**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

# Data Science Project: House Price Prediction using linear Regression (Different Models)

## Part1

Introduction:

Predicting house prices can be a difficult task due to a variety of factors that can influence the value of a property. These factors can include location, size of the property, the condition of the house, and the overall state of the housing market. It is important to consider all of these factors when making a prediction in order to ensure that the prediction is as accurate as possible.

One issue that can arise when predicting house prices is the presence of outliers. For example, a luxury home in a high-end neighborhood will likely have a much higher value than a similar home in a more modest neighborhood. These outliers can skew the data and make it more difficult to make accurate predictions. To address this issue, it may be necessary to remove outliers from the data set or to use a different model that is better suited to handling outliers. Additionally, using a variety of features like crime rate, school ranking, distance to city center, can help to capture more information and make predictions more accurate.

## Part2

Solving house prediction using linear regression, lasso regression, ridge regression, random forest regressor and XG Boost Regressor.

## Linear Regression:

Linear regression is a statistical method that can be used to predict house prices by modeling the relationship between a set of independent variables (such as location, size, and condition of the house) and the dependent variable (the price of the house). In order to use linear regression to predict house prices, the following steps can be taken:

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

Linear regression is a statistical method that is used to model the relationship between a dependent variable (y) and one or more independent variables (x). The basic mathematical formula for a simple linear regression model with one independent variable (x) is:

$y = \beta0 + \beta1*x$

where:

- y is the dependent variable (the value we are trying to predict)
- x is the independent variable (the input feature)
- β0 is the y-intercept (the value of y when x = 0)
- β1 is the slope of the line (the amount that y changes for each unit change in x)

1. Collect data: Gather data on a variety of properties, including information on location, size, condition, and the sale price.
2. Clean and preprocess the data: Remove any missing or irrelevant data and ensure that the data is in a format that can be used by the linear regression model.
3. Select features: Choose the independent variables that will be used in the model. These should be variables that are likely to have an impact on the sale price of the property.
4. Train the model: Use the data to train a linear regression model by fitting the model to the data using a suitable algorithm like gradient descent.
5. Evaluate the model: Evaluate the performance of the model using techniques such as cross-validation or using a test dataset.
6. Make predictions: Use the trained model to make predictions on new data by inputting the independent variable and getting the predicted price as output.

**Regression Models and Explanation how they works:**

It is important to note that linear regression assumes a linear relationship between the independent and dependent variables, so it may not be suitable for predictions when there is a more complex relationship. Additionally, when dealing with outliers, it is a good practice to use a more robust linear regression method like Lasso or Ridge regression to handle those cases.

Regression in Machine Learning is a supervised learning technique used to predict a continuous target variable based on one or more input features. It models the relationship between the inputs and the output by fitting a function to the data. The goal of regression is to find the best-fitting

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

function that can accurately predict the target variable for new, unseen data. There are various types of regression models such as linear regression, polynomial regression, lasso regression, ridge regression, Random Forest Regressor and XGBRegressor which are used depending on the nature of the data and the problem at hand. They are widely used in various fields such as finance, marketing, healthcare, and many more to make predictions and inform decision making.

## Real estate Business Pakistan

Real estate is a significant industry that plays a vital role in the economy. The value of real estate properties, including houses, apartments, and commercial buildings, can have a significant impact on individuals, businesses, and the economy as a whole. Here are a few reasons why accurate house price prediction is important in the real estate business:

1. Investment: Real estate is a popular investment option, and the ability to accurately predict house prices can help investors make informed decisions about when and where to buy or sell properties.
2. Mortgage lending: Accurate house price predictions are important for mortgage lenders, as they use the value of a property as collateral for loans. Inaccurate predictions can lead to lending risks and losses for the lender.
3. Government policies: Government policies and regulations, such as property taxes and zoning laws, are often based on the value of real estate. Accurate house price predictions can help ensure that these policies are fair and equitable.
4. Economic growth: The real estate market is closely tied to economic growth, as the value of properties can have a direct impact on consumer confidence and spending. Accurate house price predictions can help to identify trends in the market and anticipate potential economic fluctuations.
5. Real estate agents and brokers: Real estate agents and brokers use house price predictions to give realistic and accurate recommendations to their clients. They also use it to determine the right price for the property they're selling or renting.

