In [1]:
```
# Importing libraries for Data handling and analysis
import numpy as np
import pandas as pd
import seaborn as sns

# Libraries for plotting

import matplotlib.pyplot as plt
%matplotlib inline

# Modelling Algorithms

from sklearn.linear_model import LogisticRegression

#Model building

from patsy import dmatrices
import sklearn
```

In [4]:
```
# Importing the Dataset
df = pd.read_csv("IBM Attrition Data.csv")
```

In [5]:
```
# Display first five rows of Data
df.head()
```

Out[5]:

	Age	Attrition	Department	DistanceFromHome	Education	EducationField	EnvironmentS
0	41	Yes	Sales	1	2	Life Sciences	
1	49	No	Research & Development	8	1	Life Sciences	
2	37	Yes	Research & Development	2	2	Other	
3	33	No	Research & Development	3	4	Life Sciences	
4	27	No	Research & Development	2	1	Medical	

In [6]:
```
#Displaying no:of rows and columns
df.shape
```

Out[6]:
```
(1470, 13)
```

In [7]:
```
Names = df.columns.values
print(Names)
```

```
['Age' 'Attrition' 'Department' 'DistanceFromHome' 'Education'
 'EducationField' 'EnvironmentSatisfaction' 'JobSatisfaction'
 'MaritalStatus' 'MonthlyIncome' 'NumCompaniesWorked' 'WorkLifeBalance'
 'YearsAtCompany']
```

In [8]:
```
# Checking for Missing values
df.isnull().sum()
```

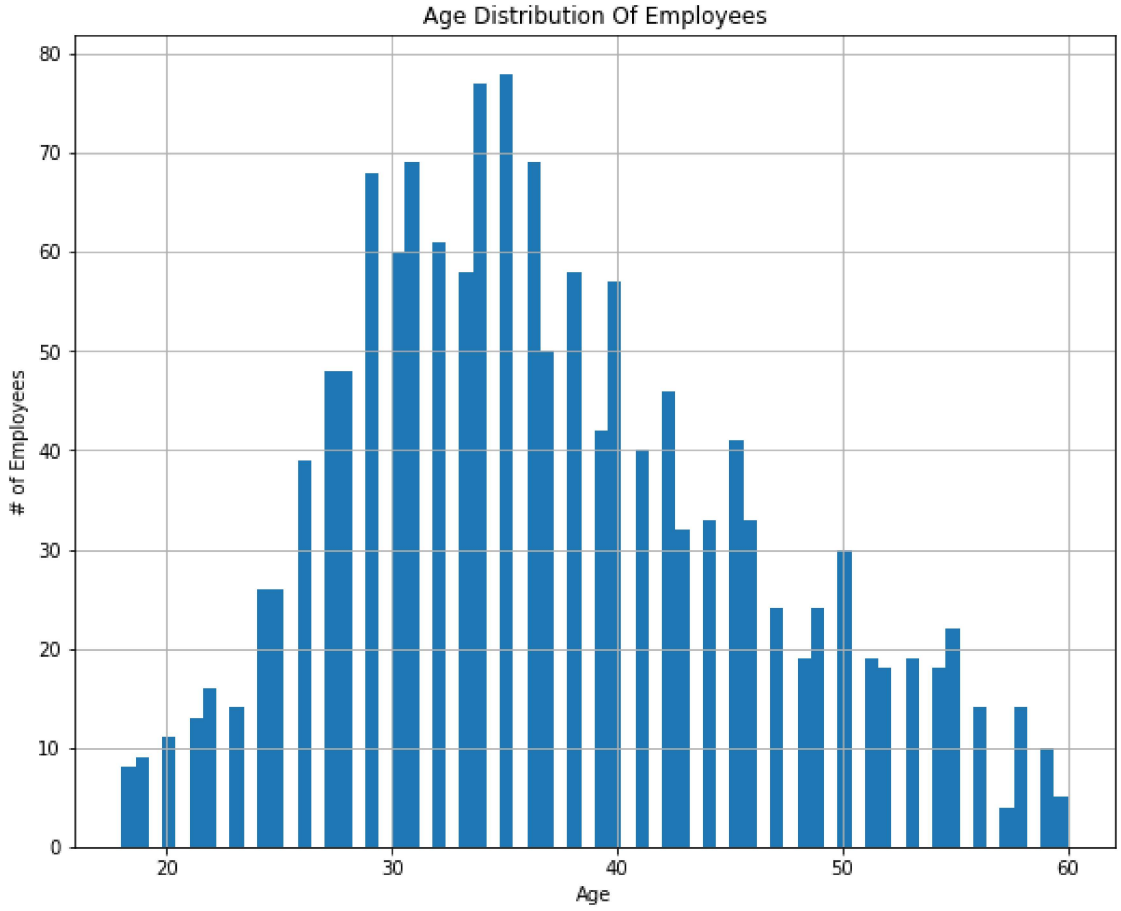
Out[8]:
```
Age                        0
Attrition                  0
Department                 0
DistanceFromHome           0
Education                  0
EducationField             0
EnvironmentSatisfaction    0
JobSatisfaction            0
MaritalStatus              0
MonthlyIncome              0
NumCompaniesWorked         0
WorkLifeBalance            0
YearsAtCompany             0
dtype: int64
```

In [9]:
```
# Displaying the count of 'yes' and 'no' values of the target variable
df['Attrition'].value_counts()
```

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

In [10]:

```
#Histogram for Age
plt.figure(figsize=(10,8))
df['Age'].hist(bins=70)
plt.title("Age Distribution Of Employees")
plt.xlabel("Age")
plt.ylabel("# of Employees")
plt.show()
```

