In [1]:

```
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
%matplotlib inline
from patsy import dmatrices
import sklearn
import seaborn as sns
```

In [2]:

```
dataframe=pd.read_csv("IBM Attrition Data.csv")
```

In [3]:

```
dataframe.head()
```

Out[3]:

	Age	Attrition	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	JobSatisfaction	MaritalStatus	MonthlyIncome	Nur
0	41	Yes	Sales	1	2	Life Sciences	2	4	Single	5993	
1	49	No	Research & Development	8	1	Life Sciences	3	2	Married	5130	
2	37	Yes	Research & Development	2	2	Other	4	3	Single	2090	
3	33	No	Research & Development	3	4	Life Sciences	4	3	Married	2909	
4	27	No	Research & Development	2	1	Medical	1	2	Married	3468	

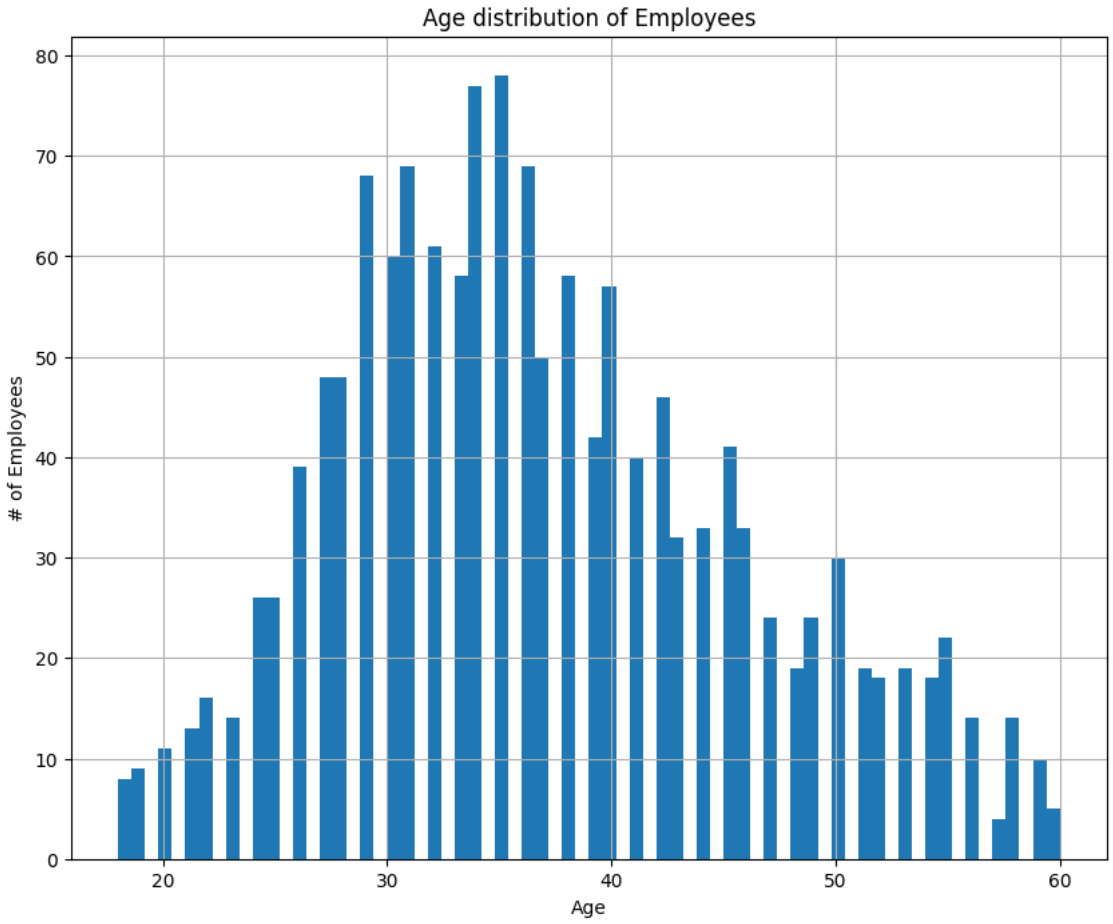
In [4]:

```
names = dataframe.columns.values
print(names)
```

