Multiple Regression Algorithm

In [81]:
```
#read excel file
import math
import numpy as np
import pandas as pd
df = pd.read_csv("FinalNew_house.csv")
```

In [82]:
```
#Load libraries for partition and LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from dmba import regressionSummary, classificationSummary, exhaustive_search,
```

Categorical Variables

In [94]:
```
# Converting to categorical variables zip code and waterfront
```

In [95]:
```
#Converting categorical variables into integers to remove decimals and then in
df[["zipcode","waterfront"]] = df[["zipcode","waterfront"]].astype("int")
df[["zipcode","waterfront"]] = df[["zipcode","waterfront"]].astype("category")
```

1. Multiple Regression

In [96]:
```
# Split the data set into predictors (X) and outcome (y - price)
predictors = df.drop(['price'],axis=1)
outcome = df['price']
```

In [97]:
```
# Partition data into predictors (x) and output (y)
X = pd.get_dummies(predictors, drop_first=True)
y = outcome
```

In [98]:
```
#Show firs 9 rows of predictors
X.head(9)
```

Out[98]:

	bedrooms	bathrooms	floors	view	condition	grade	sqft_above	sqft_basement	lat	
0	3.0	1	1	0.0	3.0	7.0	1180.0	0.0	47.5112	-1
1	3.0	2	2	0.0	3.0	7.0	2170.0	400.0	47.7210	-1
2	2.0	1	1	0.0	3.0	6.0	770.0	0.0	47.7379	-1
3	4.0	3	1	0.0	5.0	7.0	1050.0	910.0	47.5208	-1
4	3.0	2	1	0.0	3.0	8.0	1680.0	0.0	47.6168	-1
5	3.0	2	2	0.0	3.0	7.0	1715.0	0.0	47.3097	-1
6	3.0	1	1	0.0	3.0	7.0	1060.0	0.0	47.4095	-1
7	3.0	1	1	0.0	3.0	7.0	1050.0	730.0	47.5123	-1
8	3.0	2	2	0.0	3.0	7.0	1890.0	0.0	47.3684	-1

9 rows × 84 columns

In [99]:
```
X.columns
```

Out[99]:
```
Index(['bedrooms', 'bathrooms', 'floors', 'view', 'condition', 'grade',
       'sqft_above', 'sqft_basement', 'lat', 'long', 'sqft_living15',
       'sqft_lot15', 'age', 'renov_age', 'waterfront_1', 'zipcode_98002',
       'zipcode_98003', 'zipcode_98004', 'zipcode_98005', 'zipcode_98006',
       'zipcode_98007', 'zipcode_98008', 'zipcode_98010', 'zipcode_98011',
       'zipcode_98014', 'zipcode_98019', 'zipcode_98022', 'zipcode_98023',
       'zipcode_98024', 'zipcode_98027', 'zipcode_98028', 'zipcode_98029',
       'zipcode_98030', 'zipcode_98031', 'zipcode_98032', 'zipcode_98033',
       'zipcode_98034', 'zipcode_98038', 'zipcode_98039', 'zipcode_98040',
       'zipcode_98042', 'zipcode_98045', 'zipcode_98052', 'zipcode_98053',
       'zipcode_98055', 'zipcode_98056', 'zipcode_98058', 'zipcode_98059',
       'zipcode_98065', 'zipcode_98070', 'zipcode_98072', 'zipcode_98074',
       'zipcode_98075', 'zipcode_98077', 'zipcode_98092', 'zipcode_98102',
       'zipcode_98103', 'zipcode_98105', 'zipcode_98106', 'zipcode_98107',
       'zipcode_98108', 'zipcode_98109', 'zipcode_98112', 'zipcode_98115',
       'zipcode_98116', 'zipcode_98117', 'zipcode_98118', 'zipcode_98119',
       'zipcode_98122', 'zipcode_98125', 'zipcode_98126', 'zipcode_98133',
       'zipcode_98136', 'zipcode_98144', 'zipcode_98146', 'zipcode_98148',
       'zipcode_98155', 'zipcode_98166', 'zipcode_98168', 'zipcode_98177',
       'zipcode_98178', 'zipcode_98188', 'zipcode_98198', 'zipcode_98199'],
      dtype='object')
```

In [100]:
```
#Show firs 9 rows of outcomes
y.describe()
```

Out[100]:
```
count    1.735700e+04
mean     4.607963e+05
std      1.969479e+05
min      7.800000e+04
25%      3.089000e+05
50%      4.250000e+05
75%      5.775000e+05
max      1.125000e+06
Name: price, dtype: float64
```

In [101]:
```
# Split the data into training and validation
train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.4, ran
```

