Assignment: IBM HR Analytics Employee Attrition Modeling

The comments/sections provided are your cues to perform the assignment. You don't need to limit yourself to the number of rows/cells provided. You can add additional rows in each section to add more lines of code.

If at any point in time you need help on solving this assignment, view our demo video to understand the different steps of the code.

Happy coding!

IBM HR Analytics Employee Attrition Modeling.

DESCRIPTION

IBM is an American MNC operating in around 170 countries with major business vertical as computing, software, and hardware. Attrition is a major risk to service-providing organizations where trained and experienced people are the assets of the company. The organization would like to identify the factors which influence the attrition of employees.

Data Dictionary:

```
Age: Age of employee
Attrition: Employee attrition status
Department: Department of work
DistanceFromHome
Education: 1-Below College; 2- College; 3-Bachelor; 4-Master; 5-Doctor;
EducationField
EnvironmentSatisfaction: 1-Low; 2-Medium; 3-High; 4-Very High;
JobSatisfaction: 1-Low; 2-Medium; 3-High; 4-Very High;
MaritalStatus
MonthlyIncome
NumCompaniesWorked: Number of companies worked prior to IBM
WorkLifeBalance: 1-Bad; 2-Good; 3-Better; 4-Best;
YearsAtCompany: Current years of service in IBM
```

Analysis Task:

- Import attrition dataset and import libraries such as pandas, matplotlib.pyplot, numpy, and seaborn.
- Exploratory data analysis
  - Find the age distribution of employees in IBM
  - Explore attrition by age
  - Explore data for Left employees
  - Find out the distribution of employees by the education field
  - Give a bar chart for the number of married and unmarried employees
- Build up a logistic regression model to predict which employees are likely to attrite.

