# ASSIGNMENT / TASK 9

Predict retention of an employee within an organization such that whether the employee will leave the company or continue with it. An organization is only as good as its employees, and these people are the true source of its competitive advantage. Dataset is downloaded from Kaggle.

First do data exploration and visualization, after this create a logistic regression model to predict Employee Attrition Using Machine Learning & Python.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
df=pd.read_csv("/content/archive (1).zip")
df.head()
```

| | satisfaction_level | last_evaluation | number_project | average_montly_hours | time |
|---|---|---|---|---|---|
| **0** | 0.38 | 0.53 | 2 | 157 | |
| **1** | 0.80 | 0.86 | 5 | 262 | |
| **2** | 0.11 | 0.88 | 7 | 272 | |
| **3** | 0.72 | 0.87 | 5 | 223 | |
| **4** | 0.37 | 0.52 | 2 | 159 | |

```
df.shape
```

```
(14999, 10)
```

```
df.keys()
```

```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
       'promotion_last_5years', 'Department', 'salary'],
      dtype='object')
```

```
dummies=pd.get_dummies(df.Department)
merge=pd.concat([df,dummies],axis='columns')
dummies_new=pd.get_dummies(merge.salary)
merge_new=pd.concat([merge,dummies_new],axis="columns")
merge_new.head()
```

| | satisfaction_level | last_evaluation | number_project | average_montly_hours | time_spen |
|---|---|---|---|---|---|
| **0** | 0.38 | 0.53 | 2 | 157 | |
| **1** | 0.80 | 0.86 | 5 | 262 | |
| **2** | 0.11 | 0.88 | 7 | 272 | |
| **3** | 0.72 | 0.87 | 5 | 223 | |
| **4** | 0.37 | 0.52 | 2 | 159 | |

```
merge_new=merge_new.drop(["salary"],axis="columns")
merge_new.head()
```

| | satisfaction_level | last_evaluation | number_project | average_montly_hours | time_spen |
|---|---|---|---|---|---|
| **0** | 0.38 | 0.53 | 2 | 157 | |
| **1** | 0.80 | 0.86 | 5 | 262 | |
| **2** | 0.11 | 0.88 | 7 | 272 | |
| **3** | 0.72 | 0.87 | 5 | 223 | |
| **4** | 0.37 | 0.52 | 2 | 159 | |

+ Code + Text

```
x=merge_new.loc[:,["satisfaction_level","last_evaluation","number_project","average_montly_ho
print(x,"\n")
y=merge_new["left"]
print(y)
```

```
       satisfaction_level  last_evaluation  number_project  ...  high  low  medium
0                    0.38             0.53               2  ...     0    1       0
1                    0.80             0.86               5  ...     0    0       1
2                    0.11             0.88               7  ...     0    0       1
3                    0.72             0.87               5  ...     0    1       0
4                    0.37             0.52               2  ...     0    1       0
...                   ...              ...             ...  ...   ...  ...     ...
14994                0.40             0.57               2  ...     0    1       0
14995                0.37             0.48               2  ...     0    1       0
14996                0.37             0.53               2  ...     0    1       0
14997                0.11             0.96               6  ...     0    1       0
14998                0.37             0.52               2  ...     0    1       0

[14999 rows x 20 columns]

0        1
1        1
2        1
3        1
4        1
        ..
```

```
    14994    1
    14995    1
    14996    1
    14997    1
    14998    1
    Name: left, Length: 14999, dtype: int64
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.25, random_state=5)
mymodel=LogisticRegression()
mymodel.fit(x_train, y_train)
```

```
    /usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: Convergenc
    STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

    Increase the number of iterations (max_iter) or scale the data as shown in:
        https://scikit-learn.org/stable/modules/preprocessing.html
    Please also refer to the documentation for alternative solver options:
        https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
      extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
    LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                       intercept_scaling=1, l1_ratio=None, max_iter=100,
                       multi_class='auto', n_jobs=None, penalty='l2',
                       random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                       warm_start=False)
```

```
y_pred=mymodel.predict(x_test)#y_test
print(y_pred)
```

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

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
cm
```

```
    array([[2602,  229],
           [ 599,  320]])
```

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_pred))
```

```
    0.7792
```

```
print(mymodel.score(x_train,y_train))
print(mymodel.score(x_train,mymodel.predict(x_train)))
```

```
    0.8008711885500933
    1.0
```

```
print(mymodel.score(x_test,y_test))
print(mymodel.score(x_test,y_test))
print(mymodel.score(x_test,mymodel.predict(x_test)))

     0.7792
     1.0
```

```
print(classification_report(y, mymodel.predict(x)))

                   precision    recall  f1-score   support

                0       0.83      0.93      0.87     11428
                1       0.62      0.37      0.46      3571

         accuracy                           0.80     14999
        macro avg       0.72      0.65      0.67     14999
     weighted avg       0.78      0.80      0.78     14999
```