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

In [5]:

```
# histogram for age
plt.figure(figsize=(10,8))
dataframe['Age'].hist(bins=70)
plt.title("Age distribution of Employees")
plt.xlabel("Age")
plt.ylabel("# of Employees")
plt.show()
```



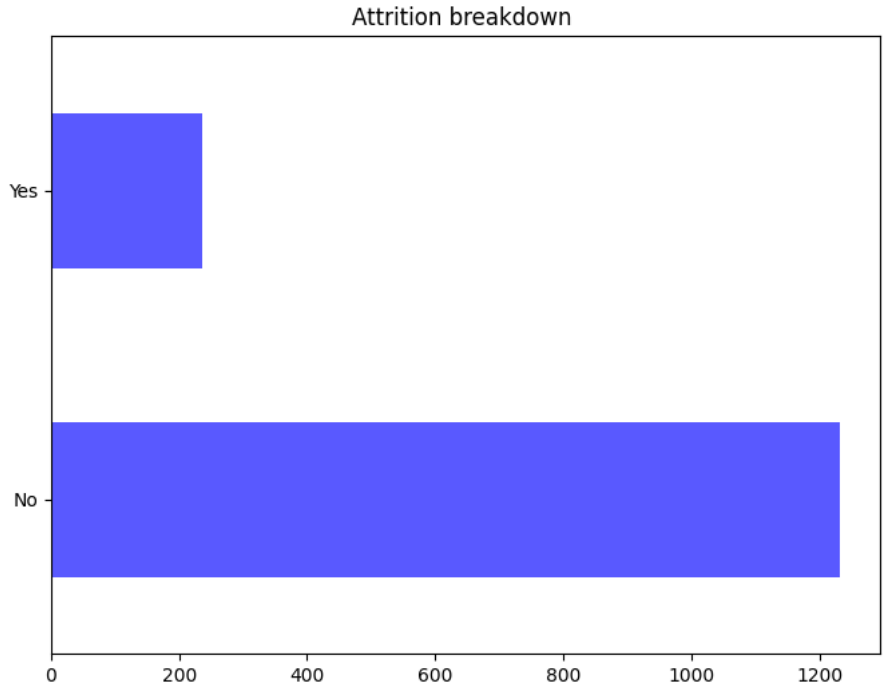
In [6]:

```
# explore data for Attrition by Age
plt.figure(figsize=(14,10))
plt.scatter(dataframe.Attrition,dataframe.Age, alpha=.55)
plt.title("Attrition by Age ")
plt.ylabel("Age")
plt.grid(b=True, which='major',axis='y')
plt.show()
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_25548\519728794.py:6: MatplotlibDeprecationWarning: The 'b' parameter of grid() ha
s been renamed 'visible' since Matplotlib 3.5; support for the old name will be dropped two minor releases later.
  plt.grid(b=True, which='major',axis='y')
```

