Logistic Regression Kaggle Data Set Processing

In [ ]:

```
```

In [4]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Used for Confusion Matrix
from sklearn import metrics
import seaborn as sns

np.set_printoptions(precision=2, suppress=True)

from sklearn.datasets import fetch_openml
#dataset = fetch_openml("mnist_784")

# Used for Splitting Training and Test Sets
from sklearn.model_selection import train_test_split

%matplotlib inline

from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
```

In [7]:

```
s_list = []

intercept_list = []
weights_list = []

df = pd.read_csv("HR.csv")
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
df['n_Gender'] = labelencoder.fit_transform(df['Gender'])
df['n_JobRole']=labelencoder.fit_transform(df['JobRole'])
df['n_Attrition'] = labelencoder.fit_transform(df['Attrition'])
df['n_BusinessTravel'] = labelencoder.fit_transform(df['BusinessTravel'])

df['n_Department'] = labelencoder.fit_transform(df['Department'])
df['n_EducationField'] = labelencoder.fit_transform(df['EducationField'])
df.head()
df.drop(['Attrition','MaritalStatus','OverTime','Over18','BusinessTravel','JobRole','Gender','Department','Educ
        axis=1, inplace=True)
df.head()
p = df['n_Attrition']
#df.drop(['n_Attrition'],axis=1, inplace=True)



from sklearn.model_selection import KFold
kf = KFold(n_splits=5, random_state=None, shuffle=True)
train = df.to_numpy()
test = p.to_numpy()
#.values.ravel()

dftemp = df
#p = df.from_dict(p,orient='index',columns=['n_Attrition'])
#p.shape()

for train_index, test_index in kf.split(df):
    #print("TRAIN:", train_index, "TEST:", test_index,"\n\n")
    #print("start TRAIN:", train_index, "TEST:", test_index,"end\n\n")

    X_train, X_test = df.iloc[train_index], df.iloc[test_index]

    y_train, y_test = X_train.loc[:,['n_Attrition']], X_test.loc[:,['n_Attrition']]
    #X_train, X_test = train_index, test_index
    #y_train, y_test = , p.values.ravel()

    X_train.drop(['n_Attrition'],axis=1, inplace=True)
    train_img =   X_train


    #train_img.drop(['n_Attrition'],axis=1, inplace=True)

    X_test.drop(['n_Attrition'],axis=1, inplace=True)
    test_img = X_test
    #test_img.drop(['n_Attrition'],axis=1, inplace=True)

    train_lbl = y_train
    test_lbl = y_test

    # test_size: what proportion of original data is used for test set
#    train_img, test_img, train_lbl, test_lbl = train_test_split(df, p, test_size=1/7.0, random_state=0)

    #train_lbl.head()
```

```
#train_img.columns

#df.drop(['n_Attrition'],axis=1, inplace=True)

X_train = train_img
X_test = test_img
Y_train = train_lbl
Y_test = test_lbl



logisticRegr = LogisticRegression( solver='lbfgs', max_iter=9000,warm_start=True)

#logisticRegr = LinearRegression()

logisticRegr
np.set_printoptions(precision=2, suppress=True)

logisticRegr.fit(train_img, train_lbl.values.ravel())

#p = logisticRegr.intercept_

#p = np.array2string(p)
#p = str(round(p, 2))
print(f'intercept: {p}')

np.set_printoptions(precision=2, suppress=True)
p = logisticRegr.coef_[0]

p = logisticRegr.coef_[0]

#p = np.array2string(p)
#p = round(p, 4)

#p = np.array2string(p)
print(f" Weights {p}")

#logisticRegr.predict(test_img)

#logisticRegr.predict(test_img[0:10])

np.set_printoptions(precision=2, suppress=True)
```

```
    #score = logisticRegr.score(test_img, test_lbl)
    #p = str(round(score, 2))

    #print(p)

    predictions = logisticRegr.predict(test_img)
    print(f'\n Predict {predictions[:10]} \n Actual {test_lbl[:10]}')