In [102]:
```
# Built the Linear Model based on the training data
df_lm = LinearRegression()
df_lm.fit(train_X, train_y)
```

Out[102]:
```
LinearRegression()
```

In [92]:
```
#Print coefficients
print(pd.DataFrame({'Predictor': X.columns, 'coefficient': df_lm.coef_}))
```

```
        Predictor    coefficient
0        bedrooms   -1588.796078
1       bathrooms   10115.824223
2          floors  -17319.404817
3            view   31304.483824
4       condition   25286.742243
..            ...            ...
79  zipcode_98177  208899.773112
80  zipcode_98178   58407.593922
81  zipcode_98188   37836.003180
82  zipcode_98198   29431.368776
83  zipcode_98199  359451.938714

[84 rows x 2 columns]
```

1. Performance Check

In [103]:
```
#Print performance measures(training data)
regressionSummary(train_y, df_lm.predict(train_X))
```

```
Regression statistics

                      Mean Error (ME) : -0.0000
       Root Mean Squared Error (RMSE) : 79132.5305
            Mean Absolute Error (MAE) : 58397.2642
          Mean Percentage Error (MPE) : -2.1369
Mean Absolute Percentage Error (MAPE) : 14.0287
```

In [109]:
```
#Calculating the R2 score for the model (training data)
df_lm_predt = df_lm.predict(train_X)
score_lr_valt = adjusted_r2_score(train_y, df_lm_predt, df_lm)*100
print("The R2 score for training data is:", score_lr_valt.round(1),"%")
```

```
The R2 score for training data is: 83.8 %
```

In [106]:
```
#Print performace measures for the validation set
regressionSummary(valid_y, df_lm.predict(valid_X))
```

```
Regression statistics

                      Mean Error (ME) : -259.8028
       Root Mean Squared Error (RMSE) : 79984.8699
            Mean Absolute Error (MAE) : 59014.6641
          Mean Percentage Error (MPE) : -2.3362
Mean Absolute Percentage Error (MAPE) : 14.4164
```

In [110]:
```
#Calculating the R2 score for the model for the validation set
df_lm_pred = df_lm.predict(valid_X)
score_lr_val = adjusted_r2_score(valid_y, df_lm_pred,df_lm)*100
print("The R2 score for validation data is:", score_lr_val.round(1),"%")
```

```
The R2 score for validation data is: 83.2 %
```

In [111]:
```
#Make prediction of 2 houses in the validation dataset
df_lm_pred = df_lm.predict(valid_X)
result = pd.DataFrame({'Predicted': df_lm_pred, 'Actual': valid_y, 'Residual':
result.sample(2)
```

Out[111]:

	Predicted	Actual	Residual
13585	397720.602425	389999	-7721.602425
2103	536464.502529	560000	23535.497471