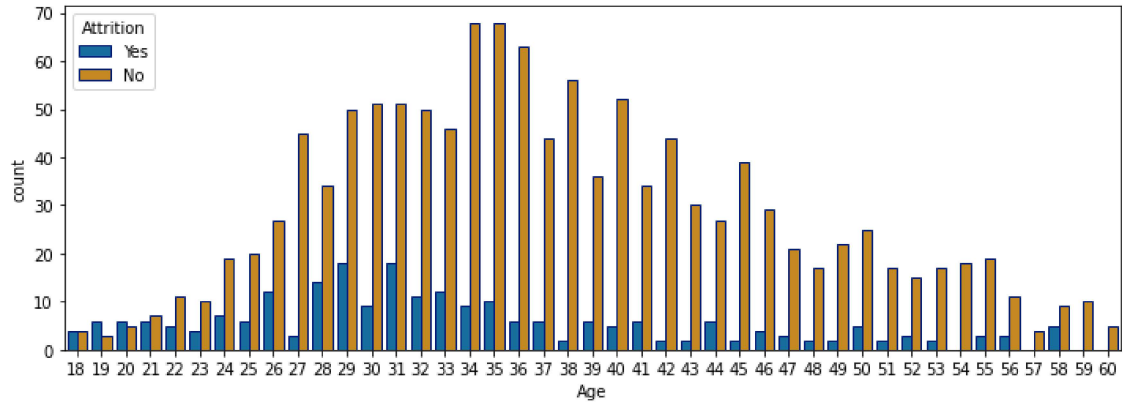


In [11]:

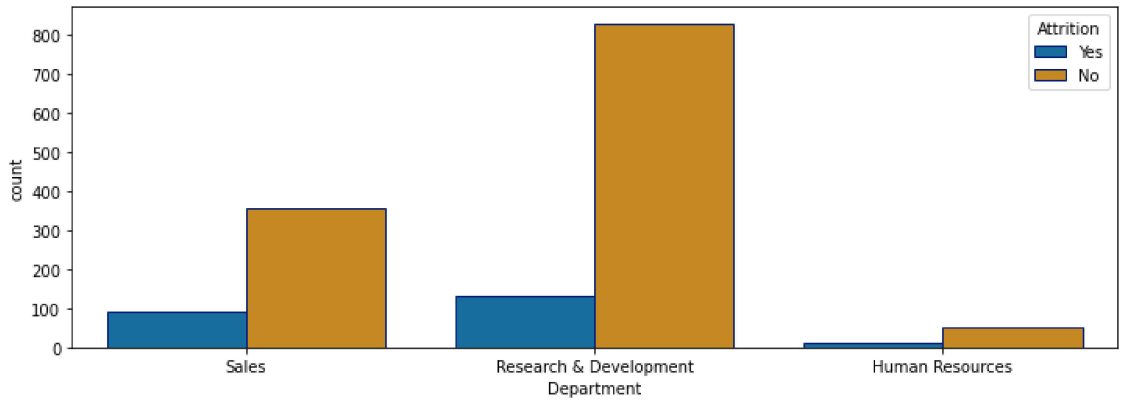
```
# Explore Data for Attrition by Age
plt.figure(figsize=(14,10))
plt.scatter(df.Attrition,df.Age, alpha=.55)
plt.title("Attrition by Age ")
plt.ylabel("Age")
plt.grid(visible=True, which='major',axis='y')
plt.show()
```

In [12]:

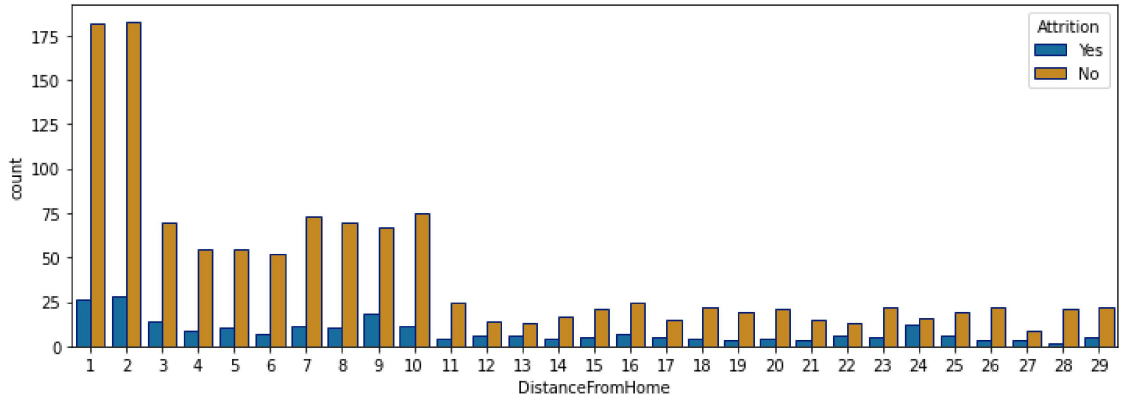
```
# Show the number of employees that left and stayed by age
fig_dims = (12, 4)
fig, ax = plt.subplots(figsize=fig_dims)
sns.countplot(x='Age', hue='Attrition', data=df, palette="colorblind", ax=
```



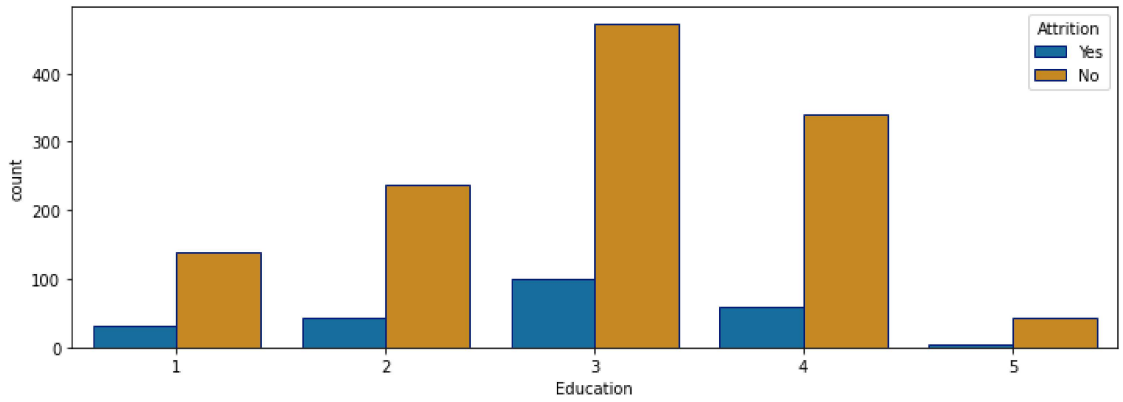
In [13]:
```
he number of employees that left and stayed by Department
 = (12, 4)
= plt.subplots(figsize=fig_dims)
tplot(x='Department', hue='Attrition', data=df, palette="colorblind", ax=ax
```