    #std_dev = [s, s]


    from sklearn.metrics import mean_squared_error
    s = mean_squared_error(test_lbl, predictions)
    p = round(s, 4)

    print(f'mean squared error {p}')
    s_list.append(p)
```

```
/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py:4163: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  return super().drop(
intercept: 0       1
1       0
2       1
3       0
4       0
       ..
1465    0
1466    0
1467    0
1468    0
1469    0
Name: n_Attrition, Length: 1470, dtype: int64
 Weights [-0.03 -0.    0.04  0.04  0.   -0.   -0.39 -0.   -0.35 -0.05 -0.35 -0.
  0.    0.14 -0.02  0.06 -0.12  0.05 -0.35 -0.05 -0.19 -0.18  0.08 -0.14
  0.14 -0.12  0.08  0.03 -0.05  0.13  0.08]

 Predict [0 0 0 0 0 0 0 0 0 0]
 Actual     n_Attrition
2             1
3             0
6             0
7             0
11            0
```

```
21            1
26            1
34            1
45            1
50            1
mean squared error 0.1463
/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py:4163: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  return super().drop(
intercept: 0.1463
 Weights [-0.03 -0.    0.03 -0.05  0.   -0.   -0.32 -0.   -0.25 -0.01 -0.35 -0.
  0.    0.17 -0.03  0.01 -0.26  0.05 -0.39 -0.04 -0.18 -0.13  0.08 -0.14
  0.16 -0.16  0.07  0.04  0.04  0.14  0.03]

 Predict [0 0 0 0 0 0 0 0 1 0]
 Actual     n_Attrition
10            0
13            0
14            1
17            0
20            0
27            0
28            0
29            0
42            1
53            0
mean squared error 0.1701
/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py:4163: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  return super().drop(
intercept: 0.1701
 Weights [-0.03 -0.    0.04  0.09  0.   -0.   -0.32 -0.   -0.43 -0.08 -0.35 -0.
  0.    0.13 -0.02  0.03 -0.17  0.05 -0.46 -0.05 -0.27 -0.28  0.08 -0.13
  0.11 -0.11  0.12  0.06  0.01  0.14  0.09]

 Predict [0 0 0 0 0 0 0 0 0 1]
 Actual     n_Attrition
5             0
19            0
25            0
30            0
31            0
```

```
33            1
35            0
40            0
46            0
47            0
mean squared error 0.1531
/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py:4163: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  return super().drop(
intercept: 0.1531
 Weights [-0.04 -0.    0.03  0.03  0.   -0.   -0.35 -0.   -0.4  -0.03 -0.32 -0.
  0.    0.11 -0.02  0.01 -0.23  0.05 -0.49 -0.03 -0.15 -0.11  0.07 -0.12
  0.12 -0.11  0.08  0.05  0.04  0.15  0.02]

 Predict [0 0 0 0 0 0 0 0 0 0]
 Actual     n_Attrition
1             0
4             0
8             0
15            0
18            0
23            0
24            1
32            0
36            1
37            0
mean squared error 0.1429
/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py:4163: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  return super().drop(
intercept: 0.1429
 Weights [-0.03 -0.    0.03 -0.02  0.   -0.   -0.42 -0.   -0.46 -0.06 -0.4  -0.
  0.    0.15 -0.02  0.07 -0.09  0.06 -0.52 -0.04 -0.14 -0.27  0.1  -0.15
  0.14 -0.14  0.02  0.02  0.05  0.12  0.06]

 Predict [0 0 1 0 0 0 0 0 0 0]
 Actual     n_Attrition
0             1
9             0
12            0
16            0
22            0
```

```
39              0
43              0
44              0
63              0
65              0
mean squared error 0.1395
```

In [9]:

```python
print(f'Mean Square Error ---> {s_list}')
std_dev = s_list


p = round(np.std(std_dev, dtype=np.float64),4)
print(f' Standard Error ----> {p}')
```

```
Mean Square Error ---> [0.1463, 0.1701, 0.1531, 0.1429, 0.1395]
 Standard Error ----> 0.0108
```

In [ ]:

In [ ]:

In [ ]:

Splitting Data into Training and Test Sets

In [10]:

```python
train_img.columns
```

Out[10]:

```
Index(['Age', 'DailyRate', 'DistanceFromHome', 'Education', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobSatisfaction', 'MonthlyIncome',
       'MonthlyRate', 'NumCompaniesWorked', 'PercentSalaryHike',
       'PerformanceRating', 'RelationshipSatisfaction', 'StandardHours',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager', 'n_Gender',
       'n_JobRole', 'n_BusinessTravel', 'n_Department', 'n_EducationField'],
      dtype='object')
```

In [11]:

```python
X_train = train_img
X_test = test_img
Y_train = train_lbl
Y_test = test_lbl
```

```
#print(f'Xtrain {X_train} X_test {X_test}')
```

In [12]:
```
print(f'x_train {X_train[0:1]}')
print(f'y_train {Y_train[0:1]}')
print(f'x_train {X_test[0:1]}')
```

```
x_train    Age  DailyRate  DistanceFromHome  Education  EmployeeCount  EmployeeNumber  \
1   49        279                 8          1              1               2   

   EnvironmentSatisfaction  HourlyRate  JobInvolvement  JobLevel  ...  \
1                        3          61               2         2  ...   

   WorkLifeBalance  YearsAtCompany  YearsInCurrentRole  \
1                3              10                   7   

   YearsSinceLastPromotion  YearsWithCurrManager  n_Gender  n_JobRole  \
1                        1                     7         1          6   

   n_BusinessTravel  n_Department  n_EducationField  
1                 1             1                 1  

[1 rows x 31 columns]
y_train    n_Attrition
1            0
x_train    Age  DailyRate  DistanceFromHome  Education  EmployeeCount  EmployeeNumber  \
0   41       1102                 1          2              1               1   

   EnvironmentSatisfaction  HourlyRate  JobInvolvement  JobLevel  ...  \
0                        2          94               3         2  ...   

   WorkLifeBalance  YearsAtCompany  YearsInCurrentRole  \
0                1               6                   4   

   YearsSinceLastPromotion  YearsWithCurrManager  n_Gender  n_JobRole  \
0                        0                     5         0          7   

   n_BusinessTravel  n_Department  n_EducationField  
0                 2             2                 1  

[1 rows x 31 columns]
```

In [ ]:

In [13]:
```
print(train_img.shape)
```

```
(1176, 31)
```

```
In [14]: print(train_lbl.shape)
```

```
(1176, 1)
```

```
In [15]: print(test_img.shape)
```

```
(294, 31)
```

```
In [16]: print(test_lbl.shape)
```

```
(294, 1)
```

Using Logistic Regression on Entire Dataset

```
In [ ]:
```

```
In [ ]:
```

```
In [17]: logisticRegr
```

```
Out[17]: LogisticRegression(max_iter=9000, warm_start=True)
```

```
In [18]: np.set_printoptions(precision=2, suppress=True)

         logisticRegr.fit(train_img, train_lbl)

         #logisticRegr = LogisticRegression(solver = 'lbfgs',max_iter=1200)
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py:72: DataConversionWarning: A column-vect
or y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
```

```
Out[18]: LogisticRegression(max_iter=9000, warm_start=True)
```

```
In [19]: print('intercept:', logisticRegr.intercept_)
```

```
intercept: [0.]
```

```
In [20]: np.set_printoptions(precision=2, suppress=True)
         p = logisticRegr.coef_[0]

         #p = np.array2string(p)
         print(f"{p}")
```

```
[-0.03 -0.    0.03 -0.02  0.   -0.   -0.42 -0.   -0.46 -0.06 -0.4  -0.
  0.    0.15 -0.02  0.07 -0.09  0.06 -0.52 -0.04 -0.14 -0.27  0.1  -0.15
  0.14 -0.14  0.02  0.02  0.05  0.12  0.06]
```

Uses the information the model learned during the model training process

In [21]:

```python
# Returns a NumPy Array
# Predict for One Observation (image)
logisticRegr.predict(test_img)
```

Out[21]:

```
array([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, 1, 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, 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, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 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, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 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, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0])
```

In [22]:

```python
# Predict for Multiple Observations (images) at Once
logisticRegr.predict(test_img[0:10])
```

Out[22]:

```
array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0])
```

Measuring Model Performance

accuracy (fraction of correct predictions): correct predictions / total number of data points

Basically, how the model performs on new data (test set)

In [23]:

```python
np.set_printoptions(precision=2, suppress=True)

score = logisticRegr.score(test_img, test_lbl)
print(score)
```

```
0.8605442176870748
```

In [ ]:

In [24]:

```python
# Make predictions on test data
```

```
predictions = logisticRegr.predict(test_img)
print(f'{predictions}')
```

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

```
In [ ]:
```

```
In [ ]:
```