Data Pre-Processing

Importing Necessary Libraries

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import metrics
from imblearn.over_sampling import SMOTE
from warnings import filterwarnings
filterwarnings("ignore")
```

Reading the Data

In [ ]:
```
Data = pd.read_csv('EmpAttrition.csv')
Data.head()
```

Out[ ]:

	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

The use of Employee Attrition Analytics is crucial as it helps organizations gain valuable insights from their data. Raw data alone lacks meaningful interpretation, much like a vast spreadsheet. HR Analytics organizes and analyzes this data to answer questions about employee turnover patterns, hiring efficiency, onboarding costs, predicting attrition, and assessing the impact of training on performance. With HR Attrition Analytics, organizations make informed decisions and manage employee attrition effectively. Here if the attrition is 'Yes', it means the employee would like to leave and 'No' means the employee would like to stay.

In [ ]:
```
Data.shape
```

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

We have 1470 rows and 35 columns

In [ ]:
```
Data.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 [ ]:
```
Data.isnull().sum()
```

Out[ ]:
```
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
```

We have no null values in our dataset

In [ ]:
```
cat = Data.select_dtypes('object')
cat
```

Out[ ]:

	Attrition	BusinessTravel	Department	EducationField	Gender	JobRole	MaritalStatus	Over18	OverTime
0	Yes	Travel_Rarely	Sales	Life Sciences	Female	Sales Executive	Single	Y	Yes
1	No	Travel_Frequently	Research & Development	Life Sciences	Male	Research Scientist	Married	Y	No
2	Yes	Travel_Rarely	Research & Development	Other	Male	Laboratory Technician	Single	Y	Yes
3	No	Travel_Frequently	Research & Development	Life Sciences	Female	Research Scientist	Married	Y	Yes
4	No	Travel_Rarely	Research & Development	Medical	Male	Laboratory Technician	Married	Y	No
...	...	...	...	...	...	...	...	...	...
1465	No	Travel_Frequently	Research & Development	Medical	Male	Laboratory Technician	Married	Y	No
1466	No	Travel_Rarely	Research & Development	Medical	Male	Healthcare Representative	Married	Y	No
1467	No	Travel_Rarely	Research & Development	Life Sciences	Male	Manufacturing Director	Married	Y	Yes
1468	No	Travel_Frequently	Sales	Medical	Male	Sales Executive	Married	Y	No
1469	No	Travel_Rarely	Research & Development	Medical	Male	Laboratory Technician	Married	Y	No

1470 rows × 9 columns

There are 9 categorical variables in the data

In [ ]:
```
num = Data.select_dtypes('int64')
num
```

Out[ ]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	J
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	
...	...	...	...	...	...	...	...	...	...	
1465	36	884	23	2	1	2061	3	41	4	
1466	39	613	6	1	1	2062	4	42	2	
1467	27	155	4	3	1	2064	2	87	4	
1468	49	1023	2	3	1	2065	4	63	2	
1469	34	628	8	3	1	2068	2	82	4	

1470 rows × 26 columns

And 26 numerical variables in our data.

In [ ]:
```
cat.describe().T
```

Out[ ]:

	count	unique	top	freq
Attrition	1470	2	No	1233
BusinessTravel	1470	3	Travel_Rarely	1043
Department	1470	3	Research & Development	961
EducationField	1470	6	Life Sciences	606
Gender	1470	2	Male	882
JobRole	1470	9	Sales Executive	326
MaritalStatus	1470	3	Married	673
Over18	1470	1	Y	1470
OverTime	1470	2	No	1054