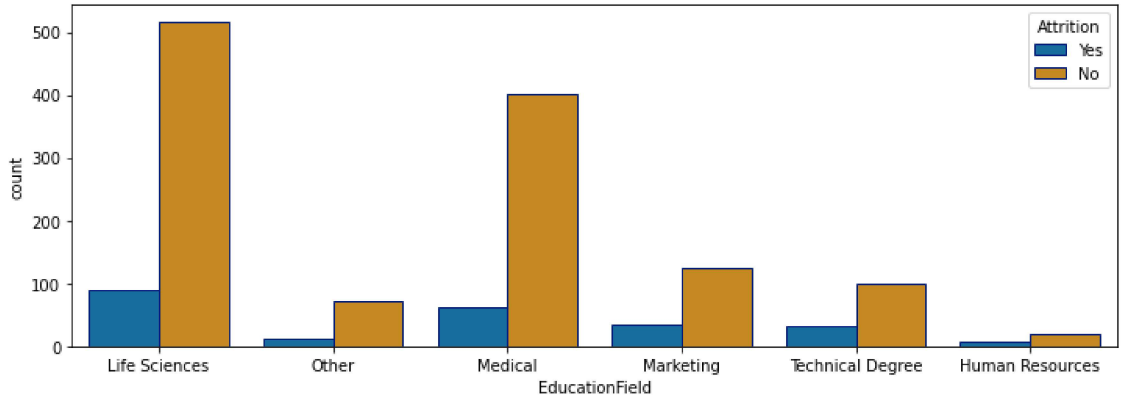
In [14]:
```
ber of employees that left and stayed by DistanceFromHome
, 4)
subplots(figsize=fig_dims)
x='DistanceFromHome', hue='Attrition', data=df, palette="colorblind", ax=ax
```



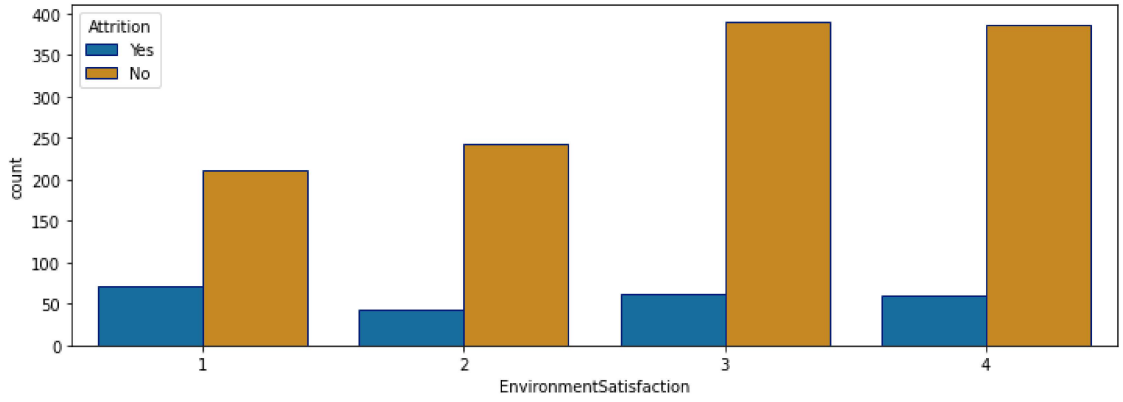
In [15]:
```
the number of employees that left and stayed by Education
s = (12, 4)
 = plt.subplots(figsize=fig_dims)
ntplot(x='Education', hue='Attrition', data=df, palette="colorblind", ax=ax
```



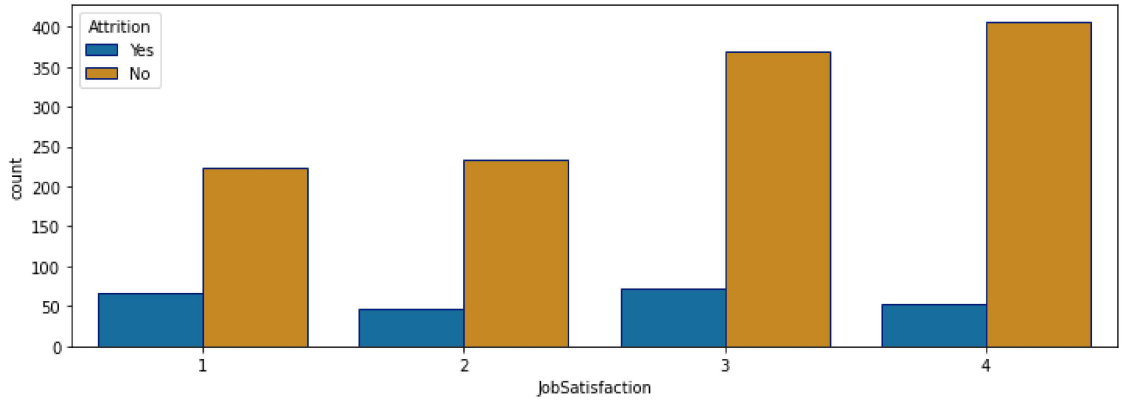
In [16]:
```
umber of employees that left and stayed by EducationField
12, 4)
t.subplots(figsize=fig_dims)
t(x='EducationField', hue='Attrition', data=df, palette="colorblind", ax=ax
```



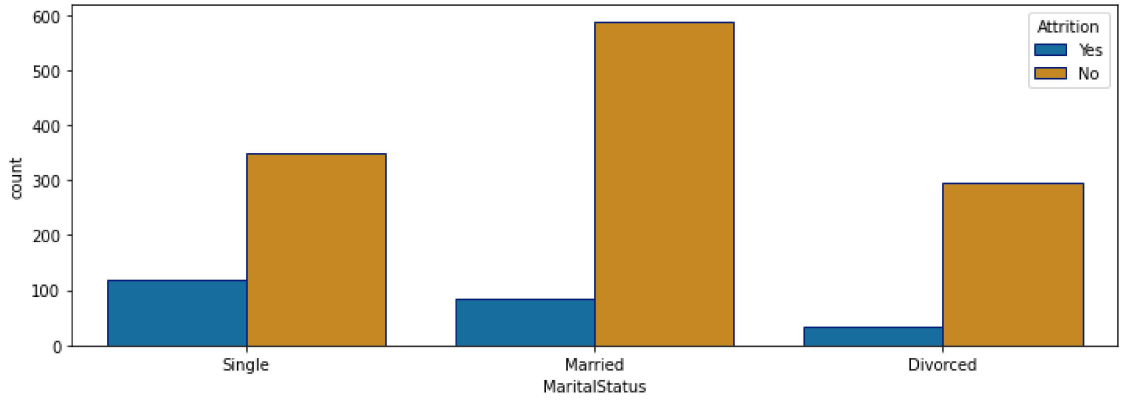
In [17]:
```
er of employees that left and stayed by EnvironmentSatisfaction
 4)
ubplots(figsize=fig_dims)
='EnvironmentSatisfaction', hue='Attrition', data=df, palette="colorblind",
```



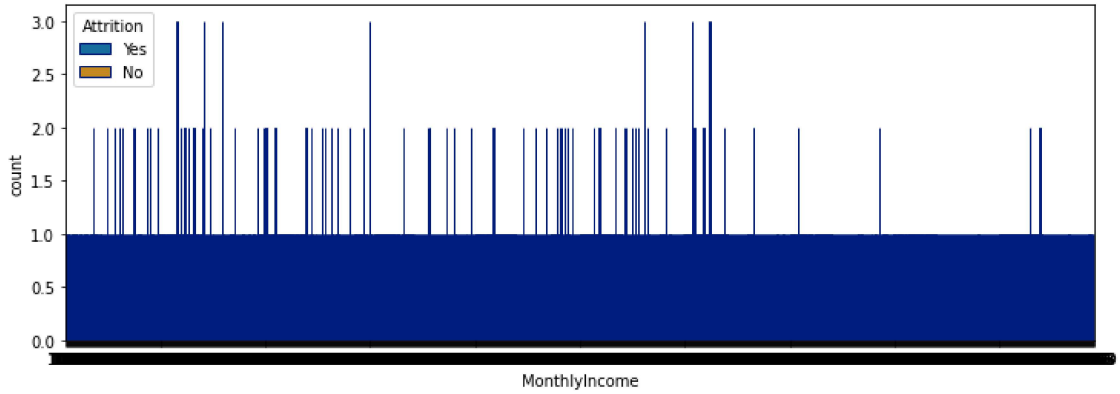
In [18]:
```
mber of employees that left and stayed by JobSatisfaction
2, 4)
.subplots(figsize=fig_dims)
(x='JobSatisfaction', hue='Attrition', data=df, palette="colorblind", ax=ax
```



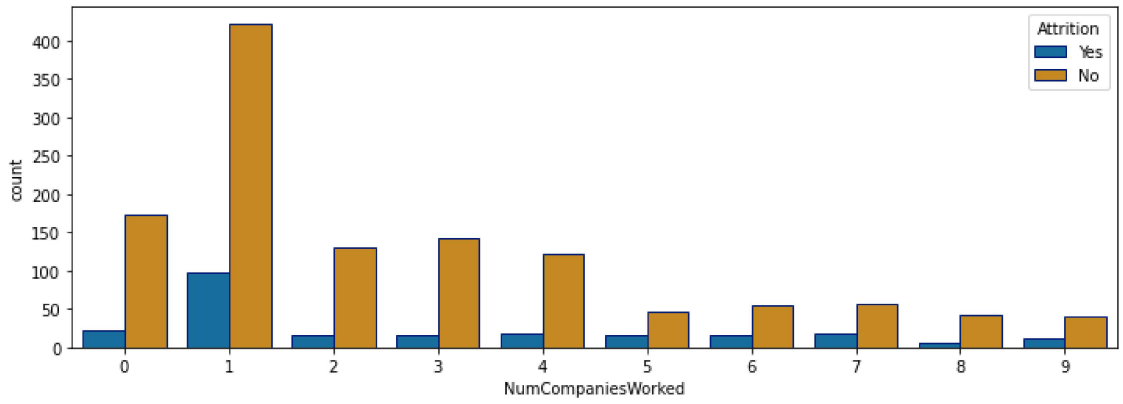
In [19]:
```
number of employees that left and stayed by MaritalStatus
(12, 4)
lt.subplots(figsize=fig_dims)
ot(x='MaritalStatus', hue='Attrition', data=df, palette="colorblind", ax=a
```



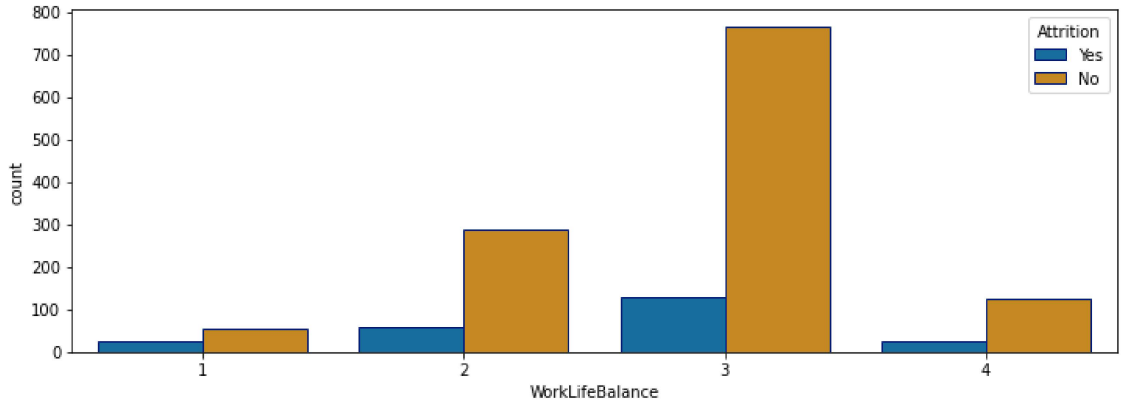
In [20]:
```
number of employees that left and stayed by MonthlyIncome
(12, 4)
lt.subplots(figsize=fig_dims)
ot(x='MonthlyIncome', hue='Attrition', data=df, palette="colorblind", ax=a
```



