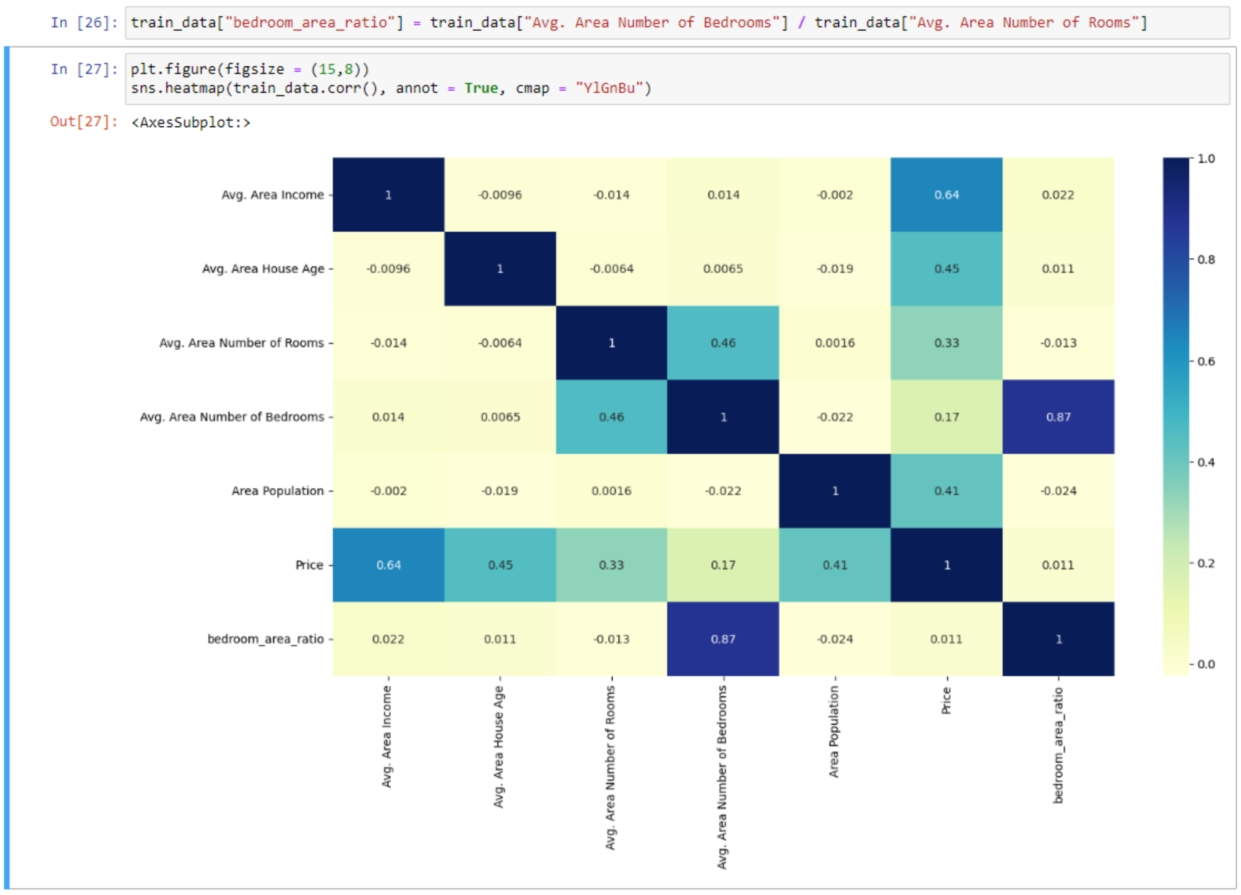
Phase 4

Feature Selection and Model Training

Feature Engineering

- First we add a new feature to the existing data set, which would show us the total area of bedrooms in a house out of all existing rooms.



```
In [26]: train_data["bedroom_area_ratio"] = train_data["Avg. Area Number of Bedrooms"] / train_data["Avg. Area Number of Rooms"]

In [27]: plt.figure(figsize = (15,8))
         sns.heatmap(train_data.corr(), annot = True, cmap = "YlGnBu")

Out[27]: <AxesSubplot:>
```

