Evaluate the Trained Model

- Predicting on validation set

```
[37] pred_y = model.predict( test_X )
```

```
[38] pred_y
```

```
array([279828.40245195, 272707.22768622, 215737.82956041, 237101.35385759,
       295851.04567483, 247070.99852961, 226419.591709  , 308313.10151485,
       254904.29077191, 295494.98693655])
```

```
test_y
```

```
6     260000
36    177600
37    236000
28    360000
43    250000
49    300000
5     300000
33    330000
20    120000
42    300000
Name: Salary, dtype: int64
```

Finding MSE and R2 score

- R2Score: A relative metric in which the higher the value, the better the model's fit. In essence, this metric represents how much of the variance between predicted and actual label values the model is able to explain.

```
from sklearn.metrics import r2_score, mean_squared_error
```

```
np.abs(r2_score(test_y, pred_y))
```

```
0.15664584974230378
```

So, the model only explains 15.6% of the variance in the validation set.

```
import numpy
```

```
np.sqrt(mean_squared_error(test_y, pred_y))
```

```
73458.043483468937
```

SLR using Decision tree

- The regression method is good, if the relationship between ind., and dep., variables are linear,
- For non-linear relationship we can used decision tree
- It uses a tree-based approach in which the features in the dataset are examined in a series of evaluations,
- Each of which results in a *branch* in a *decision tree* based on the feature value. At the end of each series of branches are leaf-nodes with the predicted label value based on the feature values.

SLR using Decision tree

- Here using decision tree algo from skit learn instead of linear regression

```
# Train the model
from sklearn.tree import DecisionTreeRegressor

# Fit a linear regression model on the training set
model = DecisionTreeRegressor().fit(train_X, train_y)
print (model)
```

```
DecisionTreeRegressor()
```

SLR using Decision tree

- It result in improvement of R2 score

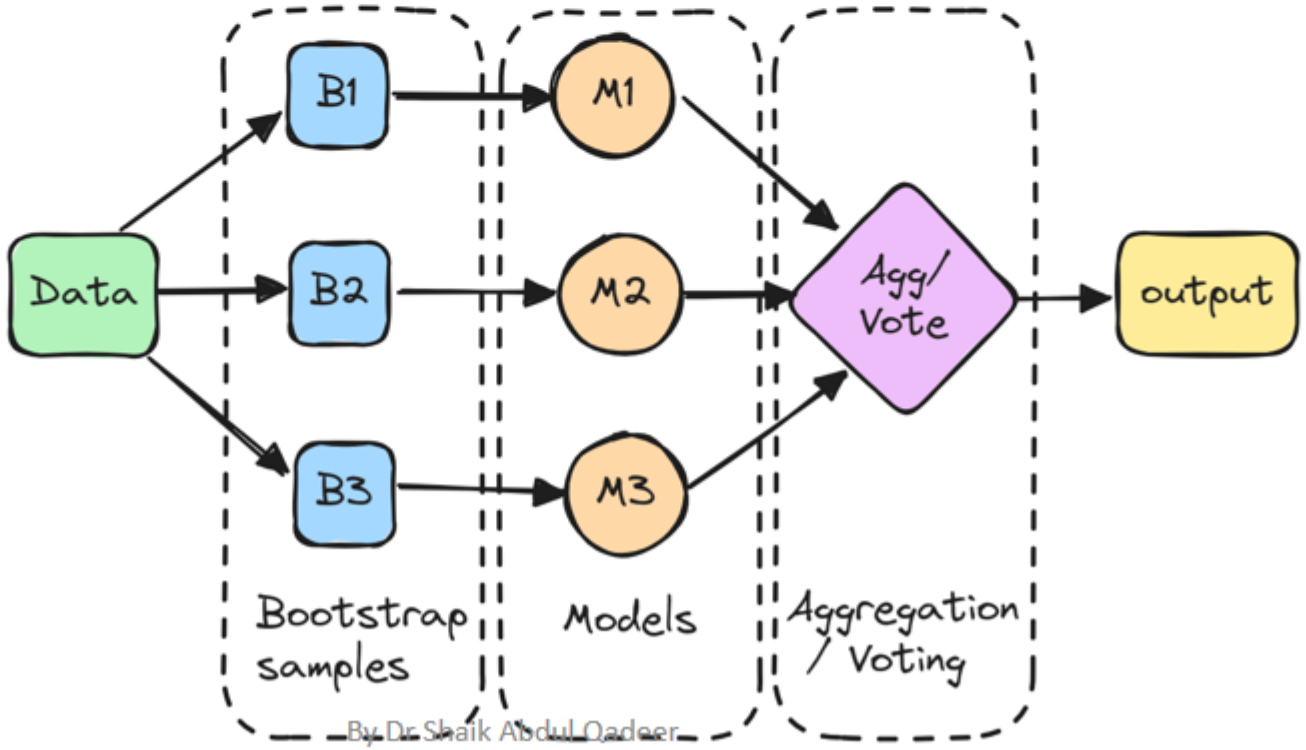
```
[14] from sklearn.metrics import r2_score, mean_squared_error
     np.abs(r2_score(test_y, pred_y))

     0.5676966322534405
```

SLR using Ensemble algorithm(ENA)

- ENA construct not just one decision tree, but a large number of trees, allowing better predictions on more complex data.
- It work by combining multiple base estimators to produce an optimal model, by applying **an aggregate function to a collection of base models** (referred to a ***bagging***)
- It work by combining multiple base estimators to produce an optimal model, by building **a sequence of models that build on one another to improve predictive performance** (referred to as ***boosting***).

SLR using Ensemble algorithm(ENA)

- It work by combining multiple base estimators to produce an optimal model, by applying **an aggregate function to a collection of base models** (referred to a ***bagging***)



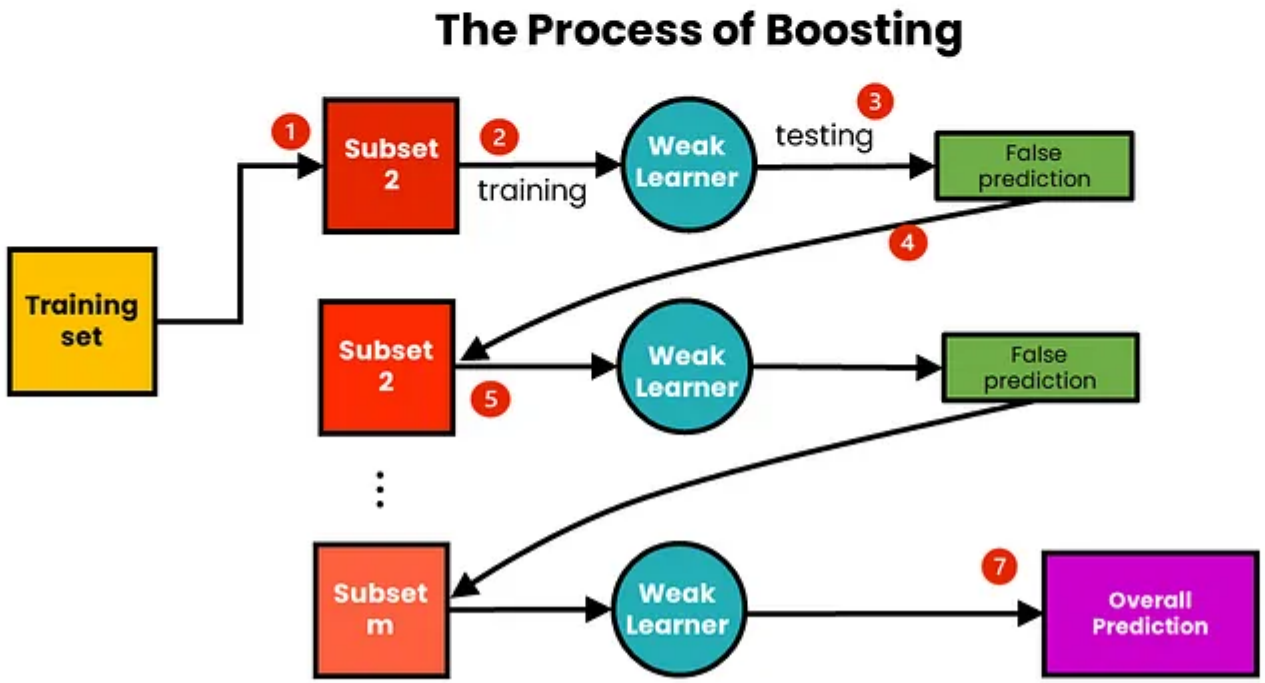
SLR using Ensemble algorithm(ENA)

- ENA construct not just one decision tree, but a large number of trees, allowing better predictions on more complex data is bagging. Ex: Random forest

```
# Train the model
from sklearn.ensemble import RandomForestRegressor

# Fit a  RandomForestRegressor model as ensemble Algorithm on the training set
model = RandomForestRegressor().fit(train_X, train_y)
print (model)

pred_y = model.predict( test_X )
```

SLR using Ensemble algorithm(ENA)

- It work by combining multiple base estimators to produce an optimal model, **by building a sequence of models that build on one another to improve predictive performance** (referred to as ***boosting***).
- Ex: Gradient boosting, cat boosting and XG boosting



SLR using Boosting

- Ex: Gradient boosting,

```
from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y = train_test_split( X ,
                                                     Y,
                                                    train_size = 0.8,
                                                    random_state = 100 )

# Train the model
from sklearn.ensemble import GradientBoostingRegressor

# Fit a GradientBoostingRegressor algorithm model on the training set
#TRy :ADAboost.XGboost
model =  GradientBoostingRegressor().fit(train_X, train_y)
print (model)

pred_y = model.predict( test_X )
```

Multiple regression

- Same process