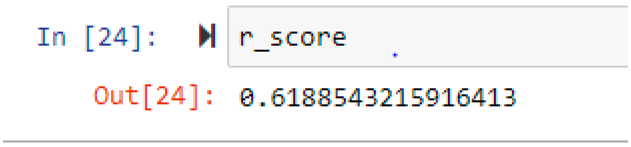
ASSIGNMENT- REGRESSION

PROBLEM STATEMENT:

A client’s requirement is, he wants to predict the concrete strength based on the several parameters. The Client has provided the dataset of the same.

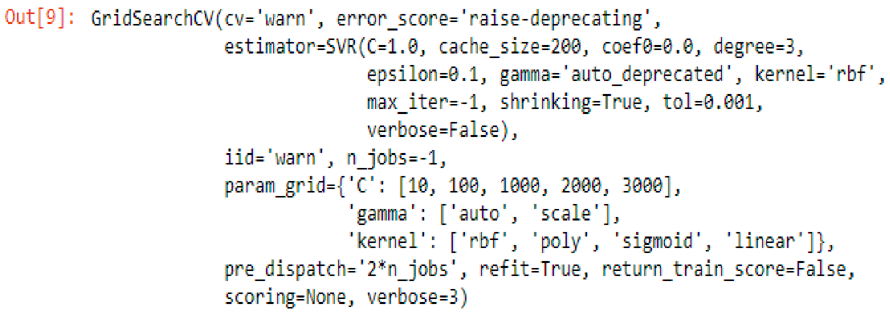
MULTIPLE LINEAR REGRESSION



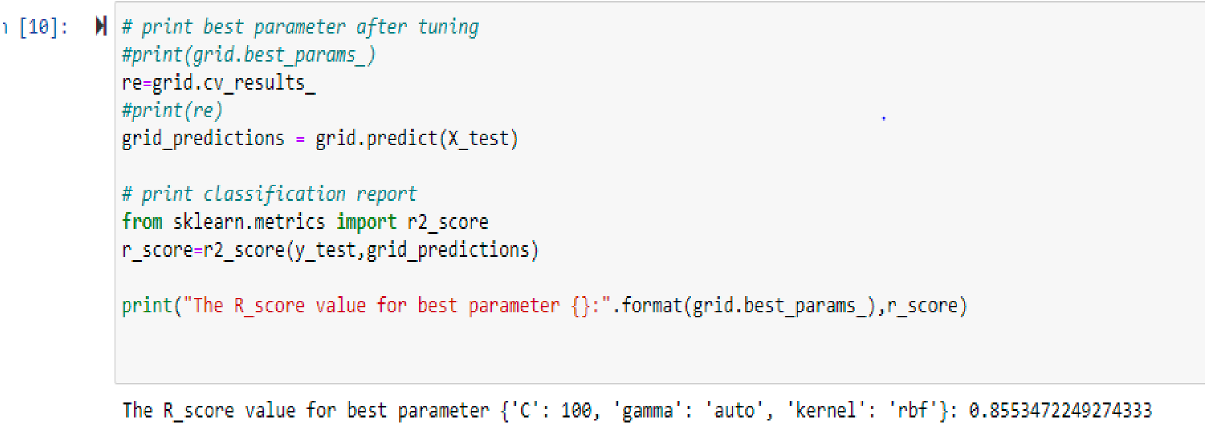
```
In [24]:  r_score

Out[24]:  0.6188543215916413
```

In Multiple Linear Regression, r squared value is **0.61.**

SVM



```
Out[9]: GridSearchCV(cv='warn', error_score='raise-deprecating',
                     estimator=SVR(C=1.0, cache_size=200, coef0=0.0, degree=3,
                                   epsilon=0.1, gamma='auto_deprecated', kernel='rbf',
                                   max_iter=-1, shrinking=True, tol=0.001,
                                   verbose=False),
                     iid='warn', n_jobs=-1,
                     param_grid={'C': [10, 100, 1000, 2000, 3000],
                                 'gamma': ['auto', 'scale'],
                                 'kernel': ['rbf', 'poly', 'sigmoid', 'linear']},
                     pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                     scoring=None, verbose=3)
```

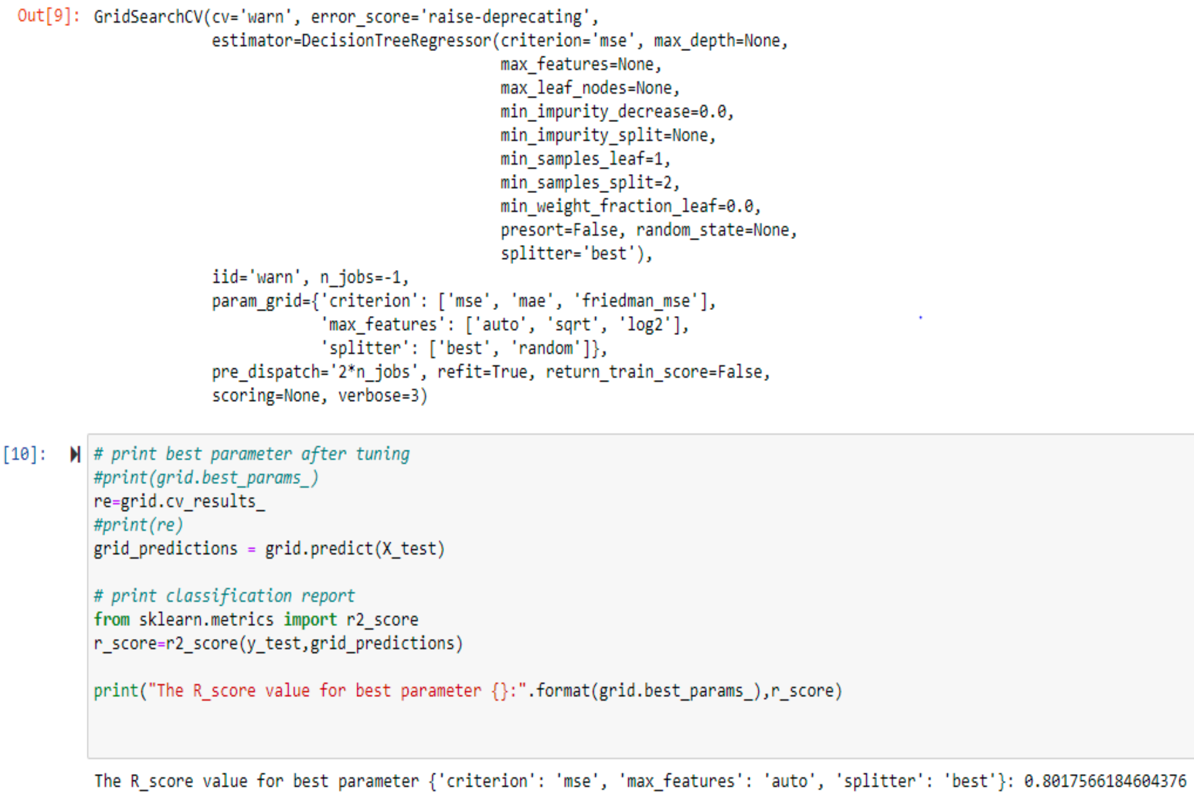


```
ı [10]:  # print best parameter after tuning
         #print(grid.best_params_)
         re=grid.cv_results_
         #print(re)
         grid_predictions = grid.predict(X_test)

         # print classification report
         from sklearn.metrics import r2_score
         r_score=r2_score(y_test,grid_predictions)

         print("The R_score value for best parameter {}:".format(grid.best_params_),r_score)

         The R_score value for best parameter {'C': 100, 'gamma': 'auto', 'kernel': 'rbf'}: 0.8553472249274333
```

C Parameter	**Gamma**	**Kernel**	**R squared value**
10	auto	rbf	0.77
		poly	0.62
		sigmoid	-10.48
		linear	0.57
	scale	rbf	0.77
		poly	0.62
		sigmoid	-10.53
		linear	0.57
100	**auto**	**rbf**	**0.85**
		poly	0.67
		sigmoid	-842.89
		linear	0.56
	scale	rbf	0.85
		poly	0.67
		sigmoid	-849.40
		linear	0.56
1000	auto	rbf	0.85
		poly	0.69
		sigmoid	-87020.46
		linear	0.56
	scale	rbf	0.85
		poly	0.69
		sigmoid	-88648.79
		linear	0.56
2000	auto	rbf	0.84
		poly	0.70
		sigmoid	-366819.51
		linear	0.56
	scale	rbf	0.84
		poly	0.69
		sigmoid	-322593.02
		linear	0.56
3000	auto	rbf	0.84
		poly	0.70
		sigmoid	-840557.54
		linear	0.56
	scale	rbf	0.84
		poly	0.70
		sigmoid	-832436.15
		linear	0.56

In SVM, when the parameter C=100, gamma=auto, kernel=rbf, the r squared value is **0.85**.

DECISION TREE



```
Out[9]: GridSearchCV(cv='warn', error_score='raise-deprecating',
                     estimator=DecisionTreeRegressor(criterion='mse', max_depth=None,
                                                     max_features=None,
                                                     max_leaf_nodes=None,
                                                     min_impurity_decrease=0.0,
                                                     min_impurity_split=None,
                                                     min_samples_leaf=1,
                                                     min_samples_split=2,
                                                     min_weight_fraction_leaf=0.0,
                                                     presort=False, random_state=None,
                                                     splitter='best'),
                     iid='warn', n_jobs=-1,
                     param_grid={'criterion': ['mse', 'mae', 'friedman_mse'],
                                 'max_features': ['auto', 'sqrt', 'log2'],
                                 'splitter': ['best', 'random']},
                     pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                     scoring=None, verbose=3)
```

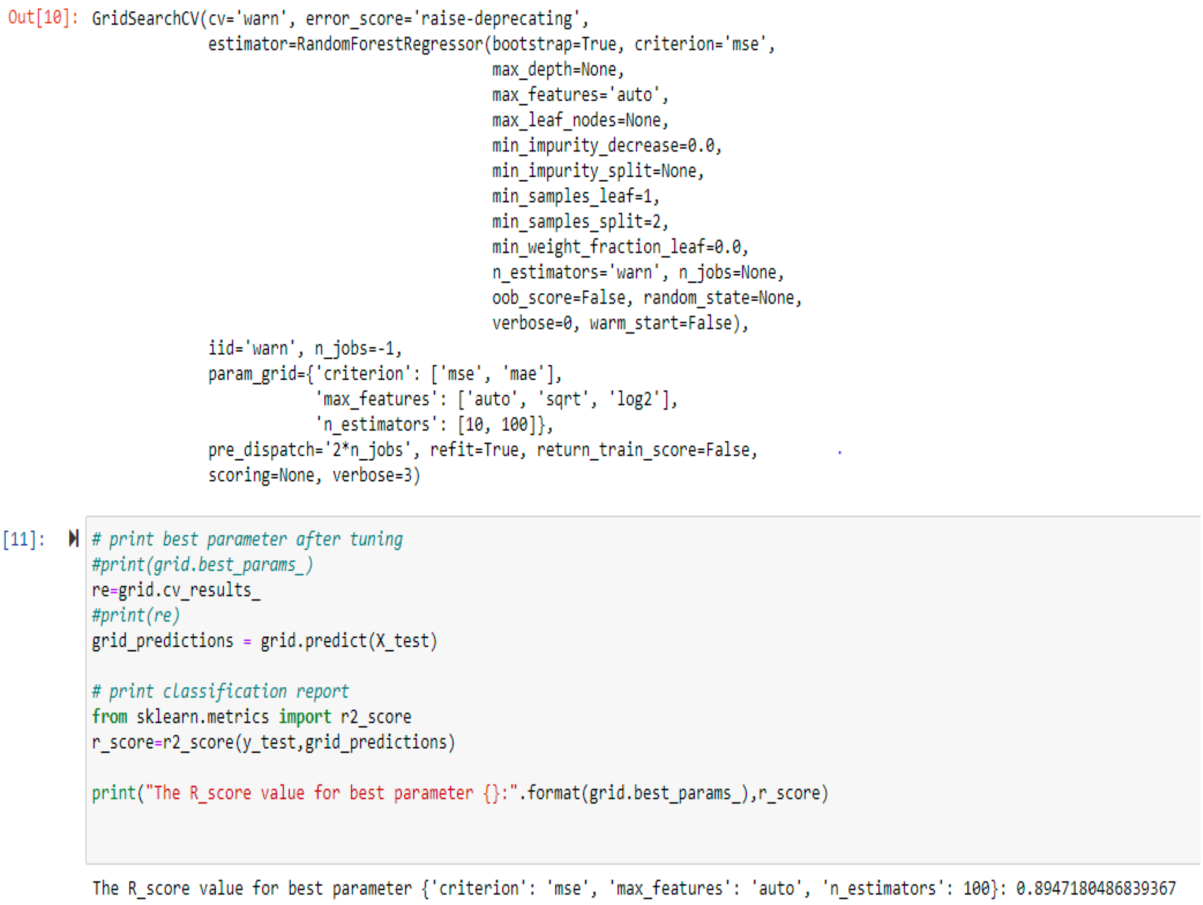
```
[10]:  # print best parameter after tuning
       #print(grid.best_params_)
       re=grid.cv_results_
       #print(re)
       grid_predictions = grid.predict(X_test)

       # print classification report
       from sklearn.metrics import r2_score
       r_score=r2_score(y_test,grid_predictions)

       print("The R_score value for best parameter {}:".format(grid.best_params_),r_score)

       The R_score value for best parameter {'criterion': 'mse', 'max_features': 'auto', 'splitter': 'best'}: 0.8017566184604376
```

Criterion	**Max_features**	**Splitter**	**R squared value**
mse	**auto**	**best**	**0.80**
		random	0.70
	sqrt	best	0.68
		random	0.54
	log2	best	0.70
		random	0.66
mae	auto	best	0.77
		random	0.72
	sqrt	best	0.67
		random	0.56
	log2	best	0.61

		random	0.59
friedman_mse	auto	best	0.80
		random	0.76
	sqrt	best	0.59
		random	0.51
	log2	best	0.64
		random	0.58

In Decision Tree Regression, when the parameter criterion=mse, max_features=auto, splitter=best, the r squared value is **0.80**.

RANDOM FOREST



```
Out[10]: GridSearchCV(cv='warn', error_score='raise-deprecating',
                      estimator=RandomForestRegressor(bootstrap=True, criterion='mse',
                                                      max_depth=None,
                                                      max_features='auto',
                                                      max_leaf_nodes=None,
                                                      min_impurity_decrease=0.0,
                                                      min_impurity_split=None,
                                                      min_samples_leaf=1,
                                                      min_samples_split=2,
                                                      min_weight_fraction_leaf=0.0,
                                                      n_estimators='warn', n_jobs=None,
                                                      oob_score=False, random_state=None,
                                                      verbose=0, warm_start=False),
                      iid='warn', n_jobs=-1,
                      param_grid={'criterion': ['mse', 'mae'],
                                  'max_features': ['auto', 'sqrt', 'log2'],
                                  'n_estimators': [10, 100]},
                      pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                      scoring=None, verbose=3)
```

```
[11]: # print best parameter after tuning
      #print(grid.best_params_)
      re=grid.cv_results_
      #print(re)
      grid_predictions = grid.predict(X_test)

      # print classification report
      from sklearn.metrics import r2_score
      r_score=r2_score(y_test,grid_predictions)

      print("The R_score value for best parameter {}:".format(grid.best_params_),r_score)
```

```
The R_score value for best parameter {'criterion': 'mse', 'max_features': 'auto', 'n_estimators': 100}: 0.8947180486839367
```

Criterion	**Max_features**	**N_estimators**	**R squared value**
mse	**auto**	10	0.89
		100	**0.89**
	sqrt	10	0.83
		100	0.86
	log2	10	0.84
		100	0.88
mae	auto	10	0.87
		100	0.87
	sqrt	10	0.84
		100	0.85
	log2	10	0.83
		100	0.87

In Random Forest, when the parameter criterion=mse, max_features=auto, n_estimators=100, the r squared value is **0.89**.

FINAL MODEL:

RANDOM FOREST is the best model to predict the strength of the cement.

RANDOM FOREST = 0.89