importing libraries

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

reading csv and viewing data

In [2]:
```
dataframe = pd.read_csv("IBM Attrition Data.csv")
dataframe.head(10)
```

Out[2]:

	Age	Attrition	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfactio
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	
5	32	No	Research & Development	2	2	Life Sciences	
6	59	No	Research & Development	3	3	Medical	
7	30	No	Research & Development	24	1	Life Sciences	
8	38	No	Research & Development	23	3	Life Sciences	
9	36	No	Research & Development	27	3	Medical	

extracting column names

In [3]:
```
column_names = dataframe.columns.values
print(column_names)
```

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

In [4]:
```
dataframe.info()
```

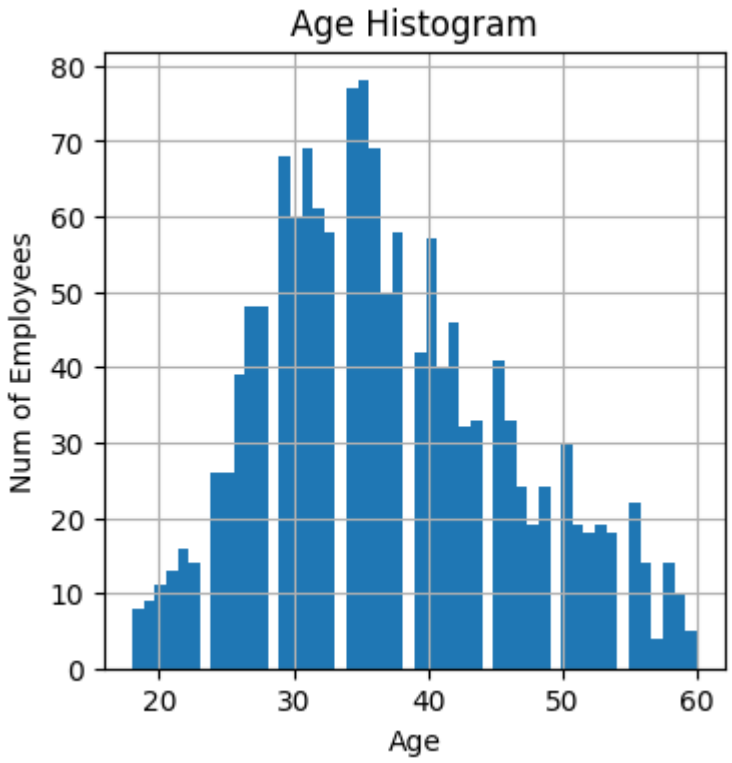
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   Age                      1470 non-null   int64 
 1   Attrition                1470 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 [5]:
```
dataframe.shape
```

