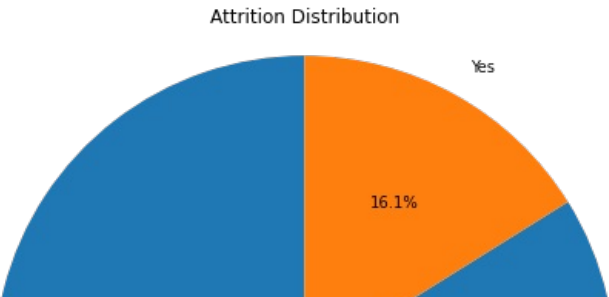


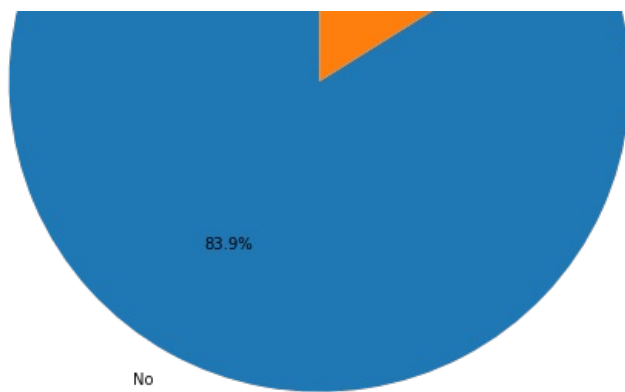
```
In [12]: plt.subplots(figsize=(25,15))
sns.heatmap(corr,annot=True,cmap="coolwarm")
```



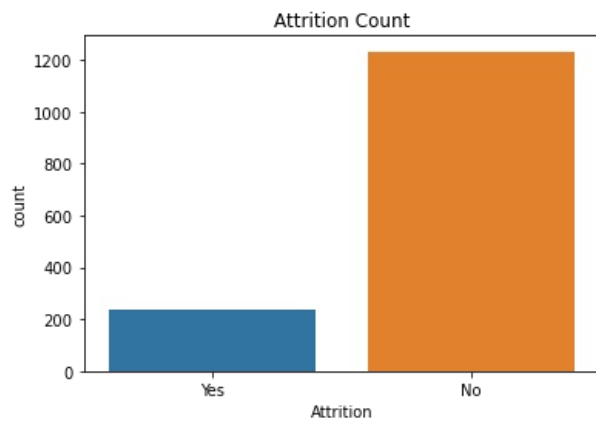
```
In [13]: attrition_counts = df['Attrition'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(attrition_counts, labels=attrition_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Attrition Distribution')
plt.axis('equal')

plt.show()
```





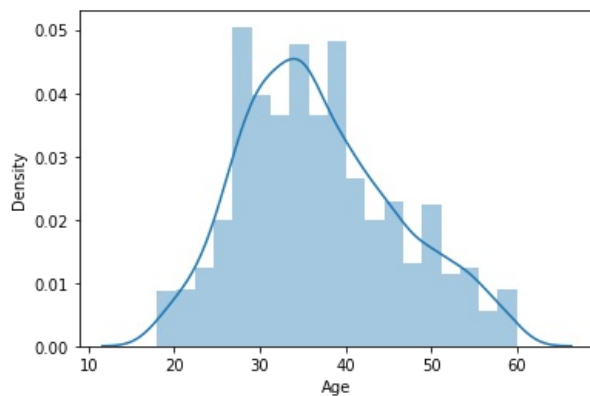
```
In [14]: sns.countplot(x="Attrition", data=df)
plt.title("Attrition Count")
plt.show()
```



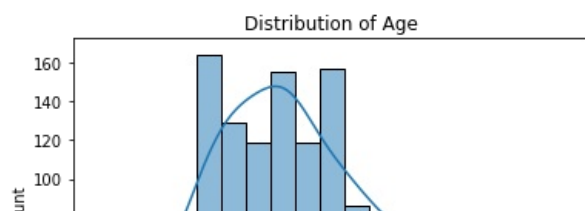
```
In [15]: sns.distplot(df["Age"])
```