In [ ]:
```
num.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	max
Age	1470.0	36.923810	9.135373	18.0	30.00	36.0	43.00	60.0
DailyRate	1470.0	802.485714	403.509100	102.0	465.00	802.0	1157.00	1499.0
DistanceFromHome	1470.0	9.192517	8.106864	1.0	2.00	7.0	14.00	29.0
Education	1470.0	2.912925	1.024165	1.0	2.00	3.0	4.00	5.0
EmployeeCount	1470.0	1.000000	0.000000	1.0	1.00	1.0	1.00	1.0
EmployeeNumber	1470.0	1024.865306	602.024335	1.0	491.25	1020.5	1555.75	2068.0
EnvironmentSatisfaction	1470.0	2.721769	1.093082	1.0	2.00	3.0	4.00	4.0
HourlyRate	1470.0	65.891156	20.329428	30.0	48.00	66.0	83.75	100.0
JobInvolvement	1470.0	2.729932	0.711561	1.0	2.00	3.0	3.00	4.0
JobLevel	1470.0	2.063946	1.106940	1.0	1.00	2.0	3.00	5.0
JobSatisfaction	1470.0	2.728571	1.102846	1.0	2.00	3.0	4.00	4.0
MonthlyIncome	1470.0	6502.931293	4707.956783	1009.0	2911.00	4919.0	8379.00	19999.0
MonthlyRate	1470.0	14313.103401	7117.786044	2094.0	8047.00	14235.5	20461.50	26999.0
NumCompaniesWorked	1470.0	2.693197	2.498009	0.0	1.00	2.0	4.00	9.0
PercentSalaryHike	1470.0	15.209524	3.659938	11.0	12.00	14.0	18.00	25.0
PerformanceRating	1470.0	3.153741	0.360824	3.0	3.00	3.0	3.00	4.0
RelationshipSatisfaction	1470.0	2.712245	1.081209	1.0	2.00	3.0	4.00	4.0
StandardHours	1470.0	80.000000	0.000000	80.0	80.00	80.0	80.00	80.0
StockOptionLevel	1470.0	0.793878	0.852077	0.0	0.00	1.0	1.00	3.0
TotalWorkingYears	1470.0	11.279592	7.780782	0.0	6.00	10.0	15.00	40.0
TrainingTimesLastYear	1470.0	2.799320	1.289271	0.0	2.00	3.0	3.00	6.0
WorkLifeBalance	1470.0	2.761224	0.706476	1.0	2.00	3.0	3.00	4.0
YearsAtCompany	1470.0	7.008163	6.126525	0.0	3.00	5.0	9.00	40.0
YearsInCurrentRole	1470.0	4.229252	3.623137	0.0	2.00	3.0	7.00	18.0
YearsSinceLastPromotion	1470.0	2.187755	3.222430	0.0	0.00	1.0	3.00	15.0
YearsWithCurrManager	1470.0	4.123129	3.568136	0.0	2.00	3.0	7.00	17.0

We have 35 Columns out of which Employee number won't be included in our analysis since it is solely based for the company's purpose to identiy it's employees. Column 'EmployeeCount' , 'StandardHours' and 'Over18' wil also be dropped since it has only 1 unique value and is same for all hence, will not be significant for our analysis. Over all we have 30 predictors and 1 target variable which is column 'Attrition'. Here we also note that our count is constant for all the cateogircal as well as numerical variables, 1470, which proves that our data is consistent and have no null values.

