Week 5:

Model Training:

1. Holdout
2. K-Fold cross validation
3. Bootstrap Sampling

In [1]:

```
import pandas as pd
import numpy as np
ColumnNames=['Hours','Calories', 'Weight']
DataValues=[[  1.0,   2500,   95],
            [  2.0,   2000,   85],
            [  2.5,   1900,   83],
            [  3.0,   1850,   81],
            [  3.5,   1600,   80],
            [  4.0,   1500,   78],
            [  5.0,   1500,   77],
            [  5.5,   1600,   80],
            [  6.0,   1700,   75],
            [  6.5,   1500,   70]]
#Create the Data Frame
GymData=pd.DataFrame(data=DataValues,columns=ColumnNames)
GymData.head()

#Separate Target Variable and Predictor Variables
TargetVariable='Weight'
Predictors=['Hours','Calories']
X=GymData[Predictors].values
y=GymData[TargetVariable].values

#### Bootstrapping ####
############################################################
# Creating empty list to hold accuracy values
AccuracyValues=[]
n_times=5

## Performing bootstrapping
for i in range(n_times):
    #Split the data into training and testing set
    from sklearn.model_selection import train_test_split
    # Chaning the seed value for each iteration
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

    ############################################################

    ###### Single Decision Tree Regression in Python #######
    from sklearn import tree
    #choose from different tunable hyper parameters
    RegModel = tree.DecisionTreeRegressor(max_depth=3,criterion='squared_error')

    #Creating the model on Training Data
    DTree=RegModel.fit(X_train,y_train)
    prediction=DTree.predict(X_test)
```

```
    #Measuring accuracy on Testing Data
    Accuracy=100- (np.mean(np.abs((y_test - prediction) / y_test)) * 100)

    # Storing accuracy values
    AccuracyValues.append(np.round(Accuracy))

################################################
# Result of all bootstrapping trials
print(AccuracyValues)

# Final accuracy
print('Final average accuracy',np.mean(AccuracyValues))
```

```
[94.0, 95.0, 98.0, 98.0, 93.0]
Final average accuracy 95.6
```

In [2]:

```
########################################
####### K-fold cross validation #######
# Defining a custom function to calculate accuracy
# Make sure there are no zeros in the Target variable if you are using MAPE
def Accuracy_Score(orig,pred):
    MAPE = np.mean(100 * (np.abs(orig-pred)/orig))
    #print('#'*70,'Accuracy:', 100-MAPE)
    return(100-MAPE)

# Custom Scoring MAPE calculation
from sklearn.metrics import make_scorer
custom_Scoring=make_scorer(Accuracy_Score, greater_is_better=True)

# Importing cross validation function from sklearn
from sklearn.model_selection import cross_val_score


###### Single Decision Tree Regression in Python #######
from sklearn import tree
#choose from different tunable hyper parameters
RegModel = tree.DecisionTreeRegressor(max_depth=3,criterion='squared_error')

# Running 10-Fold Cross validation on a given algorithm
# Passing full data X and y because the K-fold will split the data and automatically ch
Accuracy_Values=cross_val_score(RegModel, X , y, cv=10, scoring=custom_Scoring)
print('\nAccuracy values for 10-fold Cross Validation:\n',Accuracy_Values)
print('\nFinal Average Accuracy of the model:', round(Accuracy_Values.mean(),2))
```

```
Accuracy values for 10-fold Cross Validation:
 [89.47368421 95.29411765 97.59036145 97.5308642  98.75       98.07692308
 96.42857143 95.83333333 94.4        92.85714286]

Final Average Accuracy of the model: 95.62
```

In [3]:

```
# hold out

#Split the data into training and testing set
from sklearn.model_selection import train_test_split
    # Chaning the seed value for each iteration
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0

    ###########################################################
    ###########################################################
```

```
    ###### Single Decision Tree Regression in Python #######
from sklearn import tree
    #choose from different tunable hyper parameters
RegModel = tree.DecisionTreeRegressor(max_depth=3,criterion='squared_error')

    #Creating the model on Training Data
DTree=RegModel.fit(X_train,y_train)
prediction=DTree.predict(X_test)

    #Measuring accuracy on Testing Data
Accuracy=100- (np.mean(np.abs((y_test - prediction) / y_test)) * 100)
print(Accuracy)
```

```
95.24356248523506
```

In [ ]: