Cross Validation

In [70]:
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.linear_model import LinearRegression
from sklearn import datasets
from numpy import mean
from numpy import absolute
from numpy import sqrt
import pandas as pd
```

We will be using a dataset from the sklearn library using the load_boston(). It is a datset of Boston with some features. We can convert the dataset to a dataframe of the panda library which is shown below.

In [71]:
```
data = datasets.load_boston()
print(data.DESCR)
df = pd.DataFrame(data=data.data, columns=data.feature_names)
df.head()
df.dtypes
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (at
tribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 s
q.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0
otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
```

```
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of black peopl
e by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


This dataset was taken from the StatLib library which is maintained at Carnegi
e Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that
address regression
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential D
ata and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In
Proceedings on the Tenth International Conference of Machine Learning, 236-243
, University of Massachusetts, Amherst. Morgan Kaufmann.
```

Out[71]:

```
CRIM       float64
ZN         float64
INDUS      float64
CHAS       float64
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD        float64
TAX        float64
PTRATIO    float64
B          float64
LSTAT      float64
dtype: object
```

There are different type of cross validation in python

- K fold Cross validation
- LOOCV or leave one out CV
- Stratified Cross Validation

We can use the function KFold for the K fold crooss validation in python from the sklearn library. n_splits is the number of folds specifier. shuffle=TRUE helps the data to shuffle into batches. random_state is basically like setting seed we can set to an integer to make results identical. We will find out the RMSE using this.

In [72]:
```
x = df[['LSTAT']]
y= df['CRIM']

#define cross-validation method to use
cv = KFold(n_splits=10, random_state=1, shuffle=True)

#build multiple linear regression model
model = LinearRegression()

#use k-fold CV to evaluate model
scores = cross_val_score(model, x, y, scoring='neg_mean_squared_error',
                         cv=cv, n_jobs=-1)

#view RMSE
sqrt(mean(absolute(scores)))
```

Out[72]:
```
7.674771467207866
```

Now we can split the data in test and train and then do cross validation

In [73]:
```
#Split Data set
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7,random_stat
model= LinearRegression()
mymodel = model.fit(x_train,y_train)
```

In [74]:
```
#KFOLD CV
# 10 Fold Cv using the cv specs from before
scores = cross_val_score(mymodel, x_train, y_train, scoring='neg_mean_squared
                         cv=cv, n_jobs=-1)
sqrt(mean(absolute(scores)))
```

Out[74]:
```
7.5648059011571025
```

Now let's print the scores in the test dataset doing prediction

In [75]:
```
predictions = cross_val_predict(model,x_test,y_test)
predictions
scores_test = cross_val_score(model, x_test, y_test, scoring='neg_mean_square
                              cv=cv, n_jobs=-1)
sqrt(mean(absolute(scores_test)))
```

Out[75]:
```
8.099717227477791
```

KNN Classifier and Cross Validation

In [76]:
```
df = pd.read_csv('Default_Fin.csv')
df.head(n=6)
```

Out[76]:

	Index	Employed	Bank Balance	Annual Salary	Defaulted?
0	1	1	8754.36	532339.56	0
1	2	0	9806.16	145273.56	0
2	3	1	12882.60	381205.68	0
3	4	1	6351.00	428453.88	0
4	5	1	9427.92	461562.00	0
5	6	0	11035.08	89898.72	0

To use Knnclassifier we need to use the KNeighborsClassifier from the sklearn Library.

In [77]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
```

In [78]:
```
x=df[['Annual Salary','Bank Balance']]
y= df['Defaulted?']

x_train, x_test, y_train, y_test = train_test_split(x,y,random_state =5)
knnclassifier = KNeighborsClassifier(n_neighbors=5)
knnclassifier.fit(x_train,y_train)##fitting the model using the train dataset
y_pred = knnclassifier.predict(x_test)##predicting using the test dataset
metrics.accuracy_score(y_test,y_pred)##accuracy metrics is used for the class
```

Out[78]:
```
0.966
```

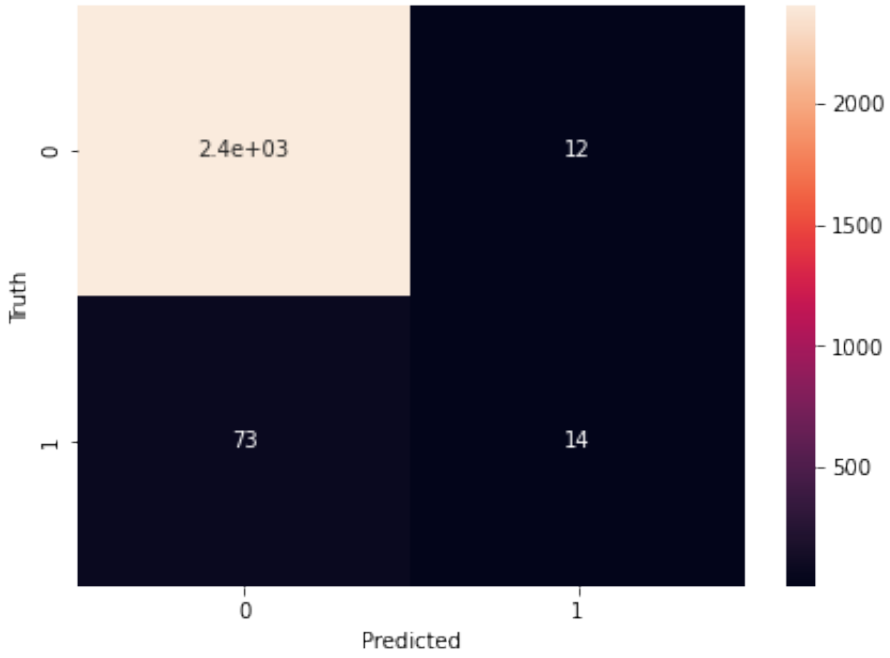
Let's Plot a consfusion matrix

In [79]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[79]:
```
array([[2401,   12],
       [  73,   14]])
```

In [80]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[80]:
```
Text(42.0, 0.5, 'Truth')
```



Let's use the CV specs from before and use cross validation in this

In [84]:
```
#define cross-validation method to use
cv = KFold(n_splits=10, random_state=1, shuffle=True)
scores_test =cross_val_score(knnclassifier, x_train, y_train, cv=cv, scoring =

mean(scores_test)
```

Out[84]:
```
0.9684000000000001
```

In [88]:
```
predictions = cross_val_predict(knnclassifier,x_test,y_test)
predictions
scores_test = cross_val_score(knnclassifier, x_test, y_test, scoring='accurac
                              cv=cv)
sqrt(mean(absolute(scores_test)))
```

Out[88]:
```
0.9834632682515396
```

Finding optimal K values

We can find the optimal k values using a function called GridSearchCV and using the n_neighbors tuning.

In [82]:
```
from sklearn.model_selection import GridSearchCV
n_neighbors = list(range(1,40))

hyperparam = dict(n_neighbors = n_neighbors)# convert to dictionary for use i

knn_new = KNeighborsClassifier()
clf = GridSearchCV(knn_new,hyperparam, cv=cv)
best_model = clf.fit(x_train,y_train)
best_model.best_estimator_.get_params()['n_neighbors']
```

Out[82]:
```
2
```

Now we can see that the best k for the training dataset is k= 2 Let's create the final model with 2 and see the test accuracy.

In [83]:
```
knnclassifier = KNeighborsClassifier(n_neighbors=2)
knnclassifier.fit(x_train,y_train)##fitting the model using the train dataset
y_pred = knnclassifier.predict(x_test)##predicting using the test dataset
metrics.accuracy_score(y_test,y_pred)##accuracy metrics is used for the class
```

Out[83]:
```
0.9648
```