Out[5]:
```
(1470, 13)
```

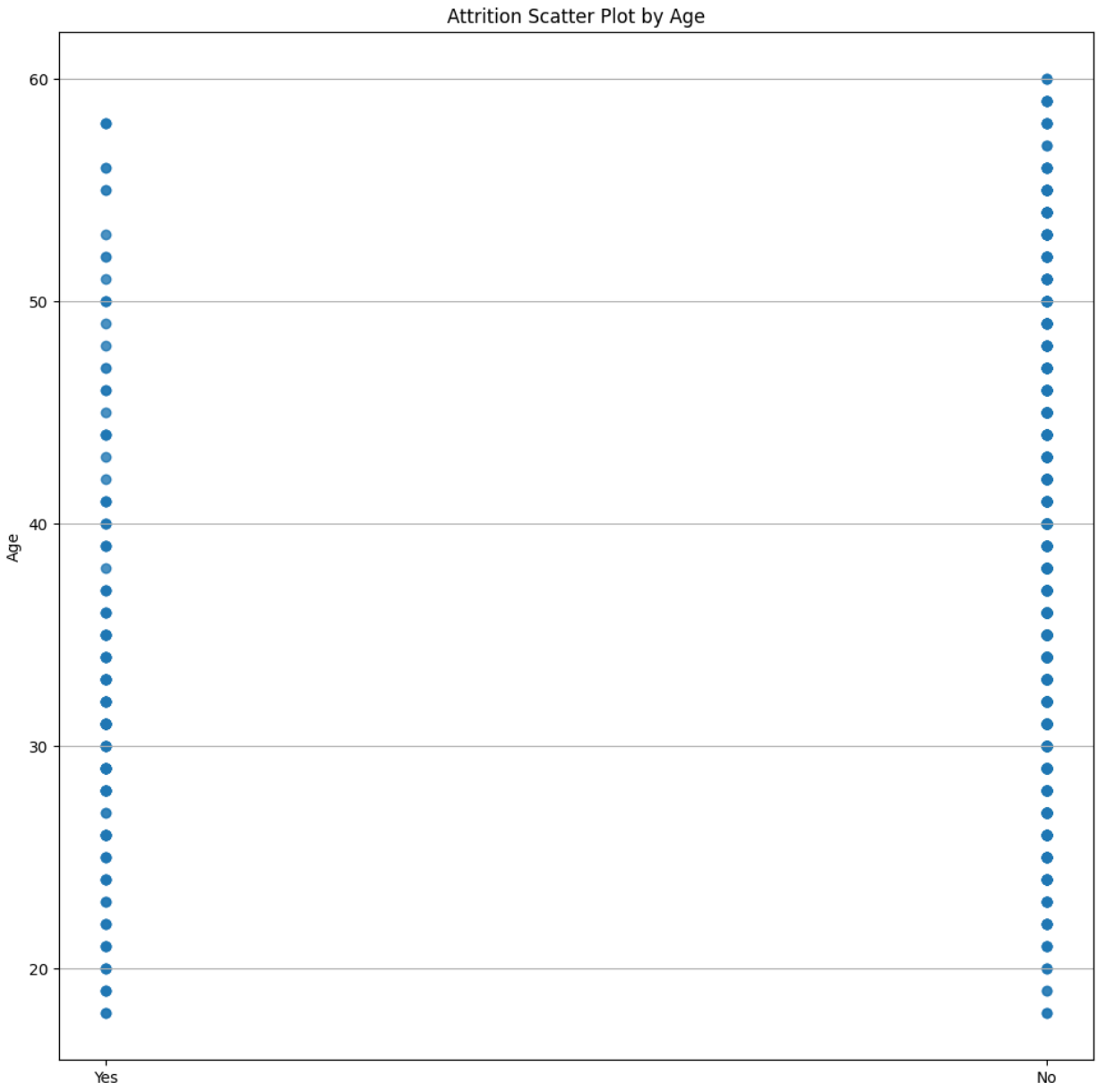
visualizing age in histogram

In [6]:
```
plt.figure(figsize=(4,4))
dataframe['Age'].hist(bins=50)
plt.title("Age Histogram")
plt.xlabel("Age")
plt.ylabel("Num of Employees")
plt.show()
```



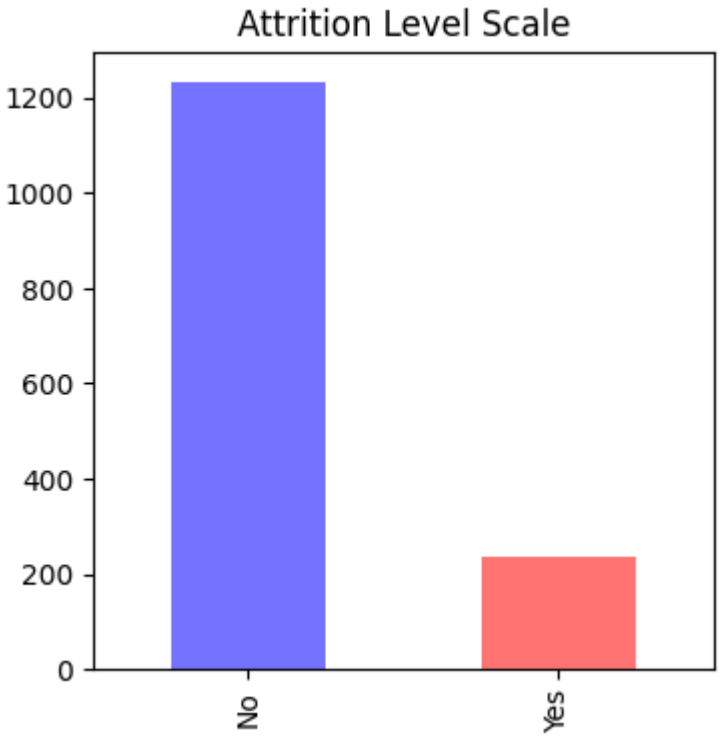
Attriration By Age Graph

In [7]:
```
plt.figure(figsize=(12,12))
plt.scatter(dataframe.Attrition,dataframe.Age,alpha=.55)
plt.title("Attrition Scatter Plot by Age")
plt.ylabel("Age")
plt.grid(visible=True,axis='y')
plt.show()
```