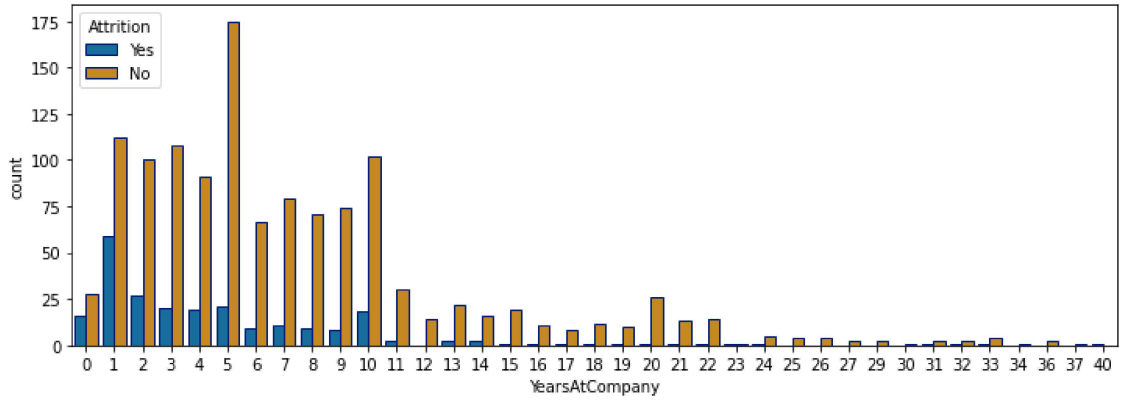
In [21]:
```
r of employees that left and stayed by Number of Companies worked
4)
bplots(figsize=fig_dims)
'NumCompaniesWorked', hue='Attrition', data=df, palette="colorblind", ax=ax
```



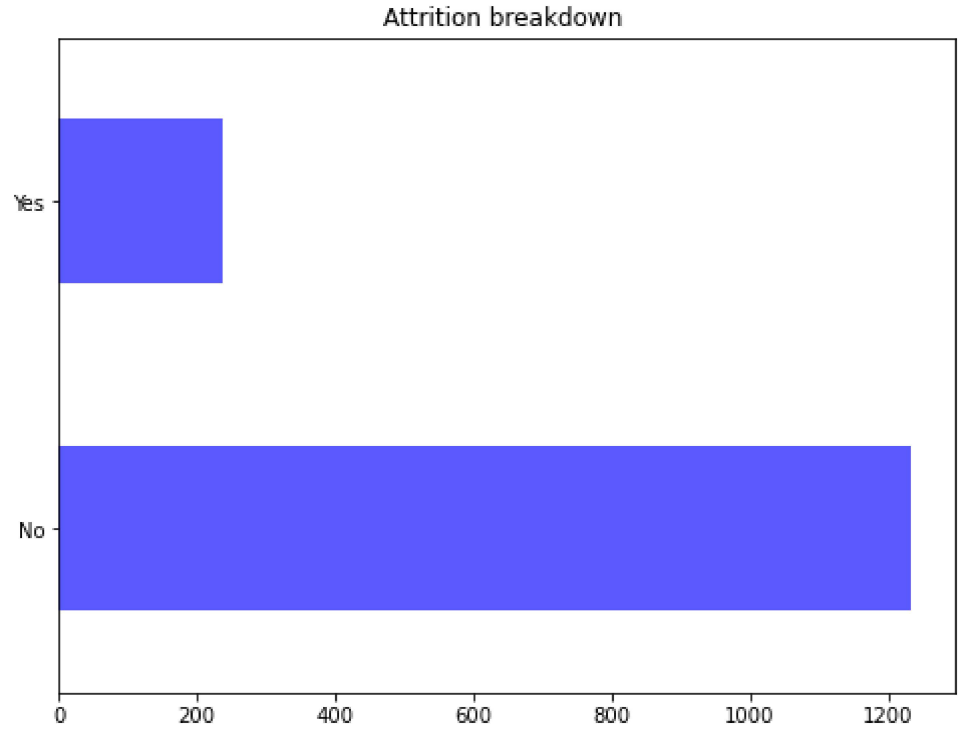
In [22]:
```
mber of employees that left and stayed by Work life balance
2, 4)
.subplots(figsize=fig_dims)
(x='WorkLifeBalance', hue='Attrition', data=df, palette="colorblind", ax=ax
```



In [23]:
```
umber of employees that left and stayed by Years worked at company
12, 4)
t.subplots(figsize=fig_dims)
t(x='YearsAtCompany', hue='Attrition', data=df, palette="colorblind", ax=a
```

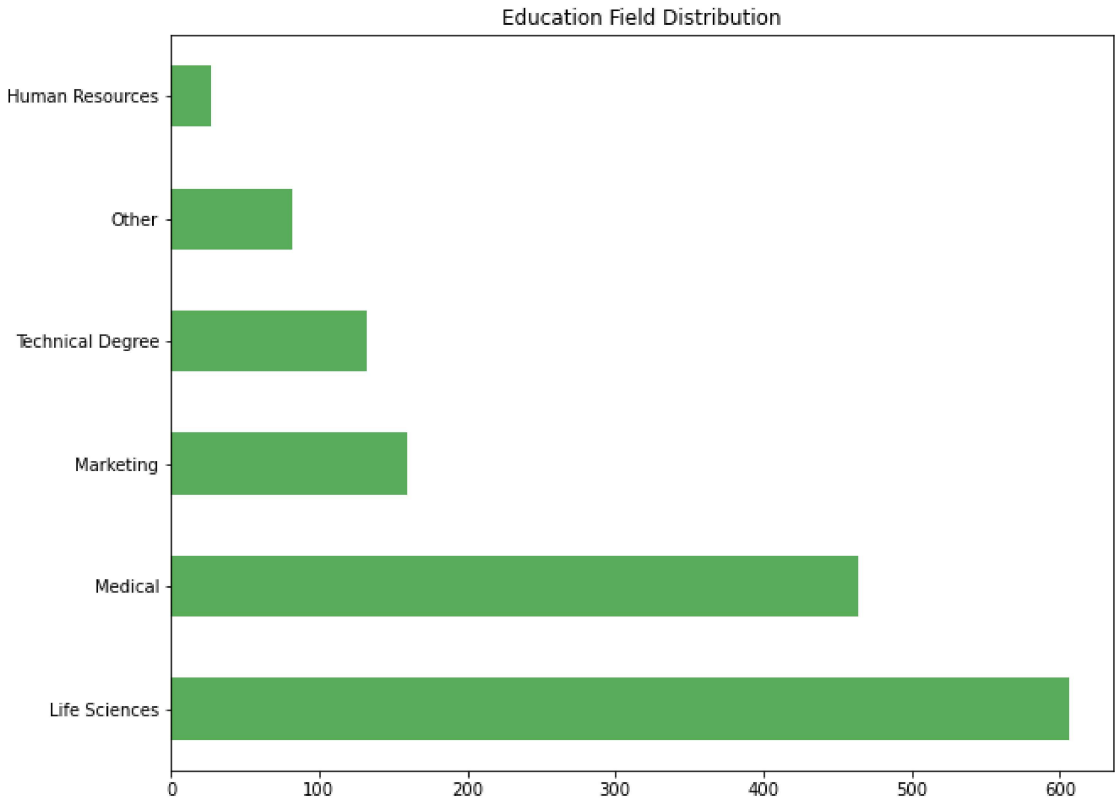


In [24]:
```
# Explore Data for left Employees Breakdown
plt.figure(figsize=(8,6))
df.Attrition.value_counts().plot(kind='barh',color='blue',alpha=.65)
plt.title("Attrition breakdown ")
plt.show()
```