In [ ]:
```
Data.drop(['Over18', 'EmployeeNumber','EmployeeCount','StandardHours'], axis = 1, inplace = True)
Data.head()
```

Out[ ]:

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

5 rows × 31 columns

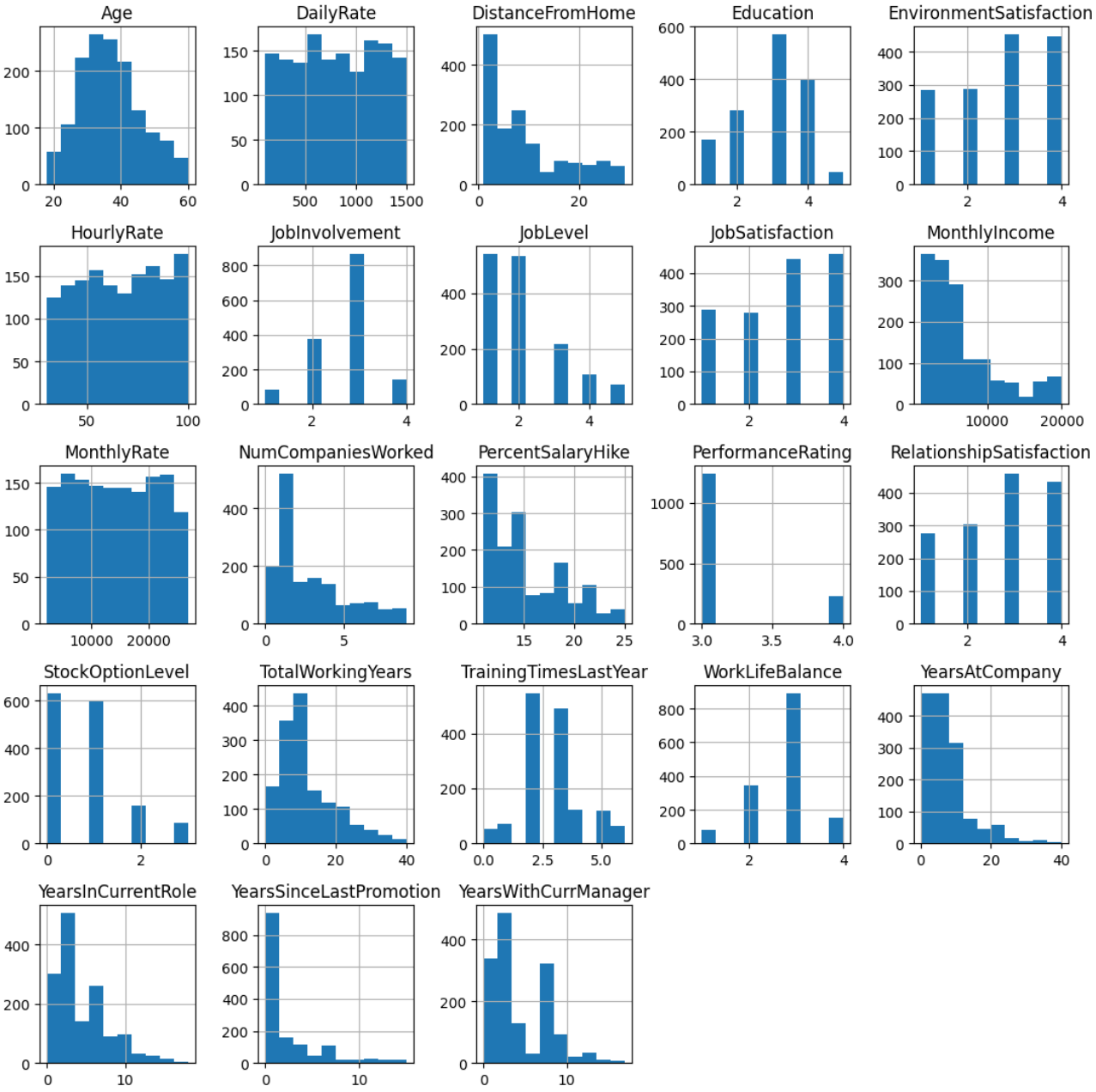
In [ ]:
```
Data.duplicated().any()
```

Out[ ]:
```
False
```

We also have no duplicated values in our data.

Exploratory Data Analysis

In [ ]:
```
Data.hist(figsize=(11,11))
plt.tight_layout()
plt.show()
```



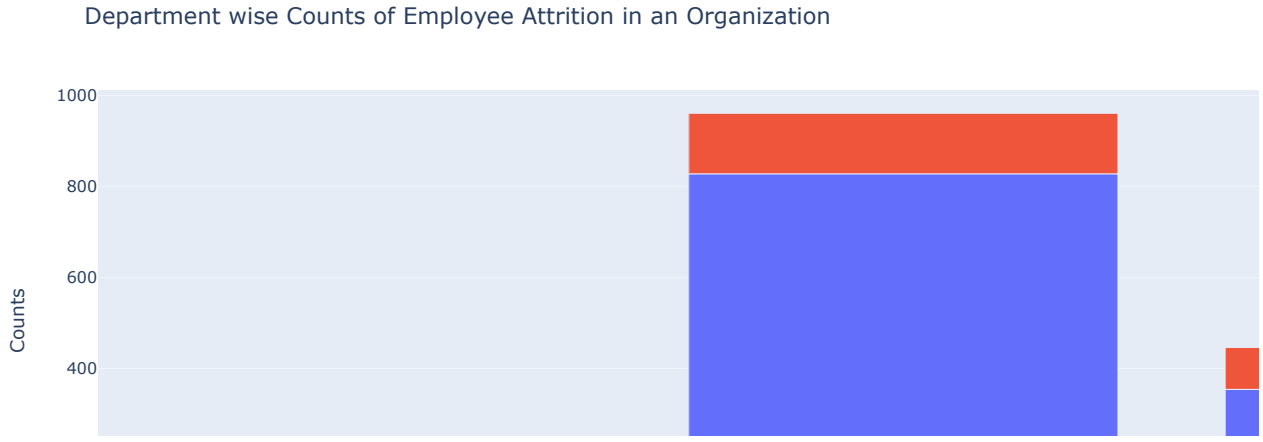
In [ ]:
```
age_att=Data.groupby(['Age','Attrition']).apply(lambda x : x['DailyRate'].count()).reset_index(name='Counts')
fig = px.line(age_att, x = 'Age', y = 'Counts', color='Attrition', title='Agewise Counts of Employee Attrition in an Organization')
fig.show(renderer='notebook')
```

We can see that the attrition is maximum between the age groups 28-35. The attrition rate keeps on falling with increasing age, as people look after stability in their jobs at these point of times. Also, at a very younger age from 18-27, the chances of an employee leaving the organization is far more since they are exploring at that point of time. It reaches a break even point at the age of 28.

In [ ]:
```
income_att = Data.groupby(['MonthlyIncome','Attrition']).apply(lambda x : x['MonthlyIncome'].count()).reset_index(name='Counts')
income_att['MonthlyIncome'] = round(income_att['MonthlyIncome'] , -3)
income_att = income_att.groupby(['MonthlyIncome','Attrition']).apply(lambda x : x['MonthlyIncome'].count()).reset_index(name='Counts'
fig = px.line(income_att, x = 'MonthlyIncome', y = 'Counts', color = 'Attrition', title='Monthly Income basis counts of Employee Attr
fig.show(renderer='notebook')
```

Here, we can note that the attrition rate is evidently high at very low income levels <5k monthly. This decreases further, but a minor spike is noticed around 10k indicating the middle class liveliood. They tend to shift towards a better standard of living, and hence move to a different job. When the monthly income is pretty decent around 11k, the chances of an employee leaving the organization is low as seen by the flat red line

