ASSIGNMENT – 08

TOPIC: RANDOM FOREST

Q.1. Check for MAE, MSE, R2-SCORE, RMSE based on different n_estimators.

Q.2. Check for minimum RMSE and returning best n_estimator.

Ans. 1)

1. **Mean absolute error:** It is the mean of the absolute value of the errors. This is the easiest of the metrics to understand since it's just average error.

2. **Mean Squared Error (MSE):** MSE is the mean of the squared error. It's more popular than Mean absolute error because the focus is geared more towards large errors. This is due to the squared term exponentially increasing larger errors in comparison to smaller ones.

3. **Root Mean Squared Error (RMSE):** RMSE is the standard deviation of the errors which occur when a prediction is made on a dataset. This is the same as MSE (Mean Squared Error) but the root of the value is considered while determining the accuracy of the model.

4. **R2-Score:** R-squared is not error, but is a popular metric for accuracy of your model. It represents how close the data are to the fitted regression line. **The higher the R-squared, the better the model fits your data. Best possible score is 1.0 and it can be negative** (because the model can be arbitrarily worse).

5. **n_estimators:** The number of trees in the forest. Since **Random Forest** is an ensemble method comprising of creating multiple **decision** trees, this parameter is used to control the number of trees to be used in the process.

Check for MAE, MSE, R2-score, RMSE based on different n_estimators

In [13]:

```
#Check for MAE, MSE, R2-score, RMSE based on different n_estimators
import random as rd
from sklearn.metrics import r2_score, mean_squared_error
i = rd.sample(range(100),10)
x,a,b,c,d = [],[],[],[],[]
for e in i:
    from sklearn.ensemble import RandomForestRegressor
    reg = RandomForestRegressor(n_estimators=e, random_state = 0)
    reg.fit(X,y)
    pred = reg.predict(X)
    x.append(e)
    a.append(np.mean(np.absolute(pred - y)).round(decimals = 2))
    b.append(np.sqrt(mean_squared_error(y, pred)).round(decimals = 2))
    c.append(r2_score(pred , y).round(decimals = 2))
    d.append(np.sqrt(np.mean((pred - y) ** 2)).round(decimals = 2))

    rand_check = pd.DataFrame({
        'n_estimators': np.array(x).flatten(),
        'MAE': np.array(a).flatten(),
        'MSE': np.array(b).flatten(),
        'R2-Score': np.array(c).flatten(),
        'RMSE': np.array(d).flatten(),
    })
rand_check
```

Out[13]:

	n_estimators	MAE	MSE	R2-Score	RMSE
0	35	26814.29	73346.44	0.89	73346.44
1	9	33000.00	60696.57	0.93	60696.57
2	80	26437.50	65478.66	0.92	65478.66
3	81	27283.95	67474.30	0.91	67474.30
4	36	26430.56	71351.60	0.90	71351.60
5	60	25658.33	66271.58	0.92	66271.58
6	63	25682.54	65660.94	0.92	65660.94
7	75	26073.33	66626.38	0.92	66626.38
8	91	26159.34	69082.49	0.91	69082.49
9	95	26915.79	71190.65	0.90	71190.65

Check for minimum RMSE and returning best best n_estimator

In [14]:

```
#Check for minimum RMSE and returning best best n_estimator
#p = np.where(d == np.amin(d))
p= d.index(min(d))
print("Best n_estimator value: ", x[p])
evaluation = pd.DataFrame({
    'n_estimators': np.array(x[p]).flatten(),
    'MAE': np.array(a[p]).flatten(),
    'MSE': np.array(b[p]).flatten(),
    'R2-Score': np.array(c[p]).flatten(),
    'RMSE': np.array(d[p]).flatten(),
})
evaluation
```

```
Best n_estimator value:  9
```

Out[14]:

	n_estimators	MAE	MSE	R2-Score	RMSE
0	9	33000.0	60696.57	0.93	60696.57