Table of Contents

Import libraries and dataset

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from patsy import dmatrices
import sklearn
import seaborn as sns
```

In [2]:
```
dataframe=pd.read_csv("D:\\NIPUN_SC_REC\\3_Practice_Project\\Course_5_Data Science wi
```

Exploratory data analysis

In [3]:
```
dataframe.head()
```

Out[3]:

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

In [4]:
```
names = dataframe.columns.values
print(names)
```

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

In [5]:
```
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: 126.4+ KB
```

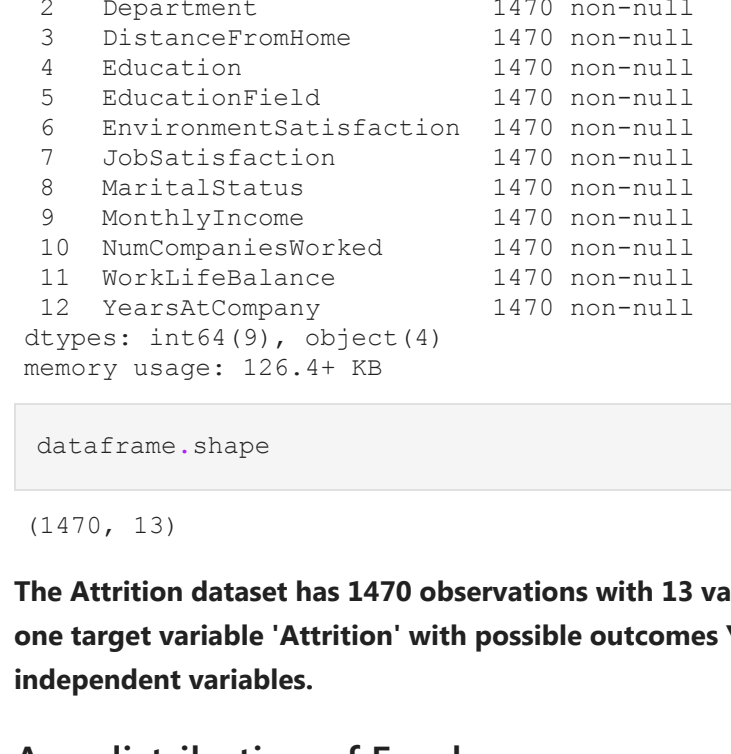
In [6]:
```
dataframe.shape
```

Out[6]: (1470, 13)

The Attrition dataset has 1470 observations with 13 variables. Out of the 13 variables, there exists one target variable 'Attrition' with possible outcomes Yes and No. The other 12 variables are independent variables.

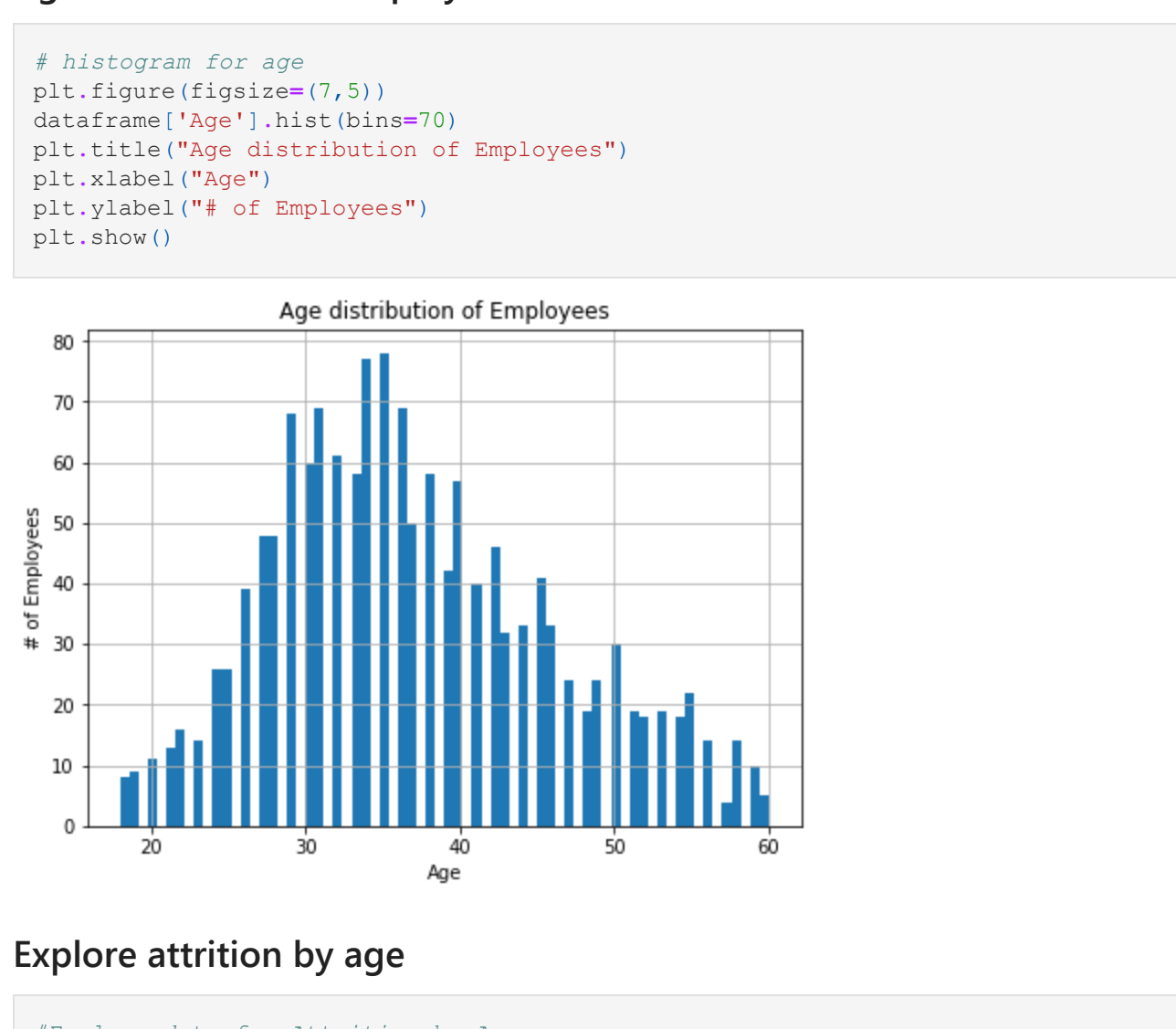
Age distribution of Employees

In [7]:
```
# histogram for age
plt.figure(figsize=(7,5))
dataframe['Age'].hist(bins=70)
plt.title("Age distribution of Employees")
plt.xlabel("Age")
plt.ylabel("# of Employees")
plt.show()
```



Explore attrition by age

In [8]:
```
#Explore data for Attrition by Age
plt.figure(figsize=(14,10))
plt.scatter(dataframe.Attrition,dataframe.Age, alpha=0.1)
plt.title("Attrition by Age ")
plt.ylabel("Age")
plt.grid(b=True, which='major',axis='y')
plt.show()
```



The Scatter plot shows most of the attritions are centered around 30 year of age.

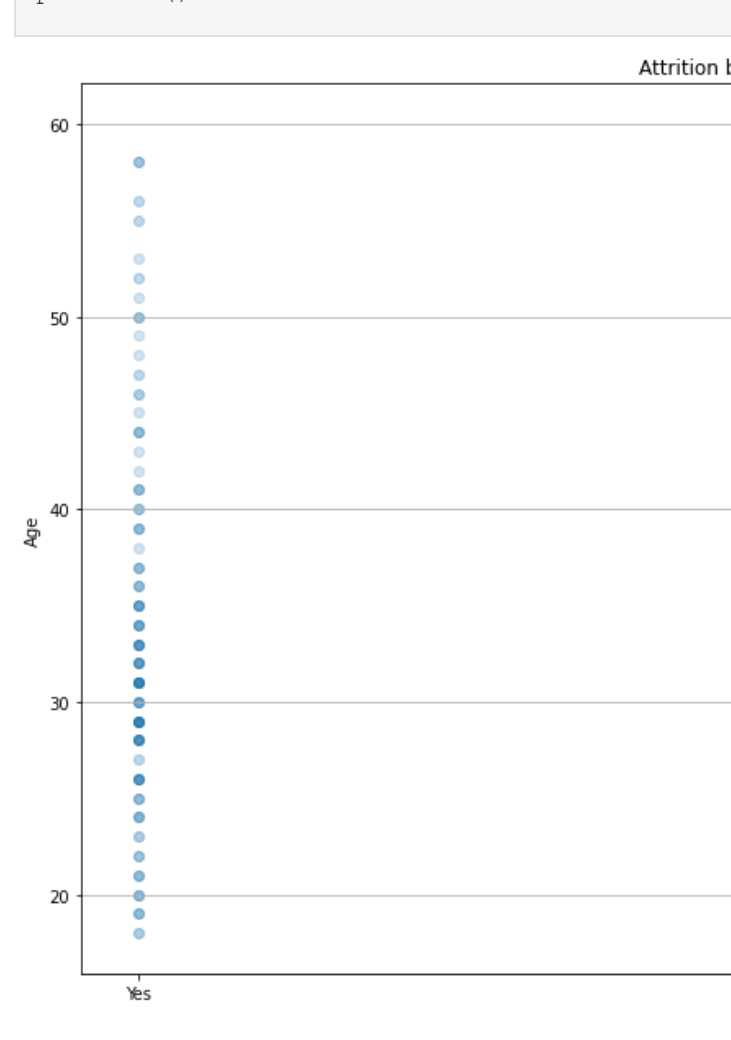
Explore data for Left employees

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