Dimension Reduction - Forward Selection

In [113]:
```
#Forward Selection

def train_model(variables):
    if len(variables) == 0:
        return None
    model = LinearRegression()
    model.fit(train_X[list(variables)], train_y)
    return model

def score_model(model, variables):
    if len(variables) == 0:
        return AIC_score(train_y, [train_y.mean()] * len(train_y), model, df=1
    return AIC_score(train_y, model.predict(train_X[variables]), model)

best_model, best_variables = forward_selection(train_X.columns, train_model, s
print("The best varibles are:", best_variables)
```

```
Variables: bedrooms, bathrooms, floors, view, condition, grade, sqft_above, s
qft_basement, lat, long, sqft_living15, sqft_lot15, age, renov_age, waterfron
t_1, zipcode_98002, zipcode_98003, zipcode_98004, zipcode_98005, zipcode_9800
6, zipcode_98007, zipcode_98008, zipcode_98010, zipcode_98011, zipcode_98014,
zipcode_98019, zipcode_98022, zipcode_98023, zipcode_98024, zipcode_98027, zi
pcode_98028, zipcode_98029, zipcode_98030, zipcode_98031, zipcode_98032, zipc
ode_98033, zipcode_98034, zipcode_98038, zipcode_98039, zipcode_98040, zipcod
e_98042, zipcode_98045, zipcode_98052, zipcode_98053, zipcode_98055, zipcode_
98056, zipcode_98058, zipcode_98059, zipcode_98065, zipcode_98070, zipcode_98
072, zipcode_98074, zipcode_98075, zipcode_98077, zipcode_98092, zipcode_9810
2, zipcode_98103, zipcode_98105, zipcode_98106, zipcode_98107, zipcode_98108,
zipcode_98109, zipcode_98112, zipcode_98115, zipcode_98116, zipcode_98117, zi
pcode_98118, zipcode_98119, zipcode_98122, zipcode_98125, zipcode_98126, zipc
ode_98133, zipcode_98136, zipcode_98144, zipcode_98146, zipcode_98148, zipcod
e_98155, zipcode_98166, zipcode_98168, zipcode_98177, zipcode_98178, zipcode_
98188, zipcode_98198, zipcode_98199
Start: score=283512.01, constant
Step: score=279063.19, add grade
Step: score=275775.39, add lat
Step: score=274534.09, add age
Step: score=273425.89, add sqft_living15
Step: score=272938.20, add bathrooms
Step: score=272516.19, add zipcode_98004
Step: score=272210.92, add zipcode_98155
Step: score=271873.59, add zipcode_98133
Step: score=271546.93, add sqft_above
Step: score=271078.70, add sqft_basement
Step: score=270740.74, add zipcode_98040
Step: score=270393.49, add view
Step: score=270062.64, add zipcode_98028
Step: score=269776.05, add zipcode_98034
Step: score=269504.69, add zipcode_98112
Step: score=269246.33, add zipcode_98019
Step: score=268961.20, add zipcode_98125
Step: score=268664.00, add zipcode_98011
Step: score=268329.06, add zipcode_98072
Step: score=268049.80, add zipcode_98177
Step: score=267765.08, add condition
Step: score=267625.11, add zipcode_98119
```

```
Step: score=267478.93, add zipcode_98109
Step: score=267344.85, add zipcode_98022
Step: score=267208.86, add zipcode_98116
Step: score=267078.14, add zipcode_98102
Step: score=266942.31, add zipcode_98006
Step: score=266815.53, add zipcode_98199
Step: score=266702.00, add zipcode_98178
Step: score=266597.96, add zipcode_98168
Step: score=266489.00, add zipcode_98058
Step: score=266396.46, add zipcode_98077
Step: score=266307.45, add zipcode_98014
Step: score=266219.94, add zipcode_98136
Step: score=266135.73, add zipcode_98122
Step: score=266051.25, add zipcode_98010
Step: score=265970.80, add zipcode_98105
Step: score=265892.28, add zipcode_98005
Step: score=265813.07, add zipcode_98027
Step: score=265746.62, add zipcode_98108
Step: score=265687.22, add zipcode_98055
Step: score=265624.76, add zipcode_98106
Step: score=265567.35, add waterfront_1
Step: score=265520.24, add zipcode_98029
Step: score=265454.06, add long
Step: score=265412.67, add zipcode_98188
Step: score=265372.52, add zipcode_98033
Step: score=265332.87, add zipcode_98031
Step: score=265294.14, add zipcode_98056
Step: score=265257.81, add zipcode_98039
Step: score=265227.79, add renov_age
Step: score=265197.39, add zipcode_98146
Step: score=265171.10, add zipcode_98052
Step: score=265145.12, add zipcode_98059
Step: score=265125.06, add zipcode_98038
Step: score=265106.94, add floors
Step: score=265087.17, add zipcode_98198
Step: score=265066.71, add zipcode_98032
Step: score=265047.69, add zipcode_98118
Step: score=265030.04, add zipcode_98053
Step: score=265016.83, add zipcode_98030
Step: score=264999.86, add zipcode_98042
Step: score=264985.62, add zipcode_98074
Step: score=264970.56, add zipcode_98065
Step: score=264961.47, add zipcode_98148
Step: score=264950.55, add zipcode_98023
Step: score=264941.04, add zipcode_98166
Step: score=264932.66, add zipcode_98092
Step: score=264925.37, add zipcode_98024
Step: score=264920.13, add zipcode_98070
Step: score=264913.84, add zipcode_98003
Step: score=264908.86, add zipcode_98126
Step: score=264906.06, add zipcode_98002
Step: score=264905.19, add zipcode_98144
Step: score=264903.76, add zipcode_98103
Step: score=264901.26, add zipcode_98115
Step: score=264895.68, add zipcode_98107
Step: score=264870.10, add zipcode_98117
```

```
Step: score=264834.04, add zipcode_98045
Step: score=264814.32, add zipcode_98007
Step: score=264787.26, add zipcode_98008
Step: score=264640.48, add zipcode_98075
Step: score=264640.48, add None
The best varibles are: ['grade', 'lat', 'age', 'sqft_living15', 'bathrooms',
'zipcode_98004', 'zipcode_98155', 'zipcode_98133', 'sqft_above', 'sqft_baseme
nt', 'zipcode_98040', 'view', 'zipcode_98028', 'zipcode_98034', 'zipcode_9811
2', 'zipcode_98019', 'zipcode_98125', 'zipcode_98011', 'zipcode_98072', 'zipc
ode_98177', 'condition', 'zipcode_98119', 'zipcode_98109', 'zipcode_98022', '
zipcode_98116', 'zipcode_98102', 'zipcode_98006', 'zipcode_98199', 'zipcode_9
8178', 'zipcode_98168', 'zipcode_98058', 'zipcode_98077', 'zipcode_98014', 'z
ipcode_98136', 'zipcode_98122', 'zipcode_98010', 'zipcode_98105', 'zipcode_98
005', 'zipcode_98027', 'zipcode_98108', 'zipcode_98055', 'zipcode_98106', 'wa
terfront_1', 'zipcode_98029', 'long', 'zipcode_98188', 'zipcode_98033', 'zipc
ode_98031', 'zipcode_98056', 'zipcode_98039', 'renov_age', 'zipcode_98146', '
zipcode_98052', 'zipcode_98059', 'zipcode_98038', 'floors', 'zipcode_98198',
'zipcode_98032', 'zipcode_98118', 'zipcode_98053', 'zipcode_98030', 'zipcode_
98042', 'zipcode_98074', 'zipcode_98065', 'zipcode_98148', 'zipcode_98023', '
zipcode_98166', 'zipcode_98092', 'zipcode_98024', 'zipcode_98070', 'zipcode_9
8003', 'zipcode_98126', 'zipcode_98002', 'zipcode_98144', 'zipcode_98103', 'z
ipcode_98115', 'zipcode_98107', 'zipcode_98117', 'zipcode_98045', 'zipcode_98
007', 'zipcode_98008', 'zipcode_98075']
```

