Code with output

Step 1 : Importing necessary libraries

Code:

```
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
import os
print(os.getcwd())
path=os.getcwd()
os.listdir(path)
```

```
[10]: 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
      import os
      print(os.getcwd())
      path=os.getcwd()
      os.listdir(path)

      /drive/notebooks
[10]: ['IBM  Data.csv',
       'IBM Attrition Data.csv',
       'IBM_Employee_Attrition_Prediction.ipynb',
       'Untitled.ipynb',
       'Untitled1.ipynb',
       'Untitled2.ipynb',
       'Untitled3.ipynb',
       'Untitled4.ipynb',
       'ibm attrition data.csv',
       'ibm data.csv',
       'raj practice (2).ipynb',
       'weather data.csv',
       'Lorenz.ipynb',
       'sqlite.ipynb',
```

Step 2 : Importing dataset

Code:

```
data=pd.read_csv("/drive/notebooks/ibm data.csv")
data.head()
```

```
[14]: data=pd.read_csv("/drive/notebooks/ibm data.csv")
      data.head()
```

[14]:

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

Step 3 : showing column names

Code:

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

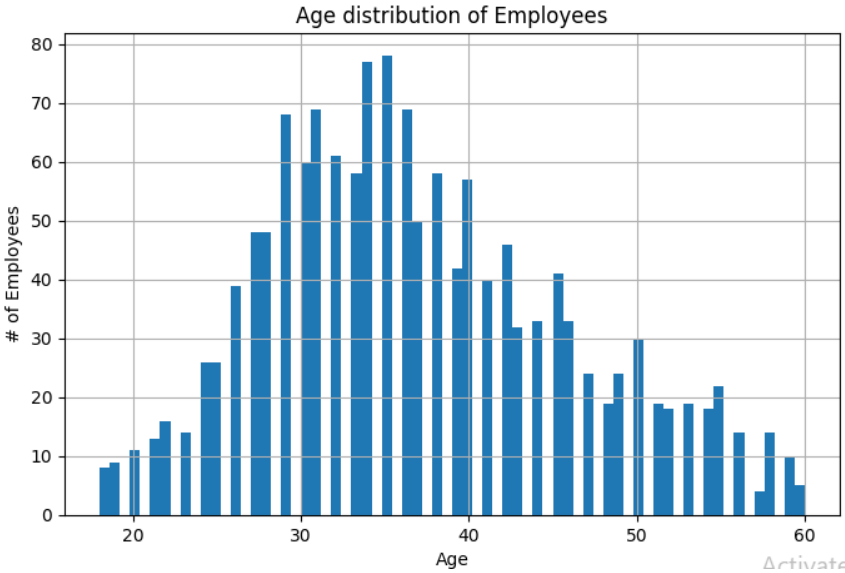
```
[15]: names = dataframe.columns.values
      print(names)

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

Step 4 : creating histogram of age for employees