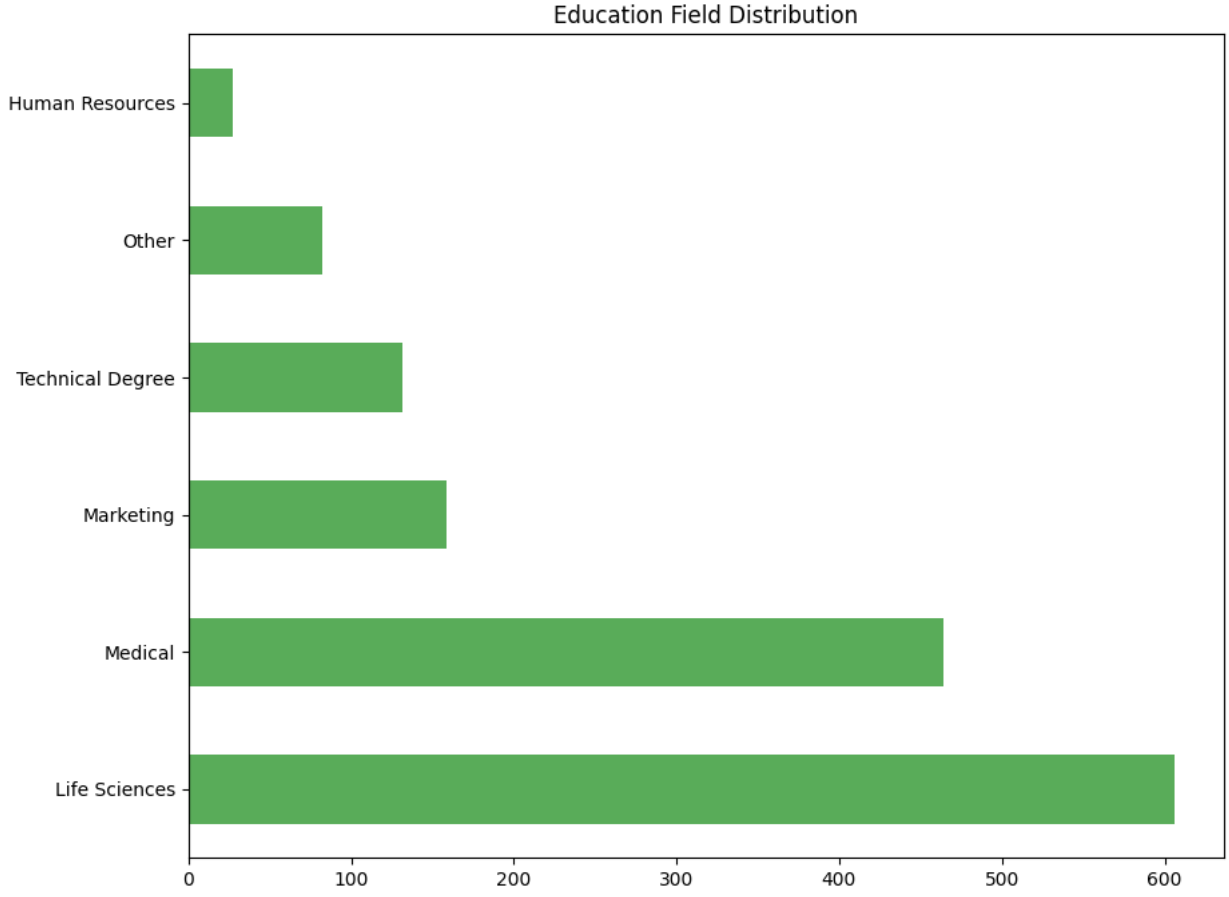
In [7]:

```
# explore data for Left employees breakdown
plt.figure(figsize=(8,6))
dataframe.Attrition.value_counts().plot(kind='barh',color='blue',alpha=.65)
plt.title("Attrition breakdown ")
plt.show()
```



