Open Data Machine Learning Use-Case:

Predicting the Saudi Agricultural Land Prices

By: Hanan Alsalamah

Types of Analytics

- **Types of Analytics:**

  - **Descriptive** Analytics, which use data aggregation and data mining to provide insight into the past and answer: “What has happened?”

  - **Predictive** Analytics, which use statistical models and forecasting techniques to understand the future and answer: “What could happen?”

  - **Prescriptive** Analytics, which use optimization and simulation algorithms to advise on possible outcomes and answer: “What should we do?”

Problem Statement

- Real estate agents get asked nearly every day about land prices. Predicting the expected average prices for the next years would help people to know the future market trends for their decision making.

- The aim of this project is to build a predictive model that can predict the agricultural land price given other factors such as whether it's residential or commercial land (Property Classification), deal date, area in square meters, and the land's region.

- In this use-case, we will use **"Agricultural Land Deals"** datasets provided by the Ministry of Justice. The deals occurred in different regions around the Kingdom of Saudi Arabia during 7 months period (From January 2019 till July 2019).

Dataset: Agricultural Lands Deals

- **No. of Features: 11**

- Region, City, AreaName, BlockNo., LotNo., Property Classification, DealDate, DealNo., AreaInSquareMeters, PricePerSquareMeters, DealPrice

- **No. of Observation:**

- 3505 records

Region	City	Area	BlockNo.	LotNo.	Classification	DealDate	DealNo.	DealPrice	AreaInSquareMeters	PricePerSquareMeters
منطقة نجران	نجران	حي/الغويلا	مخطط/أخرى	قطعة 485	تجاري	2019-01-21	8007832	90,000.000ر	20,000.00	4.5000
منطقة الجوف	القريات	حي/أخرى	مخطط/أخرى	قطعة 1	تجاري	2019-01-02	7937539	50,000.000ر	164,802.00	0.3033
منطقة الجوف	القريات	حي/أخرى	مخطط/أخرى	قطعة 4	تجاري	2019-01-31	8055899	10,000.000ر	122,890.00	0.0813
منطقة الجوف	دومة الجندل	حي/أخرى	مخطط/أخرى	قطعة 2	تجاري	2019-01-08	7955582	169,300.000ر	68,750.00	2.4625
منطقة الجوف	دومة الجندل	حي/الزراعية	مخطط/أخرى	قطعة بدون	تجاري	2019-01-31	8055590	60,000.000ر	67,000.00	0.8955
منطقة الجوف	سكاكا	حي/أخرى	مخطط/أخرى	قطعة 1	تجاري	2019-01-13	7975995	850,000.000ر	50,103.00	16.9650
منطقة الجوف	سكاكا	حي/أخرى	مخطط/أخرى	قطعة 2	تجاري	2019-01-28	8040117	1,500,050.000ر	75,000.00	20.0006
منطقة الجوف	سكاكا	حي/أخرى	مخطط/أخرى	قطعة 3	تجاري	2019-01-07	7921288	5,803,917.000ر	8,291.31	700.0000
منطقة الجوف	طبرجل	حي/أخرى	مخطط/أخرى	قطعة 3	تجاري	2019-01-29	8042911	80,000.000ر	34.13	2,343.9102
منطقة الجوف	طبرجل	حي/أخرى	مخطط/أخرى	قطعة بدون	تجاري	2019-01-27	8030872	500,000.000ر	89,550.44	5.5834
منطقة الجوف	طبرجل	حي/أخرى	مخطط/أخرى	قطعة بدون	تجاري	2019-01-27	8030989	800,000.000ر	98,428.28	8.1277
منطقة الجوف	طبرجل	حي/بسيطاء	مخطط/2732 لوحة 15 ق/ 7/ 3/ 1408	قطعة 19	تجاري	2019-01-14	7980052	100,000.000ر	1,000,000.00	0.1000
منطقة الجوف	طبرجل	حي/بسيطاء	مخطط/2732 لوحه 21 ق/ 8/ 5/ 1410	قطعة 52	تجاري	2019-01-21	8008375	100,000.000ر	1,000,000.00	0.1000
منطقة الجوف	طبرجل	حي/بسيطاء	مخطط/2732 لوحه 24 ق/ 15/ 1/ 1414	قطعة 948	تجاري	2019-01-09	7964781	50,000.000ر	1,000,000.00	0.0500
منطقة الجوف	طبرجل	حي/بسيطاء	مخطط/2732 لوحه 24 ق/ 15/ 1/ 1414	قطعة 950	تجاري	2019-01-09	7964496	50,000.000ر	1,000,000.00	0.0500
منطقة الرياض	الافلاج	حي/الغيل	مخطط/أخرى	قطعة بدون	سكنى	2019-01-30	8048854	140,000.000ر	418,191.00	0.3347
منطقة الرياض	الافلاج	حي/شرق السيح	مخطط/83	قطعة 7	سكنى	2019-01-02	7937942	60,000.000ر	54,277.00	1.1054

Exploratory Data Analysis (EDA)

```
# Print information about a DataFrame including the column dtypes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3505 entries, 0 to 3504
Data columns (total 11 columns):
Region                  3505 non-null object
City                    3505 non-null object
AreaName                3505 non-null object
BlockNo.                3505 non-null object
LotNo.                  3505 non-null object
Classification          3505 non-null object
DealDate                3505 non-null object
DealNo.                 3505 non-null int64
AreaInSquareMeters      3505 non-null float64
PricePerSquareMeters    3505 non-null float64
DealPrice               3505 non-null int64
dtypes: float64(2), int64(2), object(7)
memory usage: 301.3+ KB
```

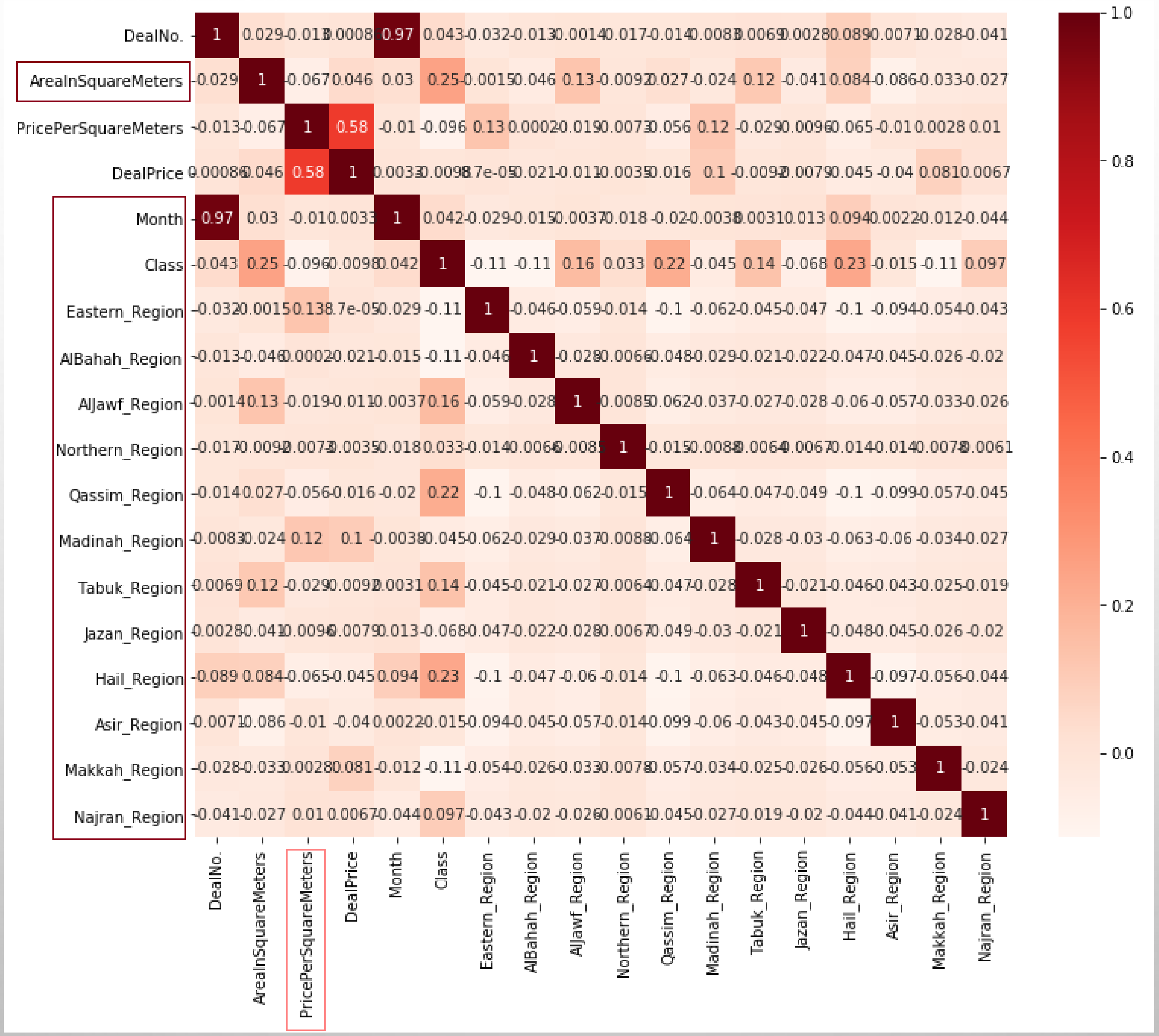
```
# Generate descriptive statistics for all numeric features
df.describe()
```

	DealNo.	AreaInSquareMeters	PricePerSquareMeters	DealPrice	Month
count	3505.00	3505.00	3505.00	3505.00	3505.00
mean	8330935.84	109544.07	79.29	682219.15	3.84
std	248401.49	332012.95	362.36	3751614.88	2.05
min	7096821.00	32.25	0.01	10000.00	1.00
25%	8121240.00	3290.00	1.80	70000.00	2.00
50%	8317340.00	10944.86	16.82	126000.00	4.00
75%	8537891.00	54400.00	62.13	400000.00	6.00
max	8771942.00	6000000.00	15004.57	135617150.00	7.00

```
1613]: # Convert to datetime
       df['DealDate'] = pd.to_datetime(df['DealDate'], format='%m/%d/%Y', errors='coerce')
```

```
# Extract the month only
df['Month'] = df['DealDate'].dt.month
```

Data Visualization (Correlation matrix)



Feature Engineering (Dummy Variables)

Encoding Categorical Features:

Dummy variables have been created using **get_dummies** from **Pandas**

- **Property Classification:** Residential/ Commercial

```
# Create dummy encoding (0/1) for the categorical variable
classification_dummies=pd.get_dummies(df.Classification, prefix='Classification')
```

Classification
تجاري
تجاري
تجاري
تجاري
تجاري
تجاري
سكني
سكني
سكني
سكني

Commercial	Residential
0	1
0	1
0	1
0	1
1	0

Class
0
0
0
0
1

- **Region:** 13 regions

```
region_dummies=pd.get_dummies(df.Region, prefix='Region')
```

Eastern_Region	AlBahah_Region	AlJawf_Region	Northern_Region	Riyadh_Region	Qassim_Region	Madinah_Region	Tabuk_Region	Jazan_Region	Hail_Region
1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1

```
# Total by Region
region_dummies.sum()
```

```
Eastern_Region      312
AlBahah_Region       75
AlJawf_Region       121
Northern_Region       7
Riyadh_Region      1589
Qassim_Region       338
Madinah_Region      131
Tabuk_Region         70
Jazan_Region         77
Hail_Region         325
Asir_Region         293
Makkah_Region       103
Najran_Region        64
dtype: int64
```