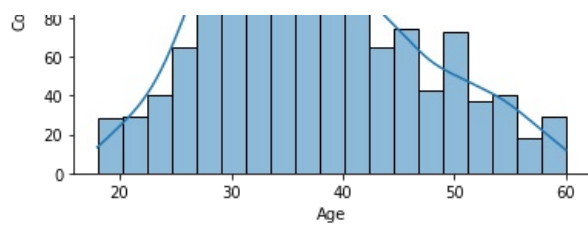
C:\Users\SRUJANA\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
warnings.warn(msg, FutureWarning)

```
Out[15]: <AxesSubplot:xlabel='Age', ylabel='Density'>
```



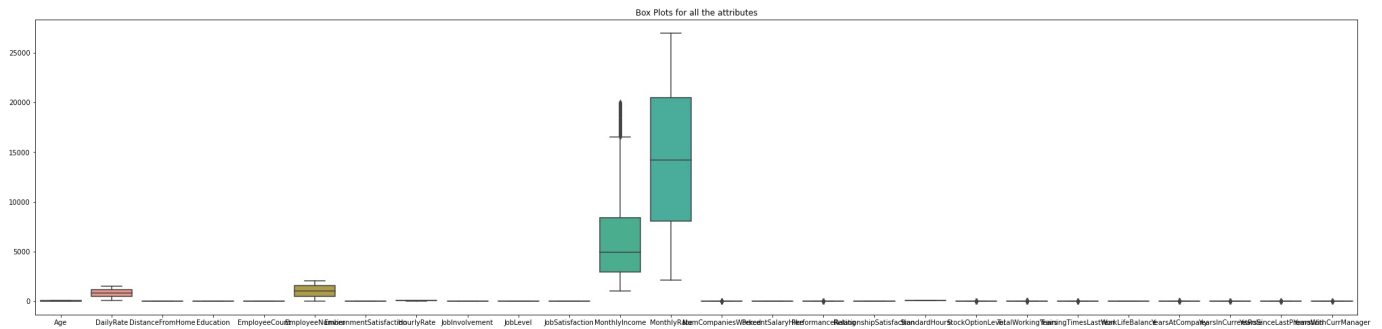
```
In [16]: sns.histplot(data=df, x="Age", kde=True)
plt.title("Distribution of Age")
plt.show()
```



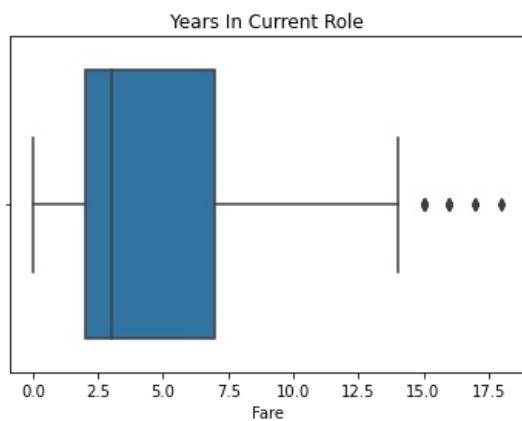


## Outlier Detection

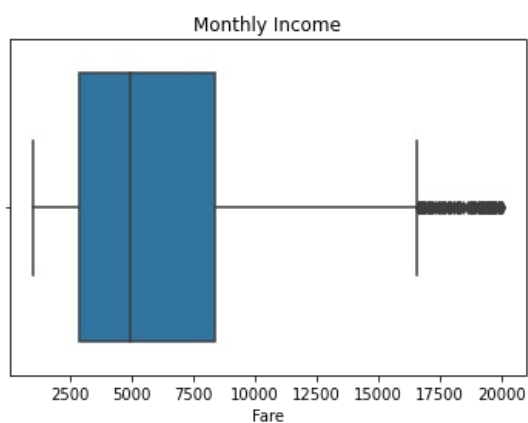
```
In [17]: plt.figure(figsize=(35, 8))
sns.boxplot(data=df)
plt.title('Box Plots for all the attributes')
plt.show()
```



```
In [18]: sns.boxplot(data=df, x='YearsInCurrentRole')
plt.title('Years In Current Role')
plt.xlabel('Fare')
plt.show()
```



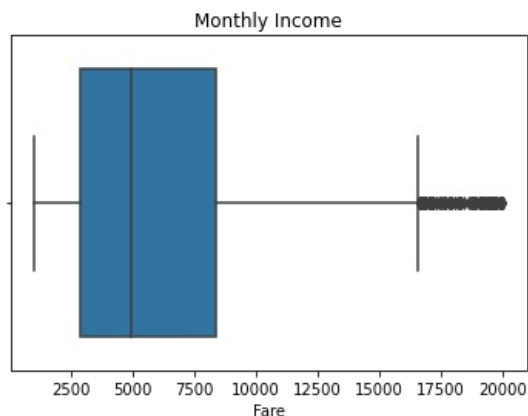
```
In [19]: sns.boxplot(data=df, x='MonthlyIncome')
plt.title('Monthly Income')
plt.xlabel('Fare')
plt.show()
```



```
In [20]: from scipy import stats

z_scores = stats.zscore(df['MonthlyIncome'])
z_score_threshold = 3
df_cleaned = df[(np.abs(z_scores) <= z_score_threshold)]
```

```
In [21]: sns.boxplot(data=df_cleaned, x='MonthlyIncome')
plt.title('Monthly Income')
plt.xlabel('Fare')
plt.show()
```



So the outliers are in large quantity, and they are inside the threshold, so let us not remove the outliers

## SPLITTING INDEPENDENT AND DEPENDENT VARIABLES

```
In [22]: x= df.drop(columns=["Attrition"])
y = df["Attrition"]
#since there are so many null values
```

```
In [23]: x.head()
```

```
Out[23]:
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	Environment
0	41	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	
1	49	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	
2	37	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	
3	33	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	
4	27	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	

5 rows × 34 columns

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

```
Out[24]:
```

0	Yes
1	No
2	Yes
3	No
4	No

Name: Attrition, dtype: object

### ENCODING

```
In [25]: categorical_features = x.select_dtypes(include=['object']).columns.tolist()
x_encoded = pd.get_dummies(x, columns=categorical_features, drop_first=True)
```

```
In [26]:
```

```
x_encoded.head()
```

```
Out[26]:
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	Job
0	41	1102	1	2	1	1	2	94	3	
1	49	279	8	1	1	2	3	61	2	
2	37	1373	2	2	1	4	4	92	2	
3	33	1392	3	4	1	5	4	56	3	
4	27	591	2	1	1	7	1	40	3	

5 rows × 47 columns

## FEATURE SCALING

```
In [27]: from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_scaled = pd.DataFrame(scaler.fit_transform(x_encoded), columns=x_encoded.columns)
x_scaled
```

```
Out[27]:
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	Job
0	0.446350	0.742527	-1.010909	-0.891688	0.0	-1.701283	-0.660531	1.383138	0.379672	
1	1.322365	-1.297775	-0.147150	-1.868426	0.0	-1.699621	0.254625	-0.240677	-1.026167	
2	0.008343	1.414363	-0.887515	-0.891688	0.0	-1.696298	1.169781	1.284725	-1.026167	
3	-0.429664	1.461466	-0.764121	1.061787	0.0	-1.694636	1.169781	-0.486709	0.379672	
4	-1.086676	-0.524295	-0.887515	-1.868426	0.0	-1.691313	-1.575686	-1.274014	0.379672	
...	...	...	...	...	...	...	...	...	...	
1465	-0.101159	0.202082	1.703764	-0.891688	0.0	1.721670	0.254625	-1.224807	1.785134	
1466	0.227347	-0.469754	-0.393938	-1.868426	0.0	1.723332	1.169781	-1.175601	-1.026167	
1467	-1.086676	-1.605183	-0.640727	0.085049	0.0	1.726655	-0.660531	1.038693	1.785134	
1468	1.322365	0.546677	-0.887515	0.085049	0.0	1.728317	1.169781	-0.142264	-1.026167	
1469	-0.320163	-0.432568	-0.147150	0.085049	0.0	1.733302	-0.660531	0.792660	1.785134	

1470 rows × 47 columns

