Job Leaving Predictor

Main Objective :

1) Determine the important/significant Features from the Dataset.
2) Encode the categorical Features.
3) Predict whether the employee will leave or not using Logistic Regression

In [116]:

```
# import all the files
import pandas as pd
import numpy as np
from sklearn import linear_model
from matplotlib import pyplot as plt
%matplotlib inline

# import sys
# !{sys.executable} -m pip install word2number

from word2number import w2n
```

Dataset is downloaded from Kaggle. Link: https://www.kaggle.com/giripujar/hr-analytics

In [2]:

```
# Dataset
data = pd.read_csv("Desktop/ML/HR.csv")
data.head()
```

Out[2]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promo
0	0.38	0.53	2	157	3	0	1	
1	0.80	0.86	5	262	6	0	1	
2	0.11	0.88	7	272	4	0	1	
3	0.72	0.87	5	223	5	0	1	
4	0.37	0.52	2	159	3	0	1	

In [3]:

```
data.shape
```

Out[3]:

```
(14999, 10)
```

There are 14999 data records and 10 features. Need to determine the dependency of the features on 'left' feature.

Data Visualization

In [10]:

```
# How many left
print('How many left : ' ,(data.left ==1).sum())
```

```
How many left :  3571
```

In [7]:

```
# Types of Salary Category
data.salary.value_counts()
```

Out[7]:

```
low       7316
medium    6446
high      1237
Name: salary, dtype: int64
```

In [9]:

```
#Types of Department
data.Department.value_counts()
```

Out[9]:

```
sales          4140
technical      2720
support        2229
IT             1227
product_mng     902
marketing       858
RandD           787
accounting      767
hr              739
management      630
Name: Department, dtype: int64
```

Average values for all columns

In [19]:

```
avg_value = data.groupby('left').mean()
avg_value
```

Out[19]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	promotio
left							
0	0.666810	0.715473	3.786664	199.060203	3.380032	0.175009	
1	0.440098	0.718113	3.855503	207.419210	3.876505	0.047326	

From the above table we can infer the following conclusions :
1) satisfaction_level : mean satisfaction_level for employees who left is lower than employees who are still there.
2) promotion_last_5years : Employees who left had a much lower mean promotion_last_5years than employees retained.
3) average_montly_hours : Mean average_montly_hours was higher for employees who left.

Features like last_evaluation , number_project and time_spend_company are almost same for both the employees who stayed or left thus we can neglect them as these are insignificant features.

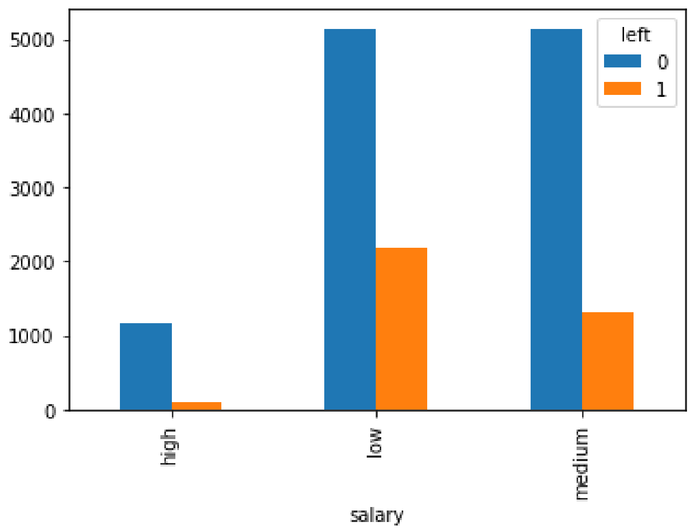
Empoyee Salary vs Left

In [11]:

```
# Bar Chart on Empoyee Salary vs Left
pd.crosstab(data.salary , data.left).plot(kind='bar')
```

Out[11]:

```
<AxesSubplot:xlabel='salary'>
```



From the bar chart it is clearly visible that the high salaried employees are least likely to leave the job whereas the low and the medium salaried employees more likely to their job.

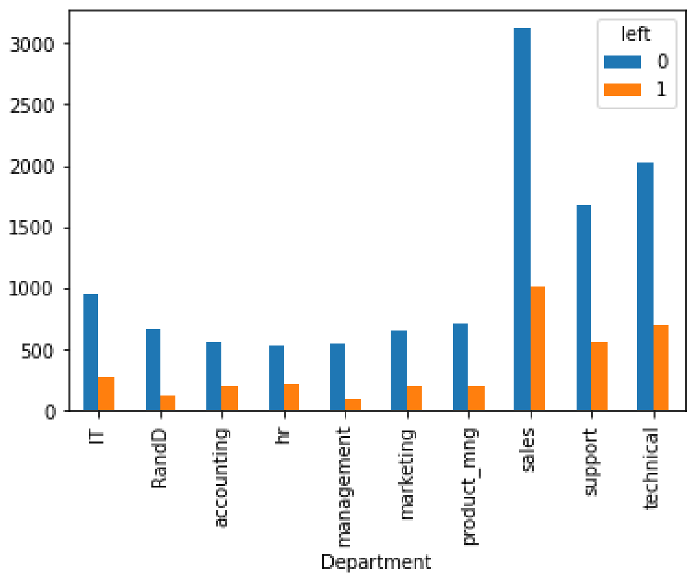
Department vs Left

In [12]:

```
# Bar Chart on Department vs Left
pd.crosstab(data.Department , data.left).plot(kind='bar')
```

Out[12]:

```
<AxesSubplot:xlabel='Department'>
```



From the bar chart we clearly see that department plays an important role in the employee leaving. Sales department has the highest employees left whereas IT , management department has the least number of employees leaving.