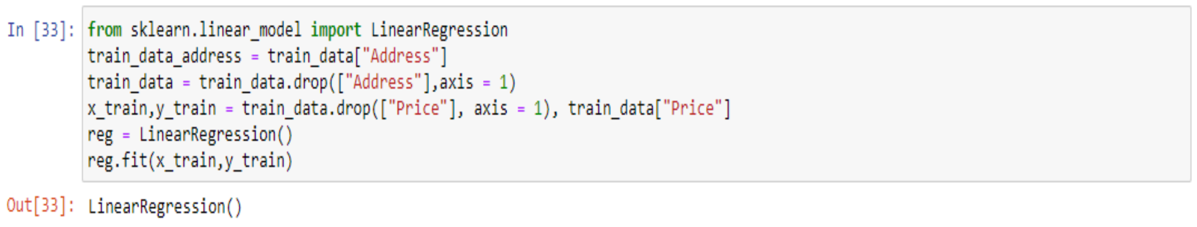
Avg. Area Income 1 -0.0096 -0.014 0.014 -0.002 0.64 0.022
Avg. Area House Age -0.0096 1 -0.0064 0.0065 -0.019 0.45 0.011
Avg. Area Number of Rooms -0.014 -0.0064 1 0.46 0.0016 0.33 -0.013
Avg. Area Number of Bedrooms 0.014 0.0065 0.46 1 -0.022 0.17 0.87
Area Population -0.002 -0.019 0.0016 -0.022 1 0.41 -0.024
Price 0.64 0.45 0.33 0.17 0.41 1 0.011
bedroom_area_ratio 0.022 0.011 -0.013 0.87 -0.024 0.011 1
Avg. Area Income
Avg. Area House Age
Avg. Area Number of Rooms
Avg. Area Number of Bedrooms
Area Population
Price
bedroom_area_ratio
1.0
0.8
0.6
0.4
0.2
0.0

- We select all features except the “Address” because it is nominal and irrelevant to the prediction of the target variable.

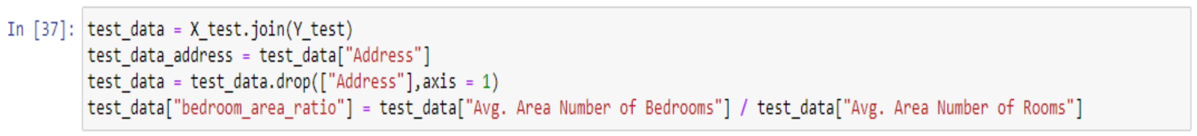
- Then, we move on with creating a Linear Regression Model.

Building Linear Regression Model

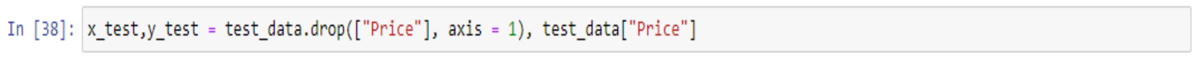
- we remove the “Address” feature from the train and test data as it is a nominal variable and not very relevant to our desired output.



```
In [33]: from sklearn.linear_model import LinearRegression
         train_data_address = train_data["Address"]
         train_data = train_data.drop(["Address"],axis = 1)
         x_train,y_train = train_data.drop(["Price"], axis = 1), train_data["Price"]
         reg = LinearRegression()
         reg.fit(x_train,y_train)

Out[33]: LinearRegression()
```



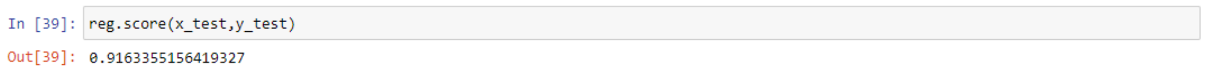
```
In [37]: test_data = X_test.join(Y_test)
         test_data_address = test_data["Address"]
         test_data = test_data.drop(["Address"],axis = 1)
         test_data["bedroom_area_ratio"] = test_data["Avg. Area Number of Bedrooms"] / test_data["Avg. Area Number of Rooms"]
```



```
In [38]: x_test,y_test = test_data.drop(["Price"], axis = 1), test_data["Price"]
```

- After the data has been fitted to the model, we will pass the testing data and check its accuracy.

Accuracy

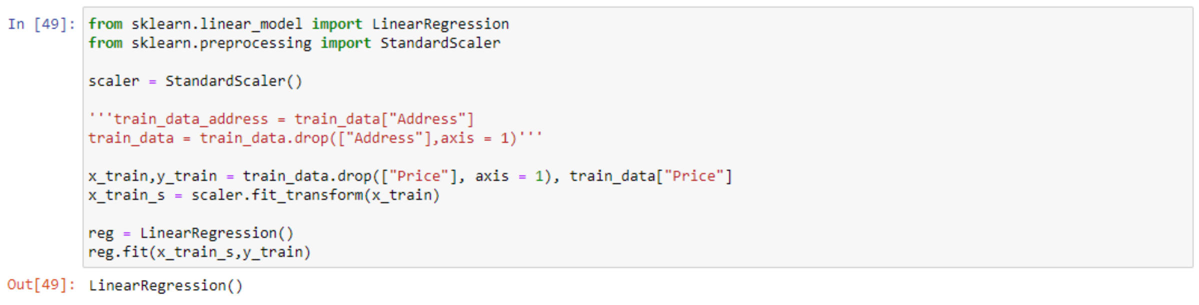


```
In [39]: reg.score(x_test,y_test)

Out[39]: 0.9163355156419327
```

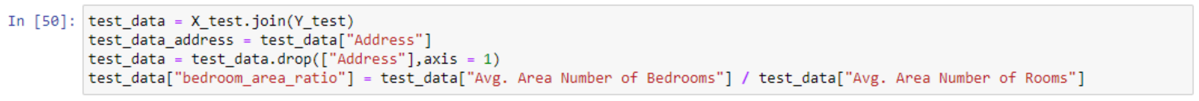
- The model is 91% accurate which is a high accuracy value. We will try and see if we can improve the accuracy by scaling our data.

Model with scaled data

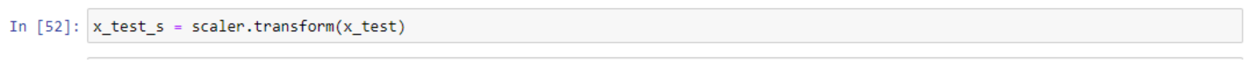
- We do the same process, but with scaled data using the StandardScaler from sklearn.



```
In [49]: from sklearn.linear_model import LinearRegression
         from sklearn.preprocessing import StandardScaler

         scaler = StandardScaler()

         '''train_data_address = train_data["Address"]
         train_data = train_data.drop(["Address"],axis = 1)'''

         x_train,y_train = train_data.drop(["Price"], axis = 1), train_data["Price"]
         x_train_s = scaler.fit_transform(x_train)

         reg = LinearRegression()
         reg.fit(x_train_s,y_train)

Out[49]: LinearRegression()
```



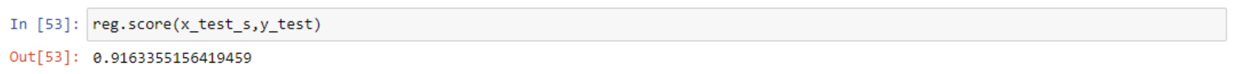
```
In [50]: test_data = X_test.join(Y_test)
         test_data_address = test_data["Address"]
         test_data = test_data.drop(["Address"],axis = 1)
         test_data["bedroom_area_ratio"] = test_data["Avg. Area Number of Bedrooms"] / test_data["Avg. Area Number of Rooms"]
```



```
In [52]: x_test_s = scaler.transform(x_test)
```

- Now, we check the accuracy with the scaled test data.