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

The dataset is well organised with no missing values. This is a Binary Classification Problem, so the Distribution of instances among the 2 classes is visualized below.

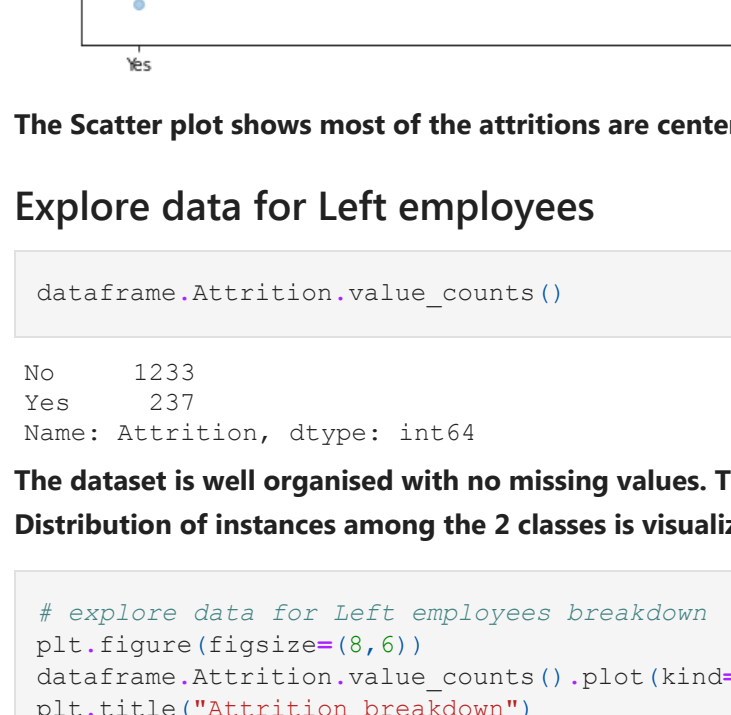
In [10]:
```
# explore data for Left employees breakdown
plt.figure(figsize=(8,6))
dataframe.Attrition.value_counts().plot(kind='bar',color='blue',alpha=.65)
plt.title("Attrition breakdown")
plt.show()
```



The above plot shows the distribution of Attrition. Out of the total of 1470 observations 1233 is No, whereas 237 is Yes. We will treat this imbalance after splitting the data into Training and Test Set.

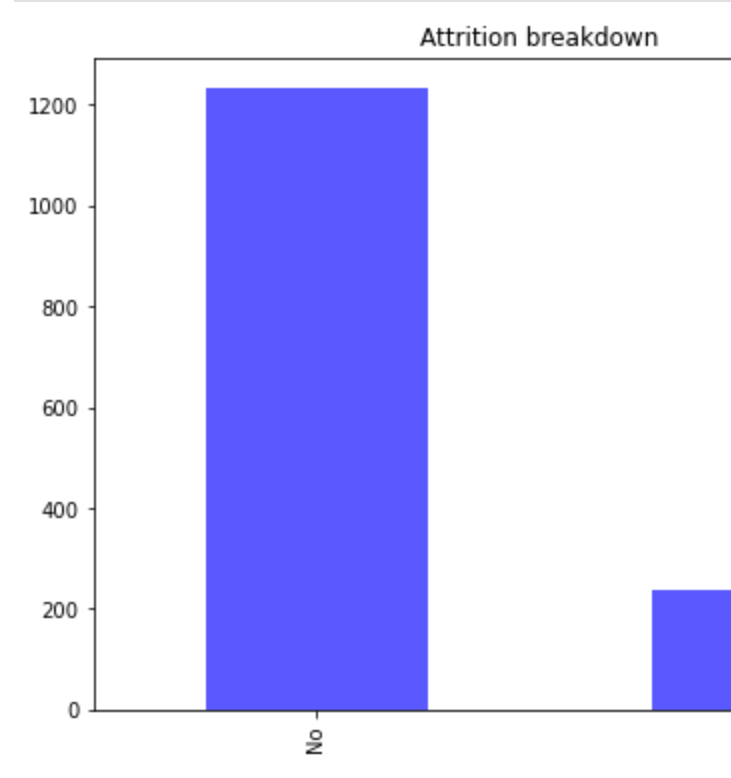
Find out the distribution of employees by the education field

In [11]:
```
# explore data for Education Field distribution
plt.figure(figsize=(7,5))
dataframe.EducationField.value_counts().plot(kind='barh',color='g',alpha=.65)
plt.title("Education Field Distribution")
plt.show()
```



Attrition by Education Field

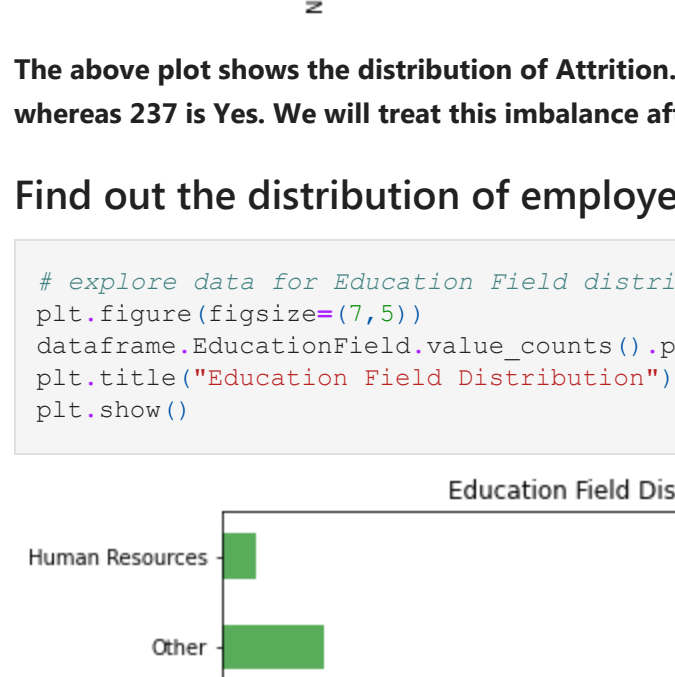
In [12]:
```
pd.crosstab(dataframe["EducationField"],dataframe["Attrition"]).plot(kind="barh",stac
plt.title("Attrition by Education Field")
plt.xlabel("Education Field")
plt.ylabel("Frequency of Attrition")
plt.show()
```



LifeSciences and Medical fields have the highest number of employees and highest number of attrition rate. The percentage of employees who have attritioned against those who have been retained seems to be approximately same in all the education fields.

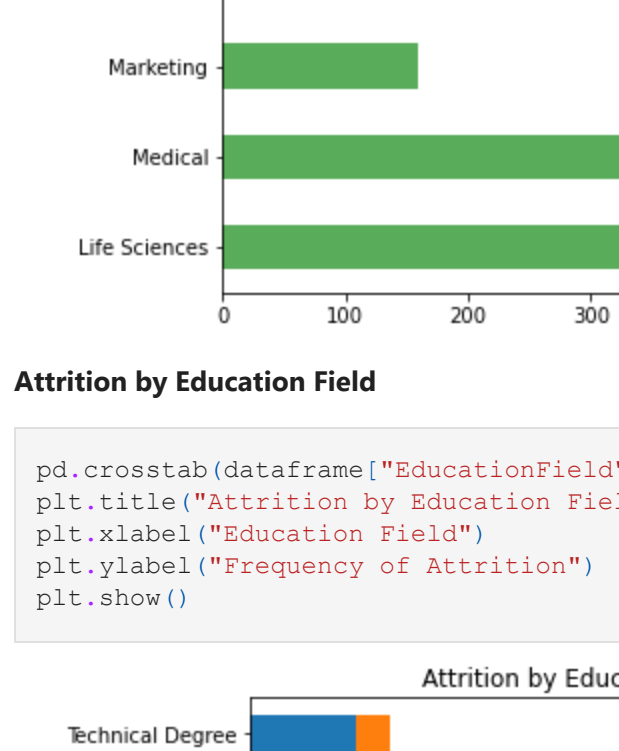
Give a bar chart for the number of married and unmarried employees

In [13]:
```