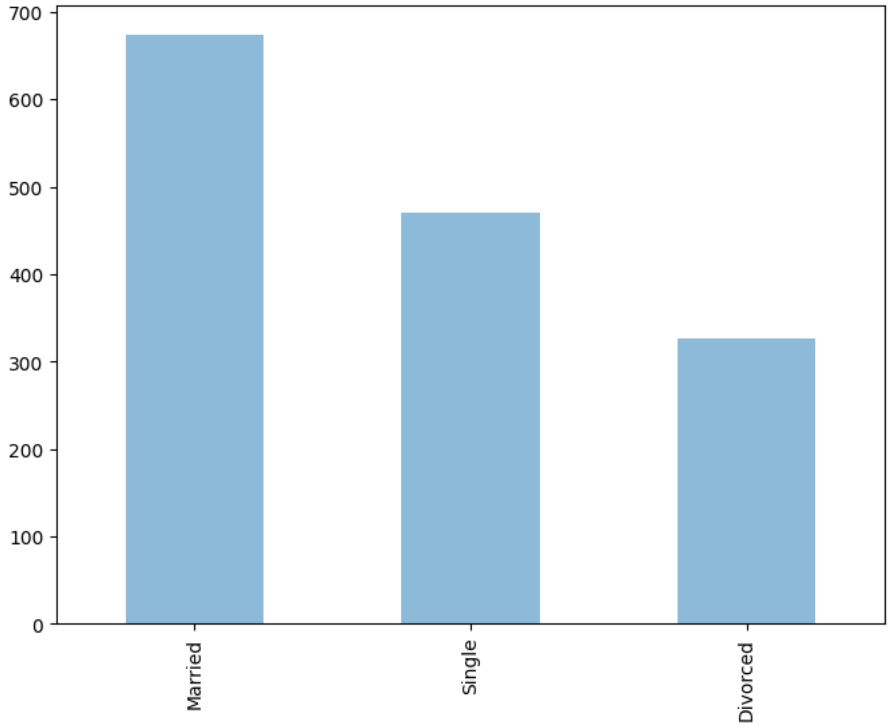
In [8]:

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



In [9]:

```
# explore data for Marital Status
plt.figure(figsize=(8,6))
dataframe.MaritalStatus.value_counts().plot(kind='bar',alpha=.5)
plt.show()
```



In [10]:

```
dataframe.describe()
```

Out[10]:

	Age	DistanceFromHome	Education	EnvironmentSatisfaction	JobSatisfaction	MonthlyIncome	NumCompaniesWorked	WorkLifeBalance	Y
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	
mean	36.923810	9.192517	2.912925	2.721769	2.728571	6502.931293	2.693197	2.761224	
std	9.135373	8.106864	1.024165	1.093082	1.102846	4707.956783	2.498009	0.706476	
min	18.000000	1.000000	1.000000	1.000000	1.000000	1009.000000	0.000000	1.000000	
25%	30.000000	2.000000	2.000000	2.000000	2.000000	2911.000000	1.000000	2.000000	
50%	36.000000	7.000000	3.000000	3.000000	3.000000	4919.000000	2.000000	3.000000	
75%	43.000000	14.000000	4.000000	4.000000	4.000000	8379.000000	4.000000	3.000000	
max	60.000000	29.000000	5.000000	4.000000	4.000000	19999.000000	9.000000	4.000000	

In [11]:

```
dataframe.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 [12]:

```
dataframe.columns
```

Out[12]:

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

In [13]:

```
dataframe.std()
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_25548\3401367348.py:1: FutureWarning: The default value of numeric_only in DataFra
me.std is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None' is depreca
ted. Select only valid columns or specify the value of numeric_only to silence this warning.
  dataframe.std()
```

Out[13]:

```
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 [14]:

```
dataframe['Attrition'].value_counts()
```

Out[14]:

```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

In [15]:

```
dataframe['Attrition'].dtypes
```

Out[15]:

```
dtype('O')
```

In [16]:

```
dataframe['Attrition'].replace('Yes',1, inplace=True)
dataframe['Attrition'].replace('No',0, inplace=True)
```

In [17]:

```
dataframe.head(10)
```

Out[17]:

	Age	Attrition	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	JobSatisfaction	MaritalStatus	MonthlyIncome	Nur
0	41	1	Sales	1	2	Life Sciences	2	4	Single	5993	
1	49	0	Research & Development	8	1	Life Sciences	3	2	Married	5130	
2	37	1	Research & Development	2	2	Other	4	3	Single	2090	
3	33	0	Research & Development	3	4	Life Sciences	4	3	Married	2909	
4	27	0	Research & Development	2	1	Medical	1	2	Married	3468	
5	32	0	Research & Development	2	2	Life Sciences	4	4	Single	3068	
6	59	0	Research & Development	3	3	Medical	3	1	Married	2670	
7	30	0	Research & Development	24	1	Life Sciences	4	3	Divorced	2693	
8	38	0	Research & Development	23	3	Life Sciences	4	3	Single	9526	
9	36	0	Research & Development	27	3	Medical	3	3	Married	5237	

In [18]:

```
# building up a logistic regression model
X = dataframe.drop(['Attrition'],axis=1)
X.head()
Y = dataframe['Attrition']
Y.head()
```

Out[18]:

```
0    1
1    0
2    1
3    0
4    0
Name: Attrition, dtype: int64
```

In [19]:

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

In [20]:

```
dataframe['EducationField'].value_counts()
```

Out[20]:

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

In [21]:

```
dataframe['Department'].value_counts()
```

Out[21]:

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

In [22]:

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

In [23]:

```
dataframe['Department'].value_counts()
```

Out[23]:

```
1    961
2    446
3     63
Name: Department, dtype: int64
```

In [24]:

```
dataframe['MaritalStatus'].value_counts()
```

Out[24]:

```
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
```

In [25]:

```
dataframe['MaritalStatus'].replace('Married',1, inplace=True)
dataframe['MaritalStatus'].replace('Single',2, inplace=True)
dataframe['MaritalStatus'].replace('Divorced',3, inplace=True)
```

In [26]:

```
dataframe['MaritalStatus'].value_counts()
```

Out[26]:

```
1    673
2    470
3    327
Name: MaritalStatus, dtype: int64
```

In [27]:

```
x=dataframe.select_dtypes(include=['int64'])
x.dtypes
```

Out[27]:

```
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 [28]:

```
x.columns
```

Out[28]:

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

In [29]:

```
y=dataframe['Attrition']
```

In [30]:

```
y.head()
```

Out[30]:

```
0    1
1    0
2    1
3    0
4    0
Name: Attrition, dtype: int64
```

In [31]:

```
y, x = dmatrices('Attrition ~ Age + Department + \
                  DistanceFromHome + Education + EducationField + YearsAtCompany',
                  dataframe, return_type="dataframe")
print (x.columns)
```

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

In [32]:

```
y = np.ravel(y)
```

In [33]:

```
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[33]:

```
0.8408163265306122
```

In [34]:

```
y.mean()
```

Out[34]:

```
0.16122448979591836
```

In [35]:

```
X_train,X_test,y_train,y_test=sklearn.model_selection.train_test_split(x,y, test_size=0.3, random_state=0)
model2=LogisticRegression()
model2.fit(X_train, y_train)
```

Out[35]:

```
▾ LogisticRegression
LogisticRegression()
```

In [36]:

```
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 [37]:

```
probs = model2.predict_proba(X_test)
print (probs)
```

```
[[0.86179629 0.13820371]
 [0.80754595 0.19245405]
 [0.74123952 0.25876048]
 [0.83441331 0.16558669]
 [0.73499943 0.26500057]
 [0.79097749 0.20902251]
 [0.85615202 0.14384798]
 [0.85699674 0.14300326]
 [0.96699053 0.03300947]
 [0.93685203 0.06314797]
 [0.95099268 0.04900732]
 [0.83101544 0.16898456]
 [0.86296552 0.13703448]
 [0.86581192 0.13418808]
 [0.88750595 0.11249405]
 [0.88892618 0.11107382]
 [0.88569719 0.11430281]
 [0.78516588 0.21483412]
 [0.79794491 0.20205509]
```

In [38]:

```
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 [39]:

```
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 [40]:

```
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 [41]:

```
#add random values to KK according to the parameters mentioned above to check the proabily of attrition of the employee
kk=[[1.0, 23.0, 1.0, 500.0, 3.0, 24.0, 1.0]]
print(model.predict_proba(kk))
```

```
[[6.25571959e-07 9.99999374e-01]]

C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have val
id feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

In [43]:

```

```

```
Note: you may need to restart the kernel to use updated packages.

ERROR: Invalid requirement: "'nbconvert[qtpdf]'"
```

In [ ]:

```

```