Exercise Download employee retention dataset from here: https://www.kaggle.com/giripujar/hr-analytics (https://www.kaggle.com/giripujar/hr-analytics).

Now do some exploratory data analysis to figure out which variables have direct and clear impact on employee retention (i.e. whether they leave the company or continue to work) Plot bar charts showing impact of employee salaries on retention Plot bar charts showing corelation between department and employee retention Now build logistic regression model using variables that were narrowed down in step 1 Measure the accuracy of the model

In [3]:
```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

In [20]:
```
df = pd.read_csv(r"C:\Users\sagar kumar\Downloads\HR_comma_sep.csv")
print(df.shape)
df.head()
```

```
(14999, 10)
```

Out[20]:

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

In [16]:
```
left = df[df.left==1]
left.shape
```

Out[16]:
```
(3571, 10)
```

In [17]:
```
retained = df[df.left==0]
retained.shape
```

Out[17]:
```
(11428, 10)
```

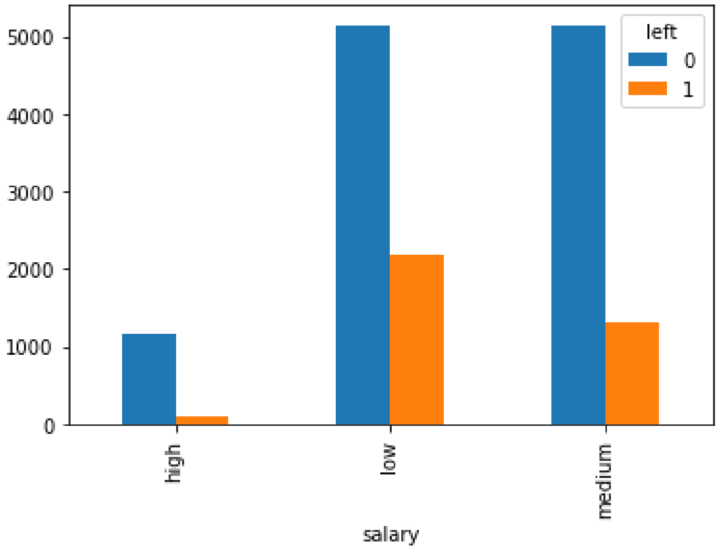
In [21]:
```
df.groupby('left').mean()
```

Out[21]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accide
left						
0	0.666810	0.715473	3.786664	199.060203	3.380032	0.1750
1	0.440098	0.718113	3.855503	207.419210	3.876505	0.0473

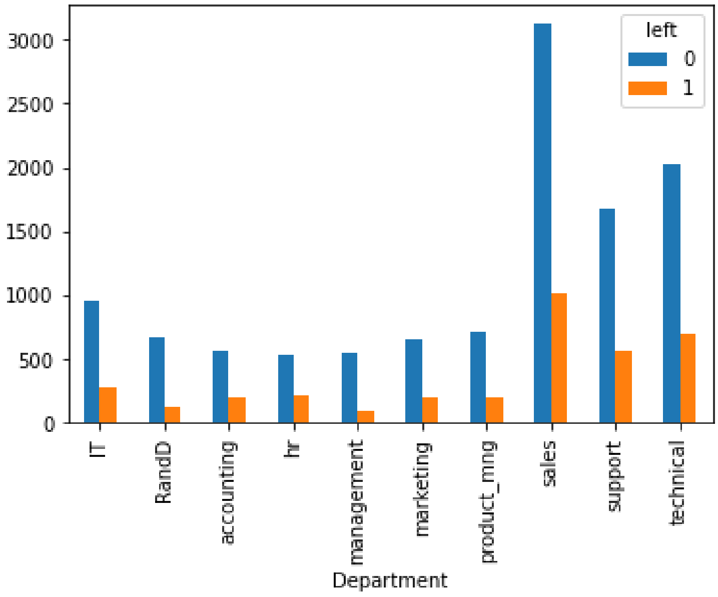
In [22]:
```
pd.crosstab(df.salary,df.left).plot(kind='bar')
```

Out[22]:
```
<AxesSubplot:xlabel='salary'>
```



In [23]:
```
pd.crosstab(df.Department,df.left).plot(kind='bar')
```

Out[23]:
```
<AxesSubplot:xlabel='Department'>
```



In [24]:

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

Out[24]:

	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 [25]:

```
salary_dummies = pd.get_dummies(subdf.salary, prefix="salary")
df_with_dummies = pd.concat([subdf,salary_dummies],axis='columns')
df_with_dummies.head()
```

Out[25]:

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

In [28]:

```
df_with_dummies.drop('salary',axis='columns',inplace=True)
df_with_dummies.head()
```

Out[28]:

	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 [29]:

```
X = df_with_dummies
X.head()
```

Out[29]:

	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 [32]:
```
y = df.left
y.head()
```

Out[32]:
```
0    1
1    1
2    1
3    1
4    1
Name: left, dtype: int64
```

In [33]:
```
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)
```

In [34]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [35]:
```
model.fit(X_train, y_train)
```

Out[35]:
```
LogisticRegression()
```

In [36]:
```
model.predict(X_test)
```

Out[36]:
```
array([0, 0, 0, ..., 0, 1, 0], dtype=int64)
```

In [37]:
```
model.score(X_test,y_test)
```

Out[37]:
```
0.772
```

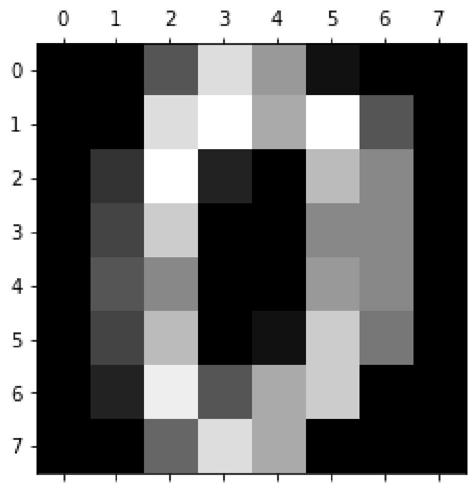
Logistic Regression: Multiclass Classification

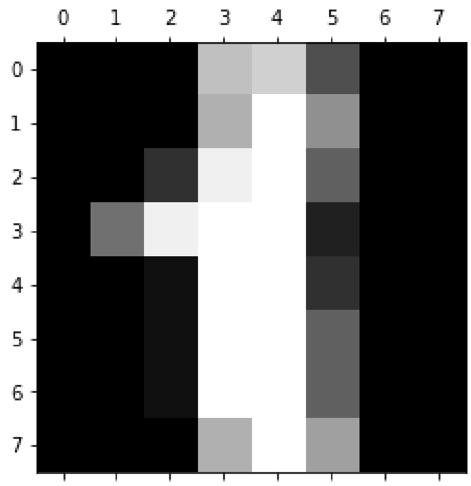
In [38]:
```
from sklearn.datasets import load_digits
%matplotlib inline
import matplotlib.pyplot as plt
digits = load_digits()
```

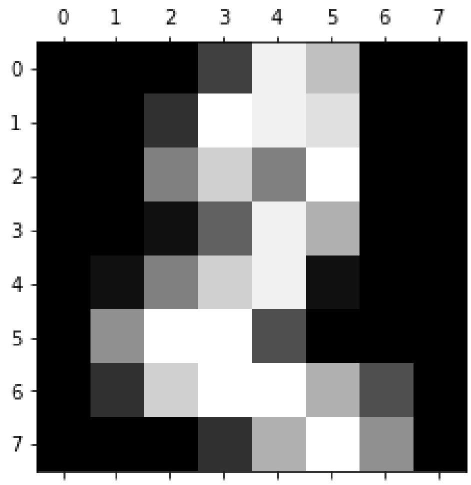
In [39]:

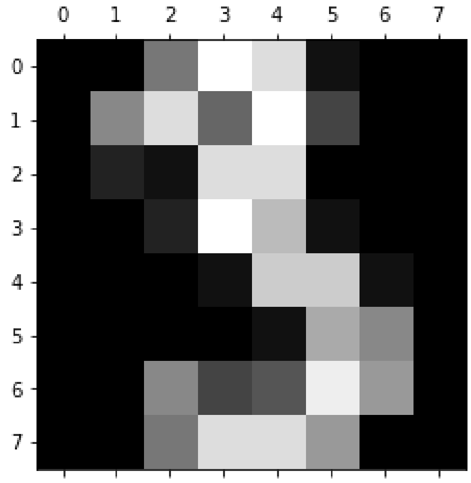
```
plt.gray()
for i in range(5):
    plt.matshow(digits.images[i])
```

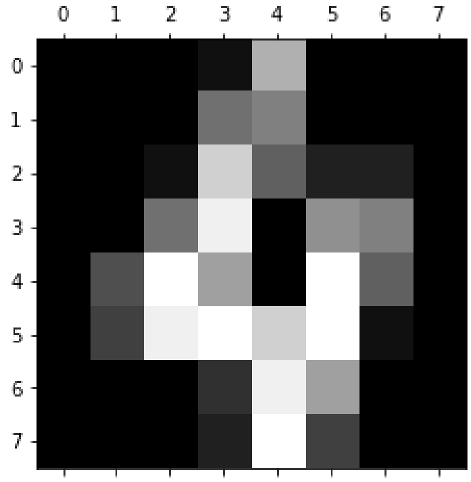
```
<Figure size 432x288 with 0 Axes>
```











In [40]:
```
dir(digits)
```

Out[40]:
```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_names']
```

In [41]:
```
digits.data[0]
```

Out[41]:
```
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13., 15., 10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,  4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,  8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,  5.,
       10., 12.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.])
```

In [42]:
```

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [43]:
```
from sklearn.model_selection import train_test_split
```

In [44]:
```
X_train, X_test, y_train, y_test = train_test_split(digits.data,digits.target, test_size=0.2
```

In [45]:
```
model.fit(X_train, y_train)
```

```
C:\Users\sagar kumar\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: Co
nvergenceWarning: lbfgs failed to converge (status=1):
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 (https://scikit-learn.org/s
table/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (http
s://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[45]:
```
LogisticRegression()
```

In [46]:
```
model.score(X_test, y_test)
```

Out[46]:
```
0.9694444444444444
```

In [48]:
```
model.predict(digits.data[0:5])
```

Out[48]:
```
array([0, 1, 2, 3, 4])
```

Confusion Matrix

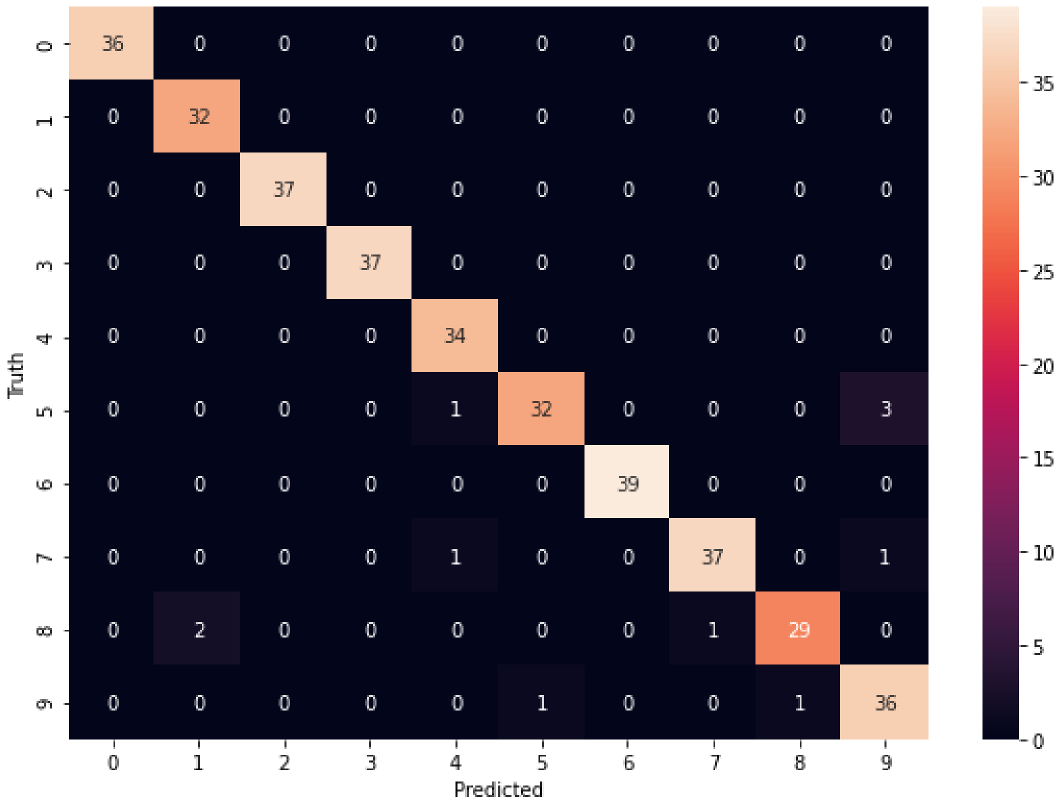
In [49]:
```
y_predicted = model.predict(X_test)
```

In [50]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_predicted)
cm
```

Out[50]:
```
array([[36,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 32,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 37,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 37,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0, 34,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  1, 32,  0,  0,  0,  3],
       [ 0,  0,  0,  0,  0,  0, 39,  0,  0,  0],
       [ 0,  0,  0,  0,  1,  0,  0, 37,  0,  1],
       [ 0,  2,  0,  0,  0,  0,  0,  1, 29,  0],
       [ 0,  0,  0,  0,  0,  1,  0,  0,  1, 36]], dtype=int64)
```

```
In [51]: import seaborn as sn
         plt.figure(figsize = (10,7))
         sn.heatmap(cm, annot=True)
         plt.xlabel('Predicted')
         plt.ylabel('Truth')

Out[51]: Text(69.0, 0.5, 'Truth')
```



```
In [ ]:
```