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

```
from sklearn.linear_model import LogisticRegression
df = pd.read_csv("/content/HR_comma_sep.csv")
```

```
df
```

	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	
...	...	...	...	...	
14994	0.40	0.57	2	151	
14995	0.37	0.48	2	160	
14996	0.37	0.53	2	143	
14997	0.11	0.96	6	280	
14998	0.37	0.52	2	158	

14999 rows × 10 columns

```
df1 = df[['salary','satisfaction_level', 'average_montly_hours', 'promotion_last_5years','lef
df1
```

	salary	satisfaction_level	average_montly_hours	promotion_last_5years	left
0	low	0.38	157	0	1
1	medium	0.80	262	0	1
2	medium	0.11	272	0	1
3	low	0.72	223	0	1
4	low	0.37	159	0	1
...	...	...	...	...	...
14994	low	0.40	151	0	1
14995	low	0.37	160	0	1
14996	low	0.37	143	0	1
14997	low	0.11	280	0	1

```
dummies = pd.get_dummies(df1.salary)
```

```
df1 = pd.concat([df1,dummies],axis = 'columns')
df1 = df1.drop(['salary','medium'],axis='columns')
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df1[['satisfaction_level', 'average_montl
model = LogisticRegression()
model.fit(X_train,y_train)
```

```
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)
```

```
model.score(X_test,y_test)
```

```
0.7828
```

```
data = pd.read_csv('/content/HR_comma_sep.csv')
```

```
# exploration of data
data.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	

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64  
 3   average_montly_hours   14999 non-null  int64  
 4   time_spend_company     14999 non-null  int64  
 5   Work_accident          14999 non-null  int64  
 6   left                   14999 non-null  int64  
 7   promotion_last_5years  14999 non-null  int64  
 8   Department             14999 non-null  object 
 9   salary                 14999 non-null  object 
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

```
# laber encoder of data
from sklearn.preprocessing import LabelEncoder
col=['Department','salary']
label_encoder =LabelEncoder()
data['Department']= label_encoder.fit_transform(data['Department'])
data['salary']= label_encoder.fit_transform(data['salary'])
print("after the laber encoder : \n",data)
```

```
after the laber encoder :
        satisfaction_level  last_evaluation  ...  Department  salary
0                     0.38             0.53  ...           7       0
1                     0.80             0.86  ...           7       0
2                     0.11             0.88  ...           7       0
3                     0.72             0.87  ...           7       0
4                     0.37             0.52  ...           7       0
...                    ...              ...  ...         ...     ...
14994                 0.40             0.57  ...           8       0
14995                 0.37             0.48  ...           8       0
14996                 0.37             0.53  ...           8       0
14997                 0.11             0.96  ...           8       0
14998                 0.37             0.52  ...           8       0

[14999 rows x 10 columns]
```

```
# LogisticRegression of data
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion matrix,accuracy score
```

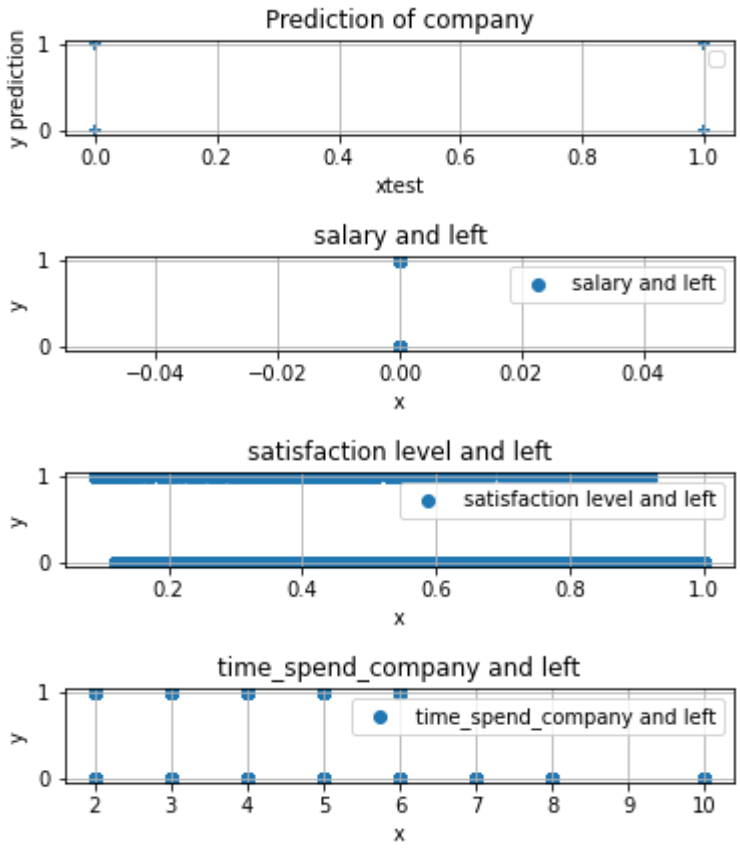
```
ft=data[['Department','satisfaction_level','salary']]
label=data['left']
xtrain,xtest,ytrain,ytest=train_test_split(ft,label)
my_model=LogisticRegression()
my_model.fit(xtrain,ytrain)
y_pred=my_model.predict(xtest) # y test
cm=confusion_matrix(ytest,y_pred)
print("confusion matrix: ",cm)
print("accuracy socre: ",accuracy_score(ytest,y_pred))
print("socre: ",my_model.score(xtrain,ytrain))
```

```
confusion matrix:  [[2659  179]
 [ 662  250]]
accuracy socre:  0.7757333333333334
socre:  0.7725131122766468
```

```
# visualization of data
import matplotlib.pyplot as plt

plt.subplot(4,1,1)
plt.scatter(ytest, y_pred, marker = '+')
plt.xlabel('xtest')
plt.ylabel('y prediction')
plt.title('Prediction of company')
plt.legend()
plt.grid()
plt.show()

plt.subplot(4,1,2)
plt.scatter(x=data['salary'], y=data['left'],label='salary and left')
plt.xlabel('x')
plt.ylabel('y')
plt.title('salary and left')
plt.legend()
plt.grid()
plt.show()

plt.subplot(4,1,3)
plt.scatter(x=data['satisfaction_level'], y=data['left'],label='satisfaction level and left')
plt.xlabel('x')
plt.ylabel('y')
plt.title('satisfaction level and left')
plt.legend()
plt.grid()
plt.show()

plt.subplot(4,1,4)
plt.scatter(x=data['time_spend_company'], y=data['left'],label='time_spend_company and left')
plt.xlabel('x')
plt.ylabel('y')
plt.title('time_spend_company and left')
```

```
plt.legend()
plt.grid()
plt.show()
```

```
No handles with labels found to put in legend.
```




```
# logistic regression model to predict Employee Attrition
#create a pipeline for Logistic Regression
from sklearn.externals import joblib
import joblib as joblib
import pickle
with open('model_save','wb') as file:
    pickle.dump(my_model,file)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/externals/joblib/__init__.py:15: FutureWa
  warnings.warn(msg, category=FutureWarning)
```

```
#load model and prediction
with open('model_save','rb') as file:
    newmodel=pickle.load(file)
# newmodel.coef_
joblib.dump(my_model,'model_joblib')
mymodel=joblib.load('model_joblib')
print("my model: ",mymodel)
print("new model: ",newmodel)
print("file is :",file)
```

```
my model:  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)
new model:  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)
file is : <_io.BufferedReader name='model_save'>
```