Accuracy

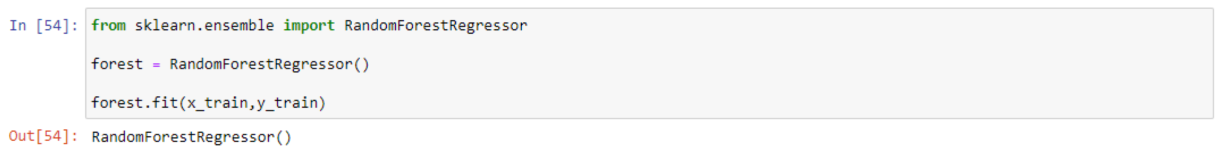


```
In [53]: reg.score(x_test_s,y_test)
Out[53]: 0.9163355156419459
```

- From the observation, there is only very minute improvement in the accuracy of the model when using scaled data. So we will try another Machine learning model like Random Forest to see if we can achieve an even higher accuracy than 91%.

Building Random Forest Model

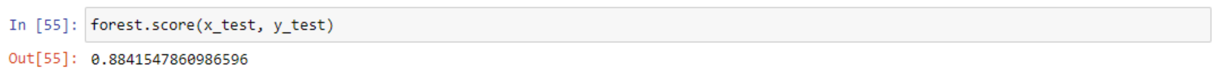
- To build the random forest model we simply just fit the already existing train data into the RandomForestRegressor.



```
In [54]: from sklearn.ensemble import RandomForestRegressor

         forest = RandomForestRegressor()

         forest.fit(x_train,y_train)
Out[54]: RandomForestRegressor()
```

- We will now check the accuracy of the model without any cross validation.

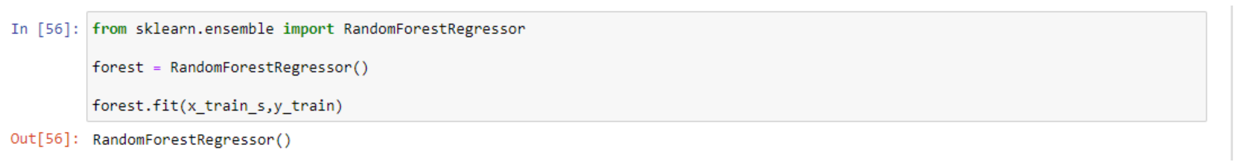
Accuracy



```
In [55]: forest.score(x_test, y_test)
Out[55]: 0.8841547860986596
```

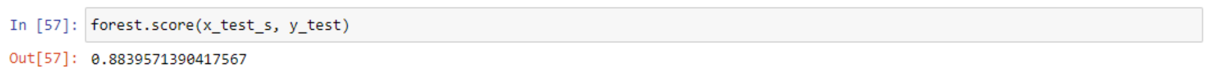
- The accuracy obtained from the random forest model is 88%, which is comparatively lower than the accuracy obtained from the Linear Regression model. Let's now try it with scaled data.

Model with scaled data



```
In [56]: from sklearn.ensemble import RandomForestRegressor

         forest = RandomForestRegressor()

         forest.fit(x_train_s,y_train)
Out[56]: RandomForestRegressor()
```