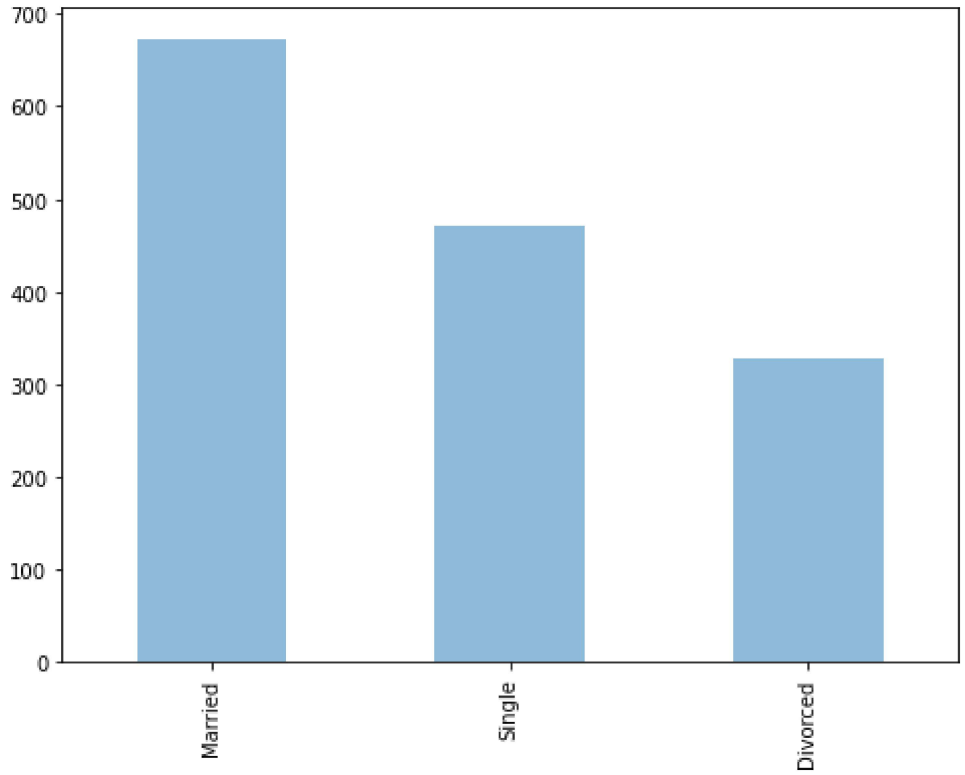


In [25]:

```
# Explore Data for Education Field distribution
plt.figure(figsize=(10,8))
df.EducationField.value_counts().plot(kind='barh',color='g',alpha=.65)
plt.title("Education Field Distribution")
plt.show()
```



In [26]:
```
# Explore Data for Marital Status
plt.figure(figsize=(8,6))
df.MaritalStatus.value_counts().plot(kind='bar',alpha=.5)
plt.show()
```



In [27]:
```
df.describe()
```

Out[27]:

	Age	DistanceFromHome	Education	EnvironmentSatisfaction	JobSatisfacti
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0000
mean	36.923810	9.192517	2.912925	2.721769	2.7285
std	9.135373	8.106864	1.024165	1.093082	1.1028
min	18.000000	1.000000	1.000000	1.000000	1.0000
25%	30.000000	2.000000	2.000000	2.000000	2.0000
50%	36.000000	7.000000	3.000000	3.000000	3.0000
75%	43.000000	14.000000	4.000000	4.000000	4.0000
max	60.000000	29.000000	5.000000	4.000000	4.0000

In [28]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   Age                      1470 non-null   int64 
 1   Attrition                1470 non-null   object
 2   Department               1470 non-null   object
 3   DistanceFromHome         1470 non-null   int64 
 4   Education                1470 non-null   int64 
 5   EducationField           1470 non-null   object
 6   EnvironmentSatisfaction  1470 non-null   int64 
 7   JobSatisfaction          1470 non-null   int64 
 8   MaritalStatus            1470 non-null   object
 9   MonthlyIncome            1470 non-null   int64 
 10  NumCompaniesWorked       1470 non-null   int64 
 11  WorkLifeBalance          1470 non-null   int64 
 12  YearsAtCompany           1470 non-null   int64 
dtypes: int64(9), object(4)
memory usage: 149.4+ KB
```

In [29]:
```
df.columns
```

Out[29]:
```
Index(['Age', 'Attrition', 'Department', 'DistanceFromHome', 'Education',
       'EducationField', 'EnvironmentSatisfaction', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked',
       'WorkLifeBalance', 'YearsAtCompany'],
      dtype='object')
```

In [30]:
```
df.std()
```

```
C:\TEMP\ipykernel_23448\3390915376.py:1: FutureWarning: Dropping of nuisa
nce columns in DataFrame reductions (with 'numeric_only=None') is depreca
ted; in a future version this will raise TypeError.  Select only valid co
lumns before calling the reduction.
  df.std()
```

Out[30]:
```
Age                          9.135373
DistanceFromHome             8.106864
Education                    1.024165
EnvironmentSatisfaction      1.093082
JobSatisfaction              1.102846
MonthlyIncome             4707.956783
NumCompaniesWorked           2.498009
WorkLifeBalance              0.706476
YearsAtCompany               6.126525
dtype: float64
```

In [31]:
```
df['Attrition'].value_counts()
```

Out[31]:
```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

In [32]:
```
df['Attrition'].dtypes
```

Out[32]:
```
dtype('O')
```

In [33]:
```
df['Attrition'].replace('Yes',1, inplace=True)
df['Attrition'].replace('No',0, inplace=True)
```

In [34]:
```
df.head(10)
```

Out[34]:

	Age	Attrition	Department	DistanceFromHome	Education	EducationField	EnvironmentS
0	41	1	Sales	1	2	Life Sciences	
1	49	0	Research & Development	8	1	Life Sciences	
2	37	1	Research & Development	2	2	Other	
3	33	0	Research & Development	3	4	Life Sciences	
4	27	0	Research & Development	2	1	Medical	
5	32	0	Research & Development	2	2	Life Sciences	
6	59	0	Research & Development	3	3	Medical	
7	30	0	Research & Development	24	1	Life Sciences	
8	38	0	Research & Development	23	3	Life Sciences	
9	36	0	Research & Development	27	3	Medical	

In [35]:
```
# Building Up a Logistic Regression Model
X = df.drop(['Attrition'],axis=1)
X.head()
Y = df['Attrition']
Y.head()
```

Out[35]:
```
0    1
1    0
2    1
3    0
4    0
Name: Attrition, dtype: int64
```

In [36]:
```
df['EducationField'].replace('Life Sciences',1, inplace=True)
df['EducationField'].replace('Medical',2, inplace=True)
df['EducationField'].replace('Marketing', 3, inplace=True)
df['EducationField'].replace('Other',4, inplace=True)
df['EducationField'].replace('Technical Degree',5, inplace=True)
df['EducationField'].replace('Human Resources', 6, inplace=True)
```

In [37]:
```
df['EducationField'].value_counts()
```

Out[37]:
```
1    606
2    464
3    159
5    132
4     82
6     27
Name: EducationField, dtype: int64
```

In [38]:
```
df['Department'].value_counts()
```

Out[38]:
```
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64
```

In [39]:
```
df['Department'].replace('Research & Development',1, inplace=True)
df['Department'].replace('Sales',2, inplace=True)
df['Department'].replace('Human Resources', 3, inplace=True)
```

In [40]:
```
df['Department'].value_counts()
```

Out[40]:
```
1    961
2    446
3     63
Name: Department, dtype: int64
```

In [41]:
```
df['MaritalStatus'].value_counts()
```

Out[41]:
```
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
```

In [42]:
```
df['MaritalStatus'].replace('Married',1, inplace=True)
df['MaritalStatus'].replace('Single',2, inplace=True)
df['MaritalStatus'].replace('Divorced',3, inplace=True)
```

In [43]:
```
df['MaritalStatus'].value_counts()
```

Out[43]:
```
1    673
2    470
3    327
Name: MaritalStatus, dtype: int64
```

In [44]:
```
x=df.select_dtypes(include=['int64'])
x.dtypes
```

Out[44]:
```
Age                        int64
Attrition                  int64
Department                 int64
DistanceFromHome           int64
Education                  int64
EducationField             int64
EnvironmentSatisfaction    int64
JobSatisfaction            int64
MaritalStatus              int64
MonthlyIncome              int64
NumCompaniesWorked         int64
WorkLifeBalance            int64
YearsAtCompany             int64
dtype: object
```

In [45]:
```
x.columns
```

Out[45]:
```
Index(['Age', 'Attrition', 'Department', 'DistanceFromHome', 'Education',
       'EducationField', 'EnvironmentSatisfaction', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked',
       'WorkLifeBalance', 'YearsAtCompany'],
      dtype='object')
```

In [46]:
```
y=df['Attrition']
```

In [47]:
```
y.head()
```

Out[47]:
```
0    1
1    0
2    1
3    0
4    0
Name: Attrition, dtype: int64
```

In [48]:
```
y, x = dmatrices('Attrition ~ Age + Department + \
                  DistanceFromHome + Education + EducationField + YearsAtC
                  df, return_type="dataframe")
print (x.columns)
```

```
Index(['Intercept', 'Age', 'Department', 'DistanceFromHome', 'Education',
       'EducationField', 'YearsAtCompany'],
      dtype='object')
```

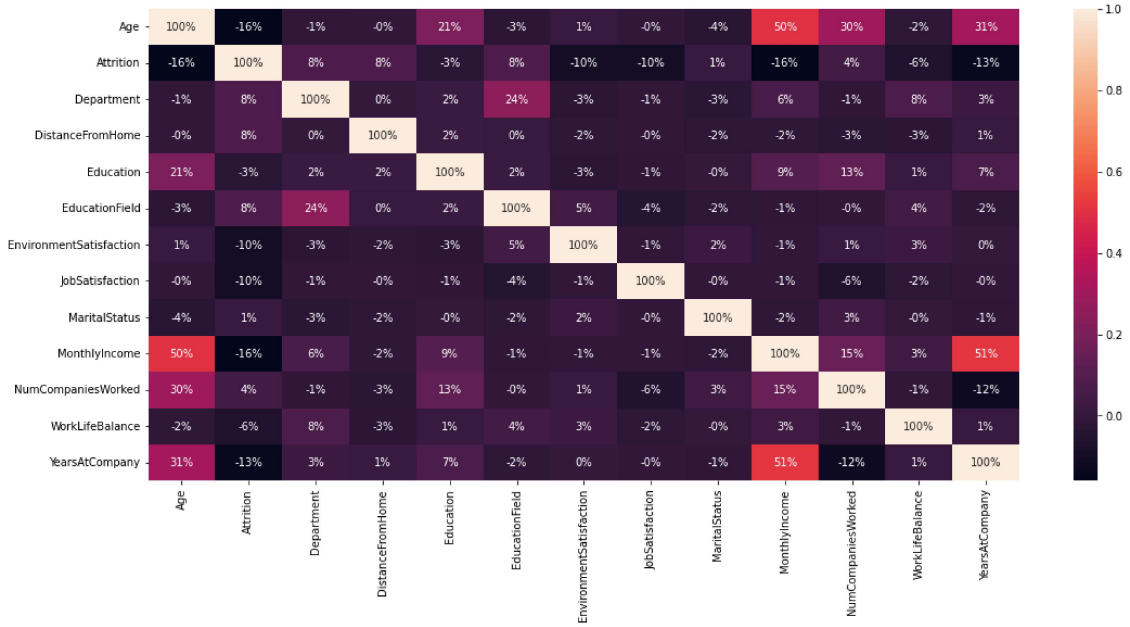
In [49]:
```
# Correlation of the columns
df.corr()
```

Out[49]:

	Age	Attrition	Department	DistanceFromHome	Education	E
Age	1.000000	-0.159205	-0.007652	-0.001686	0.208034	
Attrition	-0.159205	1.000000	0.077351	0.077924	-0.031373	
Department	-0.007652	0.077351	1.000000	0.002196	0.019636	
DistanceFromHome	-0.001686	0.077924	0.002196	1.000000	0.021042	
Education	0.208034	-0.031373	0.019636	0.021042	1.000000	
EducationField	-0.028312	0.077232	0.243641	0.004815	0.018328	
EnvironmentSatisfaction	0.010146	-0.103369	-0.026110	-0.016075	-0.027128	
JobSatisfaction	-0.004892	-0.103481	-0.006231	-0.003669	-0.011296	
MaritalStatus	-0.035466	0.011195	-0.030818	-0.021916	-0.000107	
MonthlyIncome	0.497855	-0.159840	0.056573	-0.017014	0.094961	
NumCompaniesWorked	0.299635	0.043494	-0.011261	-0.029251	0.126317	
WorkLifeBalance	-0.021490	-0.063939	0.075507	-0.026556	0.009819	
YearsAtCompany	0.311309	-0.134392	0.029752	0.009508	0.069114	

In [50]:
```
# visualize the correlation
plt.figure(figsize=(18,8))
sns.heatmap(df.corr(), annot=True, fmt='.0%')
```

Out[50]: <AxesSubplot:>



In [51]:
```python
y = np.ravel(y)
```

In [52]:
```python
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model = model.fit(x, y)

# check the accuracy on the training set
model.score(x, y)
```

Out[52]:
```
0.8408163265306122
```

In [53]:
```python
y.mean()
```

Out[53]:
```
0.16122448979591836
```

In [54]:
```python
X_train,X_test,y_train,y_test=sklearn.model_selection.train_test_split(x,y
model2=LogisticRegression()
model2.fit(X_train, y_train)
```

Out[54]:
```
LogisticRegression()
```

In [55]:
```python
predicted= model2.predict(X_test)
print (predicted)
```

```
[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. 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. 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. 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.]
```

In [56]:
```
probs = model2.predict_proba(X_test)
print (probs)
```

```
 [0.81866469 0.18133531]
 [0.74504131 0.25495869]
 [0.86779495 0.13220505]
 [0.87071139 0.12928861]
 [0.81717471 0.18282529]
 [0.71840764 0.28159236]
 [0.59825898 0.40174102]
 [0.83951549 0.16048451]
 [0.88351325 0.11648675]
 [0.7435258  0.2564742 ]
 [0.76631615 0.23368385]
 [0.98033036 0.01966964]
 [0.91857466 0.08142534]
 [0.7743284  0.2256716 ]
 [0.92514814 0.07485186]
 [0.88123383 0.11876617]
 [0.74587179 0.25412821]
 [0.90478361 0.09521639]
 [0.78685526 0.21314474]
 [0.8114777  0.1885223 ]
```

In [57]:
```
from sklearn import metrics

print (metrics.accuracy_score(y_test, predicted))
print (metrics.roc_auc_score(y_test, probs[:, 1]))
```

```
0.8435374149659864
0.6502502887947632
```

In [58]:
```
print (metrics.confusion_matrix(y_test, predicted))
print (metrics.classification_report(y_test, predicted))
```

```
[[371   0]
 [ 69   1]]
              precision    recall  f1-score   support

         0.0       0.84      1.00      0.91       371
         1.0       1.00      0.01      0.03        70

    accuracy                           0.84       441
   macro avg       0.92      0.51      0.47       441
weighted avg       0.87      0.84      0.77       441
```

In [59]:
```
print (X_train)
```

```
      Intercept   Age  Department  DistanceFromHome  Education  \
338         1.0  30.0         2.0               5.0        3.0
363         1.0  33.0         2.0               5.0        3.0
759         1.0  45.0         3.0              24.0        4.0
793         1.0  28.0         1.0              15.0        2.0
581         1.0  30.0         1.0               1.0        3.0
...         ...   ...         ...               ...        ...
763         1.0  34.0         2.0              10.0        4.0
835         1.0  35.0         3.0               8.0        4.0
1216        1.0  43.0         2.0               2.0        3.0
559         1.0  38.0         1.0               2.0        5.0
684         1.0  40.0         2.0              10.0        4.0

      EducationField  YearsAtCompany
338              3.0            10.0
363              3.0             1.0
759              2.0             6.0
793              1.0             4.0
581              1.0             2.0
...              ...             ...
763              1.0             1.0
835              5.0             5.0
1216             2.0            10.0
559              2.0             1.0
684              3.0             1.0

[1029 rows x 7 columns]
```

In [60]:
```
importance = np.abs(model.coef_[0])
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
```

```
Feature: 0, Score: 0.00060
Feature: 1, Score: 0.04095
Feature: 2, Score: 0.29101
Feature: 3, Score: 0.02636
Feature: 4, Score: 0.01227
Feature: 5, Score: 0.10509
Feature: 6, Score: 0.06505
```

In [61]:
```python
from sklearn.linear_model import LogisticRegression
import numpy as np

# Define your input features and target variable
X = [[0, 1], [1, 0], [1, 1], [0, 0]]
y = [0, 1, 1, 0]

# Initialize and fit the logistic regression model
clf = LogisticRegression(random_state=0).fit(X, y)

# Obtain the coefficients and feature importance
coefficients = clf.coef_[0]
importance = np.abs(coefficients)

# Print the results
print("Coefficients:", coefficients)
print("Feature Importance:", importance)
```

```
Coefficients: [ 8.02109869e-01 -4.71500832e-06]
Feature Importance: [8.02109869e-01 4.71500832e-06]
```

In [62]:
```python
# Add random values to KK according to the parameters mentioned above to c
kk=[[1.0, 23.0, 1.0, 500.0, 3.0, 24.0, 1.0]]
print(model.predict_proba(kk))
```

```
[[6.25571910e-07 9.99999374e-01]]

C:\Users\Imran\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarni
ng: X does not have valid feature names, but LogisticRegression was fitte
d with feature names
  warnings.warn(
```