Code:

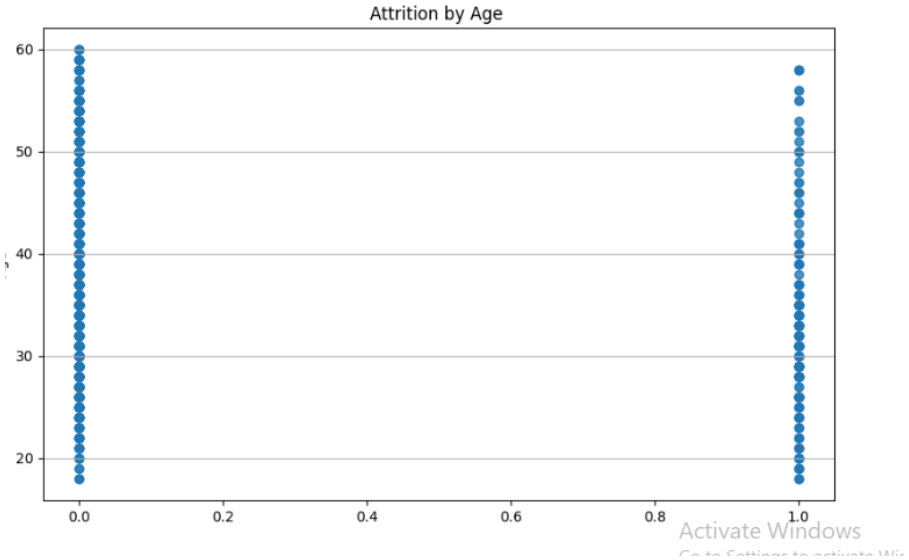
```
# histogram for age
import matplotlib.pyplot as plt
plt.figure(figsize=(8,5))
dataframe['Age'].hist(bins=70)
plt.title("Age distribution of Employees")
plt.xlabel("Age")
plt.ylabel("# of Employees")
plt.show()
```



Step 5 : explore data for attrition by age

Code:

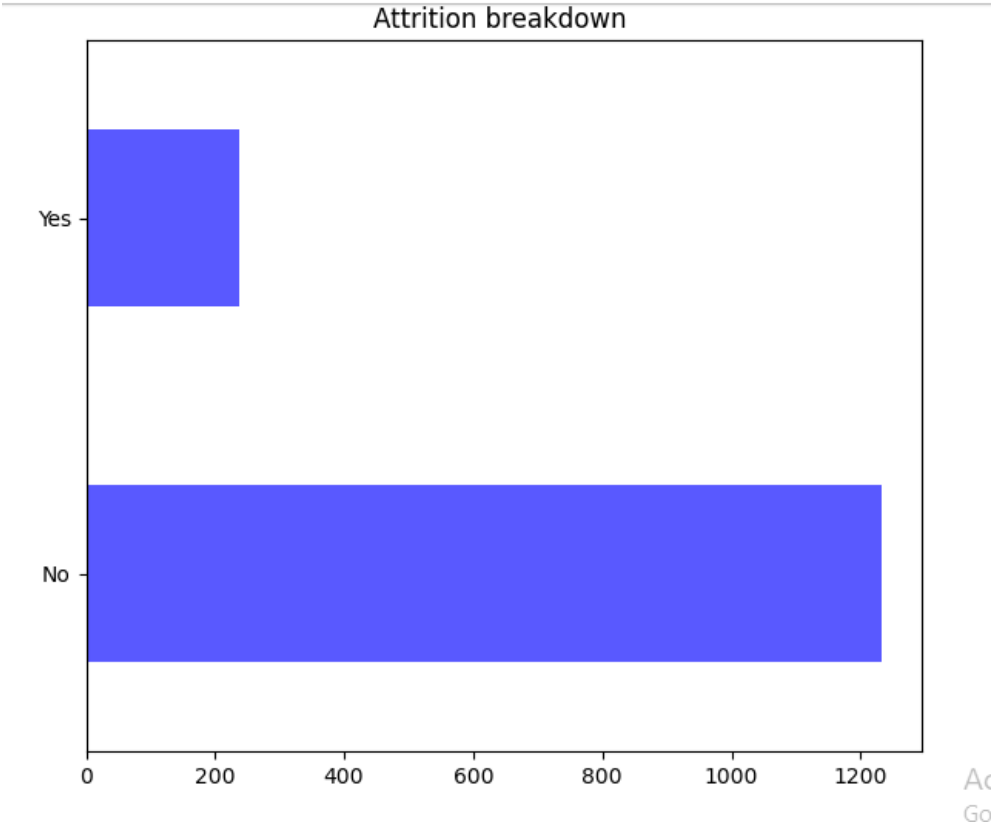
```
# explore data for Attrition by Age
plt.figure(figsize=(10,6))
plt.scatter(dataframe.Attrition,dataframe.Age, alpha=.55)
plt.title("Attrition by Age ")
plt.ylabel("Age")
plt.grid(b=True, which='major',axis='y')
plt.show()
```



Step 6 : display bar graph for data of left employess breakdown

Code:

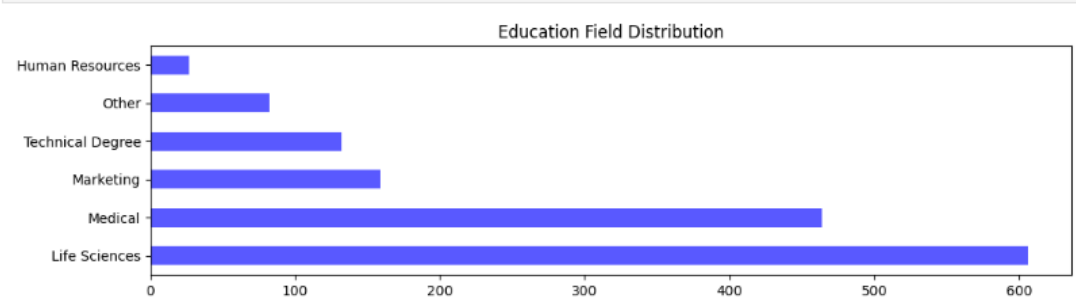
```
# explore data for Left employees breakdown
plt.figure(figsize=(7,6))
dataframe.Attrition.value_counts().plot(kind='barh',color='blue',alpha=.65)
plt.title("Attrition breakdown ")
plt.show()
```



Step 7 : plotting bargraph for education field distribution

Code:

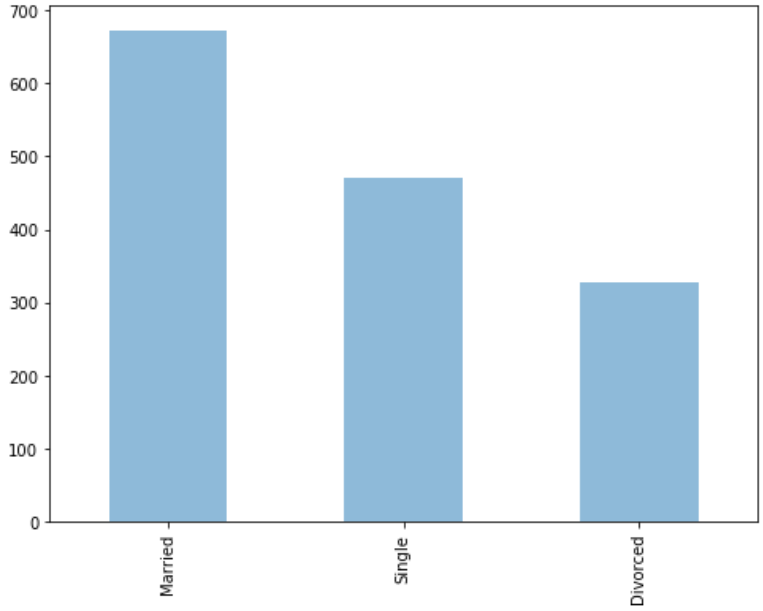
```
# explore data for Education Field distribution
plt.figure(figsize=(12,3))
dataframe.EducationField.value_counts().plot(kind='barh',color='b',alpha=.65)
plt.title("Education Field Distribution")
plt.show()
```



Step 8 : exploring data for martial status

Code:

```
# explore data for Marital Status
plt.figure(figsize=(3,6))
dataframe.MaritalStatus.value_counts().plot(kind='bar',alpha=.5)
plt.show()
```



Step 9 : describing dataframe

Code:

```
dataframe.describe()
```

```
[44]: dataframe.describe()
```

[44]:

	Age	DistanceFromHome	Education	EnvironmentSatisfaction	JobSatisfaction	MonthlyIncome	NumCompaniesWorke
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.00000
mean	36.923810	9.192517	2.912925	2.721769	2.728571	6502.931293	2.69319
std	9.135373	8.106864	1.024165	1.093082	1.102846	4707.956783	2.49800
min	18.000000	1.000000	1.000000	1.000000	1.000000	1009.000000	0.00000
25%	30.000000	2.000000	2.000000	2.000000	2.000000	2911.000000	1.00000
50%	36.000000	7.000000	3.000000	3.000000	3.000000	4919.000000	2.00000
75%	43.000000	14.000000	4.000000	4.000000	4.000000	8379.000000	4.00000
max	60.000000	29.000000	5.000000	4.000000	4.000000	19999.000000	9.00000

Step 10 : information about dataset

Code:

```
dataframe.info()
```

```
45]: 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
```

Step 11 : displaying dataframe columns

Code:

```
dataframe.columns
```

```
: dataframe.columns

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

: dataframe.std()
```

Step 12 : showing standard deviation in dataframe

Code:

```
dataframe.std()
```

```
[47]: 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
```

Showing attrition data

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

[48]: No     1233
      Yes     237
      Name: Attrition, dtype: int64

[49]: dataframe['Attrition'].dtypes

[49]: dtype('O')

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

[52]: dataframe.head(10)
```

[52]:

	Age	Attrition	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	JobSatisfaction	MaritalStatus
0	41	1	Sales	1	2	Life Sciences	2	4	Single
1	49	0	Research & Development	8	1	Life Sciences	3	2	Married
2	37	1	Research & Development	2	2	Other	4	3	Single
3	33	0	Research & Development	3	4	Life Sciences	4	3	Married
4	27	0	Research & Development	2	1	Medical	1	2	Married

Activate Windows
Go to Settings to activate Windows.