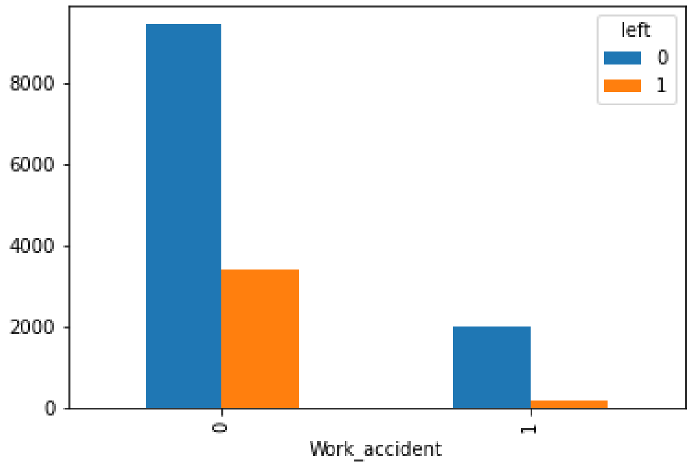
Work Accident vs Left

In [21]:

```
<br><br><br>
# Bar Chart on Work Accident vs Left
pd.crosstab( data.Work_accident, data.left  ).plot(kind='bar')
```

Out[21]:

```
<AxesSubplot:xlabel='Work_accident'>
```



Here, work accident is not the reason for the employee leaving as clearly seen above thus, we can neglect this feature as well.

Finally, satisfaction_level , promotion_last_5years, average_montly_hours and salary are the important independent feature that determine whether aan employee will leave or stay.

In [86]:

```
#Filtering only the important features
X = data[['satisfaction_level' ,'promotion_last_5years', 'average_montly_hours' , 'salar
y']]
X
```

Out[86]:

	satisfaction_level	promotion_last_5years	average_montly_hours	salary
0	0.38	0	157	low
1	0.80	0	262	medium
2	0.11	0	272	medium
3	0.72	0	223	low
4	0.37	0	159	low
...	...	...	...	...
14994	0.40	0	151	low
14995	0.37	0	160	low
14996	0.37	0	143	low
14997	0.11	0	280	low
14998	0.37	0	158	low

14999 rows × 4 columns

In [87]:

```
Y = data.left
Y
```

Out[87]:

```
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
```

Encode categorical 'salary' feature

In [27]:

```
# Method 1: Directly using get_dummies
# X1 = pd.get_dummies(X)
# X1
```

In [97]:

```
#Method 2: Encode 'salary' separately and then concatenate it with the 'X' dataset
salary_dummies = pd.get_dummies(X.salary , prefix='salary')
#Concatenate
# (NOTE: axis='columns' is very imp other wise thet will merge horizontally(table below t
able) )
concat_dummies = pd.concat([X , salary_dummies] ,axis='columns')
# Drop Salary
concat_dummies.drop('salary' , axis='columns' , inplace=True)
X1 = concat_dummies
```

In [98]:

```
X1
```

Out[98]:

	satisfaction_level	promotion_last_5years	average_montly_hours	salary_high	salary_low	salary_medium
0	0.38	0	157	0	1	0
1	0.80	0	262	0	0	1
2	0.11	0	272	0	0	1
3	0.72	0	223	0	1	0
4	0.37	0	159	0	1	0
...	...	...	...	...	...	...
14994	0.40	0	151	0	1	0
14995	0.37	0	160	0	1	0
14996	0.37	0	143	0	1	0
14997	0.11	0	280	0	1	0
14998	0.37	0	158	0	1	0

14999 rows × 6 columns

Logistic Regression

In [107]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [108]:

```
# Split data
X_train , X_test, Y_train , Y_test = train_test_split(X1, Y, test_size=0.1)
```

In [109]:

```
model = LogisticRegression()
model.fit(X_train , Y_train)
```

Out[109]:

```
LogisticRegression()
```

In [115]:

```
results = model.predict(X_test)
#Stored results as DataFrame
results_df = pd.DataFrame(results)
results
```

Out[115]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [111]:

```
print('R2 Score for the model using Logistic Regression :' , model.score(X_test , Y_test)
)
```

```
R2 Score for the model using Logistic Regression : 0.7833333333333333
```

Conclusion

Successfully designed an Employee Leaving Predictor based on Logistic Regression with an accuracy of 78.33%. The independent features considered here were satisfaction_level , promotion_last_5years, average_montly_hours and salary.