# explore data for Marital Status
plt.figure(figsize=(7,5))
dataframe.MaritalStatus.value_counts().plot(kind='bar',alpha=.5)
plt.show()
```



Attrition by Marital Status

In [14]:
```
pd.crosstab(dataframe["MaritalStatus"],dataframe["Attrition"]).plot(kind="bar",stacke
plt.title("Attrition by Marital Status")
plt.xlabel("Marital Status")
plt.ylabel("Frequency of Attrition")
plt.show()
```



The highest attrition is well correlated to 'Single' followed by 'Married' & 'Divorced.

Build up a logistic regression model to predict which employees are likely to attrite

In [15]:
```
dataframe.describe()
```

Out[15]:

	Age	DistanceFromHome	Education	EnvironmentSatisfaction	JobSatisfaction	MonthlyIncome
count	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
std	9.135373	8.106864	1.024165	1.093082	1.102846	4707.956783
min	18.000000	1.000000	1.000000	1.000000	1.000000	1009.000000
25%	30.000000	2.000000	2.000000	2.000000	2.000000	2911.000000
50%	36.000000	7.000000	3.000000	3.000000	3.000000	4919.000000
75%	43.000000	14.000000	4.000000	4.000000	4.000000	8379.000000
max	60.000000	29.000000	5.000000	4.000000	4.000000	19999.000000

In [16]:
```
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: 126.4+ KB
```

In [17]:
```
dataframe.columns
```

Out[17]:
```
Index(['Age', 'Attrition', 'Department', 'DistanceFromHome', 'Education',
       'EducationField', 'EnvironmentSatisfaction', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked',
       'WorkLifeBalance', 'YearsAtCompany'],
      dtype='object')
```

In [18]:
```
dataframe.std()
```

Out[18]:
```
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 [19]:
```
dataframe['Attrition'].value_counts()
```

Out[19]:
```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

In [20]:
```
dataframe['Attrition'].dtypes
```

Out[20]: dtype('O')

Data Preprocessing

In [21]:
```
dataframe['Attrition'].replace('Yes',1, inplace=True)
dataframe['Attrition'].replace('No',0, inplace=True)
```

In [22]:
```
dataframe.head(10)
```

Out[22]:

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

In [23]:
```
# building up a logistic regression model
X = dataframe.drop(['Attrition'],axis=1)
X.head()
Y = dataframe['Attrition']
Y.head()
```

Out[23]:
```
0    1
1    0
2    1
3    0
4    0
Name: Attrition, dtype: int64
```

In [24]:
```
dataframe['EducationField'].value_counts()
```

Out[24]:
```
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64
```

In [25]:
```
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 [26]:
```
dataframe['EducationField'].value_counts()
```

Out[26]:
```
1    606
2    464
3    159
5    132
4     82
6     27
Name: EducationField, dtype: int64
```

In [27]:
```
dataframe['Department'].value_counts()
```

Out[27]:
```
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64
```

In [28]:
```
dataframe['Department'].replace('Research & Development',1, inplace=True)
dataframe['Department'].replace('Sales',2, inplace=True)
dataframe['Department'].replace('Human Resources', 3, inplace=True)
```

In [29]:
```
dataframe['Department'].value_counts()
```

Out[29]:
```
1    961
2    446
3     63
Name: Department, dtype: int64
```

In [30]:
```
dataframe['MaritalStatus'].value_counts()
```

Out[30]:
```
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
```

In [31]:
```
dataframe['MaritalStatus'].replace('Married',1, inplace=True)
dataframe['MaritalStatus'].replace('Single',2, inplace=True)
dataframe['MaritalStatus'].replace('Divorced',3, inplace=True)
```

In [32]:
```
dataframe['MaritalStatus'].value_counts()
```

Out[32]:
```
1    673
2    470
3    327
Name: MaritalStatus, dtype: int64
```

In [33]:
```
x=dataframe.select_dtypes(include=['int64'])
x.dtypes
```

Out[33]:
```
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 [34]:
```
x.columns
```

Out[34]:
```
Index(['Age', 'Attrition', 'Department', 'DistanceFromHome', 'Education',
       'EducationField', 'EnvironmentSatisfaction', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked',
       'WorkLifeBalance', 'YearsAtCompany'],
      dtype='object')
```

In [35]:
```
y=dataframe['Attrition']
```

In [36]:
```
y.head()
```

Out[36]:
```
0    1
1    0
2    1
3    0
4    0
Name: Attrition, dtype: int64
```

In [37]:
```
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 [38]:
```
y = np.ravel(y)
```

Train and Test data

In [39]:
```
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[39]: 0.8408163265306122

In [40]:
```
y.mean()
```

Out[40]: 0.16122448979591836

In [41]:
```
X_train,X_test,y_train,y_test=sklearn.model_selection.train_test_split(x,y, test_size=
model2=LogisticRegression()
model2.fit(X_train, y_train)
```

Out[41]: LogisticRegression()

In [42]:
```
model2.score(X_train,y_train)
```

Out[42]: 0.8415937803692906

Training Accuracy of 84.15% is achieved by the model.

In [43]:
```
model2.score(X_test,y_test)
```

Out[43]: 0.8435374149659864

Validation Accuracy of 84.35% is achieved by the model

Build and Evaluate Model

In [44]:
```
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. 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. 0. 0. 0. 0. 0.]
```

In [45]:
```
probs = model2.predict_proba(X_test)
print(probs)
```