In [ ]:
```
dept_att = Data.groupby(['Department','Attrition']).apply(lambda x : x['DailyRate'].count()).reset_index(name='Counts')
fig = px.bar(dept_att, x = 'Department', y = 'Counts', color = 'Attrition', title = 'Department wise Counts of Employee Attrition in
fig.show(renderer='notebook')
```



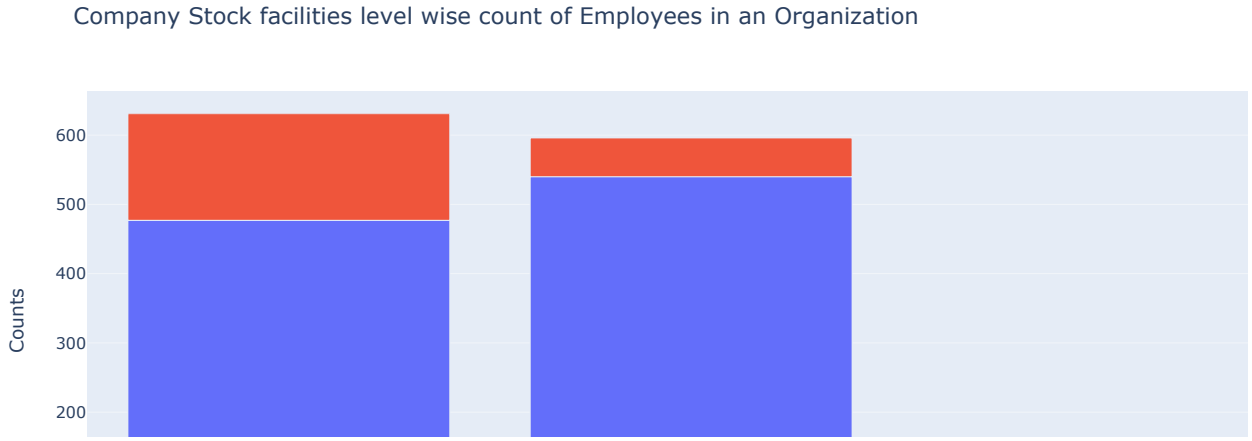
We can observe that Research & Development department has the highest attrition rates (9.04%), followed by the Sales Department (6.25%). Human Resource Department has the least attrition rates, that suggests the stability and content of the department as can be seen from the chart above (0.81%).

In [ ]:
```
env_att = Data.groupby(['EnvironmentSatisfaction','Attrition']).apply(lambda x : x['DailyRate'].count()).reset_index(name='Counts')
fig = px.line(env_att, x = 'EnvironmentSatisfaction', y = 'Counts', color = 'Attrition', title = 'Environment Satisfaction level Coun
fig.show(renderer='notebook')
```

We see that chances of employees leaving the organization slightly decreases from satisfaction level 1 to 2 which is indicative of the better hopes with which people stay in an organization. However, as we move from level 2 to 3, people tend to move on to get better opportunities and experiences. The attrition rate declines for the higher satisfaction levels.

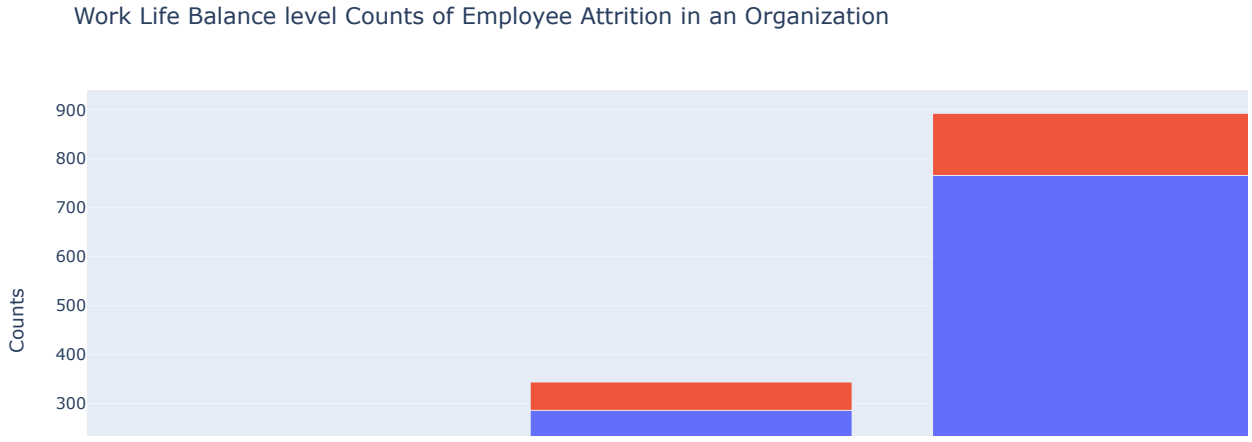
In [ ]:
```
jobsat_att = Data.groupby(['JobSatisfaction','Attrition']).apply(lambda x : x['DailyRate'].count()).reset_index(name='Counts')
fig = px.line(jobsat_att, x = 'JobSatisfaction', y = 'Counts', color = 'Attrition', title = 'Job Satisfaction level Counts of Employe
fig.show(renderer='notebook')
```

In [ ]:
```
stock_att = Data.groupby(['StockOptionLevel','Attrition']).apply(lambda x : x['DailyRate'].count()).reset_index(name='Counts')
fig = px.bar(stock_att, x = 'StockOptionLevel', y = 'Counts', color = 'Attrition', title = 'Company Stock facilities level wise count
fig.show(renderer='notebook')
```



