

classification of employee resignation

importing libraries

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

In [2]: df = pd.read_csv('Downloads/HR_comma_sep.csv')
df

Out[2]:
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	Department	salary
0	0.38	0.53	2	157	3	0	1	0	sales	low
1	0.80	0.86	5	262	6	0	1	0	sales	medium
2	0.11	0.88	7	272	4	0	1	0	sales	medium
3	0.72	0.87	5	223	5	0	1	0	sales	low
4	0.37	0.52	2	159	3	0	1	0	sales	low
...	...	...	...	...	...	...	...	...	...	...
14994	0.40	0.57	2	151	3	0	1	0	support	low
14995	0.37	0.48	2	160	3	0	1	0	support	low
14996	0.37	0.53	2	143	3	0	1	0	support	low
14997	0.11	0.96	6	280	4	0	1	0	support	low
14998	0.37	0.52	2	158	3	0	1	0	support	low

14999 rows x 10 columns

```
In [3]: df.columns

Out[3]:
```

Index(['satisfaction\_level', 'last\_evaluation', 'number\_project', 'average\_montly\_hours', 'time\_spend\_company', 'Work\_accident', 'left', 'promotion\_last\_5years', 'Department', 'salary'], dtype='object')

data exploration and visualization

```
In [4]: left = df[df.left==1]
left.shape

Out[4]: (3571, 10)

In [5]: retained = df[df.left==0]
retained.shape

Out[5]: (11428, 10)
```

average all the columns of left and retained

```
In [6]: df.groupby('left').mean()

Out[6]:
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	promotion_last_5years
left							
0	0.666810	0.715473	3.786664	199.060203	3.380032	0.175009	0.026251
1	0.440098	0.718113	3.855503	207.419210	3.876505	0.047326	0.005321

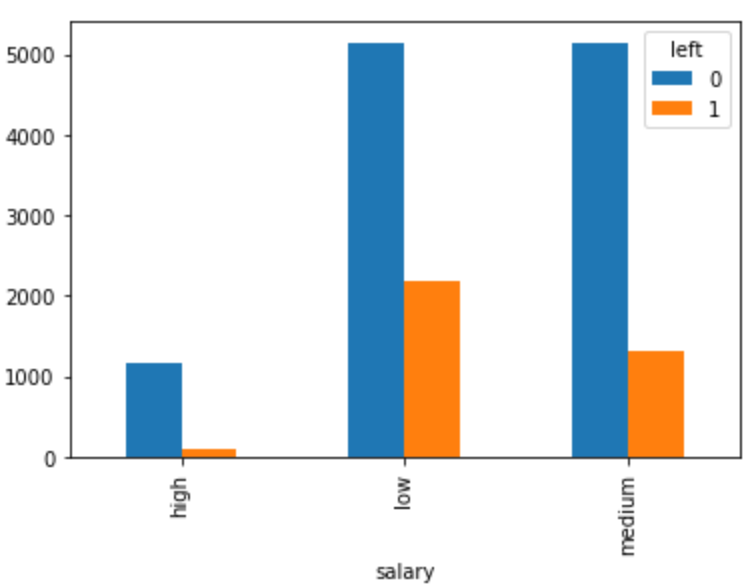
From above table we can draw following conclusions,

**Satisfaction Level:** Satisfaction level seems to be relatively low (0.44) in employees leaving the firm vs the retained ones (0.66) **Average Monthly Hours:** Average monthly hours are higher in employees leaving the firm (199 vs 207) **Promotion Last 5 Years:** Employees who are given promotion are likely to be retained at firm \*\*remaing are closely related as they cant be separated and concluded

impact of salary on employee retension

```
In [7]: pd.crosstab(df.salary,df.left).plot(kind='bar')

Out[7]: <AxesSubplot: xlabel='salary'>
```

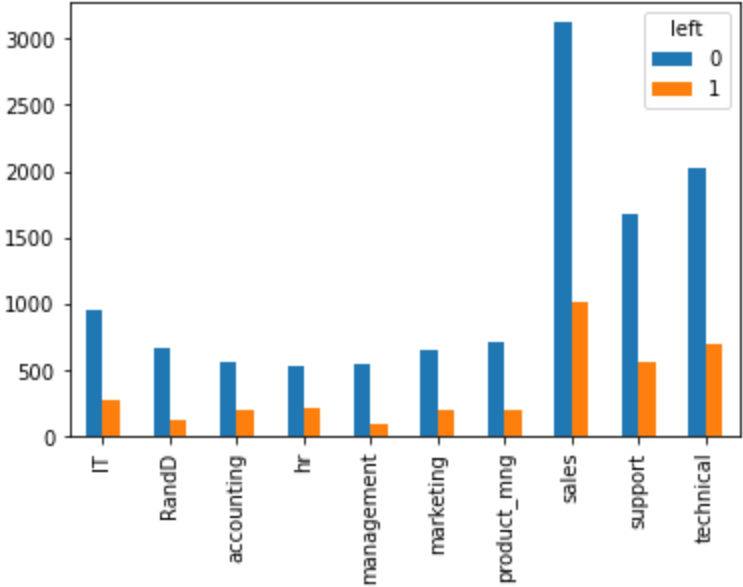


In [8]: *#Above bar chart shows employees with high salaries are likely to not leave the company*

Department wise employee retention rate