2. Multiple Regression (only variables)

In [114]:
```
# Use only the first 1000 records and select columns for regression analysis
predictors2 = df[['grade', 'lat','age','sqft_living15']]
outcome2 = df['price']
```

In [115]:
```
# Partition data into predictors (x) and output (y)
X2 = predictors2
y2 = outcome
```

In [116]:
```
# Split the data into training and validation
train_X2, valid_X2, train_y2, valid_y2 = train_test_split(X2, y2, test_size=0.
```

In [117]:
```
# Built the Linear Model based on the training data
df_lm2 = LinearRegression()
df_lm2.fit(train_X2, train_y2)
```

Out[117]:
```
LinearRegression()
```

In [118]:
```
#Print coefficients
print(pd.DataFrame({'Predictor': X2.columns, 'coefficient': df_lm2.coef_}))
```

```
       Predictor    coefficient
0          grade  109584.085093
1            lat  534014.865066
2            age    1894.846443
3  sqft_living15      94.982097
```

2. Performance Check

In [48]:
```
#Print performance measures(training data)
regressionSummary(train_y2, df_lm2.predict(train_X2))
```

```
Regression statistics

                      Mean Error (ME) : -0.0000
       Root Mean Squared Error (RMSE) : 122563.9553
            Mean Absolute Error (MAE) : 93327.2997
          Mean Percentage Error (MPE) : -6.6471
Mean Absolute Percentage Error (MAPE) : 22.6443
```

In [120]:
```
#Calculating the R2 score for the model (training data)
df_lm_predt2 = df_lm2.predict(train_X2)
score_lr_valt2 = adjusted_r2_score(train_y2, df_lm_predt2, df_lm2)*100
print("The R2 score for training data is:", score_lr_valt2.round(1),"%")
```

```
The R2 score for training data is: 62.0 %
```

In [121]:
```
#Print performace measures for the validation set
regressionSummary(valid_y2, df_lm2.predict(valid_X2))
```

```
Regression statistics

                      Mean Error (ME) : -39.6458
       Root Mean Squared Error (RMSE) : 122659.9450
            Mean Absolute Error (MAE) : 92964.2270
          Mean Percentage Error (MPE) : -6.5821
Mean Absolute Percentage Error (MAPE) : 22.5140
```

In [122]:
```
#Calculating the R2 score for the model
df_lm_pred2 = df_lm2.predict(valid_X2)
score_lr_val2 = adjusted_r2_score(valid_y2, df_lm_pred2, df_lm2)*100
print("The R2 score for validation data is:", score_lr_val2.round(1),"%")
```

```
The R2 score for validation data is: 60.9 %
```

In [123]:
```
#Make prediction of 2 houses in the validation dataset
df_lm_pred2 = df_lm2.predict(valid_X2)
result2 = pd.DataFrame({'Predicted': df_lm_pred2, 'Actual': valid_y2, 'Residua
result.sample(2)
```

Out[123]:

	Predicted	Actual	Residual
15476	202577.965503	205000	2422.034497
1371	412948.231053	338000	-74948.231053

In [ ]: