Logistic Regression Kaggle Data Set Processing

In [ ]:

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import mean_squared_error
# 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 [15]:
```
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 = 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


    from sklearn.metrics import accuracy_score







    from sklearn.metrics import accuracy_score
    p = np.round(predictions).astype(int)
    acc = accuracy_score(test_lbl,p)



    #s = mean_squared_error(test_lbl, predictions)
    p = round(acc, 2)

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

```
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 -0.003922572756349346
```

```
 Predict [0.15 0.37 0.16 0.21 0.01 0.32 0.34 0.13 0.07 0.43]
 Actual     n_Attrition
8             0
12            0
19            0
30            0
31            0
36            1
38            0
41            0
44            0
47            0
mean squared error 0.85
intercept: 0.85
 Weights -0.0037047644726774655

 Predict [0.29 0.12 0.06 0.18 0.1  0.2  0.34 0.17 0.18 0.06]
 Actual     n_Attrition
0             1
3             0
5             0
13            0
16            0
23            0
24            1
29            0
37            0
39            0
mean squared error 0.83
intercept: 0.83
 Weights -0.004673571955363548

 Predict [ 0.04  0.01  0.13  0.16  0.06  0.1  -0.19  0.09  0.13  0.25]
 Actual     n_Attrition
1             0
6             0
15            0
18            0
20            0
22            0
25            0
43            0
45            1
51            1
mean squared error 0.84
intercept: 0.84
 Weights -0.003037182499893588

 Predict [ 0.4   0.14  0.42  0.19 -0.03  0.25 -0.03  0.28  0.33  0.33]
```

```
 Actual     n_Attrition
14            1
17            0
26            1
27            0
28            0
32            0
33            1
34            1
46            0
54            0
mean squared error 0.85
intercept: 0.85
 Weights -0.0033438458684491566

 Predict [ 0.27  0.48  0.27  0.2   0.17  0.13  0.36 -0.07  0.19  0.1 ]
 Actual     n_Attrition
2             1
4             0
7             0
9             0
10            0
11            0
21            1
35            0
40            0
49            0
mean squared error 0.85
/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(
/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(
/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(
/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(
/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(
```

In [16]:
```
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 ---> [0.85, 0.83, 0.84, 0.85, 0.85]
 Standard Error ----> 0.008
```

In [ ]:

In [ ]:

In [ ]:

Splitting Data into Training and Test Sets

In [3]:
```
train_img.columns
```

Out[3]:
```
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 [4]:
```
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 [5]:
```
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 [6]:
```
print(train_img.shape)
```

```
(1176, 31)
```

In [7]:
```
print(train_lbl.shape)
```

```
(1176, 1)
```

In [8]:
```
print(test_img.shape)
```

```
(294, 31)
```

In [9]:
```
print(test_lbl.shape)
```

```
(294, 1)
```

Using Logistic Regression on Entire Dataset

In [ ]:
```

```

In [ ]:
```

```

In [10]:
```
logisticRegr
```

Out[10]:
```
LinearRegression()
```

In [11]:
```
np.set_printoptions(precision=2, suppress=True)

logisticRegr.fit(train_img, train_lbl)

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

Out[11]:
```
LinearRegression()
```

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

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

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

Uses the information the model learned during the model training process

In [ ]:
```

```

In [ ]:
```
# Returns a NumPy Array
# Predict for One Observation (image)
logisticRegr.predict(test_img)
```

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

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

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

In [ ]:
```

```

In [ ]:
```
# Make predictions on test data
predictions = logisticRegr.predict(test_img)
print(f'{predictions}')
```

In [ ]:
```

```

In [ ]:
```

```