Overall, accurate house price prediction is crucial for many different stakeholders in the real estate industry, and can help to promote stability and growth in the market.

## Importance:

The real estate sector in Pakistan is a significant contributor to the country's economy, generating employment opportunities, and driving economic growth. It is one of the leading sectors in terms of investment and construction, with a significant portion of the population investing in properties for both residential and commercial purposes. The sector also contributes to the country's GDP through various related industries, such as construction, manufacturing, and finance. The government's policies, such as the construction of low-cost housing, also plays a role in driving the growth of the sector. However, the sector also faces challenges such as lack of regulations, lack of transparency, and difficulties in accessing finance. Nevertheless, the real estate sector in Pakistan continues to be a key driver of economic growth and development.

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

Part-3:

## Dataset and Preprocessing:

## Collected from zameen.com website

- Statistical summary of dataset

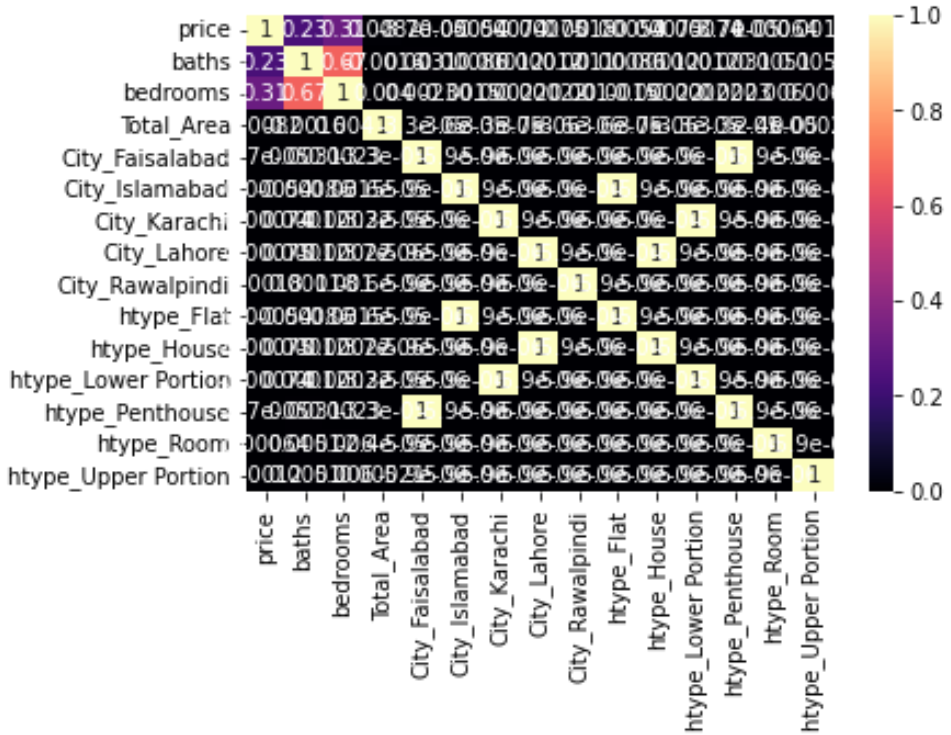
```
df.describe()
```

1 to 8 of 8 entries Filter

| index | property_id | location_id | price | latitude | longitude | baths | bedrooms | Total_Area |
|---|---|---|---|---|---|---|---|---|
| count | 168446.0 | 168446.0 | 168446.0 | 168446.0 | 168446.0 | 168446.0 | 168446.0 | 168446.0 |
| mean | 15596255.48967028 | 4375.936395046484 | 17765759.83221923 | 29.859518973506646 | 71.23980354708306 | 2.874226755161892 | 3.179422485544329 | 13942.393530193653 |
| std | 2251206.8666874063 | 3776.561581391663 | 35310032.31786422 | 3.8078697313178838 | 3.1330415440399726 | 2.463399892577542 | 1.9714009650710147 | 862364.7400401554 |
| min | 86575.0 | 1.0 | 0.0 | 11.052446 | 25.906027 | 0.0 | 0.0 | 0.0 |
| 25% | 14883201.75 | 1058.0 | 175000.0 | 24.948536 | 67.130363 | 0.0 | 2.0 | 1905.757 |
| 50% | 16658506.0 | 3286.0 | 8500000.0 | 31.459784 | 73.056182 | 3.0 | 3.0 | 4356.016 |
| 75% | 17086619.75 | 7220.0 | 19500000.0 | 33.560887 | 73.25987025 | 4.0 | 4.0 | 11979.043999999998 |
| max | 17357718.0 | 14220.0 | 2000000000.0 | 73.184088 | 80.16143 | 403.0 | 68.0 | 338798790.0 |

Show 25 per page
Like what you see? Visit the data table notebook to learn more about interactive tables.