```
[[0.86179622 0.13820378]
 [0.80754592 0.19245408]
 [0.74123931 0.25876069]
 [0.83441338 0.16558662]
 [0.73499935 0.26500065]
 [0.79097741 0.20902259]
 [0.85615196 0.14384804]
 [0.85699669 0.14300331]
 [0.96699058 0.03300942]
 [0.9368521  0.0631479 ]
 [0.95099278 0.04900722]
 [0.83101548 0.16898452]
 [0.86296556 0.13703444]
 [0.86581193 0.13418807]
 [0.88750604 0.11249396]
 [0.88892616 0.11107384]
 [0.88569727 0.11430273]
 [0.78516583 0.21483417]
 [0.7979449  0.2020551 ]
 [0.88511304 0.11488696]
 [0.70651589 0.29348411]
 [0.94676693 0.05323307]
 [0.86736254 0.13263746]
 [0.84276452 0.15723548]
 [0.60336837 0.39663163]
 [0.811292   0.188708  ]
 [0.91813731 0.08186269]
 [0.93285522 0.06714478]
 [0.68230751 0.31769249]
 [0.87027139 0.12972861]
 [0.87266386 0.12733614]
 [0.76968737 0.23031263]
 [0.86435774 0.13564226]
 [0.95758881 0.04241119]
 [0.84461487 0.15538513]
 [0.86719349 0.13280651]
 [0.90465982 0.09534018]
 [0.68936423 0.31063577]
 [0.90703618 0.09296382]
 [0.80663474 0.19336526]
 [0.91515727 0.08484273]
 [0.82351271 0.17648729]
 [0.93711517 0.06288483]
 [0.93411325 0.06588675]
 [0.89447655 0.10552345]
 [0.85317747 0.14682253]
 [0.78922387 0.21077613]
 [0.84879888 0.15120112]
 [0.66402447 0.33597553]
 [0.76252291 0.23747709]
 [0.92851112 0.07148888]
 [0.78953697 0.21046303]
 [0.86166592 0.13833408]
 [0.85837884 0.14162116]
 [0.87217674 0.12782326]
 [0.78950896 0.21049104]
 [0.87690794 0.12309206]
 [0.8416545  0.1583455 ]
 [0.72847141 0.27152859]
 [0.83181403 0.16818597]
 [0.90095034 0.09904966]
 [0.71077325 0.28922675]
 [0.92823024 0.07176976]
 [0.84375679 0.15624321]
 [0.79544106 0.20455894]
 [0.86826163 0.13173837]
 [0.91679452 0.08320548]
 [0.84763056 0.15236944]
 [0.89253708 0.10746292]
 [0.62872105 0.37127895]
 [0.93875394 0.06124606]
 [0.72620329 0.27379671]
 [0.85652974 0.14347026]
 [0.84226023 0.15773977]
 [0.77436384 0.22563616]
 [0.71899555 0.28100445]
 [0.9358739  0.0641261 ]
 [0.95710071 0.04289929]
 [0.79185834 0.20814166]
 [0.89370439 0.10629561]
 [0.9138204  0.0861796 ]
 [0.7935459  0.2064541 ]
 [0.77934019 0.22065981]
 [0.79638982 0.20361018]
 [0.83800499 0.16199501]
 [0.71395664 0.28604336]
 [0.97772717 0.02227283]
 [0.94645975 0.05354025]
 [0.88617626 0.11382374]
 [0.79620165 0.20379835]
 [0.61863823 0.38136177]
 [0.8186647  0.1813353 ]
 [0.74504124 0.25495876]
 [0.86779495 0.13220505]
 [0.8707114  0.1292886 ]
 [0.81717467 0.18282533]
 [0.71840765 0.28159235]
 [0.59825882 0.40174118]
 [0.83951552 0.16048448]
 [0.88351326 0.11648674]
 [0.74352571 0.25647429]
 [0.76631618 0.23368382]
 [0.98033038 0.01966962]
 [0.91857469 0.08142531]
 [0.7743284  0.2256716 ]
 [0.92514817 0.07485183]
 [0.88123384 0.11876616]
 [0.7458717  0.2541283 ]
 [0.90478362 0.09521638]
 [0.78685517 0.21314483]
 [0.81147771 0.18852229]
 [0.93472172 0.06527828]
 [0.93836504 0.06163496]
 [0.79411744 0.20588256]
 [0.813729   0.186271  ]
 [0.9161092  0.0838908 ]
 [0.90428346 0.09571654]
 [0.84669421 0.15330579]
 [0.95384554 0.04615446]
 [0.91283692 0.08716308]
 [0.85919605 0.14080395]
 [0.85902497 0.14097503]
 [0.87519517 0.12480483]
 [0.76114666 0.23885334]
 [0.92217687 0.07782313]
 [0.96859411 0.03140589]
 [0.94398222 0.05601778]
 [0.81780289 0.18219711]
 [0.88058705 0.11941295]
 [0.77894286 0.22105714]
 [0.97124466 0.02875534]
 [0.88807662 0.11192338]
 [0.78715262 0.21284738]
 [0.8200148  0.1799852 ]
 [0.94934545 0.05065455]
 [0.95888935 0.04111065]
 [0.73559214 0.26440786]
 [0.93416998 0.06583002]
 [0.73750627 0.26249373]
 [0.82136746 0.17863254]
 [0.821712   0.178288  ]
 [0.89896704 0.10103296]
 [0.78745764 0.21254236]
 [0.8982535  0.1017465 ]
 [0.9143382  0.0856618 ]
 [0.92724744 0.07275256]
 [0.96594967 0.03405033]
 [0.94417368 0.05582632]
 [0.93073083 0.06926917]
 [0.66320576 0.33679424]
 [0.84168649 0.15831351]
```