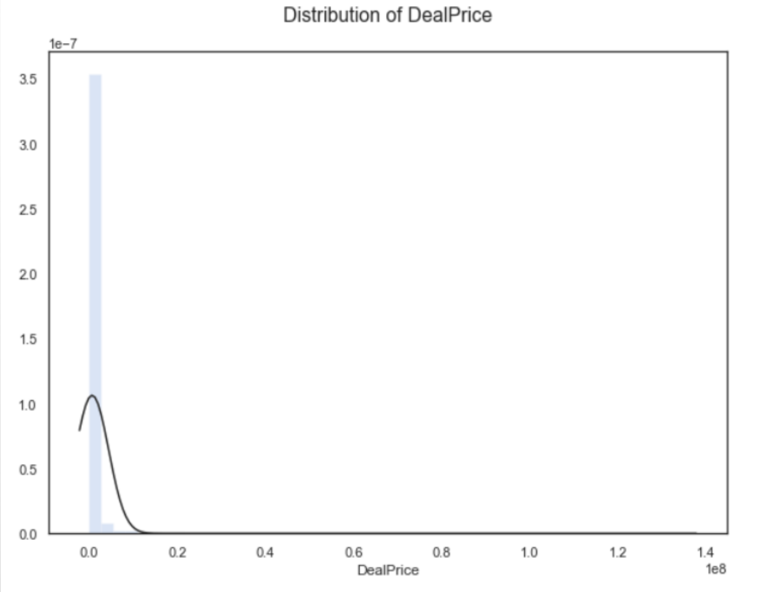
Feature Engineering (Slicing)

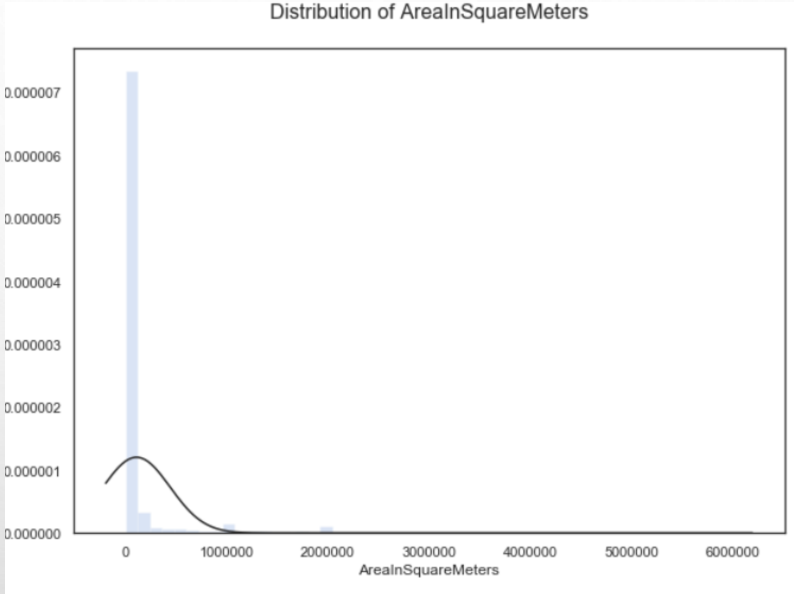
Remove Unnecessary text:

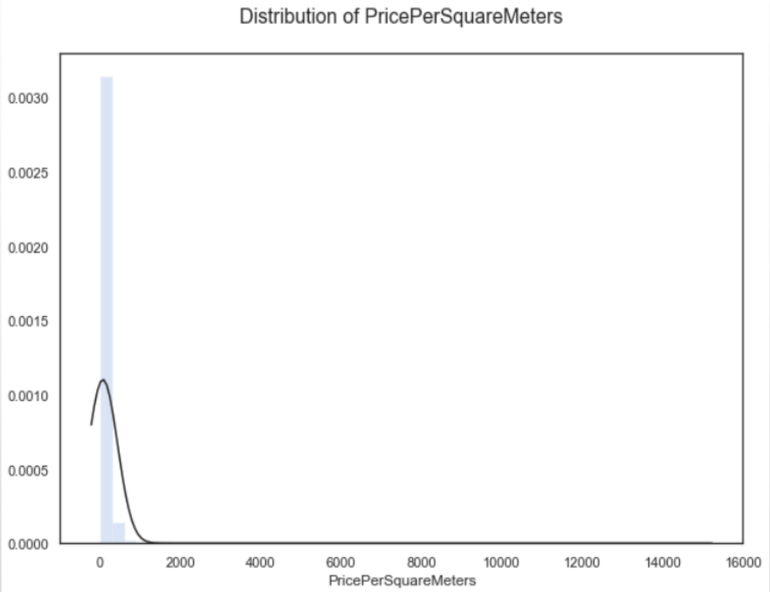
```python
# Remove the unnecessary text appears before region name in each row
df['Region'] =  df['Region'].str[6:]
# OR df.Region = df.Region.str.slice(6,)

# Remove the unnecessary text appears before Lot number in each row
df['LotNo.'] =  df['LotNo.'].str[5:]

# Remove the unnecessary text appears before Area name in each row
df['AreaName'] =  df['AreaName'].str[3:]

# Remove the unnecessary text appears before Block number in each row
df['BlockNo.'] =  df['BlockNo.'].str[5:]
```

A	B	C	D	E
Region	City	Area	BlockNo.	LotNo.
منطقة الرياض	الدرعيه	حي/العمارية	مخطط/قطعة رقم 20/ أ والقطعة رقم 1/ 20	قطعة 45
منطقة الرياض	الدرعيه	حي/العمارية	مخطط/من أصل القطعة رقم 2	قطعة 868
منطقة الرياض	الدرعيه	حي/العمارية	مخطط/من أصل القطعة رقم 61	قطعة 1220
منطقة الرياض	الدرعيه	حي/العمارية	مخطط/من أصل القطعة رقم 61	قطعة 1230
منطقة الرياض	الدرعيه	حي/العمارية	مخطط/من أصل القطعة رقم 61	قطعة 16
منطقة الرياض	الدرعيه	حي/العمارية	مخطط/من أصل القطعة رقم 61	قطعة 272

Data Visualization (Distribution Plots)







- Plot the distributions of the target (PricePerSquareMeters) and numeric features (Area In Square Meters, DealPrice)

Skewness

Skewness:

Measure of the asymmetry of the distribution

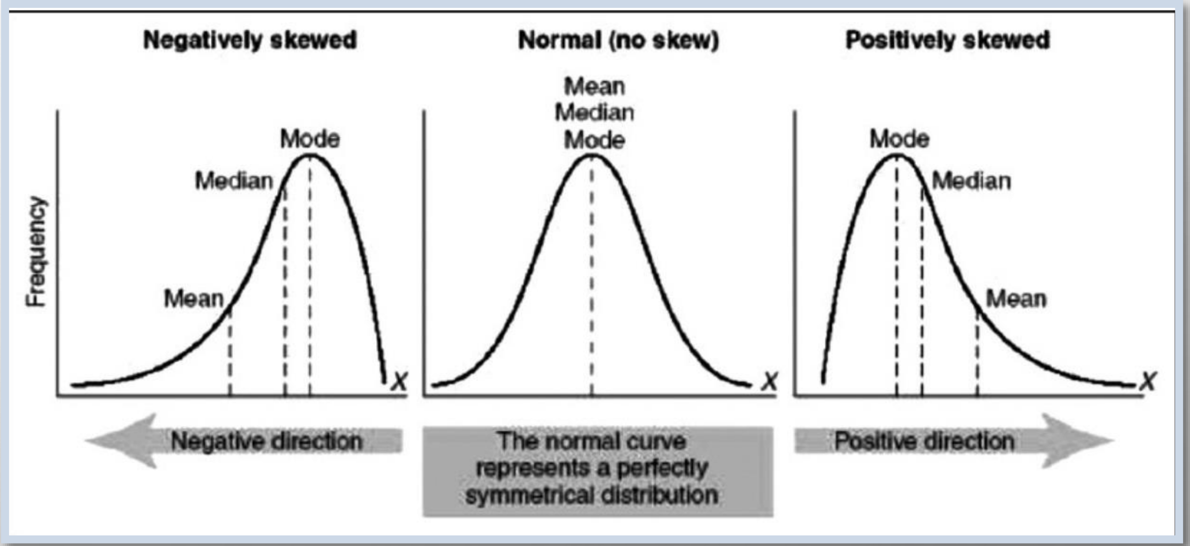
2 positively skewed variables have the highest skewness:

- Deal Price

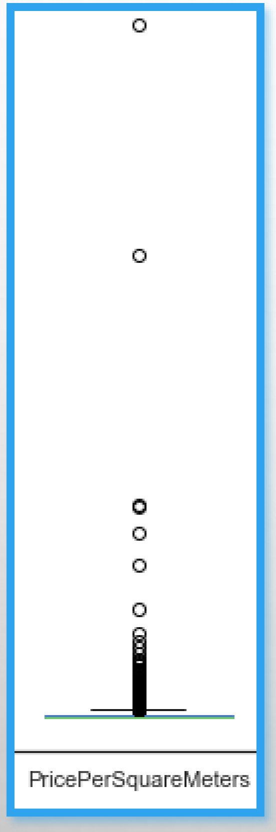
- Price Per Square Meters

```python
# check the skewness (asymmetry in the distribution) for all numeric features
for col in df._get_numeric_data():
    print(col,df[col].skew())
```

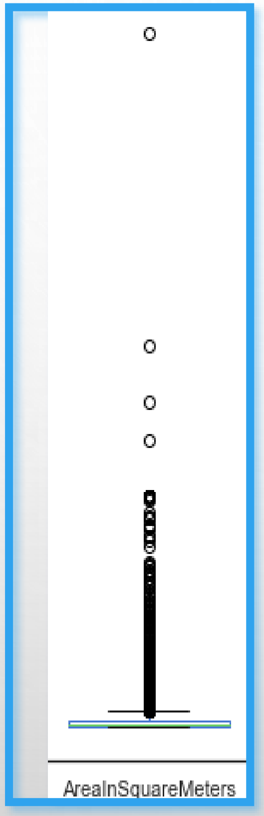
```
DealNo. 0.00951497306025
AreaInSquareMeters 5.6655051731508985
PricePerSquareMeters 27.838645617319898
DealPrice 23.101561893028354
```



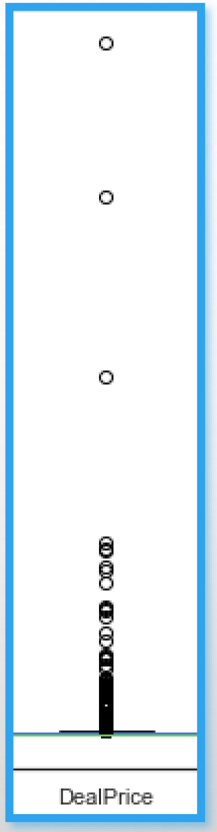
Data Visualization (Box Plots)



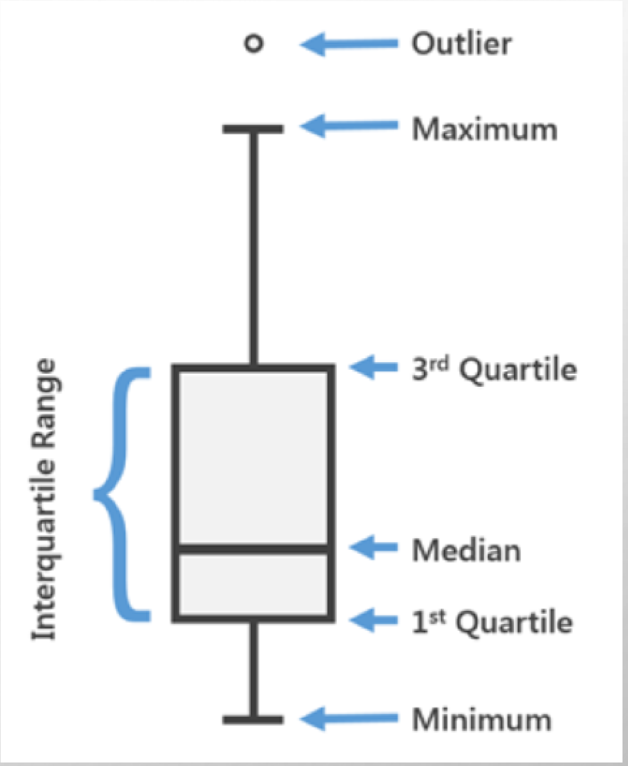
25%	1.80
50%	16.82
75%	62.13



25%	3290.00
50%	10944.86
75%	54400.00



25%	70000.00
50%	126000.00
75%	400000.00



Detect and Remove the Outliers

Methodology: IQR score

The interquartile range (IQR), is a measure of statistical dispersion, being equal to the difference between 75th and 25th percentiles, or between upper and lower quartiles.

IQR = Q3 − Q1

Applied Over:

-Price Per Square Meters

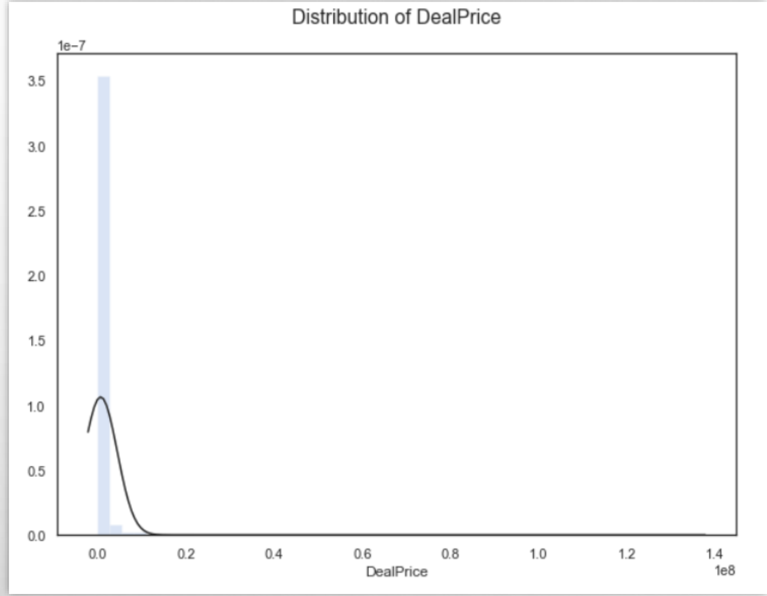
-Deal Price

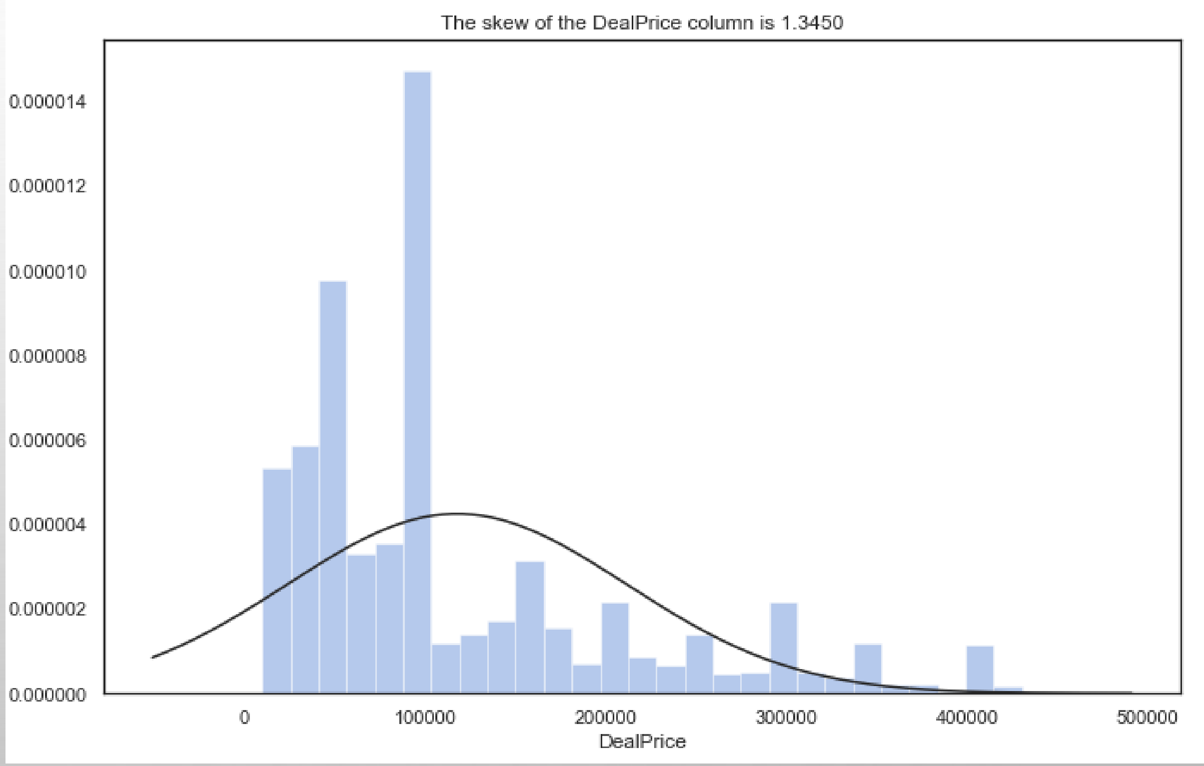
-Area In Square Meters

```
# Define a method to remove outliers
def remove_outlier(df, col):
  q1 = df[col].quantile(0.25)
  q3 = df[col].quantile(0.75)
  iqr = q3 - q1
  lower_bound  = q1 - (1.5  * iqr)
  upper_bound = q3 + (1.5 * iqr)
  out_df = df.loc[(df[col] > lower_bound) & (df[col] < upper_bound)]
  return out_df
```

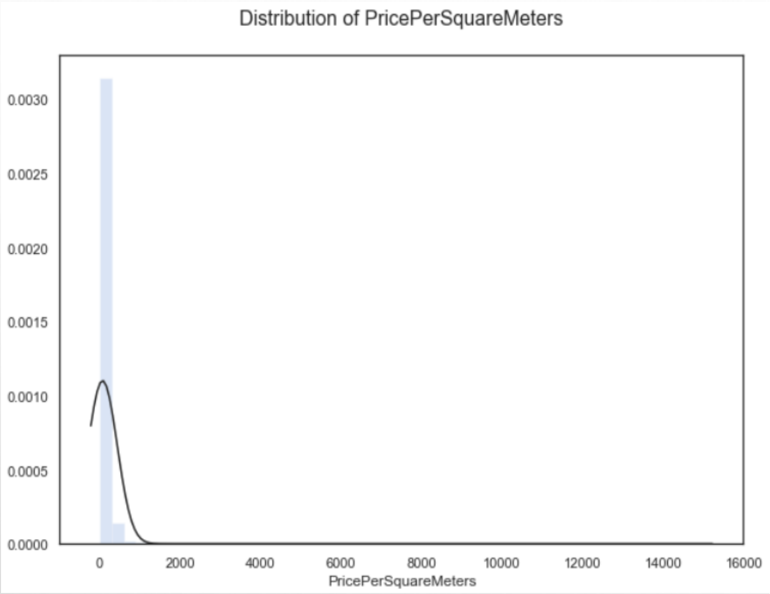
```
df = remove_outlier(df,"DealPrice")
df = remove_outlier(df,"PricePerSquareMeters")
df = remove_outlier(df,"AreaInSquareMeters")
```

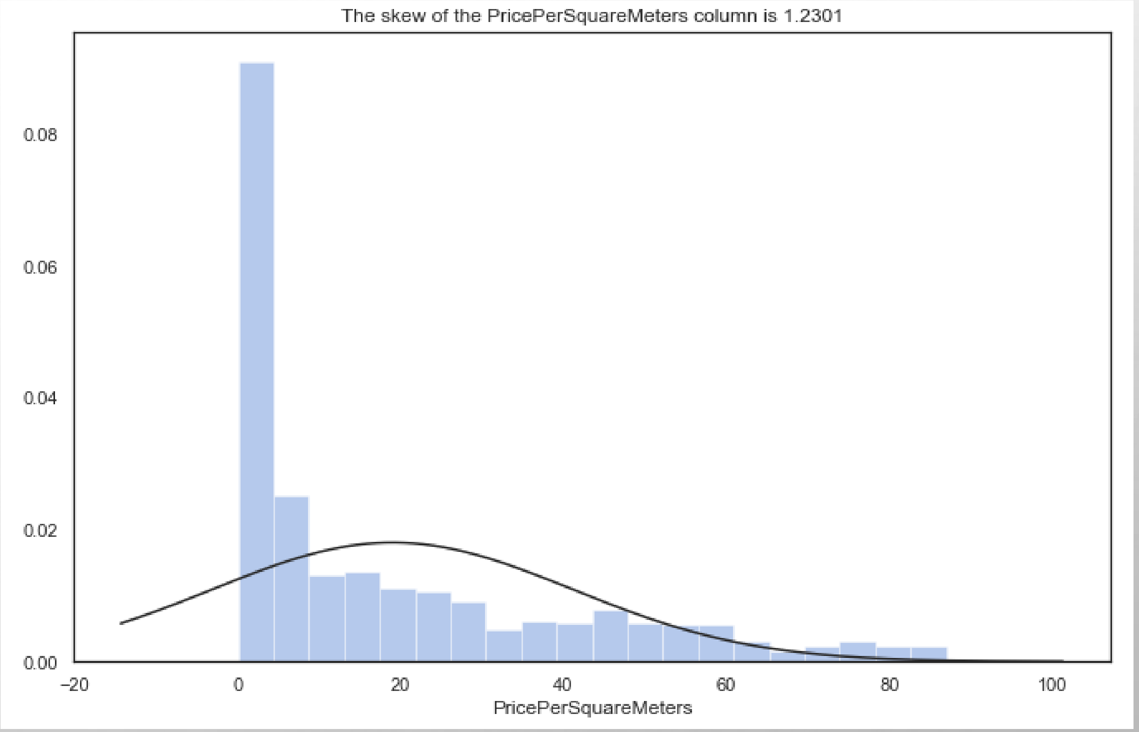
Detect and Remove the Outliers (Cont.)



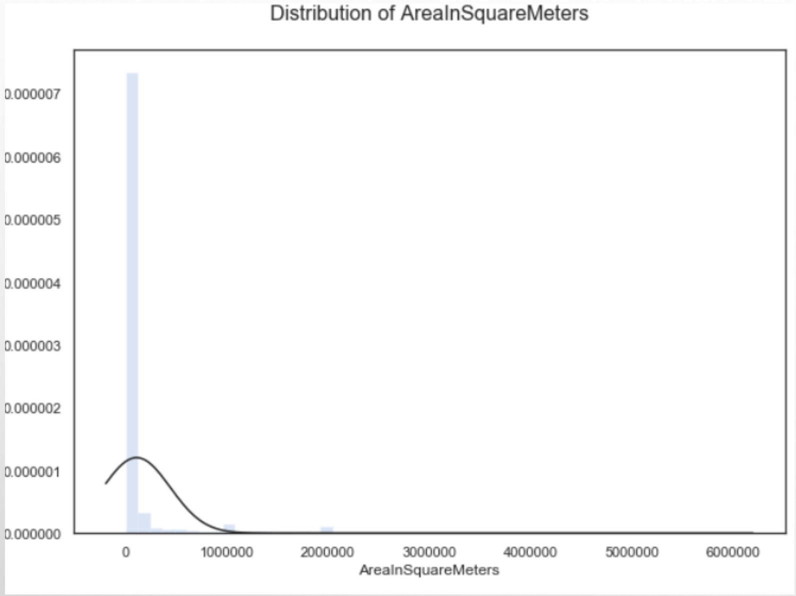


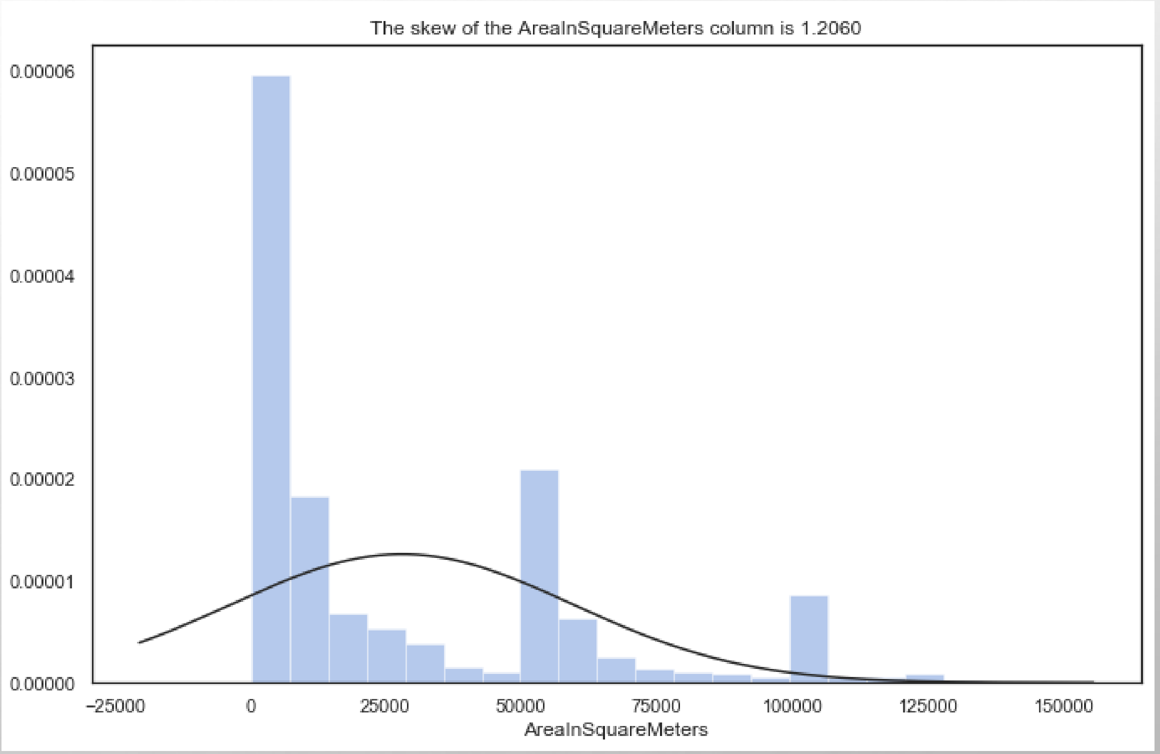
Detect and Remove the Outliers (Cont.)





Detect and Remove the Outliers (Cont.)

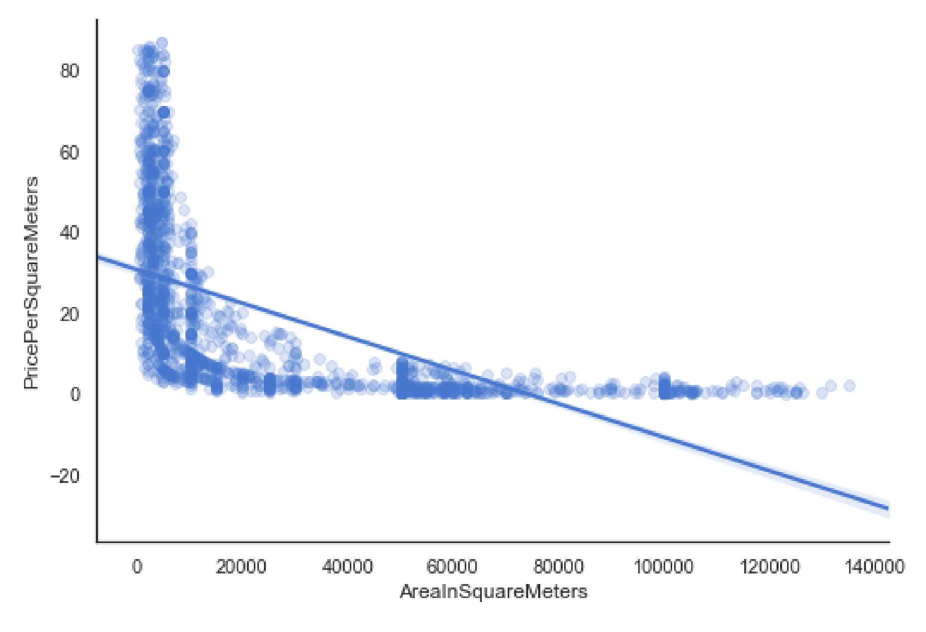




Regression Model

- **Challenge:**

  Linear Regression Algorithm:
  Linear regression requires the relation between the dependent variable and the independent variable to be linear. However, the features in the current dataset has a slight non-linear variation with the target variable (as shown in the image)



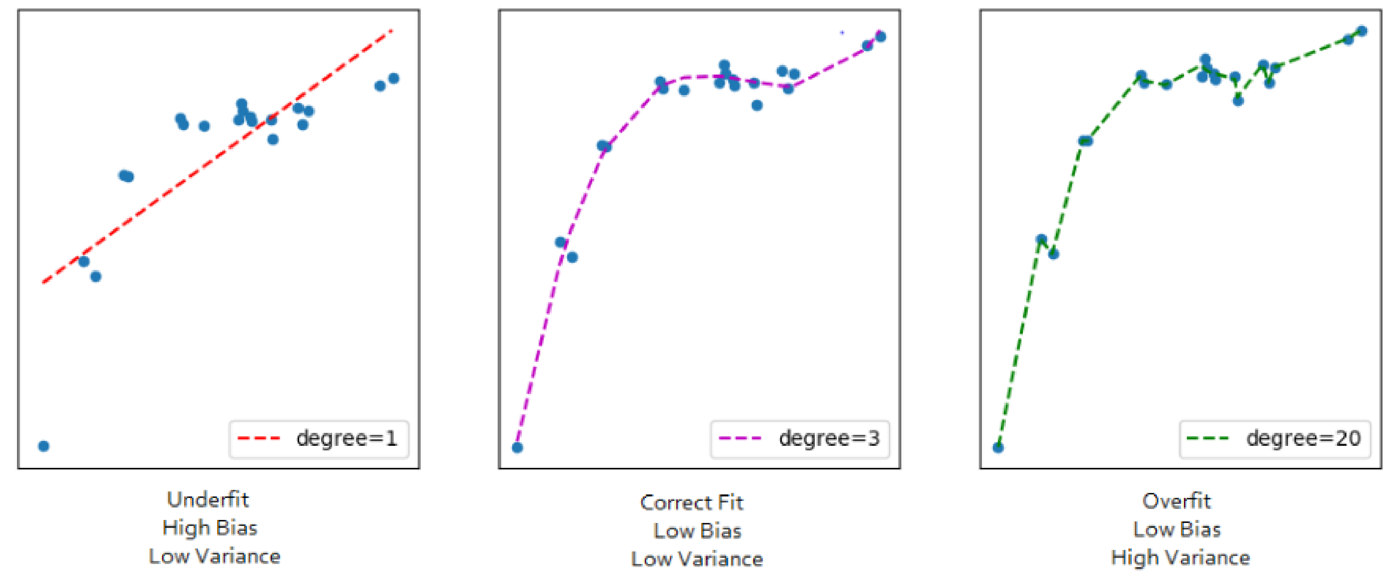
- **Best Practice:**

  1- Apply **"Polynomial regression"** to transform the original features into higher degree polynomials before training the model.

  2- Apply **"Linear Regression"**

Polynomial Regression

- **Polynomial Features:**
  Increasing the complexity of the model by applying Polynomial regression in order to generate a curve that best captures the data and overcome under-fitting.



- **(X) Features:**
  Area In Square Meters, Class, Month, Regions

- **Target (Y):** Price Per Square Meters

Train/Test Split

```
# Split x features and y into random train and test subsets
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=42)
```

Applying Polynomial Regression

```
# Generate a new feature matrix consisting of all polynomial combinations
poly_features= PolynomialFeatures(degree=2)

# transforms the existing features to higher degree features.
x_train_poly = poly_features.fit_transform(x_train)
x_test_poly = poly_features.fit_transform(x_test)
```

Linear Regression

Regression analysis is a statistical technique used to estimate the relationship between a dependent/target variable (property price) and single or multiple independent variables (predictors that impact the target variable).

Applying Linear Regression

Instantiate and fit a `LinearRegression` model on x and y from the `linear_model` section of scikit-learn.

```
# Instantiate (Make an instance of a LinearRegression object) and fit the transformed featur
lr = LinearRegression()
lr.fit(x_train_poly, y_train)
```

Using the Model for Prediction

```
# Using the Model for Prediction
y_lr_pred = lr.predict(x_test_poly)
```

PricePerSquareMeters	pred_PricePerSquareMeters
26.6666	37.59231653
44.0729	40.43386603
33	28.44457776
2.9106	3.173857461
0.9433	0.978450767
0.896	0.301554405
28.9256	24.98917879
30	37.72695171
28	33.26153678
6.3681	4.505356794
50	36.44693455
49.5049	40.18477374
17.1467	15.33862672
34	33.26153678
24	30.08857816
37.1666	32.18868343
28.6885	32.45915091
39.5583	32.62225398

Evaluation Metrics for Regression Problem

```python
# Linear metrics
from sklearn import metrics
import numpy as np
print('MAE:', metrics.mean_absolute_error(y_test, y_lr_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_lr_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_lr_pred)))
print('R2 score:', metrics.r2_score(y_test, y_lr_pred))
print('Mean absolute percentage error (MAPE):', np.mean(np.abs((y_test - y_lr_pred) / y_test)) * 100)
```

```
MAE: 10.145891661005336
MSE: 207.33587830058278
RMSE: 14.399162416633224
R2 score: 0.5418239238247964
Mean absolute percentage error (MAPE): 269.09470846054614
```

- Mean absolute error (MAE)
- Mean squared error (MSE)
- Root mean squared error (RMSE)
- R-squared (R2)
- Mean absolute percentage error (MAPE)

Building Other Regression Models

Applying other Regression Models:

- Huber Regression
- Ransac Regression
- Theil-Sen Regression
- Ridge Regression
- Lasso Regression
- Support Vector Regression
- K-Nearest Neighbors Regression
- Decision Tree Regression
- Random Forest Regression

```
# Huber metrics
get_metrics(y_test, y_huber_pred)
```

```
MAE: 17.055
MedAE: 6.455
MSE: 746.93
RMSE: 27.33
R2 Score: -0.651
MAPE: 109.43215545609408
```

```
# RANSAC metrics
get_metrics(y_test, y_ransac_pred)
```

```
MAE: 12.286
MedAE: 6.625
MSE: 343.882
RMSE: 18.544
R2 Score: 0.24
MAPE: 341.7513786051783
```

```
# Ridge metrics
get_metrics(y_test, y_ridge_pred)
```

```
MAE: 10.146
MedAE: 6.265
MSE: 207.335
RMSE: 14.399
R2 Score: 0.542
MAPE: 269.1104424002256
```

```
# Lasso metrics
get_metrics(y_test, y_lasso_pred)
```

```
MAE: 10.456
MedAE: 6.91
MSE: 212.231
RMSE: 14.568
R2 Score: 0.531
MAPE: 305.4060271520618
```

```
# Theil-Sen metrics
get_metrics(y_test, y_TheilSen_pred)
```

```
MAE: 10.86
MedAE: 6.845
MSE: 229.815
RMSE: 15.16
R2 Score: 0.492
MAPE: 366.7002608616875
```

```
# SVR metrics
get_metrics(y_test, y_svr_pred)
```

```
MAE: 13.539
MedAE: 6.965
MSE: 456.083
RMSE: 21.356
R2 Score: -0.008
MAPE: 412.80070600461187
```

```
# KNN metrics
get_metrics(y_test, y_neigh_pred)
```

```
MAE: 9.087
MedAE: 3.563
MSE: 218.211
RMSE: 14.772
R2 Score: 0.518
MAPE: 110.17348591704084
```

```
# Decision Tree metrics
get_metrics(y_test, y_dtree_pred)
```

```
MAE: 8.309
MedAE: 2.848
MSE: 214.356
RMSE: 14.641
R2 Score: 0.526
MAPE: 116.82133029399684
```

```
# Random Forest metrics
get_metrics(y_test, y_rf_pred)
actual_error_rate = np.square(np.subtract(y_test,y_rf_pred)).mean()
print("Actual Error Rate  = ",actual_error_rate)
```

```
MAE: 7.757
MedAE: 2.672
MSE: 172.82
RMSE: 13.146
R2 Score: 0.618
MAPE: 111.36134591926891
Actual Error Rate  =  172.8196333370651
```

Future Work

- Add more features that significantly explain the target (e.g., data about neighborhood, amenities, schools, hospitals etc.).

- Add more data records.

- Examine multiple algorithms

- Deployment of a model into a software system or application.

THANK YOU