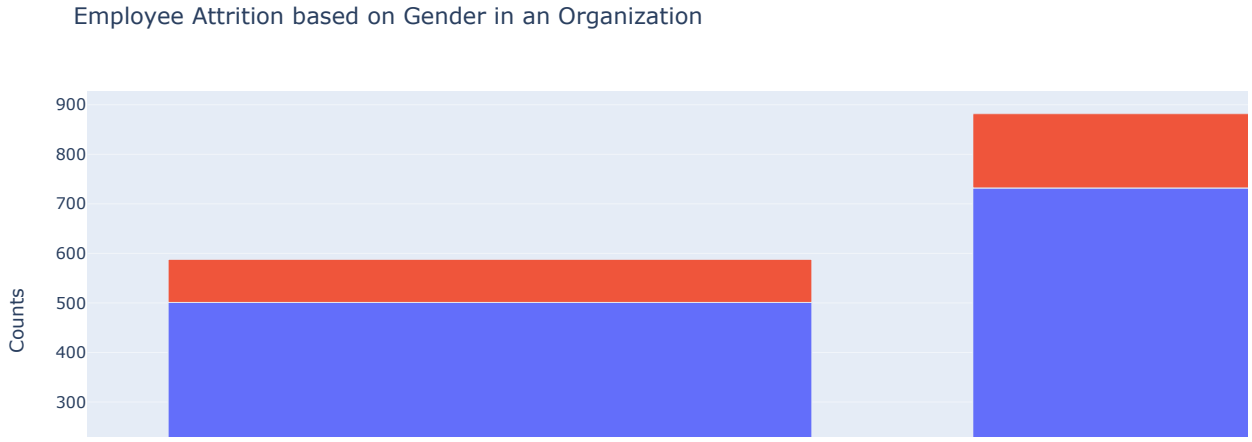
Here, the tendency of employees to leave the organization is much more when the stock availing options are zero since, the stocks constitute to a huge amount of money while staying for a few years, people do not want to lose that opportunity. People with very limited stock options have a freedom to leave the organization as well. When stock option level increases from 2 to 3, we can see that very few employees still prefer to leave the company and majority decide to stay.

In [ ]:
```
worklifebal_att = Data.groupby(['WorkLifeBalance','Attrition']).apply(lambda x : x['DailyRate'].count()).reset_index(name='Counts')
fig = px.bar(worklifebal_att, x = 'WorkLifeBalance', y = 'Counts', color = 'Attrition', title = 'Work Life Balance level Counts of Em
fig.show(renderer='notebook')
```



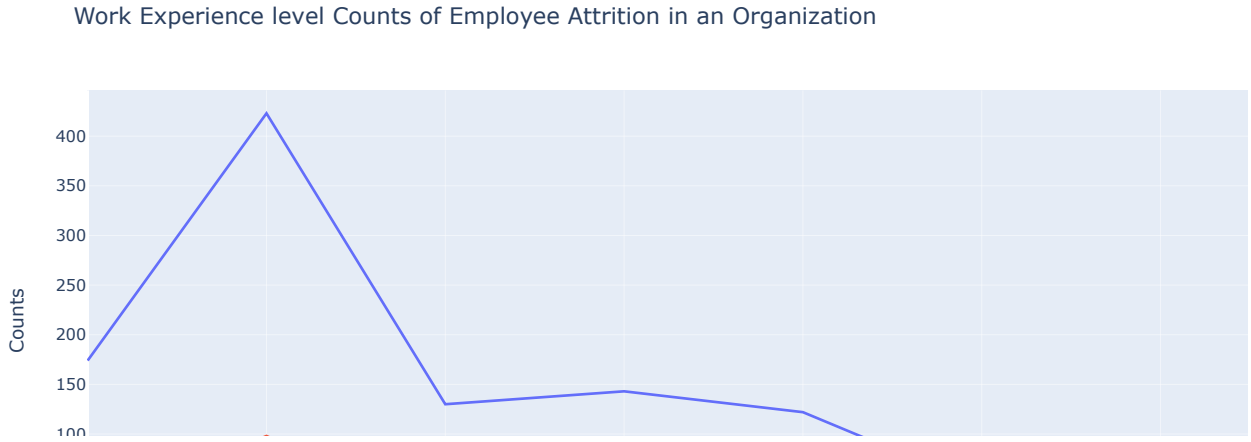
Employees with very poor levels of Work life balance or very good level of Work Life Balance have adjusted themselves to their jobs, but as seen for the above parameters with a better work life score, people are more accustomed to the better life and want to switch jobs more. But this trend perishes when the work life balance is really good, and people are satisfied with the work they are doing.

In [ ]:
```
gen_att = Data.groupby(['Gender','Attrition']).apply(lambda x : x['DailyRate'].count()).reset_index(name='Counts')
fig = px.bar(gen_att, x = 'Gender', y = 'Counts', color = 'Attrition', title = 'Employee Attrition based on Gender in an Organization
fig.show(renderer='notebook')
```



Females tend to stick with their companies than males perhaps, looking for better oppotunities.

In [ ]:
```
exp_att = Data.groupby(['NumCompaniesWorked','Attrition']).apply(lambda x:x['DailyRate'].count()).reset_index(name='Counts')
fig = px.line(exp_att, x = 'NumCompaniesWorked', y = 'Counts', color = 'Attrition', title = 'Work Experience level Counts of Employee
fig.show(renderer='notebook')
```



Employees who started their career or have switched to the company in the initial years of their career tend to switch companies more. People who have gained much experience- working in multiple companies tend to stay in the company they join as seen above.