Step 13 : building a logistic regression model

Code:

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

```
[53]: 0    1
      1    0
      2    1
      3    0
      4    0
      Name: Attrition, dtype: int64
```

Step 14 : replacing columns names with numeric values

Code:

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

```
54]: 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)

55]: dataframe['EducationField'].value_counts()

55]: 1    606
     2    464
     3    159
     5    132
     4     82
     6     27
     Name: EducationField, dtype: int64

56]: dataframe['Department'].value_counts()

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

Step 15 : showing the values of department column

Code:

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

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

8]: 1    961
    2    446
    3     63
    Name: Department, dtype: int64
```

Step 16 : displaying values of martial status

Code:

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

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

[59]: Married     673
      Single      470
      Divorced    327
      Name: MaritalStatus, dtype: int64

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

[61]: dataframe['MaritalStatus'].value_counts()

[61]: 1    673
      2    470
      3    327
      Name: MaritalStatus, dtype: int64
```

Step 17 : displaying datatypes of various columns

Code:

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

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

```
[62]: 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
```

Step 18 : displaying columns in x(dataframe named as x)

Code:

```
x.columns
```

```
•[63]: x.columns
```

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

Step 19 : displaying showing attrition and data of y

Code:

```
y=dataframe['Attrition']
y.head()
```

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

```
•[65]: y.head()
```

```
[65]: 0    1
      1    0
      2    1
      3    0
      4    0
      Name: Attrition, dtype: int64
```

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

[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')
```

Step 21 : testing accuracy of model

Code:

```
y = np.ravel(y)
from sklearn.linear_model import LogisticRegression

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

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

[32]:
```
y = np.ravel(y)
```

[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)
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver wi
ll be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.

  FutureWarning)
```

[33]:
```
0.8408163265306122
```


Step 22 : testing mean of y

Code:

```
y.mean()
```

0.16122448979591836

Step 23:

Code:

```
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)
```

```
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)

     /opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver wi
     ll be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.

       FutureWarning)

35]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
     ,          intercept_scaling=1, max_iter=100, multi_class='warn',
     ,          n_jobs=None, penalty='l2', random_state=None, solver='warn',
     ,          tol=0.0001, verbose=0, warm_start=False)
```

Step 24 : checking model prediction

Code:

```
predicted= model2.predict(X_test)
print (predicted)
```

```
6]: 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.
```

Step 25 : checking model for x_test

Code:

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

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

      [[0.86257761 0.13742239]
       [0.80710189 0.19289811]
       [0.7429987  0.2570013 ]
       [0.83583504 0.16416496]
       [0.73307035 0.26692965]
       [0.78942615 0.21057385]
       [0.85718191 0.14281809]
       [0.85697723 0.14302277]
       [0.96732187 0.03267813]
       [0.93781765 0.06218235]
       [0.95112889 0.04887111]
       [0.83140356 0.16859644]
       [0.86069144 0.13930856]
```

Step 26 : displaying metrics of predicted and tested probability

Code:

```
from sklearn import metrics

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

```
3]: from sklearn import metrics

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

    0.8435374149659864

    0.6500577589526376
```

Step 27 : displaying models2 data

Code:

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

```
[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

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

Step 28 : display values for x_train

Code:

```
print (X_train)
```

```
[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	
320	1.0	27.0	2.0	2.0	3.0	
452	1.0	45.0	2.0	2.0	3.0	
195	1.0	37.0	1.0	21.0	3.0	
776	1.0	20.0	2.0	9.0	3.0	
1295	1.0	41.0	2.0	4.0	1.0	
70	1.0	59.0	2.0	1.0	1.0	
1135	1.0	46.0	2.0	1.0	4.0	
1011	1.0	36.0	2.0	3.0	4.0	

Step 28 : checking values to according to parameters to check the probability of attrition of employee

Code:

```
#add random values to 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))
```

```
[[7.14139240e-07 9.99999286e-01]]
```