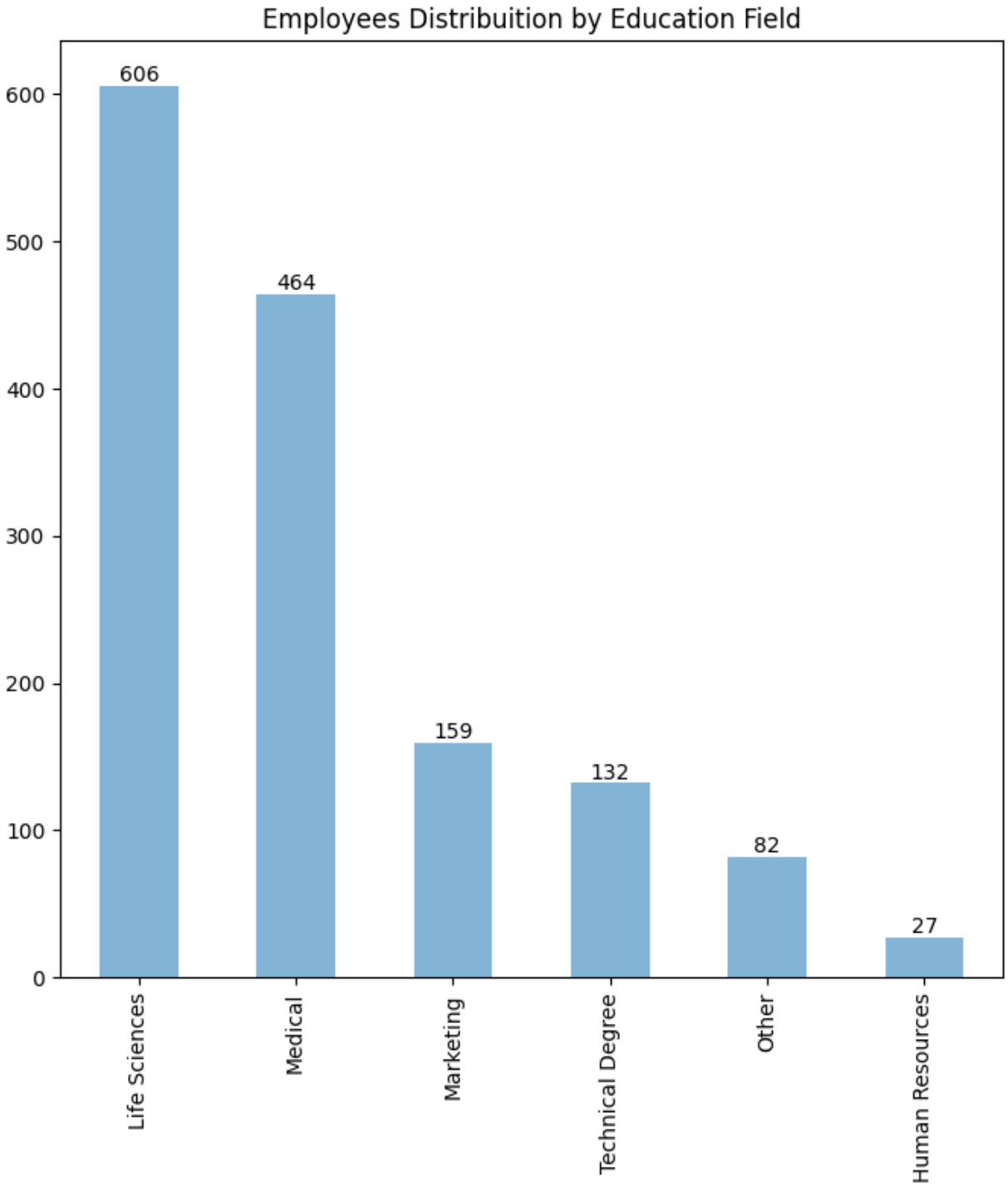
Remaining Employees Graph

In [8]:
```
plt.figure(figsize=(4,4,))
colors = ['blue', 'red']
dataframe.Attrition.value_counts().plot(kind='bar',color=colors,alpha=.55)
plt.title("Attrition Level Scale")
plt.show()
```



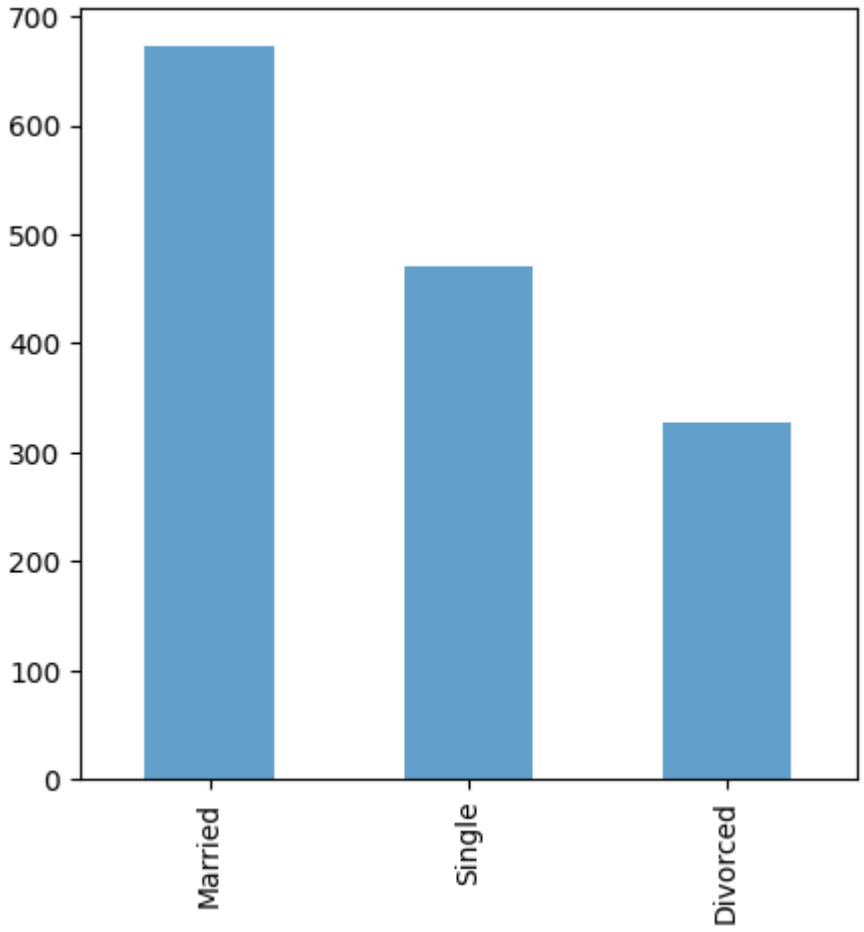
Distribuition by Education

In [9]:
```
plt.figure(figsize=(8,8))
dataframe.EducationField.value_counts().plot(kind='bar',alpha=.55)
plt.title("Employees Distribuition by Education Field")
for i, count in enumerate(dataframe.EducationField.value_counts()):
    plt.text(i, count, str(count), ha='center', va='bottom')
plt.show()
```



Distribuition by RelationShip Status

In [10]:
```
plt.figure(figsize=(5,5))
dataframe.MaritalStatus.value_counts().plot(kind='bar',alpha=.70)
plt.show()
```



Quartiles of DataFrame with Discriptive Statistics

In [11]:
```
dataframe.describe()
```

Out[11]:

	Age	DistanceFromHome	Education	EnvironmentSatisfaction	JobSatisfaction	Month
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	147
mean	36.923810	9.192517	2.912925	2.721769	2.728571	65(
std	9.135373	8.106864	1.024165	1.093082	1.102846	47(
min	18.000000	1.000000	1.000000	1.000000	1.000000	10(
25%	30.000000	2.000000	2.000000	2.000000	2.000000	29'
50%	36.000000	7.000000	3.000000	3.000000	3.000000	49'
75%	43.000000	14.000000	4.000000	4.000000	4.000000	83'
max	60.000000	29.000000	5.000000	4.000000	4.000000	199

In [12]:
```
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 [13]:
```
dataframe.std()
```

```
C:\Users\fast laptop\AppData\Local\Temp\ipykernel_5204\3401367348.py:1: FutureWarnin
g: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is
deprecated; in a future version this will raise TypeError.  Select only valid columns
before calling the reduction.
  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'].dtypes
```

Out[14]:
```
dtype('O')
```

In [15]:
```
dataframe['Attrition'].value_counts()
```

Out[15]:
```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

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	EnvironmentSatisfactio
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	

LabelEncoding Manually For LogisiticRegression

In [18]:

```
x = dataframe.drop(['Attrition'],axis=1)
print(x.head())
print("================================================")
y = dataframe['Attrition']
print(y.head())
```

```
   Age              Department  DistanceFromHome  Education EducationField  \
0   41                   Sales                 1          2  Life Sciences
1   49  Research & Development                 8          1  Life Sciences
2   37  Research & Development                 2          2          Other
3   33  Research & Development                 3          4  Life Sciences
4   27  Research & Development                 2          1        Medical

   EnvironmentSatisfaction  JobSatisfaction MaritalStatus  MonthlyIncome  \
0                        2                4        Single           5993
1                        3                2       Married           5130
2                        4                3        Single           2090
3                        4                3       Married           2909
4                        1                2       Married           3468

   NumCompaniesWorked  WorkLifeBalance  YearsAtCompany
0                   8                1               6
1                   1                3              10
2                   6                3               0
3                   1                3               8
4                   9                3               2
==================================================
0    1
1    0
2    1
3    0
4    0
Name: Attrition, dtype: int64
```

encoding EducationField column

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]:
```
print(dataframe.EducationField.head())
```