In [ ]:
```
role_att = Data.groupby(['YearsInCurrentRole','Attrition']).apply(lambda x : x['DailyRate'].count()).reset_index(name='Counts')
fig = px.line(role_att, x = 'YearsInCurrentRole', y = 'Counts', color = 'Attrition', title = 'Employee Attrition based on working yea
fig.show(renderer='notebook')
```

Employees are more prone to leave the company in starting years of their current role that is less than 2 years on an average. Employees tend to stay longer in their current role when they are in it for a long period of time.

In [ ]:
```
hike_att= Data.groupby(['PercentSalaryHike','Attrition']).apply(lambda x : x['DailyRate'].count()).reset_index(name='Counts')
fig = px.line(hike_att, x ='PercentSalaryHike', y ='Counts', color = 'Attrition', title = 'Employee Attrition based on Salary Percenta
fig.show(renderer='notebook')
```

We see here that higher salary hikes motivate people to work better and stay in the organization which is logically very obvious and hence, we see that chances of an employee leaving the organization is much more when the hike is lower which can be easily predicted.

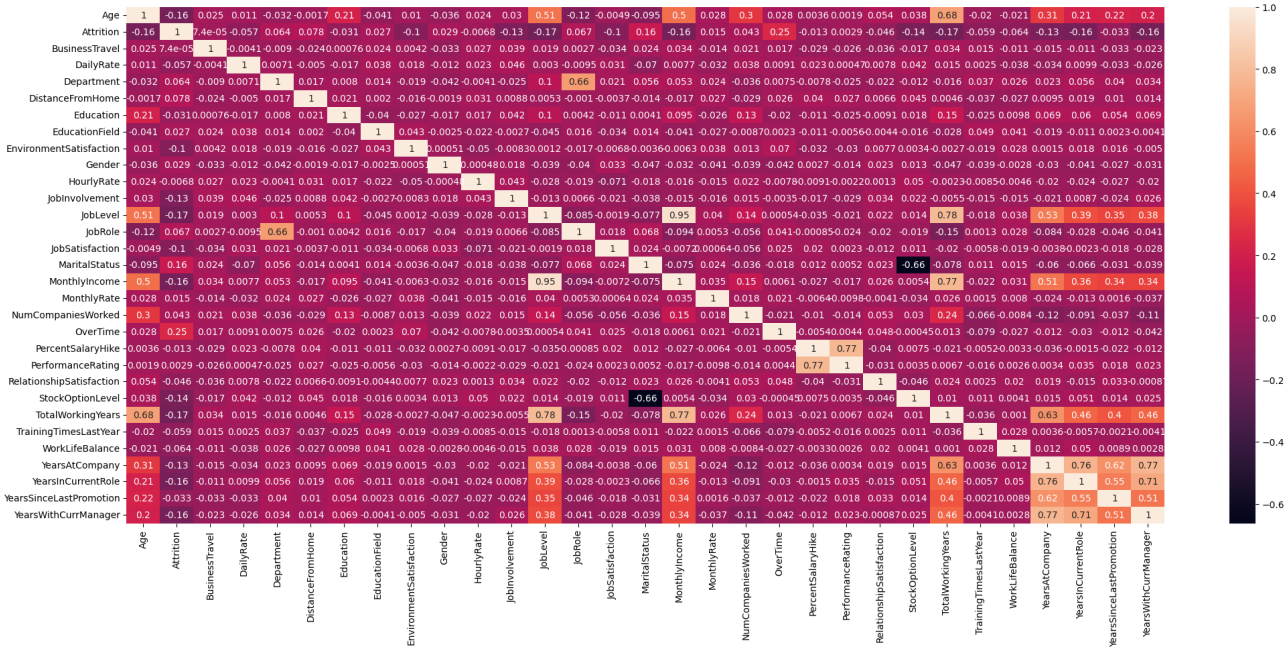
In [ ]:
```
boss_att = Data.groupby(['YearsWithCurrManager','Attrition']).apply(lambda x : x['DailyRate'].count()).reset_index(name='Counts')
fig = px.line(boss_att, x = 'YearsWithCurrManager', y = 'Counts', color = 'Attrition', title = 'Employee Attrition based on Years Spe
fig.show(renderer='notebook')
```

Clearly, we can see three major spikes in attrition rate. First, when the time spent with the manager is relatively very less, people tend to leave their jobs considering their relationship with their previous managers. Second, when the time spend is 2 years on an average perhaps employees feel the need to improve, they tend to go for a change. Third,when the time spent with the manager is more about 7 years, employees tend to leave and switch again. In addition, when the relative time spent with a manager is very high, employees tend to stick to the company adapting to the manager's behaviour. Hence, the chances of an employee resigning then is significantly low.