Accuracy

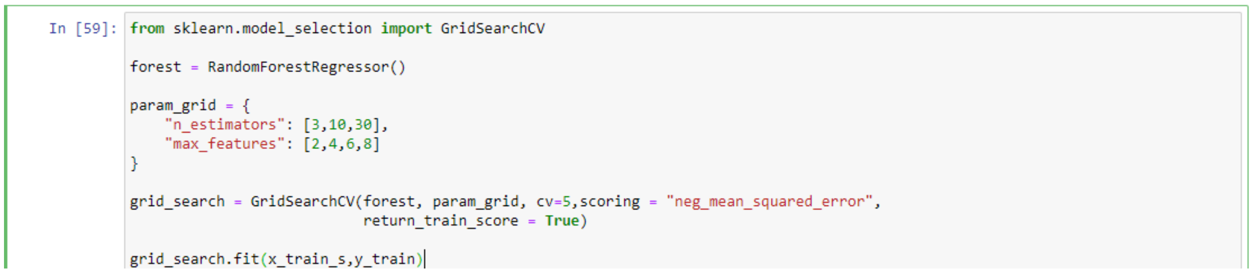


```
In [57]: forest.score(x_test_s, y_test)
Out[57]: 0.8839571390417567
```

- With the scaled data the accuracy has actually decreased (This is common during model evaluation) . So we will try it with cross validation.

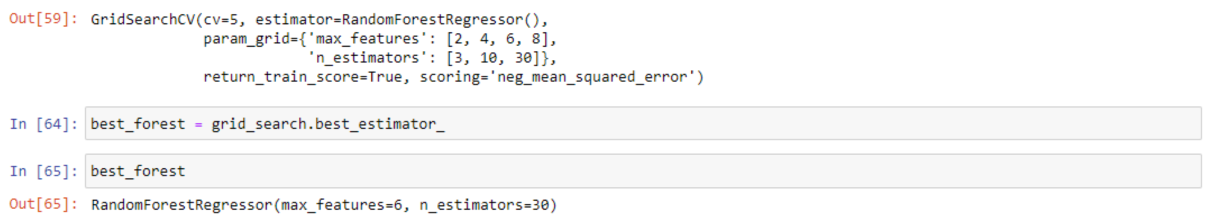
Cross Validation

- We pass estimators and max features into parameters, and we fit the data into the grid search with our forest.



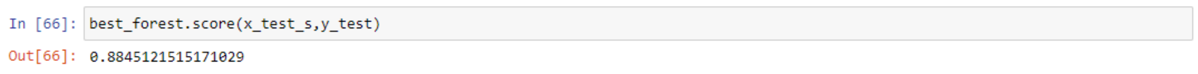
```
In [59]: from sklearn.model_selection import GridSearchCV

         forest = RandomForestRegressor()

         param_grid = {
             "n_estimators": [3,10,30],
             "max_features": [2,4,6,8]
         }

         grid_search = GridSearchCV(forest, param_grid, cv=5,scoring = "neg_mean_squared_error",
                                    return_train_score = True)

         grid_search.fit(x_train_s,y_train)
```

- We can see which parameter was the best estimator.



```
Out[59]: GridSearchCV(cv=5, estimator=RandomForestRegressor(),
                      param_grid={'max_features': [2, 4, 6, 8],
                                  'n_estimators': [3, 10, 30]},
                      return_train_score=True, scoring='neg_mean_squared_error')

In [64]: best_forest = grid_search.best_estimator_

In [65]: best_forest
Out[65]: RandomForestRegressor(max_features=6, n_estimators=30)
```