```
In [28]: x_scaled.head()
```

```
Out[28]:
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	Job
0	0.446350	0.742527	-1.010909	-0.891688	0.0	-1.701283	-0.660531	1.383138	0.379672	
1	1.322365	-1.297775	-0.147150	-1.868426	0.0	-1.699621	0.254625	-0.240677	-1.026167	
2	0.008343	1.414363	-0.887515	-0.891688	0.0	-1.696298	1.169781	1.284725	-1.026167	
3	-0.429664	1.461466	-0.764121	1.061787	0.0	-1.694636	1.169781	-0.486709	0.379672	
4	-1.086676	-0.524295	-0.887515	-1.868426	0.0	-1.691313	-1.575686	-1.274014	0.379672	

5 rows × 47 columns

## Train test and split

```
In [29]: from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.2, random_state=42)
```

```
In [30]: x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
Out[30]: ((1176, 47), (294, 47), (1176,), (294,))
```



# MODEL BUILDING

```
In [31]: # Import the necessary libraries
from sklearn.linear_model import LogisticRegression
logreg_model = LogisticRegression(random_state=42)

from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier(random_state=42)
```

```
In [32]: logreg_model.fit(x_train, y_train)
dt model.fit(x_train, y_train)
```

```
Out[32]: DecisionTreeClassifier(random_state=42)
```

```
In [33]: logreg_predictions = logreg_model.predict(x_test)
logreg_predictions
```

[illegible]

```
In [34]: dt_predictions = dt_model.predict(x_test)
dt_predictions
```

```
Out[34]: array(['No', 'No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'No',  
      'No', 'Yes', 'No', 'No', 'No', 'No', 'Yes', 'No', 'Yes', 'No',  
      'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No',  
      'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No',  
      'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No',  
      'No', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No',  
      'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'No',  
      'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'Yes', 'Yes', 'No',  
      'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',  
      'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No',  
      'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',  
      'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No',  
      'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No',  
      'Yes', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No',  
      'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'Yes', 'No',  
      'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'No',  
      'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',  
      'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No',  
      'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',  
      'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',  
      'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No',  
      'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No',  
      'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No',  
      'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No',  
      'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',  
      'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'Yes',  
      'No', 'No'], dtype=object)
```

```
In [35]: y_test

Out[35]: 1041    No
184      No
1222    Yes
67       No
220     No
...
567     No
560     No
945     No
522     No
651     No
Name: Attrition, Length: 294, dtype: object
```

```
In [36]: df

Out[36]:
```

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

1470 rows × 35 columns

# Evaluation of the model

```
In [59]: from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score, roc_curve
from joblib import dump

In [60]: logreg_accuracy = accuracy_score(y_test, logreg_predictions)
print("Logistic Regression Accuracy:", logreg_accuracy)

Logistic Regression Accuracy: 0.8809523809523809

In [61]: dt_accuracy = accuracy_score(y_test, dt_predictions)
print("Decision Tree Accuracy:", dt_accuracy)

Decision Tree Accuracy: 0.7721088435374149

In [62]: logreg_report = classification_report(y_test, logreg_predictions)
print("Classification Report for Logistic Regression:\n", logreg_report)

Classification Report for Logistic Regression:
precision    recall  f1-score   support
```



```
In [ ]: #roc aoc curve
fpr,tpr,thresholda=roc_curve(y_test,probability)
```

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

# Random Forest

```
In [45]: from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
```

```
In [46]: forest_params = [{'max_depth': list(range(10, 15)), 'max_features': list(range(0,14))}]
```

```
In [51]: from sklearn.model_selection import GridSearchCV
rfc_cv= GridSearchCV(rfc,param_grid=forest_params,cv=10,scoring="accuracy")
rfc_cv
```

```
Out[51]: GridSearchCV(cv=10, estimator=RandomForestClassifier(),
    param_grid=[{'max_depth': [10, 11, 12, 13, 14],
    'max_features': [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
    12, 13]}],
    scoring='accuracy')
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js