```
 [0.82636815 0.17363185]
 [0.80616637 0.19383363]
 [0.96157634 0.03842366]
 [0.9351516  0.0648484 ]
 [0.94778627 0.05221373]
 [0.97337827 0.02662173]
 [0.79297051 0.20702949]
 [0.87770196 0.12229804]
 [0.86103661 0.13896339]
 [0.95185479 0.04814521]
 [0.93131494 0.06868506]
 [0.75685354 0.24314646]
 [0.74997796 0.25002204]
 [0.95527867 0.04472133]
 [0.86950723 0.13049277]
 [0.81376767 0.18623233]
 [0.76990616 0.23009384]
 [0.80077907 0.19922093]
 [0.92801122 0.07198878]
 [0.90949247 0.09050753]
 [0.94557628 0.05442372]
 [0.93330468 0.06669532]
 [0.69145897 0.30854103]
 [0.93056885 0.06943115]
 [0.74525418 0.25474582]
 [0.78594339 0.21405661]
 [0.93308602 0.06691398]
 [0.80878291 0.19121709]
 [0.85096457 0.14903543]
 [0.66956273 0.33043727]
 [0.9033838  0.0966162 ]
 [0.91158934 0.08841066]
 [0.87300498 0.12699502]
 [0.92939057 0.07060943]
 [0.66661713 0.33338287]
 [0.89098514 0.10901486]
 [0.86216705 0.13783295]
 [0.78838555 0.21161445]
 [0.53099595 0.46900405]
 [0.73344323 0.26655677]
 [0.7105398  0.2894602 ]
 [0.85530494 0.14469506]
 [0.86960594 0.13039406]
 [0.75470005 0.24529995]
 [0.89823514 0.10176486]
 [0.79247837 0.20752163]
 [0.90643145 0.09356855]
 [0.77650777 0.22349223]
 [0.88387788 0.11612212]
 [0.85404309 0.14595691]
 [0.81899356 0.18100644]
 [0.7444851  0.2555149 ]
 [0.86259433 0.13740567]
 [0.77822623 0.22177377]
 [0.76928322 0.23071678]
 [0.79679225 0.20320775]
 [0.92096591 0.07903409]
 [0.74573765 0.25426235]
 [0.87499229 0.12500771]
 [0.85503289 0.14496711]
 [0.77292149 0.22707851]
 [0.87227272 0.12772728]
 [0.67395346 0.32604654]
 [0.93621665 0.06378335]
 [0.82479478 0.17520522]
 [0.95146047 0.04853953]
 [0.83462449 0.16537551]
 [0.81106481 0.18893519]
 [0.80828254 0.19171746]
 [0.87664154 0.12335846]
 [0.6678699  0.3321301 ]
 [0.59616847 0.40383153]
 [0.98979366 0.01020634]
 [0.70634394 0.29365606]
 [0.91633093 0.08366907]
 [0.92133476 0.07866524]
 [0.71383908 0.28616092]
 [0.62453934 0.37546066]
 [0.76440576 0.23559424]
 [0.95401655 0.04598345]
 [0.88186724 0.11813276]
 [0.86019901 0.13980099]
 [0.92117381 0.07882619]
 [0.88019168 0.11980832]
 [0.80451424 0.19548576]
 [0.80585181 0.19414819]
 [0.91345494 0.08654506]
 [0.72101808 0.27898192]
 [0.94508005 0.05491995]
 [0.90906909 0.09093091]
 [0.73381746 0.26618254]
 [0.98109185 0.01890815]
 [0.85490447 0.14509553]
 [0.89817071 0.10182929]
 [0.82436992 0.17563008]
 [0.83387098 0.16612902]
 [0.88136648 0.11863352]
 [0.88031306 0.11968694]
 [0.87560621 0.12439379]
 [0.81617201 0.18382799]
 [0.88028848 0.11971152]
 [0.61530828 0.38469172]
 [0.88826159 0.11173841]
 [0.89692815 0.10307185]
 [0.85328926 0.14671074]
 [0.98297043 0.01702957]
 [0.77341236 0.22658764]
 [0.62184875 0.37815125]
 [0.82739135 0.17260865]
 [0.84083338 0.15916662]
 [0.84808604 0.15191396]
 [0.84937029 0.15062971]
 [0.75540981 0.24459019]
 [0.86188857 0.13811143]
 [0.9068469  0.0931531 ]
 [0.8466327  0.1533673 ]
 [0.81290902 0.18709098]
 [0.7405357  0.2594643 ]
 [0.87035277 0.12964723]
 [0.83938089 0.16061911]
 [0.86267947 0.13732053]
 [0.66362889 0.33637111]
 [0.90671716 0.09328284]
 [0.87056301 0.12943699]
 [0.92580792 0.07419208]
 [0.84386849 0.15613151]
 [0.8987814  0.1012186 ]
 [0.91239497 0.08760503]
 [0.7986763  0.2013237 ]
 [0.64052596 0.35947404]
 [0.84683942 0.15316058]
 [0.75503191 0.24496809]
 [0.85264542 0.14735458]
 [0.99251433 0.00748567]
 [0.86138218 0.13861782]
 [0.88140142 0.11859858]
 [0.82757085 0.17242915]
 [0.93027076 0.06972924]
 [0.87307441 0.12692559]
 [0.88802398 0.11197602]
 [0.83753315 0.16246685]
 [0.86481029 0.13518971]
 [0.86515551 0.13484449]
 [0.89754792 0.10245208]
 [0.78409085 0.21590915]
 [0.79165111 0.20834889]
 [0.88388121 0.11611879]
 [0.65418947 0.34581053]
 [0.94059381 0.05940619]
 [0.89913437 0.10086563]
 [0.72525609 0.27474391]
 [0.69085507 0.30914493]
 [0.87591631 0.12408369]
 [0.86562248 0.13437752]
 [0.97508878 0.02491122]
 [0.86265129 0.13734871]
 [0.54036059 0.45963941]
 [0.91302278 0.08697722]
 [0.74609581 0.25390419]
 [0.86758146 0.13241854]
 [0.88852755 0.11147245]
 [0.87911718 0.12088282]
 [0.85664962 0.14335038]
 [0.77281957 0.22718043]