In [ ]:
```
label=LabelEncoder()
cat = Data.select_dtypes('object')
for i in cat:
    Data[i] = label.fit_transform(Data[i])
```

In [ ]:
```
plt.figure(figsize=(25,10))
sns.heatmap(Data.corr(), annot=True)
```

Out[ ]: <Axes: >



Since Job Level and Monthly Income have high correlation, we will consider dropping Job level in our further analysis as it has comparatively high correlations with many variables.

In [ ]:
```
Data.columns
```

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

Specifying Target and Predictor Variables

In [ ]:
```
X = Data[['Age','DailyRate','DistanceFromHome','Education','EducationField','EnvironmentSatisfaction','Gender','HourlyRate','JobInvol
          'MaritalStatus','MonthlyIncome','MonthlyRate','NumCompaniesWorked','OverTime','PercentSalaryHike','PerformanceRating','Rela
          'TotalWorkingYears','TrainingTimesLastYear','WorkLifeBalance','YearsAtCompany','YearsInCurrentRole','YearsSinceLastPromotio
Y = Data['Attrition']
# Normalising Dataset
scale = MinMaxScaler(feature_range=(0,1))
X = scale.fit_transform(X)
```

In ML, we often come across a term called Imbalanced Data Distribution, when observations in one class are much higher or lower than the other classes. Standard ML techniques such as Logistic Regression have a bias towards the majority class & they tend to ignore the minority class. In short, if we have imbalanced data distribution in our dataset then our model becomes more prone to the case when minority class has negligible or very lesser recall. SMOTE (synthetic minority oversampling technique) is a method to solve the imbalance problem by randomly increasing minority class examples by replicating them. It generates the virtual training records by linear interpolation for the minority class. After the oversampling process, the data is reconstructed and several classification models can be applied for the processed data.

In [ ]:
```
sampler = SMOTE(random_state=42)
smote_predcitor, smote_target = sampler.fit_resample(X,Y)
```

Employee Attrition Prediction Model

Splitting the Datasets

In [ ]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.3 , random_state=48)
```

Selecting Models

In [ ]:
```
LR = LogisticRegression()
RF = RandomForestClassifier()
GB = GradientBoostingClassifier()
```

Training Models

In [ ]:
```
LR = LR.fit(X_train,Y_train)
RF = RF.fit(X_train,Y_train)
GB = GB.fit(X_train,Y_train)
```

Evaluating Models

In [ ]:
```
print('LOGISTIC REGRESSION')
print('Traning Model accuracy scores: {:.3f}'.format(LR.score(X_train, Y_train)))
print('Test Model accuracy scores: {:.3f}'.format(LR.score(X_test, Y_test)))
print('---------------------------------------')
print('RANDOM FOREST CLASSIFIER')
print('Traning Model accuracy scores: {:.3f}'.format(RF.score(X_train, Y_train)))
print('Test Model accuracy scores: {:.3f}'.format(RF.score(X_test, Y_test)))
print('---------------------------------------')
print('GRADIENT BOOSTING CLASSIFIER')
print('Traning Model accuracy scores: {:.3f}'.format(GB.score(X_train, Y_train)))
print('Test Model accuracy scores: {:.3f}'.format(GB.score(X_test, Y_test)))
```

```
LOGISTIC REGRESSION
Traning Model accuracy scores: 0.877
Test Model accuracy scores: 0.855
---------------------------------------
RANDOM FOREST CLASSIFIER
Traning Model accuracy scores: 1.000
Test Model accuracy scores: 0.859
---------------------------------------
GRADIENT BOOSTING CLASSIFIER
Traning Model accuracy scores: 0.960
Test Model accuracy scores: 0.868
```

We were able to use three models namely Logistic Regression, Random Forest Classifier and Gradient Boosting Classifier and all the three models show impressively high accuracy scores. Random Forest Classifier showing 100% accuracy for train data but only 86% accuracy for test data. Logistic Regression shows 87% for train and 85% for test which is also acceptable. Lastly, Gradient Boosting Classifier shows around 96% accuracy for test data whereas 87% accuracy for test data hence proving to be the best model out of all.