```
In [9]: pd.crosstab(df.Department,df.left).plot(kind='bar')

Out[9]: <AxesSubplot: xlabel='Department'>
```



In [10]: *#From above chart there seem to be some impact of department on employee retention but it is not major hence we will ignore department in our analysis*

```
In [11]: subdf = df[['satisfaction_level', 'average_montly_hours', 'promotion_last_5years', 'salary']]
subdf.head()

Out[11]:
```

	satisfaction_level	average_montly_hours	promotion_last_5years	salary
0	0.38	157	0	low
1	0.80	262	0	medium
2	0.11	272	0	medium
3	0.72	223	0	low
4	0.37	159	0	low

```
In [12]: #convert salary to numeric using one hot encoding

In [13]: salary_dummies = pd.get_dummies(subdf.salary, prefix='salary')

In [14]: df_with_dummies = pd.concat([subdf, salary_dummies], axis='columns')

In [15]: df_with_dummies.head()

Out[15]:
```

	satisfaction_level	average_montly_hours	promotion_last_5years	salary	salary_high	salary_low	salary_medium
0	0.38	157	0	low	0	1	0
1	0.80	262	0	medium	0	0	1
2	0.11	272	0	medium	0	0	1
3	0.72	223	0	low	0	1	0
4	0.37	159	0	low	0	1	0

```
In [16]: df_with_dummies.drop('salary', axis='columns', inplace=True)
df_with_dummies.head()

Out[16]:
```

	satisfaction_level	average_montly_hours	promotion_last_5years	salary_high	salary_low	salary_medium
0	0.38	157	0	0	1	0
1	0.80	262	0	0	0	1
2	0.11	272	0	0	0	1
3	0.72	223	0	0	1	0
4	0.37	159	0	0	1	0

```
In [17]: X = df_with_dummies
X.head()

Out[17]:
```

	satisfaction_level	average_montly_hours	promotion_last_5years	salary_high	salary_low	salary_medium
0	0.38	157	0	0	1	0
1	0.80	262	0	0	0	1
2	0.11	272	0	0	0	1
3	0.72	223	0	0	1	0
4	0.37	159	0	0	1	0

```
In [18]: y = df.left
y

Out[18]:
```

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

splitting of dataset

```
In [19]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.3)
```

importing required model

```
In [20]: from sklearn.linear_model import LogisticRegression
model = LogisticRegression()

In [21]: model.fit(X_train, y_train)

Out[21]: LogisticRegression()

In [26]: y_pred = model.predict(X_test)
y_pred

Out[26]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)

In [27]: model.score(X_test, y_test)

Out[27]: 0.7790476190476191

In [ ]:
```

confusion matrix

```
In [28]: from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm

Out[28]:
```

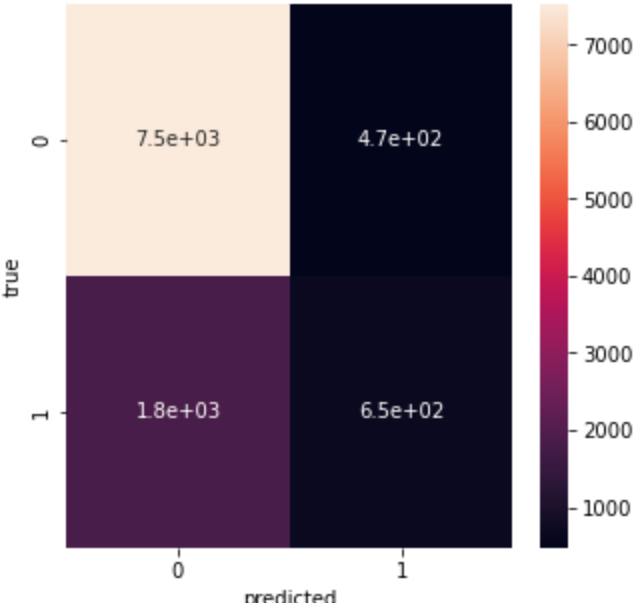
array([[7527, 474],  
[1846, 653]], dtype=int64)

```
In [29]: import seaborn as sns
import matplotlib.pyplot as plt

In [31]: plt.figure(figsize=(5,5))
sns.heatmap(data=cm, annot=True)
plt.xlabel('predicted')
plt.ylabel('true')

Out[31]:
```

Text(24.0, 0.5, 'true')



```
In [ ]:
```