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

In [21]:
```
dataframe.EducationField.value_counts()
```

Out[21]:
```
1    606
2    464
3    159
5    132
4     82
6     27
Name: EducationField, dtype: int64
```

In [22]:
```
dataframe.Department.value_counts()
```

Out[22]:
```
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64
```

Encoding Department column

In [23]:
```
dataframe['Department'].replace('Research & Development',1, inplace=True)
dataframe['Department'].replace('Sales',2, inplace=True)
dataframe['Department'].replace('Human Resources', 3, inplace=True)
```

In [24]:
```
dataframe.Department.head()
```

Out[24]:
```
0    2
1    1
2    1
3    1
4    1
Name: Department, dtype: int64
```

Encoding MaritalStatus column

In [25]:
```
dataframe.MaritalStatus.value_counts()
```

Out[25]:
```
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
```

In [26]:
```
dataframe['MaritalStatus'].replace('Married',1, inplace=True)
dataframe['MaritalStatus'].replace('Single',2, inplace=True)
dataframe['MaritalStatus'].replace('Divorced',3, inplace=True)
```

In [27]:
```
dataframe.MaritalStatus.head()
```

Out[27]:
```
0    2
1    1
2    2
3    1
4    1
Name: MaritalStatus, dtype: int64
```

In [28]:
```
X = dataframe.select_dtypes(include=['int64'])
X.dtypes
```

Out[28]:
```
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 [29]:
```
X.columns
```

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

In [41]:
```
Y = dataframe.Attrition
Y.head()
```

Out[41]:
```
0    1
1    0
2    1
3    0
4    0
Name: Attrition, dtype: int64
```

In [31]:
```
Y, X = dmatrices('Attrition ~ Age + Department + DistanceFromHome + Education + Educat
```

In [42]:
```
print(X.columns)
X.head()
```

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

Out[42]:

	Intercept	Age	Department	DistanceFromHome	Education	EducationField	YearsAtCompany
0	1.0	41.0	2.0	1.0	2.0	1.0	6.0
1	1.0	49.0	1.0	8.0	1.0	1.0	10.0
2	1.0	37.0	1.0	2.0	2.0	4.0	0.0
3	1.0	33.0	1.0	3.0	4.0	1.0	8.0
4	1.0	27.0	1.0	2.0	1.0	2.0	2.0

Sequencing Array to 1 dimensional array

In [33]:
```python
Y = np.ravel(Y)
print(Y)
```

```
[1. 0. 1. ... 0. 0. 0.]
```

Creating LogisiticRegression

In [43]:
```python
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model = model.fit(X,Y)

model.score(X,Y)
```

Out[43]: 0.8408163265306122

In [44]:
```python
Y.mean()
```

Out[44]: 0.16122448979591836

In [54]:
```python
X_train,X_test,Y_train,Y_test=skl.model_selection.train_test_split(X,Y, test_size=0.4,
model2=LogisticRegression()
model2.fit(X_train, Y_train)
```

Out[54]:

```
▾ LogisticRegression
LogisticRegression()
```

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

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 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]
```

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

```
[[0.86256067 0.13743933]
 [0.82947283 0.17052717]
 [0.74749496 0.25250504]
 ...
 [0.85489844 0.14510156]
 [0.83122171 0.16877829]
 [0.82330024 0.17669976]]
```

In [57]:
```
print(skl.metrics.accuracy_score(Y_test, predVals))
```

```
0.8452380952380952
```

In [59]:
```
print(skl.metrics.roc_auc_score(Y_test,prob[:, 1]))
```

```
0.6458564135983491
```

In [60]:
```
print(skl.metrics.confusion_matrix(Y_test,predVals))
```

```
[[494   1]
 [ 90   3]]
```

In [61]:
```
print(skl.metrics.classification_report(Y_test,predVals))
```

```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92       495
           1       0.75      0.03      0.06        93

    accuracy                           0.85       588
   macro avg       0.80      0.52      0.49       588
weighted avg       0.83      0.85      0.78       588
```