 [0.80713984 0.19286016]
 [0.85285505 0.14714495]
 [0.77595544 0.22404456]
 [0.70409845 0.29590155]
 [0.88971656 0.11028344]
 [0.4896307  0.5103693 ]
 [0.92370652 0.07629348]
 [0.75572309 0.24427691]
 [0.67788847 0.32211153]
 [0.91301004 0.08698996]
 [0.94029214 0.05970786]
 [0.8800979  0.1199021 ]
 [0.8859117  0.1140883 ]
 [0.95656829 0.04343171]
 [0.89909425 0.10090575]
 [0.94784649 0.05215351]
 [0.83255771 0.16744229]
 [0.87822141 0.12177859]
 [0.81945797 0.18054203]
 [0.81668783 0.18331217]
 [0.95036172 0.04963828]
 [0.86940138 0.13059862]
 [0.90541601 0.09458399]
 [0.83661189 0.16338811]
 [0.84583646 0.15416354]
 [0.79307976 0.20692024]
 [0.81808123 0.18191877]
 [0.81598408 0.18401592]
 [0.83677553 0.16322447]
 [0.91443034 0.08556966]
 [0.91533088 0.08466912]
 [0.68503441 0.31496559]
 [0.99073759 0.00926241]
 [0.76795004 0.23204996]
 [0.79871922 0.20128078]
 [0.73176829 0.26823171]
 [0.67465182 0.32534818]
 [0.79712455 0.20287545]
 [0.84935619 0.15064381]
 [0.86467083 0.13532917]
 [0.85844153 0.14155847]
 [0.84533264 0.15466736]
 [0.83038803 0.16961197]
 [0.92192756 0.07807244]
 [0.8318972  0.1681028 ]
 [0.97707553 0.02292447]
 [0.90418236 0.09581764]
 [0.92691203 0.07308797]
 [0.84797355 0.15202645]
 [0.7671257  0.2328743 ]
 [0.94895159 0.05104841]
 [0.94782089 0.05217911]
 [0.75181501 0.24818499]
 [0.87880469 0.12119531]
 [0.80770785 0.19229215]
 [0.93864303 0.06135697]
 [0.86008491 0.13991509]
 [0.76056691 0.23943309]
 [0.90892476 0.09107524]
 [0.75402141 0.24597859]
 [0.94271844 0.05728156]
 [0.9182732  0.0817268 ]
 [0.90545128 0.09454872]
 [0.76879454 0.23120546]
 [0.92358139 0.07641861]
 [0.80705877 0.19294123]
 [0.90079529 0.09920471]
 [0.87933333 0.12066667]
 [0.80585688 0.19414312]
 [0.83150053 0.16849947]
 [0.53816763 0.46183237]
 [0.95031826 0.04968174]
 [0.73291897 0.26708103]
 [0.89182871 0.10817129]
 [0.80079854 0.19920146]
 [0.87739806 0.12260194]
 [0.96805232 0.03194768]
 [0.81741597 0.18258403]
 [0.86150092 0.13849908]
 [0.59407547 0.40592453]
 [0.82625734 0.17374266]
 [0.92534979 0.07465021]
 [0.81692203 0.18307797]
 [0.92586785 0.07413215]
 [0.89094605 0.10905395]
 [0.70071749 0.29928251]
 [0.82181894 0.17818106]
 [0.96589942 0.03410058]
 [0.86996989 0.13003011]
 [0.89918552 0.10081448]
 [0.88983802 0.11016198]
 [0.8143246  0.1856754 ]
 [0.8583004  0.1416996 ]
 [0.83878656 0.16121344]
 [0.84056579 0.15943421]
 [0.82661173 0.17338827]
 [0.9407517  0.0592483 ]
 [0.83183534 0.16816466]
 [0.77561353 0.22438647]
 [0.69399537 0.30600463]
 [0.85962129 0.14037871]
 [0.82513942 0.17486058]
 [0.84107714 0.15892286]
 [0.87191016 0.12808984]
 [0.89447655 0.10552345]
 [0.82945346 0.17054654]
 [0.72856787 0.27143213]
 [0.94692822 0.05307178]
 [0.960743   0.039257  ]
 [0.90500199 0.09499801]
 [0.88599991 0.11400009]
 [0.84865643 0.15134357]
 [0.79086971 0.20913029]
 [0.67302962 0.32697038]
 [0.9339021  0.0660979 ]
 [0.65644877 0.34355123]
 [0.74382846 0.25617154]
 [0.94248416 0.05751584]
 [0.78365678 0.21634322]
 [0.90655672 0.09344328]
 [0.81578728 0.18421272]
 [0.89149737 0.10850263]
 [0.85791874 0.14208126]
 [0.67453851 0.32546149]
 [0.93130398 0.06869602]
 [0.89999985 0.10000015]]
```

In [46]:

```
from sklearn import metrics

print(metrics.accuracy_score(y_test, predicted))
```

```
0.8435374149659864
```

In [47]:

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

```
0.6502502887947632
```

In [48]:

```
print(metrics.confusion_matrix(y_test, predicted))
```

```
[[371   0]
 [ 69   1]]
```

Classification Report of the model

In [49]:

```
print (metrics.classification_report(y_test, predicted))
```

```
              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 [50]:

```
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 [51]:

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

```
[[6.25571917e-07 9.99999374e-01]]
```

According to the Performance Analysis, it can be concluded that the Machine Learning Predictive Model has been successful in effectively classifying 84.35% unknown (Validation Set) examples correctly and has shown quite descent statistical figures for different performance metrics.