Accuracy

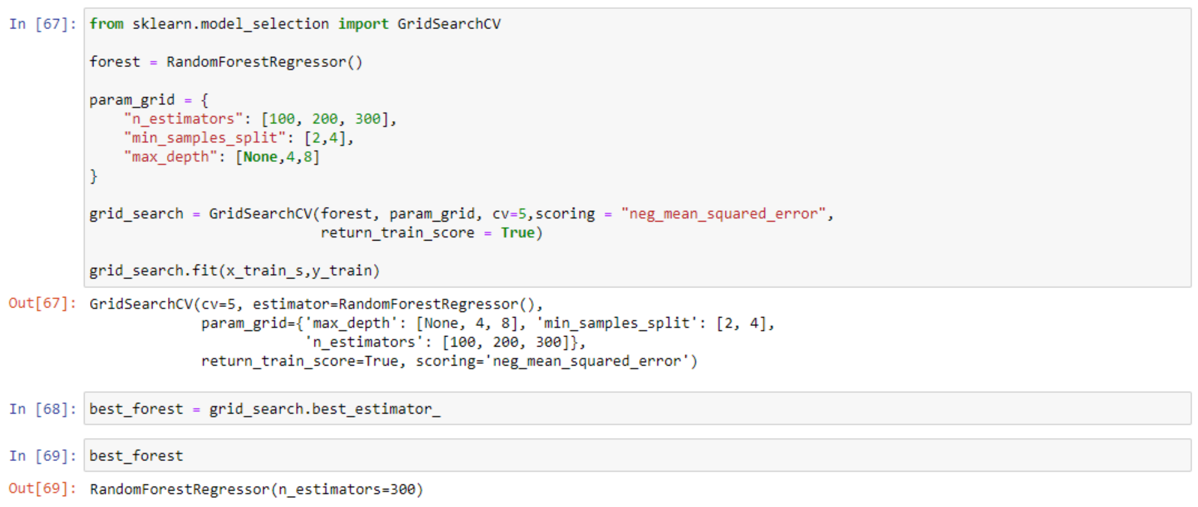


```
In [66]: best_forest.score(x_test_s,y_test)
Out[66]: 0.8845121515171029
```

- But still we only seem to be getting slight improvements in the accuracy of the model.

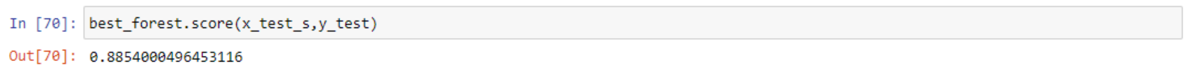
- We will perform another CV with different parameters to check the accuracy of the model once again.

- We will use minimum sample split and max depth parameters



```
In [67]: from sklearn.model_selection import GridSearchCV

         forest = RandomForestRegressor()

         param_grid = {
             "n_estimators": [100, 200, 300],
             "min_samples_split": [2,4],
             "max_depth": [None,4,8]
         }

         grid_search = GridSearchCV(forest, param_grid, cv=5,scoring = "neg_mean_squared_error",
                                    return_train_score = True)

         grid_search.fit(x_train_s,y_train)
Out[67]: GridSearchCV(cv=5, estimator=RandomForestRegressor(),
                      param_grid={'max_depth': [None, 4, 8], 'min_samples_split': [2, 4],
                                  'n_estimators': [100, 200, 300]},
                      return_train_score=True, scoring='neg_mean_squared_error')

In [68]: best_forest = grid_search.best_estimator_

In [69]: best_forest
Out[69]: RandomForestRegressor(n_estimators=300)
```

Accuracy



```
In [70]: best_forest.score(x_test_s,y_test)
Out[70]: 0.8854000496453116
```

- Even after cross validation the random forest model does not seem to produce a better accuracy score than the Linear regression model.

- Therefore, The linear regression model proves to be superior in predicting the house prices for our project.

Inference

- The linear regression model was trained with training data and produced an accuracy value of over **91%** (0.9163355156419327). With scaled data the accuracy was improved very very slightly but still at an overall **91%** (0.9163355156419459).
- The random forest model produced an accuracy value of **88%** (0.8844666295835167). Lower than that of the Linear regression model. Even after cross validation the final accuracy was still recorded to be **88%** (0.8855340019730495). It was still lower than our linear regression model.
- Therefore, for our project, the linear regression model proves to be superior in predicting the house prices with the given features in the dataset.

Conclusion

From the observed results of both the linear regression model and the random forest model, through training and testing both models. The linear regression model proved to be superior with an accuracy value of 91%. Which is 3% more than the accuracy score of the random forest models.

Precise accuracy scores of both models:

- **Linear Regression Model :** 0.9163355156419327
- **Random Forest Model :** 0.8855340019730495

We conclude that the Linear Regression model works better and accurately for our House price prediction project with the given dataset.

https://www.kaggle.com/datasets/vedavyasv/usa-housing