Assignment 1

LINEAR REGRESSION

Submitted By: Neel Vijay Patil (11436004)

Submitted To: Dr. Junhua Ding

INFO 5505

Applied Machine Learning for Data Science

UNT

The main objective of this assignment is to design a predictive model which helps to predict the dependent variable (DV) price based on the other independent variables (IV) such as height, width etc. The Monet dataset consist of the 6 different variables/attributes. price, height, width, signed, picture, house.

In this assignment I have utilized 2 types of the linear regression models

I. Simple Linear Regression
II. Multivariate Linear Regression

1)Importing Libraries

I have imported the required libraries.

```
[1] # Importing libraries
    import pandas as pd
    import numpy as np
    import seaborn as sb
    import matplotlib.pyplot as plt
    %matplotlib inline
    import missingno as mn
    from scipy.stats import skew
```

```
[21] #importing libraries
     from sklearn import linear_model
     from sklearn import preprocessing
     from sklearn.model_selection import train_test_split
     from sklearn.model_selection import cross_val_score
```

2) Imported the data into google Colab

```
#importing data
from google.colab import files
uploaded = files.upload()
```

```
Choose Files monet.csv
• monet.csv(text/csv) - 16875 bytes, last modified: 7/16/2022 - 100% done
Saving monet.csv to monet.csv
```

3) Read the csv file

```python
# Read Data
import io
data = pd.read_csv('monet.csv')
```

```python
data.head()
```

	PRICE	HEIGHT	WIDTH	SIGNED	PICTURE	HOUSE
0	3.993780	21.3	25.6	1	1	1
1	8.800000	31.9	25.6	1	2	2
2	0.131694	6.9	15.9	0	3	3
3	2.037500	25.7	32.0	1	4	2
4	1.487500	25.7	32.0	1	4	2

```python
data.tail()
```

	PRICE	HEIGHT	WIDTH	SIGNED	PICTURE	HOUSE
425	5.2825	25.6	39.4	1	375	1
426	9.3525	25.9	39.6	1	375	2
427	8.2525	25.6	39.6	1	375	1
428	3.4100	25.6	39.4	1	386	2
429	1.5425	25.7	32.0	1	387	1

Checking the shape of dataset

The Monet dataset has 6 features and 430 observations.

```python
#checking the dimensions of dataset
data.shape
```

```
(430, 6)
```

4) Checking the Monet dataset information like column, non-Null content and the data type

The Monet dataset has `float64(3), int64(3)` datatypes and does not have any null Values.

If we have the missing values, we can either replace with the mean or median or delete the particular observation

```
#checking for the datatypes, non-null counts and columnname
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 430 entries, 0 to 429
Data columns (total 6 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   PRICE    430 non-null    float64
 1   HEIGHT   430 non-null    float64
 2   WIDTH    430 non-null    float64
 3   SIGNED   430 non-null    int64  
 4   PICTURE  430 non-null    int64  
 5   HOUSE    430 non-null    int64  
dtypes: float64(3), int64(3)
memory usage: 20.3 KB
```

5)Checking the Statistics of dataset

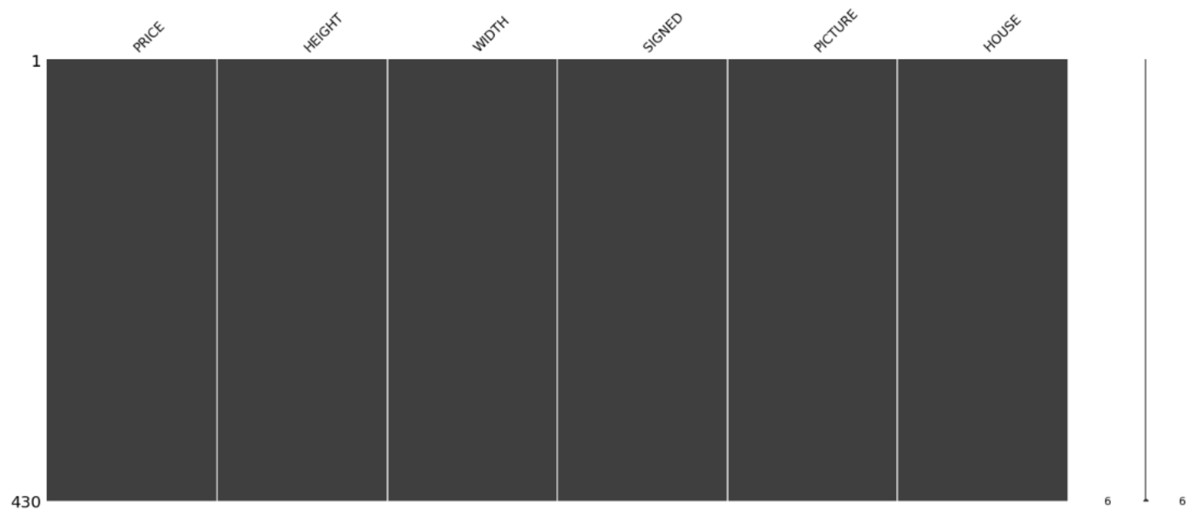
```
#Checking statistics of dataset
data.describe()
```

	PRICE	HEIGHT	WIDTH	SIGNED	PICTURE	HOUSE
count	430.000000	430.000000	430.000000	430.000000	430.000000	430.000000
mean	3.089996	27.646977	32.111395	0.820930	182.644186	1.611628
std	4.311260	10.097013	10.459677	0.383857	109.091529	0.591592
min	0.010413	3.900000	6.700000	0.000000	1.000000	1.000000
25%	0.600153	23.125000	28.525000	1.000000	87.250000	1.000000
50%	1.312782	25.600000	31.900000	1.000000	179.500000	2.000000
75%	3.850000	31.450000	36.200000	1.000000	274.750000	2.000000
max	33.013504	78.700000	89.000000	1.000000	387.000000	3.000000

6**) Representation of the missing values using the missingno**

```
[9] #checking for missing values
mn.matrix(data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3132be6cd0>
```



7) Creating the size column we multiplied the height and width column and added to the same data frame.

```
[10] # Creating size column
     data['SIZE'] = data['HEIGHT']*data['WIDTH']
```

```
[11] data.head()
```

	PRICE	HEIGHT	WIDTH	SIGNED	PICTURE	HOUSE	SIZE
0	3.993780	21.3	25.6	1	1	1	545.28
1	8.800000	31.9	25.6	1	2	2	816.64
2	0.131694	6.9	15.9	0	3	3	109.71
3	2.037500	25.7	32.0	1	4	2	822.40
4	1.487500	25.7	32.0	1	4	2	822.40

8) Calculating the Skewness coefficient

The skewness coefficient gives me an idea whether the variable is positively/negatively skewed, or it is normally distributed that means when plotted follows the gaussian curve. If the attribute is not normally distributed, then I carried out log transformation to make it evenly distributed.

```
[12] #Checking for skewness
for column in data.columns:
  if data.dtypes[column] != np.object:
    print(column ,' = ' , skew(data[column], axis=0, bias=True, nan_policy='omit'))
```

```
PRICE  =  2.8309820163618893
HEIGHT  =  1.8263892242940685
WIDTH  =  2.048113094558297
SIGNED  =  -1.674080612166068
PICTURE  =  0.07007980158027305
HOUSE  =  0.3733324102363447
SIZE  =  3.8831941413460154
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this warning, use `object` by itself.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  This is separate from the ipykernel package so we can avoid doing imports until
```

Here we can see features price, height, width, size are positively skewed while the attribute signed is negatively skewed.

9) Log transformation

We can say that log transformation is the method to make data normally distributed if the data is left/right skewed. The log transformation converts the y variable into log(y).

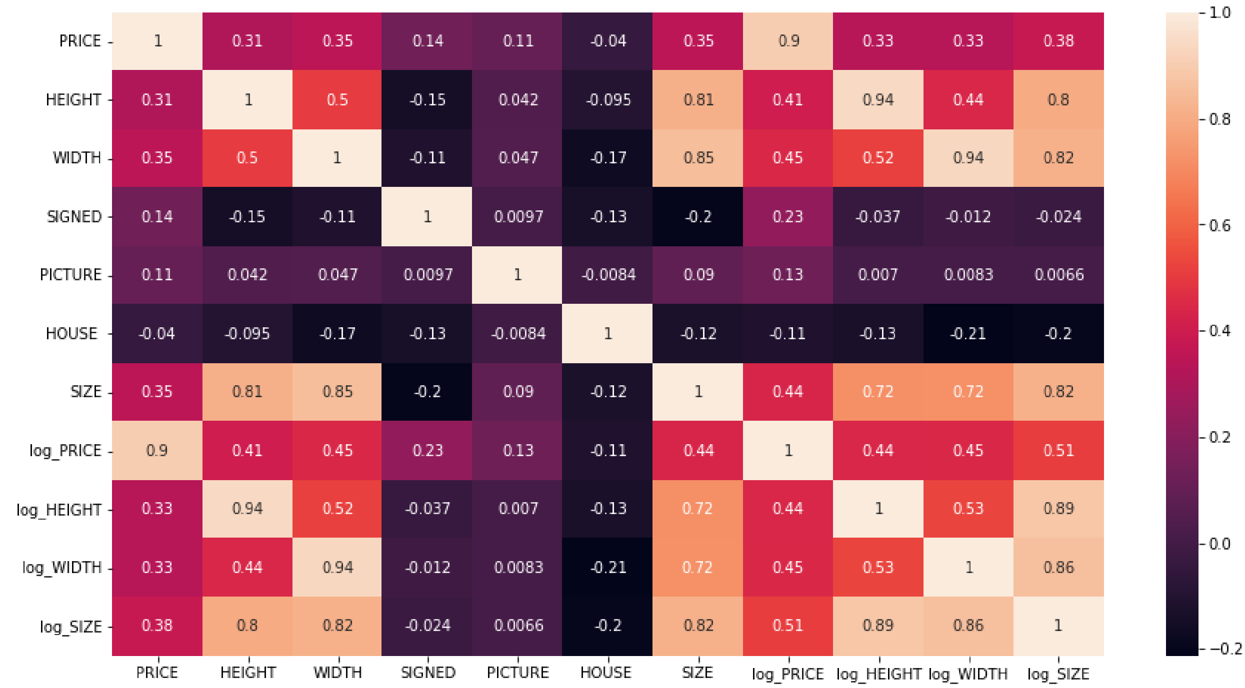
```
[13] # Log transformation

     to_be_transformed = ['PRICE','HEIGHT','WIDTH','SIZE']
     data_log = data[to_be_transformed].applymap(lambda x: np.log(x+1))
```

```
     # columns renaming
     data_log.columns = 'log_' + data_log.columns
```

```
[15] #Checking Skewness of log attributes
     data_log.skew()
```

```
log_PRICE     0.837605
log_HEIGHT   -0.469233
log_WIDTH    -0.590784
log_SIZE     -0.824549
dtype: float64
```

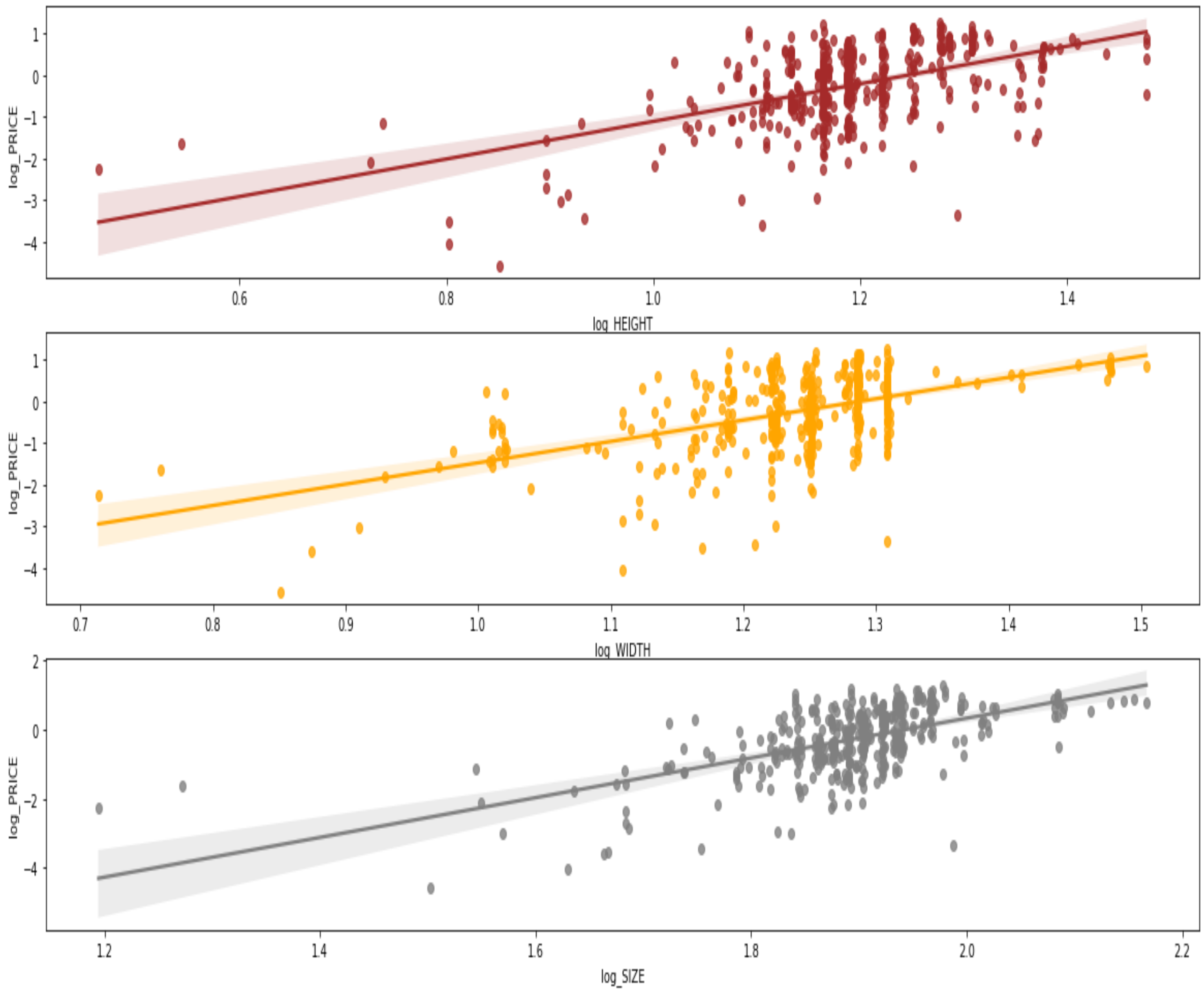
10) Correlation Matrix between the variables.



I have used heatmap to show find relationship between different features, Herby, I concluded that price of the painting is mostly correlated firstly with the size and then with width and third with the height. While, other 3 features are very less correlated.

11) Relationship between the dependent and independent variable using scatter plot to find corelation

```
[20] # Converting numerical features into log normal values
     #Relation between IV and DV
     fig, (ax1,ax2,ax3) = plt.subplots(nrows=3, ncols=1, figsize=(20,10))
     sb.regplot(x='log_HEIGHT',y='log_PRICE',data=monetdataset.apply(np.log), scatter=True, fit_reg=True, ax=ax1, color='brown')
     sb.regplot(x='log_WIDTH',y='log_PRICE',data=monetdataset.apply(np.log), scatter=True, fit_reg=True, ax=ax2, color='orange')
     sb.regplot(x='log_SIZE',y='log_PRICE',data=monetdataset.apply(np.log), scatter=True, fit_reg=True, ax=ax3, color='grey')
```



12) Simple linear regression 1

Independent variable= `log_HEIGHT`

Dependent variable= `log_PRICE`

The log_ height is taken as X-Dataset and y-dataset log_price and used train_test_split into 75% training data and 25% testing data.

```python
# Simple Linear Regression model 1

X = monetdataset[['log_HEIGHT']].values
y= monetdataset['log_PRICE'].values
```

```python
[23] #Train_Test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

```python
[24] lr_model1 = linear_model.LinearRegression()
lr_model1.fit(X_train, y_train)
```

```
LinearRegression()
```

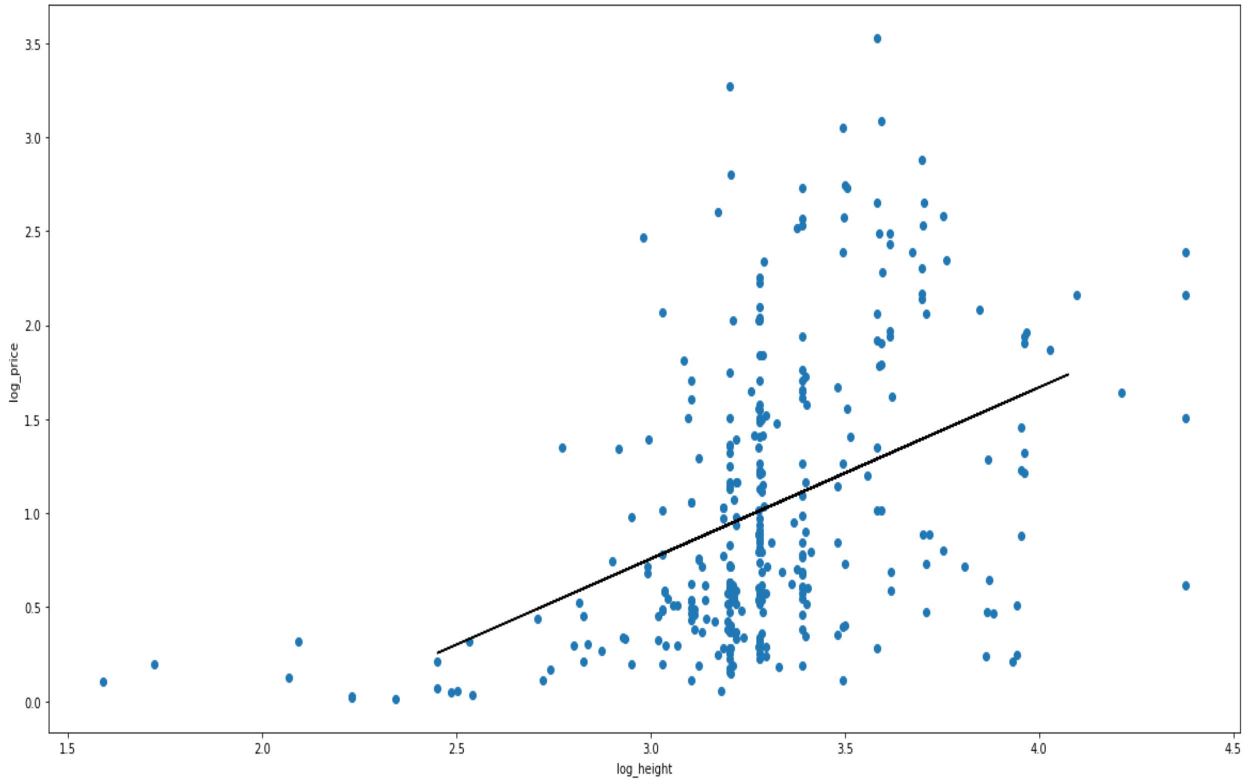
```python
[25] y_pred = lr_model1.predict(X_test)
```

```python
[26] price_data = pd.DataFrame({'Actual Price': y_test, 'Predicted Price': y_pred})
price_data
```

	Actual Price	Predicted Price
0	0.646056	1.012690
1	0.634458	0.490241
2	0.566889	1.065961
3	0.301585	0.748035
4	1.556564	1.012690

Regression Plot for the model

```python
[27] # Linear Regression plot for this model 1
fig, ax = plt.subplots(figsize=(20,10))
plt.scatter(X_train, y_train)
plt.plot(X_test, y_pred, color = 'black')
plt.xlabel("log_height")
plt.ylabel("log_price")
plt.show()
```



From the regression plot it can be concluded that log height and log_price are corelated with each other linearly. The given regression line indicates the predicted price for the respective height values. The blue dots represent the actual price values.

The root mean square error is the AVG difference in the predicted painting prices, actual painting prices.

Here RMSE indicates the prediction error using the testing data set.

```
[28] # Calculating RMSE
import sklearn.metrics as metrics
root_mean_square_error = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print(root_mean_square_error)
```

```
0.7241978635131306
```

```
[29] # Checcking mean absolute error

mean_absolute_error = metrics.mean_absolute_error(y_test, y_pred)
print(mean_absolute_error)
```

```
0.5744860226075414
```

The RMSE Value was found to be 0.72

13) Simple linear regression 2

Independent variable is log_size

Dependent variable is log_price

The log_ size is taken as X-Dataset and y-dataset log_price and used train_test_split into 75% training data and 25% testing data.

```
# Simple Linear Regression model 2
X = monetdataset[['log_SIZE']].values
y= monetdataset['log_PRICE'].values
```

```
[31] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

```
[32] lr_model2 = linear_model.LinearRegression()
     lr_model2.fit(X_train, y_train)

     LinearRegression()
```

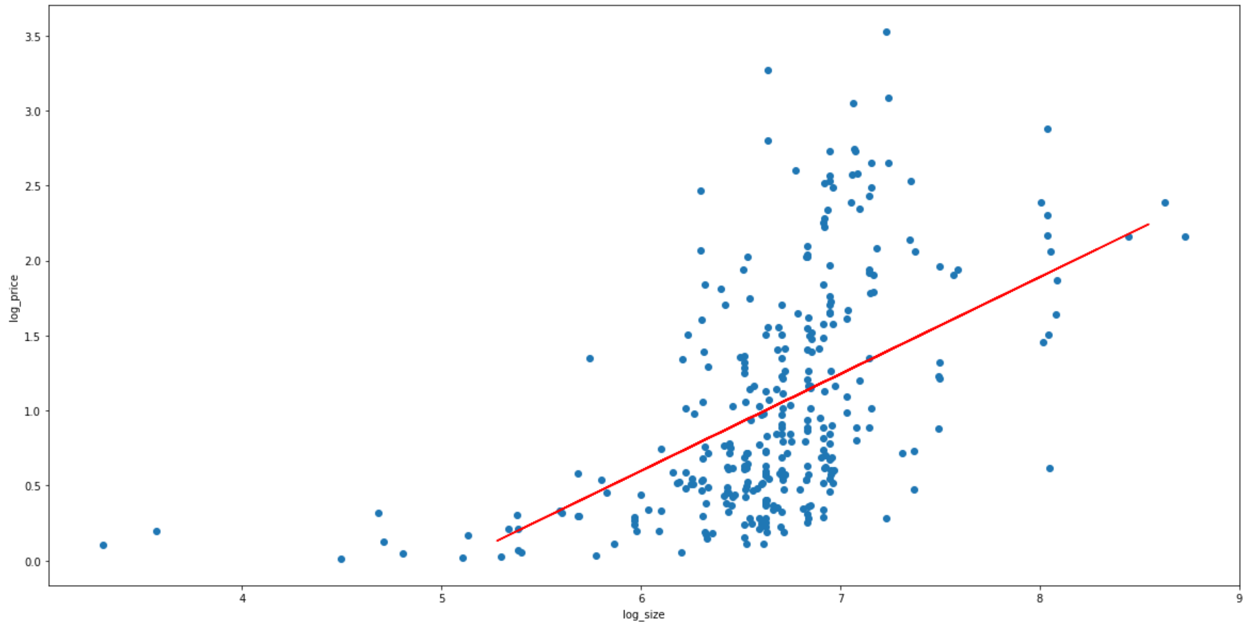
```
[33] y_pred = lr_model2.predict(X_test)
```

```
[34] price_data = pd.DataFrame({'Actual Price': y_test, 'Predicted Price': y_pred})
     price_data
```

	Actual Price	Predicted Price
0	0.646056	1.189646
1	0.634458	0.596115
2	0.566889	1.179432
3	0.301585	0.695389
4	1.556564	1.138545
...	...	...

```
[35] # Linear Regression plot for this model 2
     fig, ax = plt.subplots(figsize=(20,10))
     plt.scatter(X_train, y_train)
     plt.plot(X_test, y_pred, color = 'red')
     plt.xlabel("log_size")
     plt.ylabel("log_price")
     plt.show()
```

Regression Plot



From the regression plot it can be concluded that log size and log_price are corelated with each other linearly. The given regression line indicates the predicted price for the respective size values. The blue dots represent the actual price values and find to accumulated near the regression line

```
[36] #Checking RMSE
     root_mean_square_error = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
     print(root_mean_square_error)

     0.7242002612492014
```

```
[37] #Checking Mean Absolute error
     mean_absolute_error = metrics.mean_absolute_error(y_test, y_pred)
     print(mean_absolute_error)

     0.5686707923594233
```

The RMSE values was found to be 0.72 and mean absolute error was found to be 0.56

14) Simple linear regression 3

Independent variable is log width

Dependent variable is log_price

The log width is taken as X-Dataset and y-dataset log_price and used train_test_split into 75% training data and 25% testing data.

[38]
```python
# Simple Linear Regression model 3

X = monetdataset[['log_WIDTH']].values
y= monetdataset['log_PRICE'].values
```

[39]
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

[40]
```python
lr_model3 = linear_model.LinearRegression()
lr_model3.fit(X_train, y_train)
```

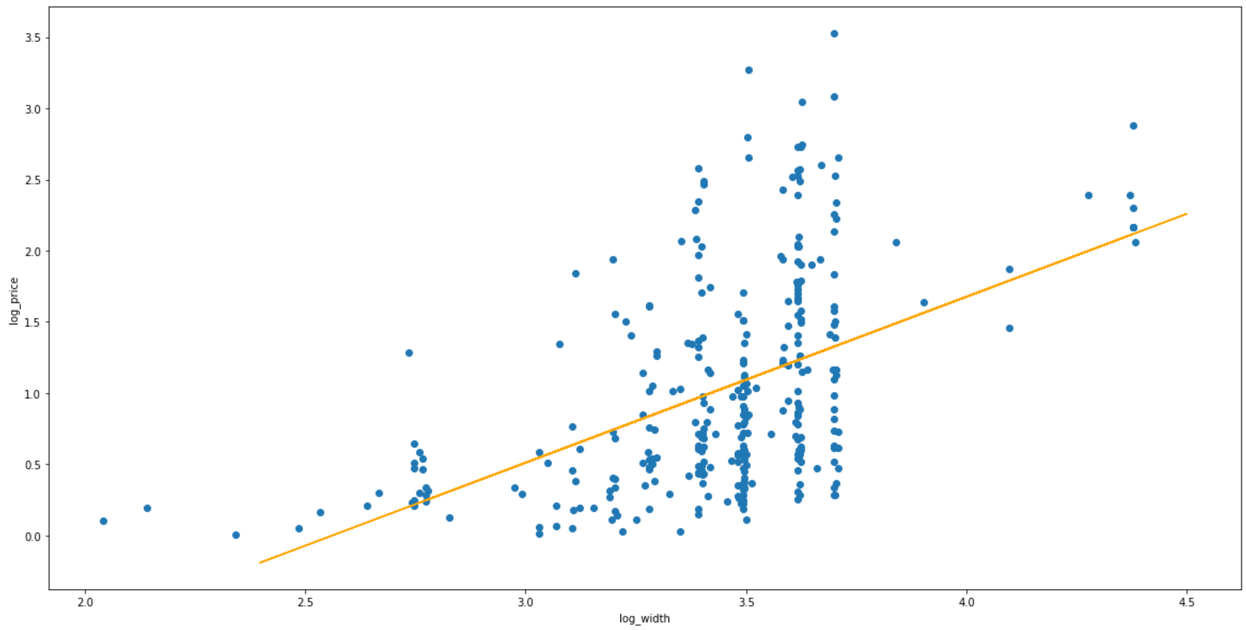
```
LinearRegression()
```

[41]
```python
y_pred = lr_model3.predict(X_test)
```

[42]
```python
price_data = pd.DataFrame({'Actual Price': y_test, 'Predicted Price': y_pred})
price_data
```

	Actual Price	Predicted Price
0	0.646056	1.325829
1	0.634458	0.967305
2	0.566889	1.239031
3	0.301585	0.803271
4	1.556564	1.235920

[43]
```python
# Linear Regression plot for this model 3
fig, ax = plt.subplots(figsize=(20,10))
plt.scatter(X_train, y_train)
plt.plot(X_test, y_pred, color = 'orange')
plt.xlabel("log_width")
plt.ylabel("log_price")
plt.show()
```



From the regression plot it can be concluded that log width and log_price are corelated with each other linearly. The given regression line indicates the predicted price for the respective log width values. The blue dots represent the actual price values.

```
#calculating RMSE
root_mean_square_error = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print(root_mean_square_error)
```

```
0.7725163537150288
```

```
[45] mean_absolute_error = metrics.mean_absolute_error(y_test, y_pred)
print(mean_absolute_error)
```

```
0.617090379577124
```

The RMSE values was found to be 0.77 and mean absolute error is 0.61

15) Multivariate linear regression

Independent variable is log width, signed, log height, picture, log size, house

Dependent variable is log_price

The log width, signed, log height, picture, log size, house is taken as X-Dataset and y-dataset as log_price and used train_test_split into 75% training data and 25% testing data.

```
[46] # Multivariate Linear Regression model

     X = monetdataset[['SIGNED','PICTURE', 'HOUSE ','log_WIDTH', 'log_HEIGHT','log_SIZE']].values
     y= monetdataset['log_PRICE'].values
```

```
[47] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

```
[48] lr_model4 = linear_model.LinearRegression()
     lr_model4.fit(X_train, y_train)

     LinearRegression()
```

```
[49] y_pred = lr_model4.predict(X_test)
```

```
[50] price_data = pd.DataFrame({'Actual Price': y_test, 'Predicted Price': y_pred})
     price_data
```

	Actual Price	Predicted Price
0	0.646056	1.264055
1	0.634458	0.731277
2	0.566889	1.155242
3	0.301585	0.655573
4	1.556564	1.133473

```
[51] #Calculate RMSE
     root_mean_square_error = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
     print(root_mean_square_error)

     0.6921640821860559
```

```
[52] #Calculating mean absolute error
     mean_absolute_error = metrics.mean_absolute_error(y_test, y_pred)
     print(mean_absolute_error)

     0.5383370901785762
```

The RMSE value is 0.69 and mean absolute error is 0.53

16)Conclusion: It can be stated that the multivariate linear regression model has the lowest RMSE and mean absolute error score compared to the other simple linear regression models. Which illustrates that price of painting can be estimated with greater accuracy using all the Monet attributes instead of evaluating them individually.