In [1]:

```
import pandas as pd
```

In [2]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [4]:

```
df = pd.read_csv('attrition_employee.csv')
```

In [5]:

```
df.head()
```

Out[5]:

	id	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EnvironmentSatisfaction	.
0	0	36	Travel_Frequently	599	Research & Development	24	3	Medical	1	4	.
1	1	35	Travel_Rarely	921	Sales	8	3	Other	1	1	.
2	2	32	Travel_Rarely	718	Sales	26	3	Marketing	1	3	.
3	3	38	Travel_Rarely	1488	Research & Development	2	3	Medical	1	3	.
4	4	50	Travel_Rarely	1017	Research & Development	5	4	Medical	1	2	.

5 rows × 35 columns

In [6]:

```
df.tail()
```

Out[6]:

	id	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EnvironmentSatisfac
1672	1672	30	Travel_Rarely	945	Sales	1	3	Life Sciences	1	
1673	1673	32	Travel_Rarely	1303	Research & Development	2	3	Life Sciences	1	
1674	1674	29	Travel_Frequently	1184	Human Resources	24	3	Human Resources	1	
1675	1675	36	Travel_Rarely	441	Sales	9	2	Marketing	1	
1676	1676	36	Travel_Rarely	1141	Research & Development	20	3	Life Sciences	1	

5 rows × 35 columns

In [7]:

```
df.shape
```

Out[7]:

(1677, 35)

In [8]:

```
df.columns
```

Out[8]:

```
Index(['id', 'Age', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       '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', 'Attrition'],
      dtype='object')
```

In [9]:

```
df.duplicated().sum()
```

Out[9]:

```
0
```

In [10]:

```
df.isnull().sum()
```

Out[10]:

```
id                          0
Age                         0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               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
Attrition                   0
dtype: int64
```

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1677 entries, 0 to 1676
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   id                        1677 non-null   int64 
 1   Age                       1677 non-null   int64 
 2   BusinessTravel            1677 non-null   object
 3   DailyRate                 1677 non-null   int64 
 4   Department                1677 non-null   object
 5   DistanceFromHome          1677 non-null   int64 
 6   Education                 1677 non-null   int64 
 7   EducationField            1677 non-null   object
 8   EmployeeCount             1677 non-null   int64 
 9   EnvironmentSatisfaction   1677 non-null   int64 
 10  Gender                    1677 non-null   object
 11  HourlyRate                1677 non-null   int64 
 12  JobInvolvement            1677 non-null   int64 
 13  JobLevel                  1677 non-null   int64 
 14  JobRole                   1677 non-null   object
 15  JobSatisfaction           1677 non-null   int64 
 16  MaritalStatus             1677 non-null   object
 17  MonthlyIncome             1677 non-null   int64 
 18  MonthlyRate               1677 non-null   int64 
 19  NumCompaniesWorked        1677 non-null   int64 
 20  Over18                    1677 non-null   object
 21  OverTime                  1677 non-null   object
 22  PercentSalaryHike         1677 non-null   int64 
 23  PerformanceRating         1677 non-null   int64 
 24  RelationshipSatisfaction  1677 non-null   int64 
 25  StandardHours             1677 non-null   int64 
 26  StockOptionLevel          1677 non-null   int64 
 27  TotalWorkingYears         1677 non-null   int64 
 28  TrainingTimesLastYear     1677 non-null   int64 
 29  WorkLifeBalance           1677 non-null   int64 
 30  YearsAtCompany            1677 non-null   int64 
 31  YearsInCurrentRole        1677 non-null   int64 
 32  YearsSinceLastPromotion   1677 non-null   int64 
 33  YearsWithCurrManager      1677 non-null   int64 
 34  Attrition                 1677 non-null   int64 
dtypes: int64(27), object(8)
memory usage: 458.7+ KB
```

In [12]:

```
df.describe()
```

Out[12]:

	id	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EnvironmentSatisfaction	HourlyRate	JobInvol
count	1677.000000	1677.000000	1677.000000	1677.000000	1677.000000	1677.0	1677.000000	1677.000000	1677
mean	838.000000	36.036971	892.749553	8.683959	2.937984	1.0	2.757901	67.798450	2
std	484.252517	8.507112	374.496259	7.826143	1.039078	0.0	1.086835	19.435928	0
min	0.000000	18.000000	107.000000	1.000000	1.000000	1.0	1.000000	30.000000	1
25%	419.000000	30.000000	589.000000	2.000000	2.000000	1.0	2.000000	51.000000	2
50%	838.000000	35.000000	890.000000	7.000000	3.000000	1.0	3.000000	69.000000	3
75%	1257.000000	41.000000	1223.000000	12.000000	4.000000	1.0	4.000000	84.000000	3
max	1676.000000	60.000000	3921.000000	29.000000	15.000000	1.0	4.000000	100.000000	4

8 rows × 27 columns

In [13]:

```
df.nunique()
```

Out[13]:

```
id                          1677
Age                           43
BusinessTravel                 3
DailyRate                    625
Department                     3
DistanceFromHome              29
Education                      6
EducationField                 6
EmployeeCount                  1
EnvironmentSatisfaction        4
Gender                         2
HourlyRate                    71
JobInvolvement                 4
JobLevel                       6
JobRole                        9
JobSatisfaction                4
MaritalStatus                  3
MonthlyIncome                895
MonthlyRate                  903
NumCompaniesWorked            10
Over18                         1
OverTime                       2
PercentSalaryHike             15
PerformanceRating              2
RelationshipSatisfaction       4
StandardHours                  1
StockOptionLevel               4
TotalWorkingYears             41
TrainingTimesLastYear          7
WorkLifeBalance                4
YearsAtCompany                34
YearsInCurrentRole            19
YearsSinceLastPromotion       16
YearsWithCurrManager          18
Attrition                      2
dtype: int64
```

In [14]:

```
df_new = df[['BusinessTravel', 'Department', 'Education', 'EducationField',
            'EnvironmentSatisfaction', 'Gender', 'JobInvolvement', 'JobRole',
            'JobSatisfaction', 'MaritalStatus', 'NumCompaniesWorked', 'OverTime',
             'PerformanceRating', 'RelationshipSatisfaction', 'StockOptionLevel',
             'TrainingTimesLastYear', 'WorkLifeBalance', 'Attrition']]
```

In [15]:

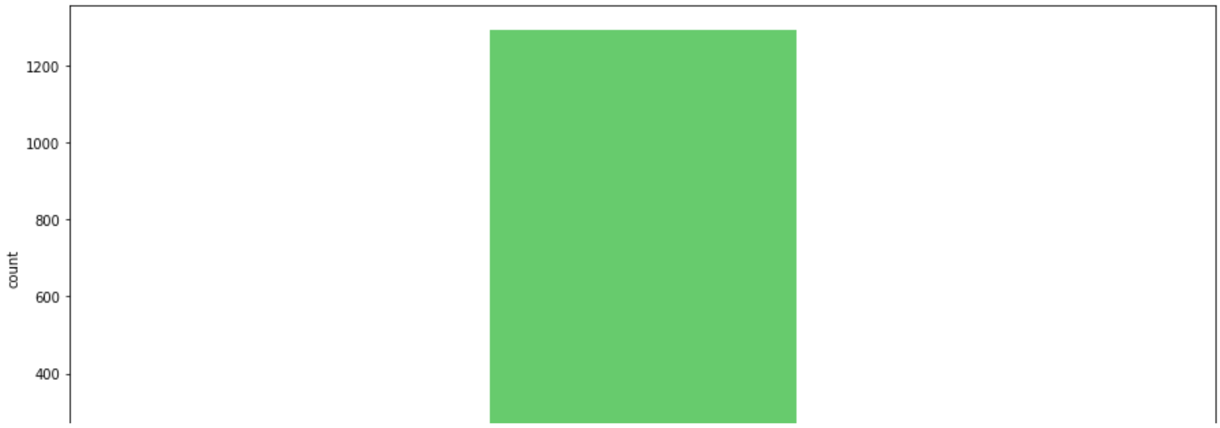
```
for i in df_new.columns:
    print(i,':')
    print('Unique Values:', df_new[i].unique())
    print('Value Counts:')
    print(df_new[i].value_counts())
    print('\n')
```

```
BusinessTravel :
Unique Values: ['Travel_Frequently' 'Travel_Rarely' 'Non-Travel']
Value Counts:
Travel_Rarely        1290
Travel_Frequently     261
Non-Travel            126
Name: BusinessTravel, dtype: int64


Department :
Unique Values: ['Research & Development' 'Sales' 'Human Resources']
Value Counts:
Research & Development    1167
Sales                      471
Human Resources             39
Name: Department, dtype: int64


Education :
Unique Values: [ 3  4  1  2  5 15]
```

In [16]:

```
for i in df_new.columns:
    plt.figure(figsize=[15,7],)
    print('Countplot for:', i)
    sns.countplot(df_new[i], data = df_new, palette = 'hls')
    plt.xticks(rotation = 0)
    plt.show()
    print('\n')
```

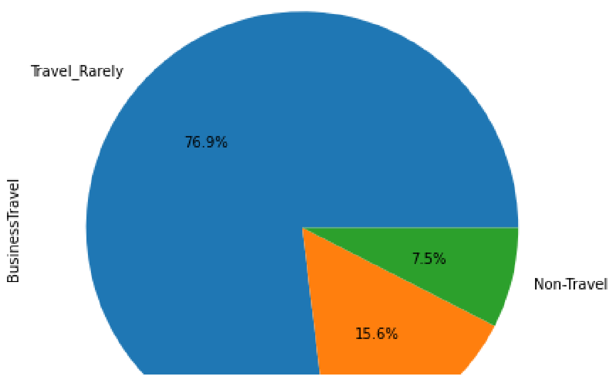
Countplot for: BusinessTravel



In [17]:

```
for i in df_new.columns:
    plt.figure(figsize=[15,7],)
    print('Pieplot for:', i)
    df_new[i].value_counts().plot(kind='pie',autopct='%1.1f%%')
    plt.show()
    print('\n')
```

Pieplot for: BusinessTravel



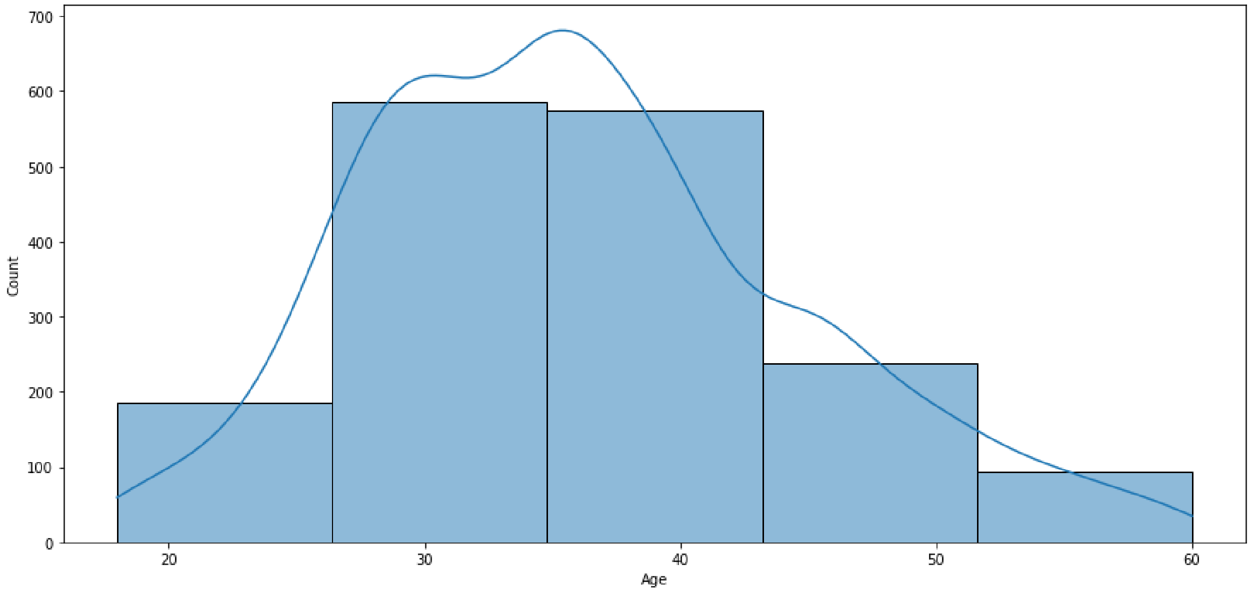
In [18]:

```
int_cols = [col for col in df.columns if df[col].dtype == 'int64']
print('Integer columns:', int_cols)

print('\n')

obj_cols = [col for col in df.columns if df[col].dtype == 'object']
print('Object columns:', obj_cols)
```

```
Integer columns: ['id', 'Age', 'DailyRate', 'DistanceFromHome', 'Education', 'EmployeeCount', 'EnvironmentSatisfact
ion', 'HourlyRate', 'JobInvolvement', 'JobLevel', 'JobSatisfaction', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesW
orked', 'PercentSalaryHike', 'PerformanceRating', 'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinc
eLastPromotion', 'YearsWithCurrManager', 'Attrition']


Object columns: ['BusinessTravel', 'Department', 'EducationField', 'Gender', 'JobRole', 'MaritalStatus', 'Over18',
'OverTime']
```

In [19]:

```
plt.figure(figsize=[15,7],)
sns.histplot(df['Age'], kde = 'True', bins = 5, palette = 'hls')
plt.xticks(rotation = 0)
plt.show()
```

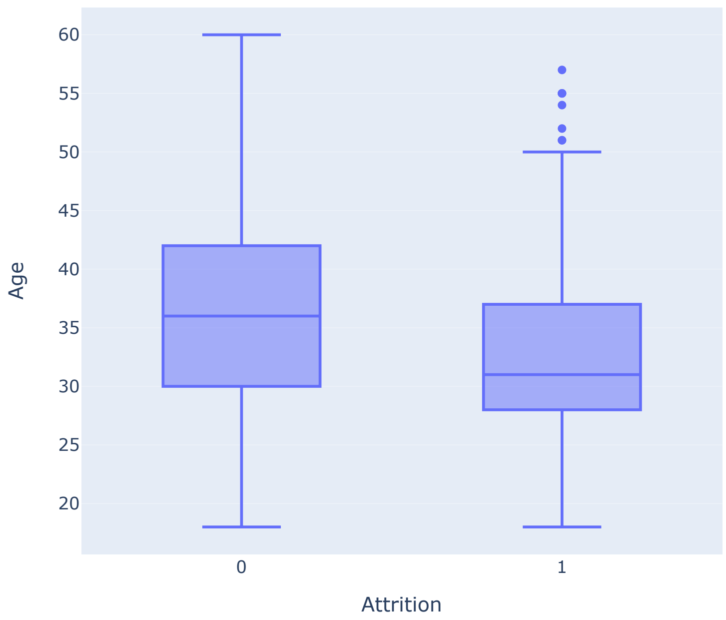


In [20]:

```
import plotly.express as px
```

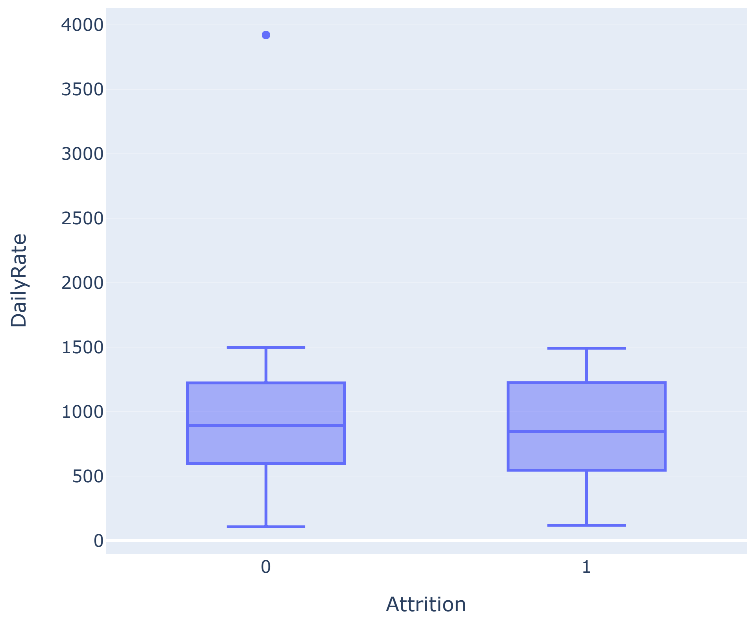
In [21]:

```
fig = px.box(df, x = 'Attrition', y = 'Age')
fig.show()
```



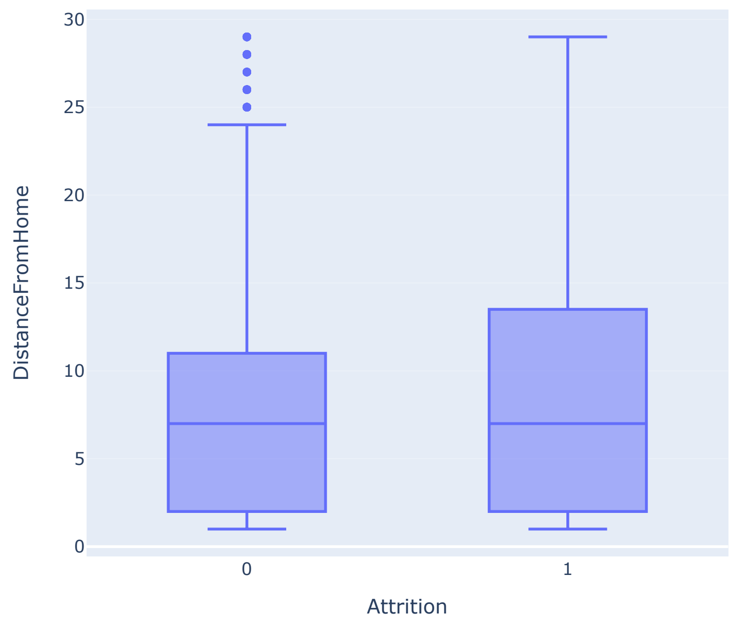
In [22]:

```
fig = px.box(df, x = 'Attrition', y = 'DailyRate')
fig.show()
```



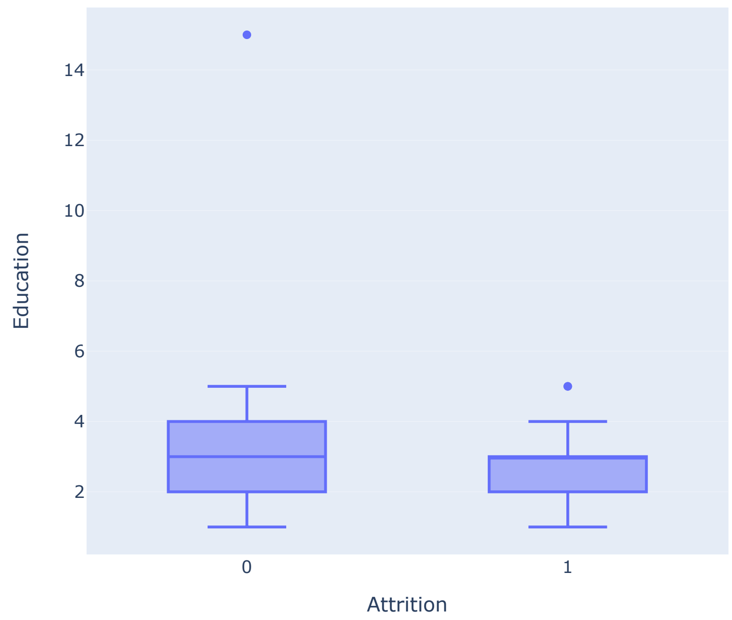
In [23]:

```
fig = px.box(df, x = 'Attrition', y = 'DistanceFromHome')
fig.show()
```



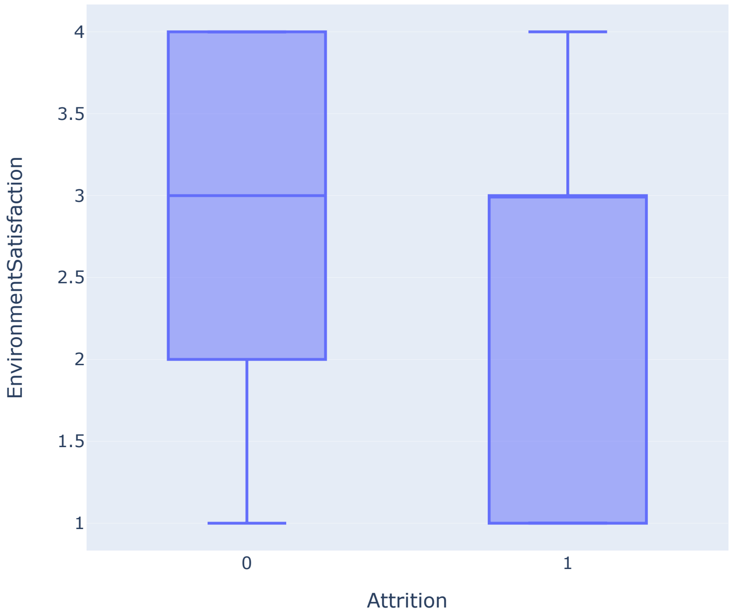
In [24]:

```
fig = px.box(df, x = 'Attrition', y = 'Education')
fig.show()
```



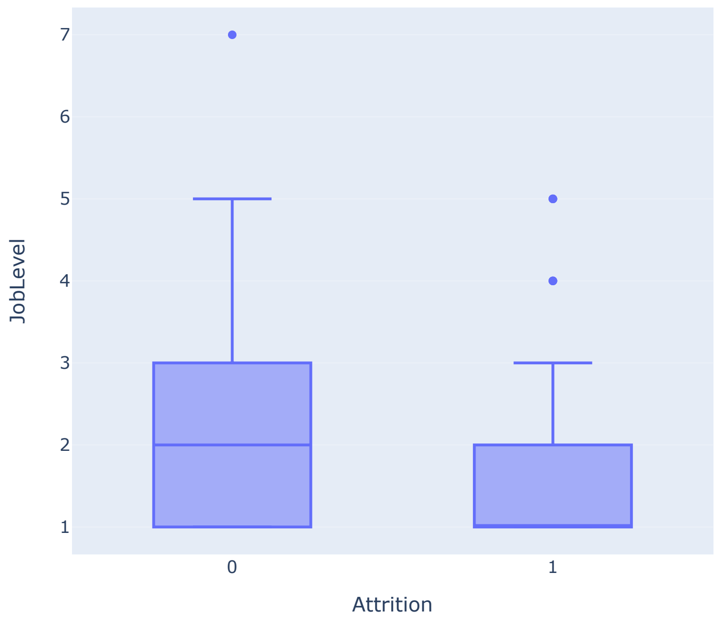
In [25]:

```
fig = px.box(df, x = 'Attrition', y = 'EnvironmentSatisfaction')
fig.show()
```



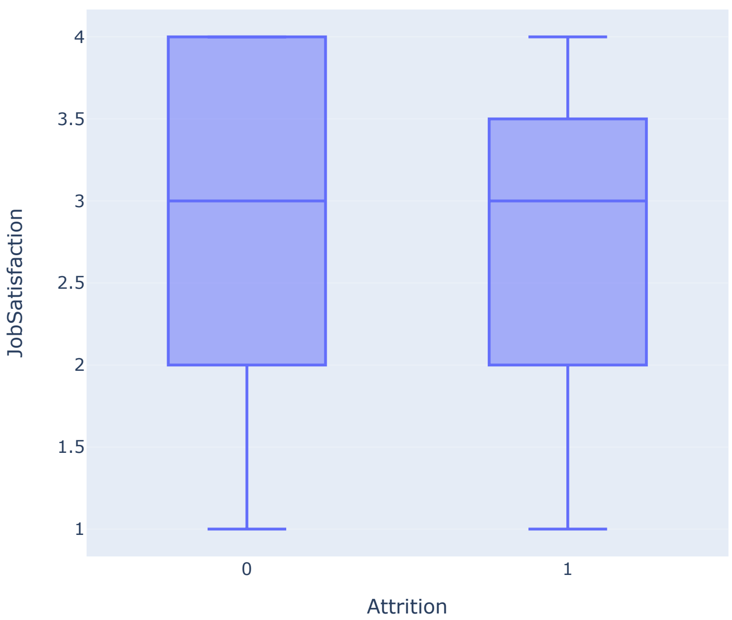
In [26]:

```
fig = px.box(df, x = 'Attrition', y = 'JobLevel')
fig.show()
```



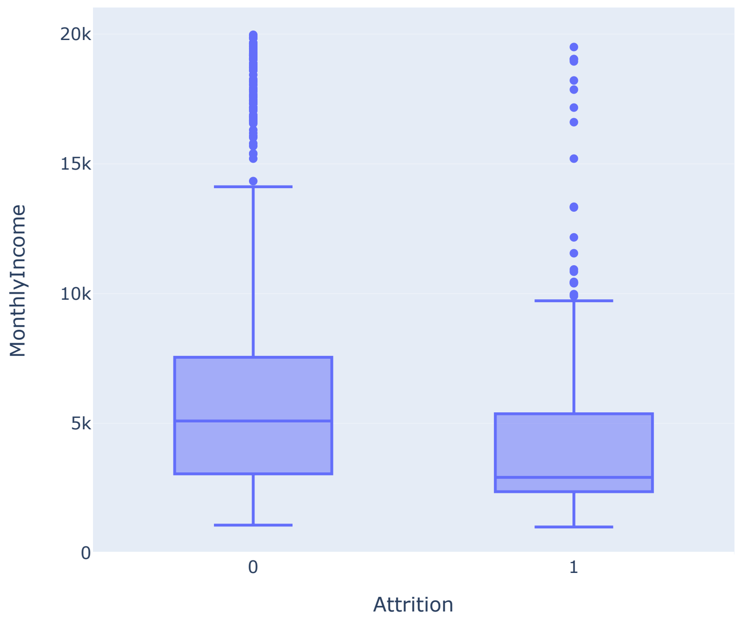
In [27]:

```
fig = px.box(df, x = 'Attrition', y = 'JobSatisfaction')
fig.show()
```



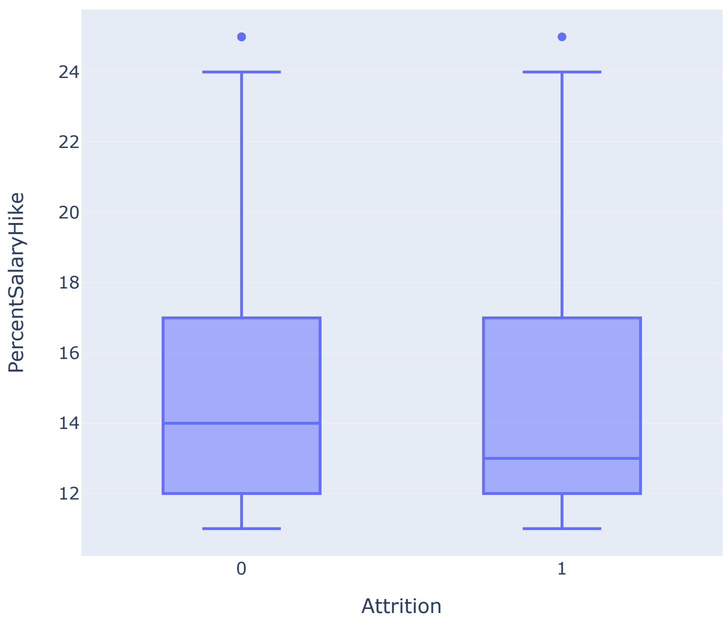
In [28]:

```
fig = px.box(df, x = 'Attrition', y = 'MonthlyIncome')
fig.show()
```



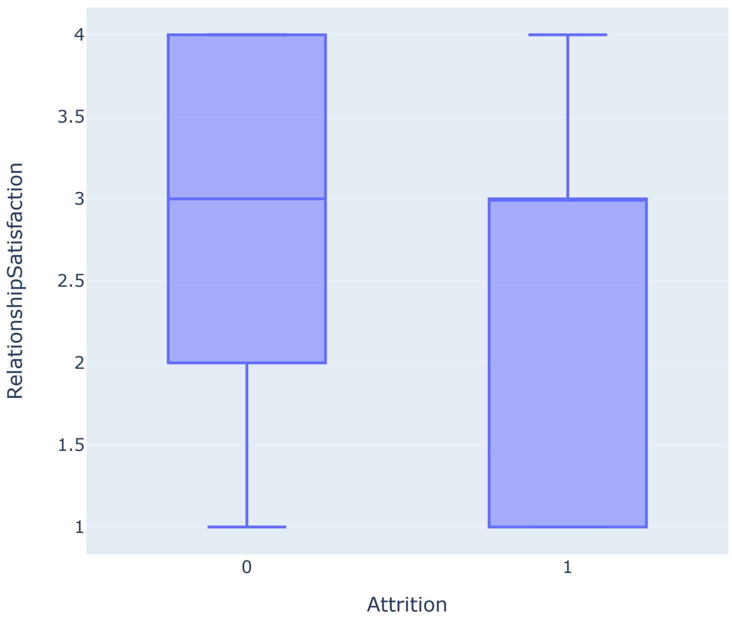
In [29]:

```
fig = px.box(df, x = 'Attrition', y = 'PercentSalaryHike')
fig.show()
```



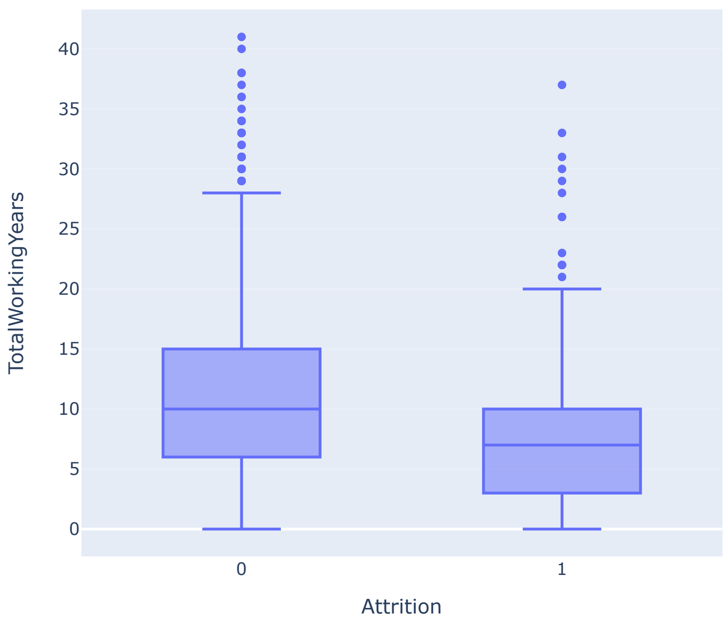
In [30]:

```
fig = px.box(df, x = 'Attrition', y = 'RelationshipSatisfaction')
fig.show()
```



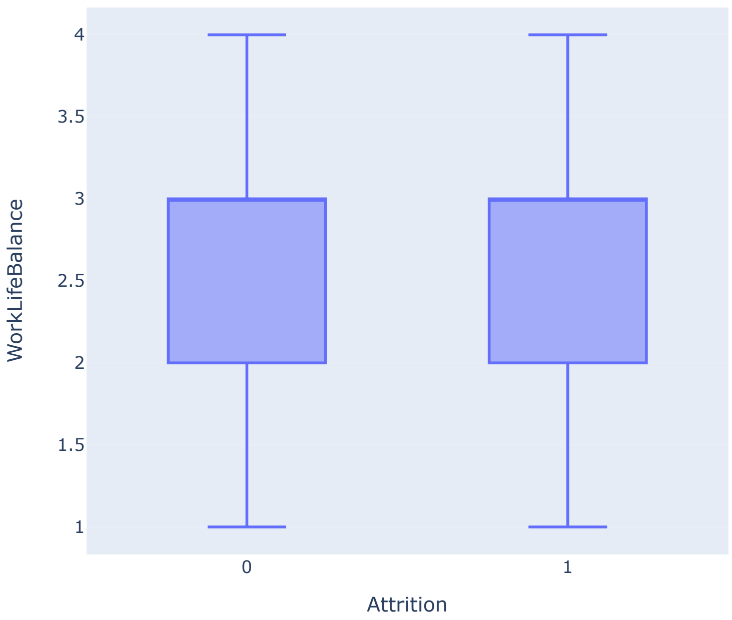
In [31]:

```
fig = px.box(df, x = 'Attrition', y = 'TotalWorkingYears')
fig.show()
```



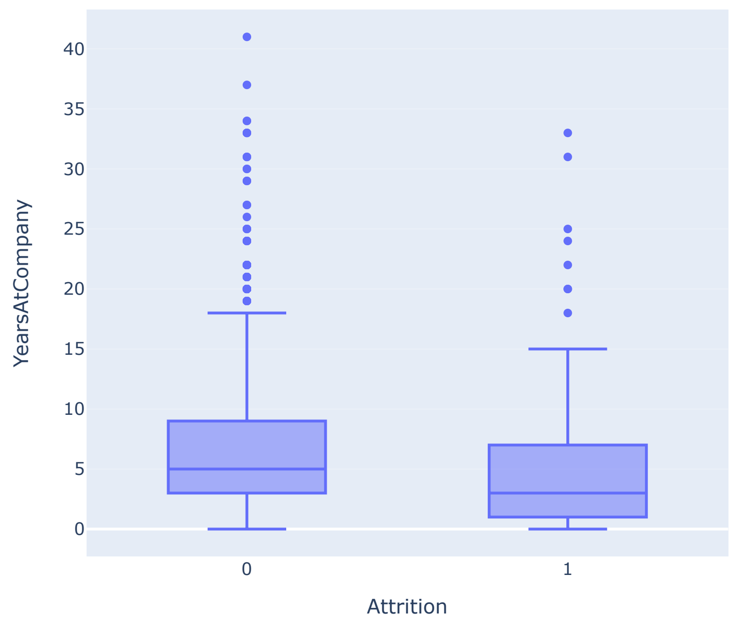
In [32]:

```
fig = px.box(df, x = 'Attrition', y = 'WorkLifeBalance')
fig.show()
```



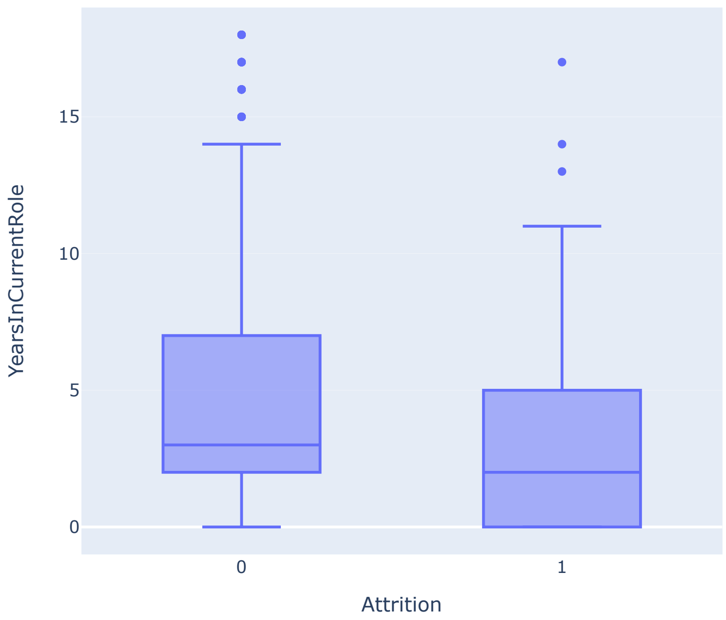
In [33]:

```
fig = px.box(df, x = 'Attrition', y = 'YearsAtCompany')
fig.show()
```



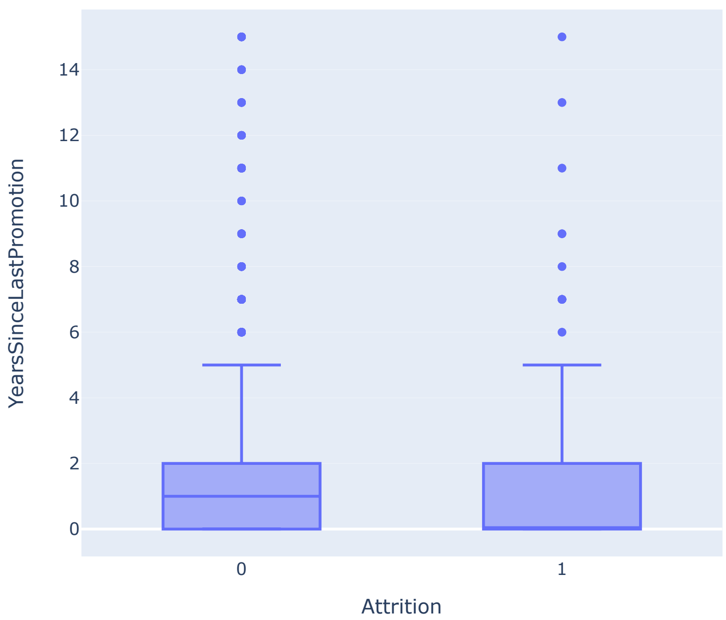
In [34]:

```
fig = px.box(df, x = 'Attrition', y = 'YearsInCurrentRole')
fig.show()
```



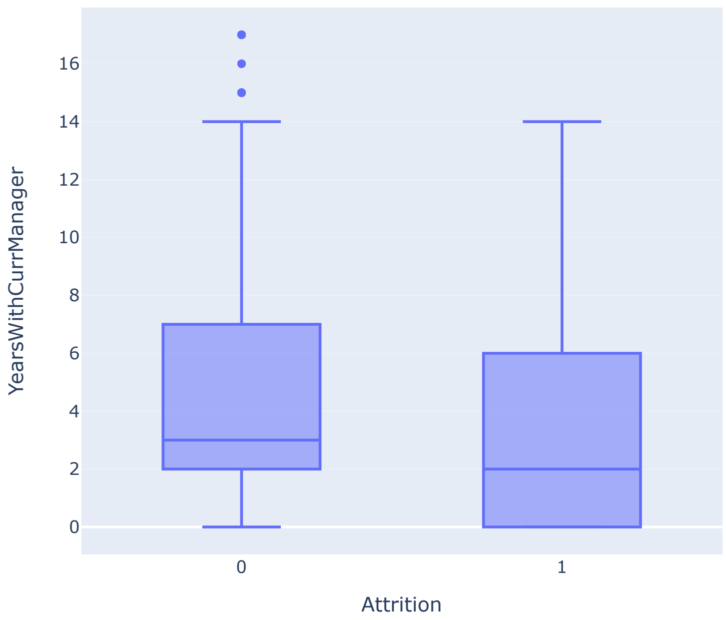
In [35]:

```
fig = px.box(df, x = 'Attrition', y = 'YearsSinceLastPromotion')
fig.show()
```



In [36]:

```
fig = px.box(df, x = 'Attrition', y = 'YearsWithCurrManager')
fig.show()
```



In [37]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['BusinessTravel'] = le.fit_transform(df['BusinessTravel'])
df['Department'] = le.fit_transform(df['Department'])
df['EducationField'] = le.fit_transform(df['EducationField'])
df['JobRole'] = le.fit_transform(df['JobRole'])
df['Gender'] = le.fit_transform(df['Gender'])
df['MaritalStatus'] = le.fit_transform(df['MaritalStatus'])
df['Over18'] = le.fit_transform(df['Over18'])
df['OverTime'] = le.fit_transform(df['OverTime'])
```

In [38]:

```
df.head()
```

Out[38]:

	id	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EnvironmentSatisfaction	...
0	0	36	1	599	1	24	3	3	1	4	...
1	1	35	2	921	2	8	3	4	1	1	...
2	2	32	2	718	2	26	3	2	1	3	...
3	3	38	2	1488	1	2	3	3	1	3	...
4	4	50	2	1017	1	5	4	3	1	2	...

5 rows × 35 columns

In [39]:

```
corr = df.corr()
```

In [40]:

```
corr
```

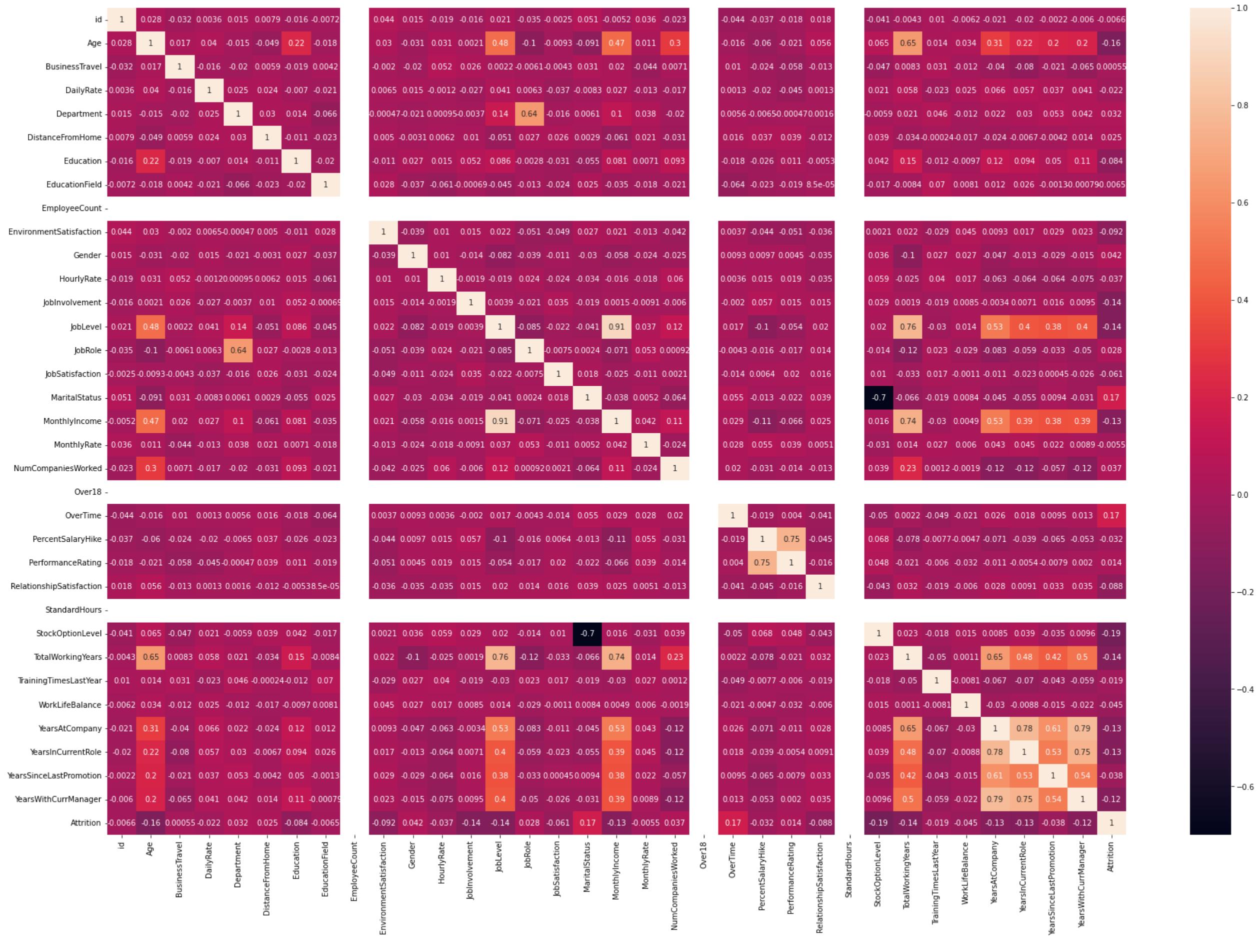
Out[40]:

	id	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Employee
id	1.000000	0.027563	-0.031831	0.003572	0.014652	0.007871	-0.015997	-0.007152	
Age	0.027563	1.000000	0.017462	0.039686	-0.014663	-0.049025	0.223545	-0.018181	
BusinessTravel	-0.031831	0.017462	1.000000	-0.015708	-0.020004	0.005933	-0.018889	0.004240	
DailyRate	0.003572	0.039686	-0.015708	1.000000	0.025107	0.024168	-0.007035	-0.021046	
Department	0.014652	-0.014663	-0.020004	0.025107	1.000000	0.029781	0.013881	-0.065715	
DistanceFromHome	0.007871	-0.049025	0.005933	0.024168	0.029781	1.000000	-0.011436	-0.023405	
Education	-0.015997	0.223545	-0.018889	-0.007035	0.013881	-0.011436	1.000000	-0.019603	
EducationField	-0.007152	-0.018181	0.004240	-0.021046	-0.065715	-0.023405	-0.019603	1.000000	
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
EnvironmentSatisfaction	0.044271	0.029557	-0.002002	0.006483	-0.000465	0.004959	-0.011189	0.028366	
Gender	0.014820	-0.031498	-0.019574	0.015338	-0.020542	-0.003123	0.027398	-0.036537	
HourlyRate	-0.018994	0.030628	0.051914	-0.001213	0.000948	0.006191	0.014862	-0.060686	
JobInvolvement	-0.016039	0.002101	0.026406	-0.026725	-0.003713	0.010035	0.052390	-0.000689	
JobLevel	0.021245	0.479015	0.002184	0.041369	0.138378	-0.051008	0.085823	-0.045207	
JobRole	-0.035070	-0.102034	-0.006113	0.006331	0.643463	0.026561	-0.002808	-0.012677	
JobSatisfaction	-0.002511	-0.009273	-0.004297	-0.037459	-0.015714	0.026309	-0.030686	-0.023946	
MaritalStatus	0.051459	-0.091312	0.030960	-0.008254	0.006123	0.002864	-0.054694	0.024995	
MonthlyIncome	-0.005224	0.470758	0.019567	0.027375	0.099545	-0.061019	0.081054	-0.034944	
MonthlyRate	0.036047	0.010959	-0.044104	-0.013332	0.037719	0.020542	0.007133	-0.018051	
NumCompaniesWorked	-0.022619	0.300044	0.007145	-0.017337	-0.020384	-0.031303	0.092789	-0.021372	
Over18	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
OverTime	-0.044391	-0.016417	0.010134	0.001339	0.005621	0.016349	-0.017772	-0.063571	
PercentSalaryHike	-0.036811	-0.060012	-0.024196	-0.020007	-0.006533	0.036970	-0.025858	-0.022959	
PerformanceRating	-0.018226	-0.021206	-0.057984	-0.045213	-0.000466	0.039206	0.010790	-0.019065	
RelationshipSatisfaction	0.018472	0.056115	-0.013382	0.001315	0.001612	-0.011868	-0.005253	0.000085	
StandardHours	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
StockOptionLevel	-0.041337	0.064863	-0.047337	0.021273	-0.005868	0.039086	0.041722	-0.016655	
TotalWorkingYears	-0.004288	0.648047	0.008332	0.058044	0.021490	-0.033573	0.153291	-0.008423	
TrainingTimesLastYear	0.010423	0.014303	0.031099	-0.023140	0.046363	-0.000239	-0.011924	0.070101	
WorkLifeBalance	-0.006236	0.034138	-0.012355	0.025152	-0.012399	-0.017184	-0.009697	0.008050	
YearsAtCompany	-0.020820	0.306628	-0.040091	0.066057	0.022345	-0.023564	0.116723	0.012349	
YearsInCurrentRole	-0.020064	0.219880	-0.079881	0.057011	0.029925	-0.006670	0.094065	0.025646	
YearsSinceLastPromotion	-0.002203	0.204357	-0.021144	0.037035	0.053127	-0.004215	0.050483	-0.001262	
YearsWithCurrManager	-0.005955	0.201601	-0.064886	0.040969	0.041859	0.013749	0.109573	-0.000790	
Attrition	-0.006598	-0.161044	0.000552	-0.022380	0.031996	0.024741	-0.084305	-0.006513	

35 rows × 35 columns

In [41]:

```
plt.figure(figsize=[30,20],)
sns.heatmap(corr, annot = True)
plt.show()
```



In [42]:

```
df1 = df.copy()
```

In [43]:

```
X = df1.drop(['Attrition'], axis = 1)
y = df1['Attrition']
```

In [44]:

```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
X = scaler.fit_transform(X)
```

In [45]:

```
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(X,y)
print(model.feature_importances_)
```

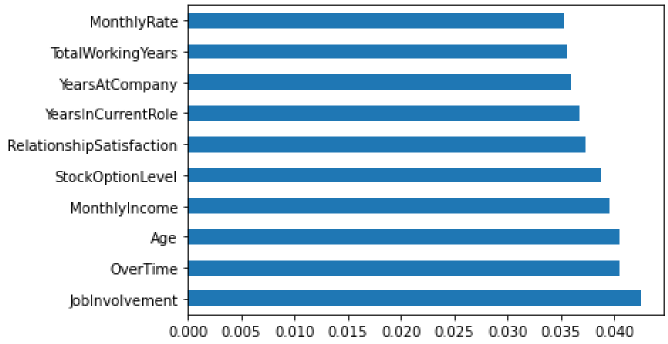
```
[0.03198568 0.04044102 0.02410835 0.03375663 0.02151207 0.03273632
 0.0303788  0.02818188 0.         0.03522922 0.01800049 0.03412623
 0.04250434 0.02814186 0.03392777 0.03168117 0.03413634 0.03949013
 0.03528703 0.03095158 0.         0.04054669 0.03360471 0.01302054
 0.03729354 0.         0.03881399 0.03549759 0.02990741 0.03018425
 0.03598791 0.03669997 0.027918   0.03394849]
```

In [46]:

```
X = df1.iloc[:,:-1]
```

In [47]:

```
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```



In [48]:

```
X_new = X[['StockOptionLevel', 'MonthlyIncome', 'JobInvolvement', 'OverTime',
          'Age', 'YearsInCurrentRole', 'YearsAtCompany', 'TotalWorkingYears',
          'JobRole', 'RelationshipSatisfaction']]
```

In [49]:

```
X_new.shape
```

Out[49]:

```
(1677, 10)
```

In [50]:

```
y.shape
```

Out[50]:

```
(1677,)
```

In [51]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(X_new, y, test_size= 0.30, random_state=0)
```

In [52]:

```
from sklearn.linear_model import LogisticRegression
log_r= LogisticRegression(random_state=0)
log_r.fit(X_train, y_train)
```

Out[52]:

```
LogisticRegression
LogisticRegression(random_state=0)
```

In [53]:

```
y_pred_lr= log_r.predict(X_test)
```

In [54]:

```
from sklearn.metrics import accuracy_score
```

In [55]:

```
accuracy_score(y_test, y_pred_lr)
```

Out[55]:

```
0.873015873015873
```

In [56]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred_lr)
```

In [57]:

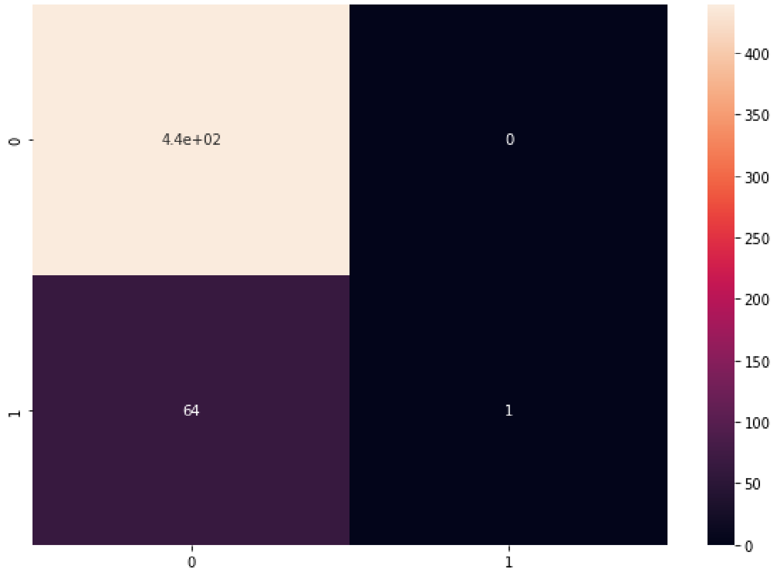
```
cm
```

Out[57]:

```
array([[439,   0],
       [ 64,   1]], dtype=int64)
```

In [58]:

```
plt.figure(figsize=[10,7],)
sns.heatmap(cm, annot = True)
plt.show()
```



In [59]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_lr))
```

```
              precision    recall  f1-score   support

           0       0.87      1.00      0.93       439
           1       1.00      0.02      0.03        65

    accuracy                           0.87       504
   macro avg       0.94      0.51      0.48       504
weighted avg       0.89      0.87      0.82       504
```

In [60]:

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
```

Out[60]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [61]:

```
y_pred_dt= dt.predict(X_test)
```

In [62]:

```
accuracy_score(y_test, y_pred_dt)
```

Out[62]:

```
0.8432539682539683
```

In [63]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred_dt)
```

In [64]:

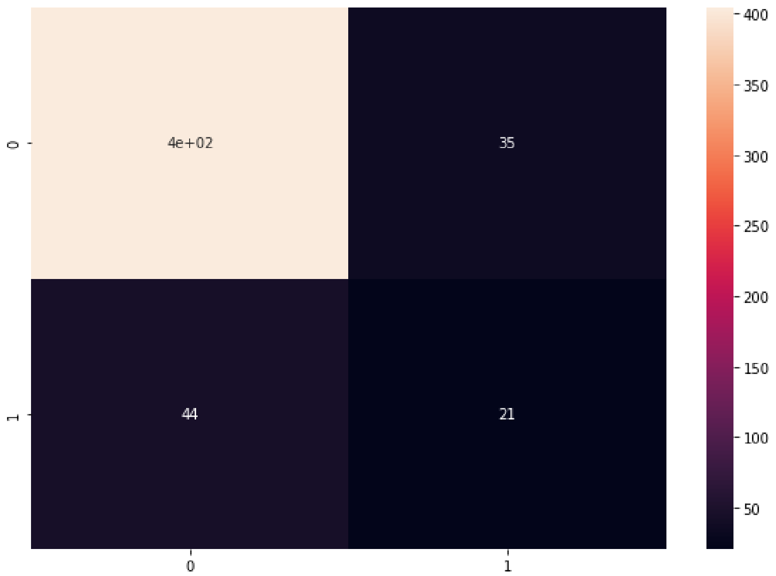
```
cm
```

Out[64]:

```
array([[404,  35],
       [ 44,  21]], dtype=int64)
```

In [65]:

```
plt.figure(figsize=[10,7],)
sns.heatmap(cm, annot = True)
plt.show()
```



In [66]:

```
print(classification_report(y_test, y_pred_dt))
```

```
              precision    recall  f1-score   support

           0       0.90      0.92      0.91       439
           1       0.38      0.32      0.35        65

    accuracy                           0.84       504
   macro avg       0.64      0.62      0.63       504
weighted avg       0.83      0.84      0.84       504
```

In [67]:

```
from sklearn.ensemble import RandomForestClassifier
rf_c = RandomForestClassifier(n_estimators= 10, criterion="entropy")
rf_c.fit(X_train, y_train)
```

Out[67]:

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

In [68]:

```
y_pred_rf_c= rf_c.predict(X_test)
```

In [69]:

```
accuracy_score(y_test, y_pred_rf_c)
```

Out[69]:

```
0.8611111111111112
```

In [70]:

```
cm= confusion_matrix(y_test, y_pred_rf_c)
```

In [71]:

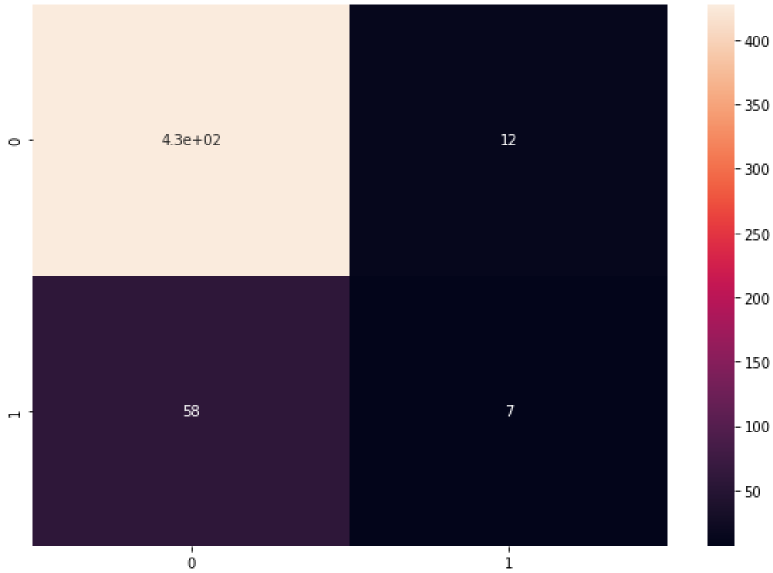
```
cm
```

Out[71]:

```
array([[427,  12],
       [ 58,   7]], dtype=int64)
```

In [72]:

```
plt.figure(figsize=[10,7],)
sns.heatmap(cm, annot = True)
plt.show()
```



In [73]:

```
print(classification_report(y_test, y_pred_rf_c))
```

```
              precision    recall  f1-score   support

           0       0.88      0.97      0.92       439
           1       0.37      0.11      0.17        65

    accuracy                           0.86       504
   macro avg       0.62      0.54      0.55       504
weighted avg       0.81      0.86      0.83       504
```

In [74]:

```
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
```

In [75]:

```
folds = StratifiedKFold(n_splits = 5, shuffle = True, random_state = 40)
```

In [76]:

```
def grid_search(model,folds,params,scoring):
    grid_search = GridSearchCV(model,
                            cv=folds,
                            param_grid=params,
                            scoring=scoring,
                            n_jobs=-1, verbose=1)
    return grid_search
```

In [77]:

```
def print_best_score_params(model):
    print("Best Score: ", model.best_score_)
    print("Best Hyperparameters: ", model.best_params_)
```

In [78]:

```
log_reg = LogisticRegression()
log_params = {'C': [0.01, 1, 10],
          'penalty': ['l1', 'l2'],
          'solver': ['liblinear','newton-cg','saga']
         }
grid_search_log = grid_search(log_reg, folds, log_params, scoring=None)
grid_search_log.fit(X_train, y_train)
print_best_score_params(grid_search_log)
```

```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
Best Score:  0.8934315330060011
Best Hyperparameters:  {'C': 1, 'penalty': 'l2', 'solver': 'newton-cg'}
```

In [79]:

```
dtc = DecisionTreeClassifier(random_state=40)
dtc_params = {
    'max_depth': [5,10,20,30],
    'min_samples_leaf': [5,10,20,30]
}
grid_search_dtc = grid_search(dtc, folds, dtc_params, scoring='roc_auc_ovr')
grid_search_dtc.fit(X_train, y_train)
print_best_score_params(grid_search_dtc)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best Score:  0.7096681152538641
Best Hyperparameters:  {'max_depth': 5, 'min_samples_leaf': 10}
```

In [80]:

```
rfc = RandomForestClassifier(random_state=40, n_jobs = -1,oob_score=True)
rfc_params = {'max_depth': [10,20,30,40],
          'min_samples_leaf': [5,10,15,20,30],
          'n_estimators': [100,200,500,700]
        }
grid_search_rfc = grid_search(rfc, folds, rfc_params, scoring='roc_auc_ovr')
grid_search_rfc.fit(X_train, y_train)
print('OOB SCORE :',grid_search_rfc.best_estimator_.oob_score_)
```

```
Fitting 5 folds for each of 80 candidates, totalling 400 fits
OOB SCORE : 0.896845694799659
```

In [81]:

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [82]:

```
model.fit(X_train, y_train, epochs=50, batch_size=32)
```

```
Epoch 1/50
37/37 [==============================] - 1s 2ms/step - loss: 3.0136 - accuracy: 0.7801
Epoch 2/50
37/37 [==============================] - 0s 1ms/step - loss: 3.1524 - accuracy: 0.8031
Epoch 3/50
37/37 [==============================] - 0s 1ms/step - loss: 1.0043 - accuracy: 0.7621
Epoch 4/50
37/37 [==============================] - 0s 1ms/step - loss: 4.4103 - accuracy: 0.8201
Epoch 5/50
37/37 [==============================] - 0s 960us/step - loss: 3.6005 - accuracy: 0.8005
Epoch 6/50
37/37 [==============================] - 0s 892us/step - loss: 3.6550 - accuracy: 0.8090
Epoch 7/50
37/37 [==============================] - 0s 970us/step - loss: 1.6914 - accuracy: 0.8031
Epoch 8/50
37/37 [==============================] - 0s 971us/step - loss: 3.9877 - accuracy: 0.7971
Epoch 9/50
37/37 [==============================] - 0s 920us/step - loss: 4.7610 - accuracy: 0.8210
Epoch 10/50
```

In [83]:

```
test_loss, test_acc = model.evaluate(X_test, y_test)
print('Test accuracy:', test_acc)
```

```
16/16 [==============================] - 0s 945us/step - loss: 2.3749 - accuracy: 0.8710
Test accuracy: 0.8710317611694336
```

In [84]:

```
df2 = pd.read_csv('employee_test.csv')
```

In [85]:

```
df2['BusinessTravel'] = le.fit_transform(df2['BusinessTravel'])
df2['Department'] = le.fit_transform(df2['Department'])
df2['EducationField'] = le.fit_transform(df2['EducationField'])
df2['JobRole'] = le.fit_transform(df2['JobRole'])
df2['Gender'] = le.fit_transform(df2['Gender'])
df2['MaritalStatus'] = le.fit_transform(df2['MaritalStatus'])
df2['Over18'] = le.fit_transform(df2['Over18'])
df2['OverTime'] = le.fit_transform(df2['OverTime'])
```

In [86]:

```
X_new = df2[['StockOptionLevel', 'MonthlyIncome', 'JobInvolvement', 'OverTime',
          'Age', 'YearsInCurrentRole', 'YearsAtCompany', 'TotalWorkingYears',
          'JobRole', 'RelationshipSatisfaction']]
```

In [87]:

```
X = scaler.fit_transform(X_new)
```

In [88]:

```
y_lr = grid_search_log.predict(X)
```

In [89]:

```
y_lr
```

Out[89]:

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

In [90]:

```
y_dt = grid_search_dtc.predict(X_new)
```

In [91]:

```
y_dt
```

Out[91]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [92]:

```
y_rfc = grid_search_rfc.predict(X_new)
```

In [93]:

```
y_rfc
```

Out[93]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [94]:

```
y_nn = model.predict(X_new)
```

```
35/35 [==============================] - 0s 911us/step
```

In [95]:

```
y_nn
```

Out[95]:

```
array([[1.8085128e-04],
       [1.2094801e-09],
       [1.0392133e-05],
       ...,
       [4.5603679e-10],
       [2.9436662e-09],
       [2.5529080e-04]], dtype=float32)
```