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

import seaborn as sns
```

```
dataframe=pd.read_csv("/content/IBM HR Analytics Employee Attrition Modeling .zip")
```

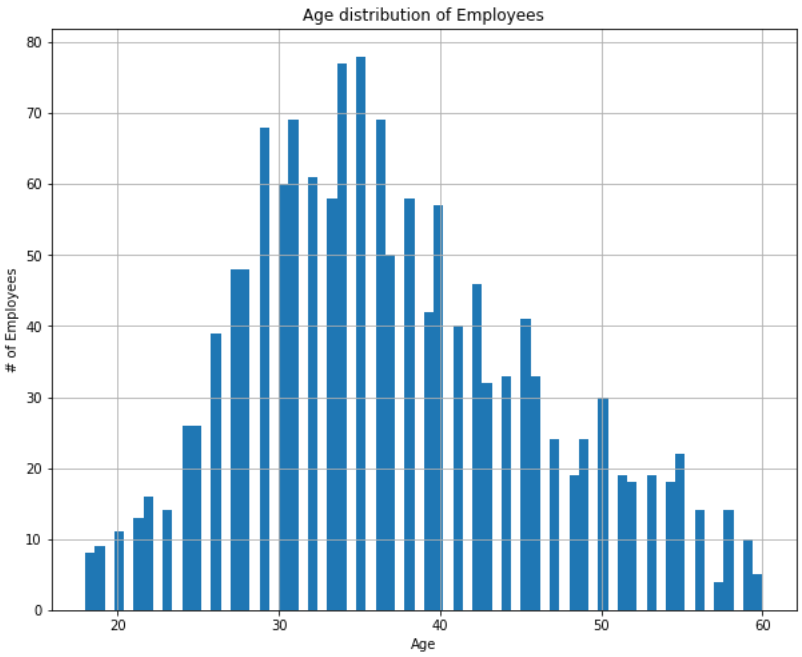
```
dataframe.head()
```

	Age	Attrition	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	JobSatisfact
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	

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

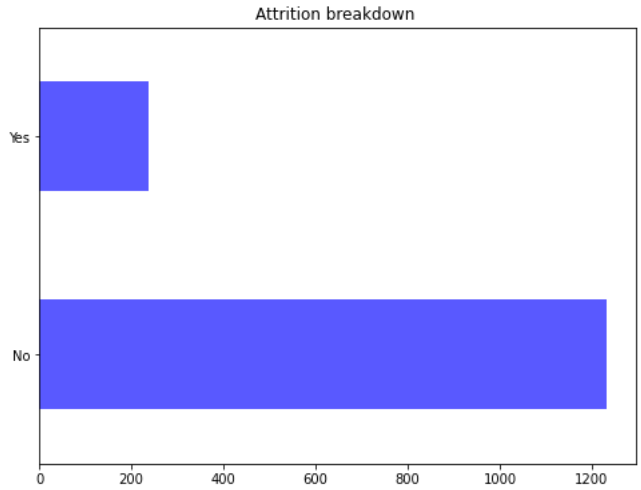
```
['Age' 'Attrition' 'Department' 'DistanceFromHome' 'Education'
 'EducationField' 'EnvironmentSatisfaction' 'JobSatisfaction'
 'MaritalStatus' 'MonthlyIncome' 'NumCompaniesWorked' 'WorkLifeBalance'
 'YearsAtCompany']
```

```
# 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()
```




```
# 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()
```

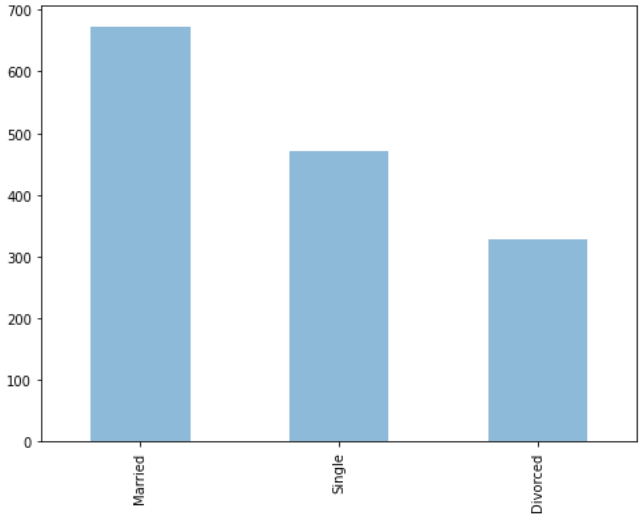

```
# explore data for Left employees breakdown
plt.figure(figsize=(8,6))
dataframe.Attrition.value_counts().plot(kind='barh',color='blue',alpha=.65)
plt.title("Attrition breakdown ")
plt.show()
```




```
# 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()
```

Education Field Distribution

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




```
dataframe.describe()
```

	Age	DistanceFromHome	Education	EnvironmentSatisfaction	JobSatisfaction	MonthlyIncome	NumCom
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	

```
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
```

```
dataframe.columns
```

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

```
dataframe.std()
```

```
<ipython-input-16-90bdd6cc73ba>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; i
  dataframe.std()
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
```

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

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

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

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

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

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

	Age	Attrition	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfa
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	

```
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()
```

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

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

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

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

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

```
Married     673
Single      470
```

```
Divorced    327
Name: MaritalStatus, dtype: int64
```

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

```
Age                        int64
Attrition                  int64
Department                 int64
DistanceFromHome           int64
Education                  int64
EducationField             int64
EnvironmentSatisfaction    int64
JobSatisfaction            int64
MonthlyIncome              int64
NumCompaniesWorked         int64
WorkLifeBalance            int64
YearsAtCompany             int64
dtype: object
```

```
x.columns
```

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

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

```
y.head()
```

```
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)
```

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

```
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)
```

```
0.8408163265306122
```

```
y.mean()
```

```
0.16122448979591836
```

```
import sklearn.model_selection
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)
```

```
LogisticRegression()
```

```
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. 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.]
```

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

```
[0.80585706 0.19414294]
[0.83150068 0.16849932]
[0.53816801 0.46183199]
[0.95031821 0.04968179]
[0.73291921 0.26708079]
[0.89182874 0.10817126]
[0.80079859 0.19920141]
[0.87739802 0.12260198]
[0.96805228 0.03194772]
[0.81741596 0.18258404]
[0.86150085 0.13849915]
[0.59407578 0.40592422]
[0.82625729 0.17374271]
[0.92534969 0.07465031]
[0.81692206 0.18307794]
[0.9258678  0.0741322 ]
[0.8909459  0.1090541 ]
[0.70071778 0.29928222]
[0.82181894 0.17818106]
[0.96589935 0.03410065]
[0.8699699  0.1300301 ]
[0.89918545 0.10081455]
[0.88983808 0.11016192]
[0.81432466 0.18567534]
[0.85830032 0.14169968]
[0.8387865  0.1612135 ]
[0.84056579 0.15943421]
[0.82661166 0.17338834]
[0.94075161 0.05924839]
[0.83183533 0.16816467]
[0.7756136  0.2243864 ]
[0.69399555 0.30600445]
[0.8596213  0.1403787 ]
[0.8251394  0.1748606 ]
[0.84107717 0.15892283]
[0.87191013 0.12808987]
[0.8944765  0.1055235 ]
[0.82945355 0.17054645]
[0.72856817 0.27143183]
[0.94692819 0.05307181]
[0.96074297 0.03925703]
[0.90500199 0.09499801]
[0.88599986 0.11400014]
[0.84865641 0.15134359]
[0.79086983 0.20913017]
[0.67302986 0.32697014]
[0.93390204 0.06609796]
[0.65644896 0.34355104]
[0.74382857 0.25617143]
[0.94248406 0.05751594]
[0.78365668 0.21634332]
[0.90655654 0.09344346]
[0.81578726 0.18421274]
[0.89149735 0.10850265]
[0.85791873 0.14208127]
[0.67453868 0.32546132]
[0.93130389 0.06869611]
[0.89999981 0.10000019]]
```

```
from sklearn import metrics

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

```
0.8435374149659864
0.6502502887947632
```

```
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
```

```
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]
```

```
#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))
```

```
[[6.25571946e-07 9.99999374e-01]]
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but LogisticRegression was fitted wi
  warnings.warn(
```