Heat_Map:



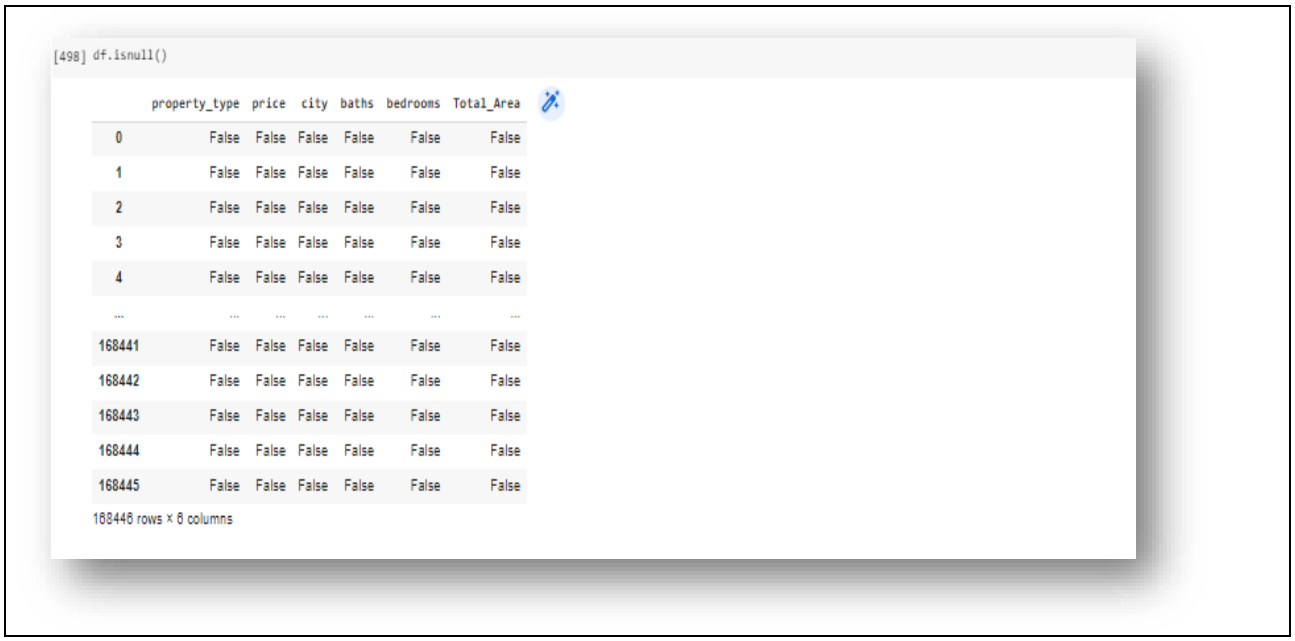
**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

Challenges:

1. Missing Values

```
Missing Values by Column
--------------------------------
property_id                 0
location_id                 0
page_url                    0
property_type               0
price                       0
location                    0
city                        0
province_name               0
latitude                    0
longitude                   0
baths                       0
purpose                     0
bedrooms                    0
date_added                  0
agency                  44071
agent                   44072
Total_Area                  0
dtype: int64
--------------------------------
TOTAL MISSING VALUES: 88143
```

2. NaN Values/ffill(method)



[498] df.isnull()

| | property_type | price | city | baths | bedrooms | Total_Area |
|---|---|---|---|---|---|---|
| 0 | False | False | False | False | False | False |
| 1 | False | False | False | False | False | False |
| 2 | False | False | False | False | False | False |
| 3 | False | False | False | False | False | False |
| 4 | False | False | False | False | False | False |
| ... | ... | ... | ... | ... | ... | ... |
| 168441 | False | False | False | False | False | False |
| 168442 | False | False | False | False | False | False |
| 168443 | False | False | False | False | False | False |
| 168444 | False | False | False | False | False | False |
| 168445 | False | False | False | False | False | False |

168446 rows × 6 columns

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

3. Dropping Unnecessary Columns

## Unnecessory Data Columns

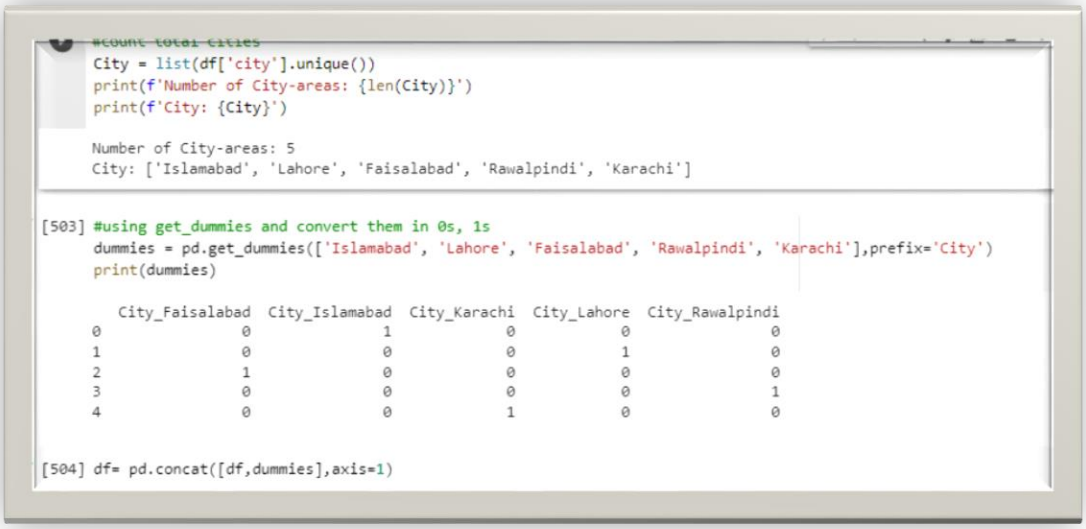
```
df=df.drop(['province_name','property_id','location_id',
            'page_url','latitude','purpose','date_added','longitude','agency','agent','location'], axis=1 )
```

```
[496] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 168446 entries, 0 to 168445
Data columns (total 6 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   property_type 168446 non-null  object 
 1   price         168446 non-null  int64  
 2   city          168446 non-null  object 
 3   baths         168446 non-null  int64  
 4   bedrooms      168446 non-null  int64  
 5   Total_Area    168446 non-null  float64
dtypes: float64(1), int64(3), object(2)
memory usage: 9.0+ MB
```

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

4. Conversion in continuous data columns (Categorical Values(nominal-Values))



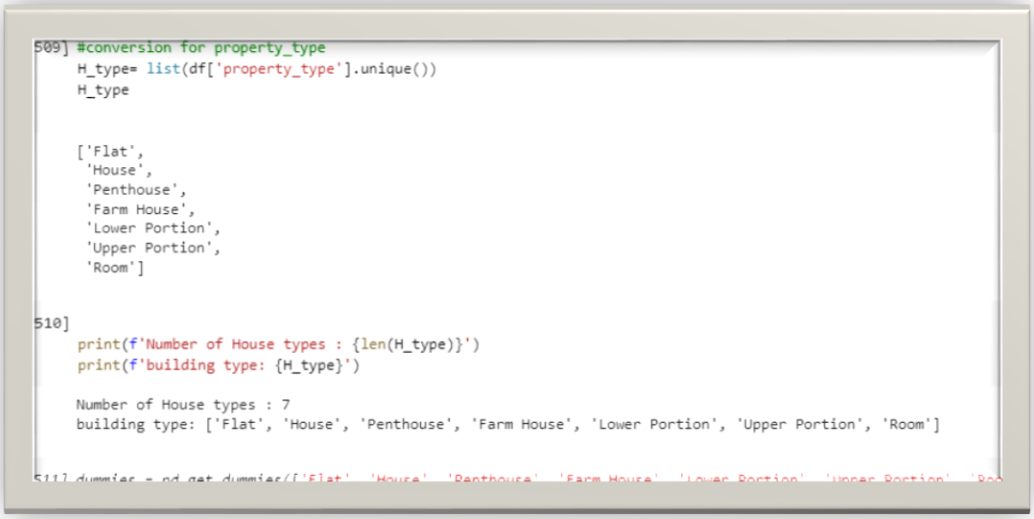
```
#count total cities
City = list(df['city'].unique())
print(f'Number of City-areas: {len(City)}')
print(f'City: {City}')

Number of City-areas: 5
City: ['Islamabad', 'Lahore', 'Faisalabad', 'Rawalpindi', 'Karachi']
```

```
[503] #using get_dummies and convert them in 0s, 1s
dummies = pd.get_dummies(['Islamabad', 'Lahore', 'Faisalabad', 'Rawalpindi', 'Karachi'],prefix='City')
print(dummies)

   City_Faisalabad  City_Islamabad  City_Karachi  City_Lahore  City_Rawalpindi
0                0               1             0            0                0
1                0               0             0            1                0
2                1               0             0            0                0
3                0               0             0            0                1
4                0               0             1            0                0
```

```
[504] df= pd.concat([df,dummies],axis=1)
```



```
509] #conversion for property_type
H_type= list(df['property_type'].unique())
H_type


['Flat',
 'House',
 'Penthouse',
 'Farm House',
 'Lower Portion',
 'Upper Portion',
 'Room']
```

```
510]
print(f'Number of House types : {len(H_type)}')
print(f'building type: {H_type}')

Number of House types : 7
building type: ['Flat', 'House', 'Penthouse', 'Farm House', 'Lower Portion', 'Upper Portion', 'Room']
```

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

```
dummies = pd.get_dummies(['Flat', 'House', 'Penthouse', 'Farm House',
                          'Lower Portion', 'Upper Portion', 'Room'], prefix='htype', drop_first=True)
print(dummies)
```

```
   htype_Flat  htype_House  htype_Lower Portion  htype_Penthouse  htype_Room  \
0           1            0                    0                0           0
1           0            1                    0                0           0
2           0            0                    0                1           0
3           0            0                    0                0           0
4           0            0                    1                0           0
5           0            0                    0                0           0
6           0            0                    0                0           1

   htype_Upper Portion
0                    0
1                    0
2                    0
3                    0
4                    0
5                    1
6                    0
```

- ✓ Whole Dataset is now in numerical values, so we can apply ‘Machine Learning’ algorithms.

| | price | baths | bedrooms | Total_Area | City_Faisalabad | City_Islamabad | City_Karachi | City_Lahore | City_Rawalpindi | htype_Flat | htype_House | htype_Lower Portion | htype_Penthouse | htype_Room | htype_Upper Portion |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 10000000 | 2 | 2 | 1089.004 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1 | 6900000 | 3 | 3 | 15246.056 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2 | 16500000 | 6 | 5 | 2178.008 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 3 | 43500000 | 4 | 4 | 10890.000 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 4 | 7000000 | 3 | 3 | 2178.008 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 168441 | 26500000 | 0 | 6 | 26136.096 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 168442 | 12500000 | 0 | 3 | 2178.008 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 168443 | 27000000 | 0 | 6 | 26136.096 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 168444 | 11000000 | 0 | 3 | 21235.578 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 168445 | 9000000 | 3 | 3 | 25591.594 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

168446 rows × 15 columns

```
[52] X['price'].describe()
```

```
count    1.684460e+05
mean     1.776575e+07
std      3.531003e+07
min      0.000000e+00
25%      1.750000e+05
50%      8.500000e+06
75%      1.950000e+07
max      2.000000e+09
Name: price, dtype: float64
```

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

5. Training the dataset on different machine learning models of Linear Regression

## Train-Test Split

```
[530] from sklearn.model_selection import train_test_split, cross_val_score
      X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.4,random_state=42)
```

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

6. Lasso Regressor

## Lsso Regression

```
[538] lasso = Lasso()
      lasso.fit(X_train, y_train)
      pred_lasso =lasso.predict(X_test)
      mae, mse, rmse, r_squared = evaluation(y_test, pred_lasso) #model Evaluation
      print('Lasso_Regression model Evaluation')
      print('--'*30)
      print("MAE:", mae)
      print("MSE:", mse)
      print("RMSE:", rmse)
      print("R2 Score:", r_squared)
```

```
Lasso_Regression model Evaluation
------------------------------------------------------------
MAE: 2.286122336186487e-08
MSE: 2.181108954372505e-15
RMSE: 4.6702344206394024e-08
R2 Score: 1.0
```

$y = \beta0 + \beta1x1 + \beta2x2 + \ldots + \beta n*xn + \varepsilon$

Where:

- y is the dependent variable (the value we are trying to predict)
- x1, x2, ..., xn are the independent variables
- β0, β1, β2, ..., βn are the model coefficients
- ε is the error term (the difference between the predicted value and the actual value)

In Lasso regression, the objective is to minimize the sum of the squared residuals (the difference between the predicted values and the actual values) as well as the sum of the absolute values of the coefficients (the L1 regularization term)

Objective function = (1/(2*N)) * SUM(y-y^)^2 + α * SUM|β|

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

7. Ridge Regressor

## Ridge Regression

```
[539] ridge = Ridge()
      ridge.fit(X_train, y_train)
      pred_ridge=ridge.predict(X_test)
      mae, mse, rmse, r_squared = evaluation(y_test, pred_ridge) #model Evaluation
      print('Ridge_Regression model Evaluation')
      print('--'*30)
      print("MAE:", mae)
      print("MSE:", mse)
      print("RMSE:", rmse)
      print("R2 Score:", r_squared)
```

```
Ridge_Regression model Evaluation
------------------------------------------------------------
MAE: 1.2611139641794805e-08
MSE: 4.632218562161218e-16
RMSE: 2.1522589440309496e-08
R2 Score: 1.0
/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_ridge.py:157: LinAlgWarning: Ill
  return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
```

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

Ridge regression is a variation of linear regression that is used to deal with the problem of overfitting, which can occur when there are a large number of independent variables. The basic mathematical formula for Ridge regression is similar to that of linear regression but with an additional term that penalizes the square of the coefficients (also called L2 regularization). The formula for Ridge regression is as follows:

$y = \beta 0 + \beta 1x1 + \beta 2x2 + ... + \beta n*xn + \varepsilon$

Where:

- y is the dependent variable (the value we are trying to predict)
- x1, x2, ..., xn are the independent variables
- β0, β1, β2, ..., βn are the model coefficients
- ε is the error term (the difference between the predicted value and the actual value)

In Ridge regression, the objective is to minimize the sum of the squared residuals (the difference between the predicted values and the actual values) as well as the sum of the squares of the coefficients (the L2 regularization term)

Objective function = (1/(2*N)) * SUM(y-y^)^2 + α * SUM(β^2)

Where:

- N is the number of observations
- α is the regularization parameter
- SUM(y-y^)^2 is the sum of the squared residuals
- SUM(β^2) is the sum of the squares of the coefficients

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

8. Linear Regressor

## Linear Regression

```
[536] lin_reg = LinearRegression()
      lin_reg.fit(X_train,y_train)
      y_pred = lin_reg.predict(X_test)

      mae, mse, rmse, r_squared = evaluation(y_test, y_pred) #model Evaluation
      print('Linear_Regression model Evaluation')
      print('--'*30)
      print("MAE:", mae)
      print("MSE:", mse)
      print("RMSE:", rmse)
      print("R2 Score:", r_squared)
```

```
Linear_Regression model Evaluation
------------------------------------------------------------
MAE: 1.1400431006850747e-08
MSE: 5.664804585779822e-16
RMSE: 2.3800849954948714e-08
R2 Score: 1.0
```

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

Linear regression is a statistical method that is used to model the relationship between a dependent variable (y) and one or more independent variables (x). The basic mathematical formula for a simple linear regression model with one independent variable (x) is:

$y = \beta0 + \beta1*x$

where:

- y is the dependent variable (the value we are trying to predict)
- x is the independent variable (the input feature)
- β0 is the y-intercept (the value of y when x = 0)
- β1 is the slope of the line (the amount that y changes for each unit change in x)

The values of β0 and β1 are called the model coefficients and can be estimated using the data using methods such as least squares. Once the coefficients are estimated, the linear regression model can be used to make predictions for new data by plugging in the values of the independent variable(s) and solving for the dependent variable.

For multiple independent variables, the formula becomes:

$y = \beta0 + \beta1x1 + \beta2x2 + ... + \beta n*xn$

Where x1, x2, ..., xn are the independent variables and β1, β2, ..., βn are the respective coefficients, it's also called multivariate linear regression.

It's important to note that the linear regression model assumes a linear relationship between the independent and dependent variables, so it may not be suitable for predictions when

Conclusion:

Linear regression is a basic and widely used type of regression model that assumes a linear relationship between the input features and the target variable. The model finds the best-fitting straight line through the data points by minimizing the sum of the squared differences between the predicted and actual values. Linear regression can be used for both simple and multiple regression problems.

Lasso regression is a variation of linear regression that adds a constraint to the model to make some of the coefficients equal to zero. This is known as L1 regularization, and it can be useful for feature selection in high-dimensional datasets. Lasso regression tends to produce sparse models, where only a subset of the input features are used to make predictions.

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

Ridge regression is another variation of linear regression that adds a constraint to the model to keep the coefficients small. This is known as L2 regularization, and it can be useful for preventing overfitting in high-dimensional datasets. Ridge regression tends to produce models with small but non-zero coefficients for all input features.

Both Lasso and Ridge regression are used to overcome the problem of overfitting in linear regression by adding a penalty term to the cost function, L1 regularization term (lasso) adds penalty equivalent to the absolute value of the magnitude of coefficients, and L2 regularization term (ridge) adds penalty equivalent to the square of the magnitude of coefficients.

## Note:
## Model Comparison

- The less the Root Mean Squared Error (RMSE), the better the model is.

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

Evaluation:

XGBoost Regressor model Evaluation:

.....

MAE: 19293.307195062167

MSE: 4697463596152.285

RMSE: 2167363.281997802

R2 Score: 0.9961415242827611

Random Forest_Regression model Evaluation

.....

MAE: 15024.17982561332

MSE: 5468849645590.729

RMSE: 2338557.171760128

R2 Score: 0.9955079112106316

Ridge_Regression model Evaluation

---

MAE: 1.2611139641794805e-08

MSE: 4.6322185621612l8e-16

RMSE: 2.1522589440309496e-08

R2 Score: 1.0

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

```
Linear_Regression model Evaluation
-----------------------------------
MAE: 1.1400431006850747e-08
MSE: 5.664804585779822e-16
RMSE: 2.3800849954948714e-08
R2 Score: 1.0
```

```
XGBoost Regressor model Evaluation:
-----------------------------------
MAE: 19293.307195062167
MSE: 4697463596152.285
RMSE: 2167363.281997802
R2 Score: 0.996141524282761 1
```

## Evaluating a regression model:

Evaluating a regression model involves comparing the predicted values to the true values of the target variable. There are several metrics commonly used to evaluate the performance of a regression model:

- Mean Absolute Error (MAE) measures the average magnitude of the errors in the predicted values, without considering their direction. It is calculated as the sum of the absolute differences between the predicted and true values, divided by the number of observations.

**M_HUZAIFA_KHILJI**
**My Portfolio Project**
**Email: huzaifakhilji7@gmail.com**

- Mean Squared Error (MSE) measures the average squared differences between the predicted and true values. It is calculated as the sum of the squared differences between the predicted and true values, divided by the number of observations.
- Root Mean Squared Error (RMSE) is the square root of the MSE and it measures the average magnitude of the errors in the predicted values in the same units as the target variable.
- R-squared (R2) measures the proportion of the variance in the target variable that is explained by the model. It ranges from 0 to 1, where a higher value indicates a better fit.
- Adjusted R-squared is similar to R-squared, but it takes into account the number of input features and the sample size. It adjusts the R-squared value by penalizing models with a large number of input features.

Another important aspect of evaluating a regression model is to check for assumptions, for example linearity and independence of errors, and if the assumptions are not met then it's better to use a different model or transform the data.

It's important to use multiple evaluation metrics to get a comprehensive understanding of the model performance, and also to validate the model with a separate test dataset.

---