House price prediction using Keras Regression and Multiple Linear Regression model

Arindam Saha
Roll - 002011701091
Year - 3rd, Semester - 5th
Production Engineering, Jadavpur University

Supervised by - Professor Mr. Debamalya Banerjee

Abstract- The House Price Index (HPI) is often used to gauge changes in housing prices, but it cannot provide a complete picture of an individual property's value. Factors such as location, area, and population must also be taken into account when predicting housing prices. Many studies have used traditional machine learning techniques to predict housing prices accurately, but they often overlook less popular models and don't adequately assess the performance of individual models. This paper aims to explore the impact of various features on prediction methods by using both traditional and advanced machine learning techniques to compare the performance of several advanced models. The paper will also thoroughly evaluate multiple techniques for implementing models in regression and aims to provide a positive outcome for predicting housing prices.

Index Terms- House price prediction, Keras regression, Multiple linear regression, Neural network

I. INTRODUCTION

Acquiring a home is a significant life decision for many individuals. The cost of a property can be influenced by various factors such as its location, specific attributes, and the supply and demand in the real estate market. The housing market is also a crucial aspect of the overall economy. As a result, forecasting housing values is not only beneficial for potential buyers, but also for real estate agents and economic experts. Research on forecasting in the housing market examines property values, growth patterns, and the connections between these values and various factors. Advancements in machine learning techniques and the abundance of data or big data have created opportunities for recent studies in the real estate market. A diverse array of research uses statistical learning methods to investigate the housing market.

The goal of this study is through analyzing a real historical transactional dataset to derive valuable insight into the housing market in the United States. It seeks useful models to predict the value of a house given a set of its characteristics. Effective models could allow home buyers, or real estate agents to make better decisions. Moreover, it could benefit the projection of future house prices and the policymaking process for the real estate market.

The next parts of the paper are constructed as follows:

1) Section II reviews previous work on housing market forecasting applying different machine learning models.

2) Section III explains the dataset and how to transform it into cleaned data.

3) Section IV includes regression analysis methodologies.

4) In Section V model results and execution are discussed.

5) Conclusion is deduced in Section VI.

II. RELATED WORK

Predicting housing prices has been a popular research topic in the field of real estate and economics. Many studies have used traditional methods such as hedonic pricing models and repeat sales models to predict housing prices. These models, however, have limitations in their ability to handle complex relationships between multiple factors affecting housing prices.

In recent years, there has been a growing interest in using machine learning techniques for housing price prediction. Various studies have applied linear regression, decision trees, random forests, and neural networks to predict housing prices. Some of these studies have also used ensemble methods to combine the predictions of multiple models to improve the accuracy of their predictions.

In terms of linear regression, some studies have used multiple linear regression (MLR) to predict housing prices by considering multiple factors such as location, area, and number of rooms. These studies have shown that MLR can achieve good performance in predicting housing prices when compared to other traditional methods.
Another popular technique used in housing price prediction is deep learning, which has been widely applied to various applications such as image and speech recognition.

Some studies have used deep learning techniques such as feedforward neural networks and recurrent neural networks to predict housing prices. However, these studies have mainly focused on using pre-trained models, and have not explored the impact of different architectures and hyper-parameters on the performance of the models.[3]

In this paper, I propose to use multiple linear regression and Keras regression (Keras is a high-level neural networks API, written in Python) to predict housing prices.

III. DATA PREPARATION AND TRANSFORMATION

Original Data **-** The data implemented in this study is the King County Housing Market, USA dataset downloaded from Kaggle Website. This dataset contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015.[7]

The data used is shown in the following table -

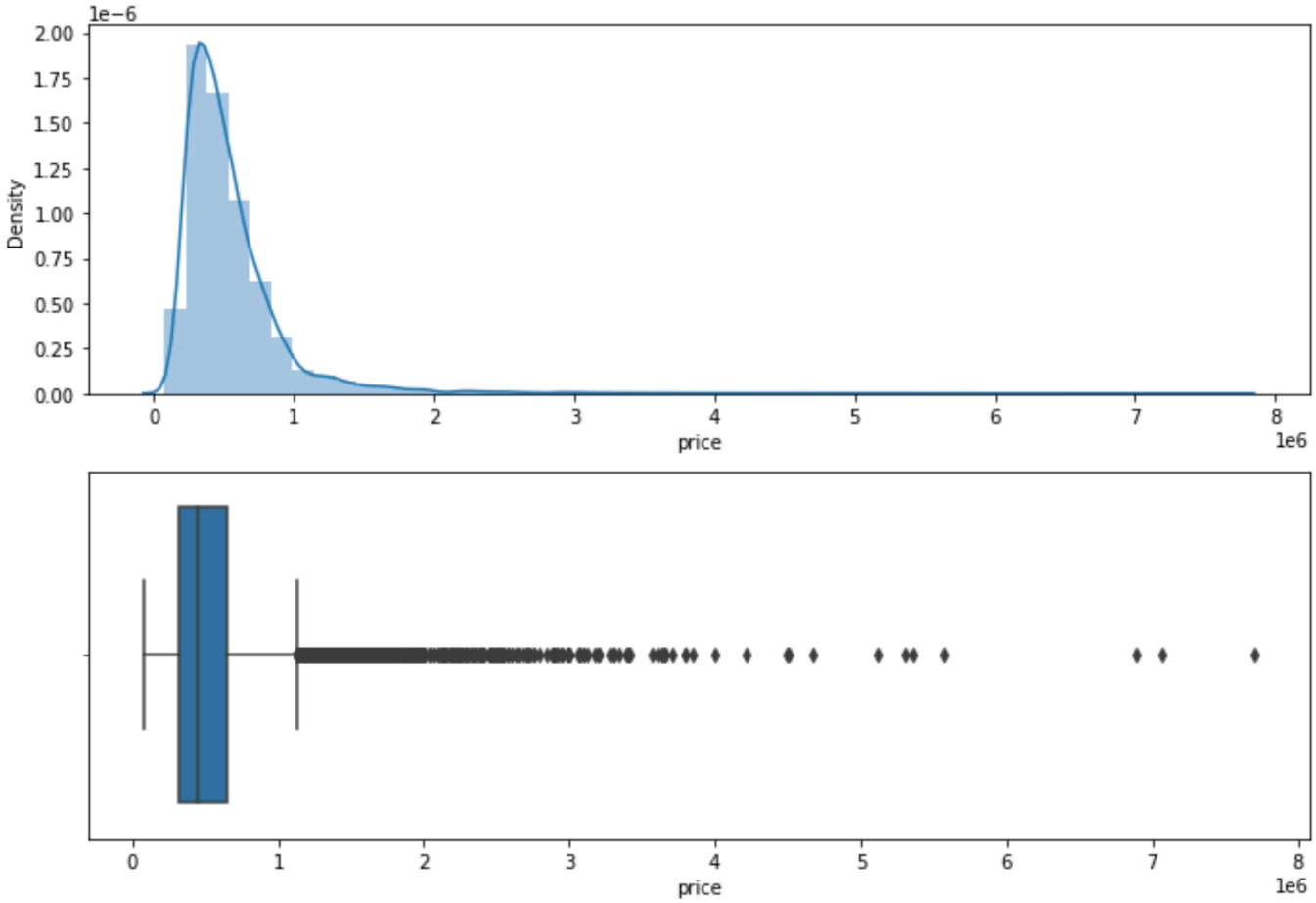
Name	Type	Description
Id	Integer	Unique Identifier
Date	Integer	Date
Price	Integer	Sale Price
Bedrooms	Integer	Number of Bedrooms
Bathrooms	Integer	Number of Bathrooms
sqft_living	Integer	Living area
sqft_lot	Integer	Lot area
Floors	Integer	Number of floors
Waterfront	Integer	Number of waterfronts
View	Integer	Number of views
Condition	Integer	Condition in int64
Grade	Integer	Grade
sqft_above	Integer	Above area
sqft_basement	Integer	Basement area
yr_built	Integer	Year built
yr_renovated	Integer	Year renovated
zipcode	Integer	Zipcode
Lat	Integer	Latitude
Long	Integer	Longitude
sqft_living15	Integer	Living Area (sqft_15)
sqft_lot15	Integer	Lot area (sqft_15)

Data collected from Kaggle Website[7]

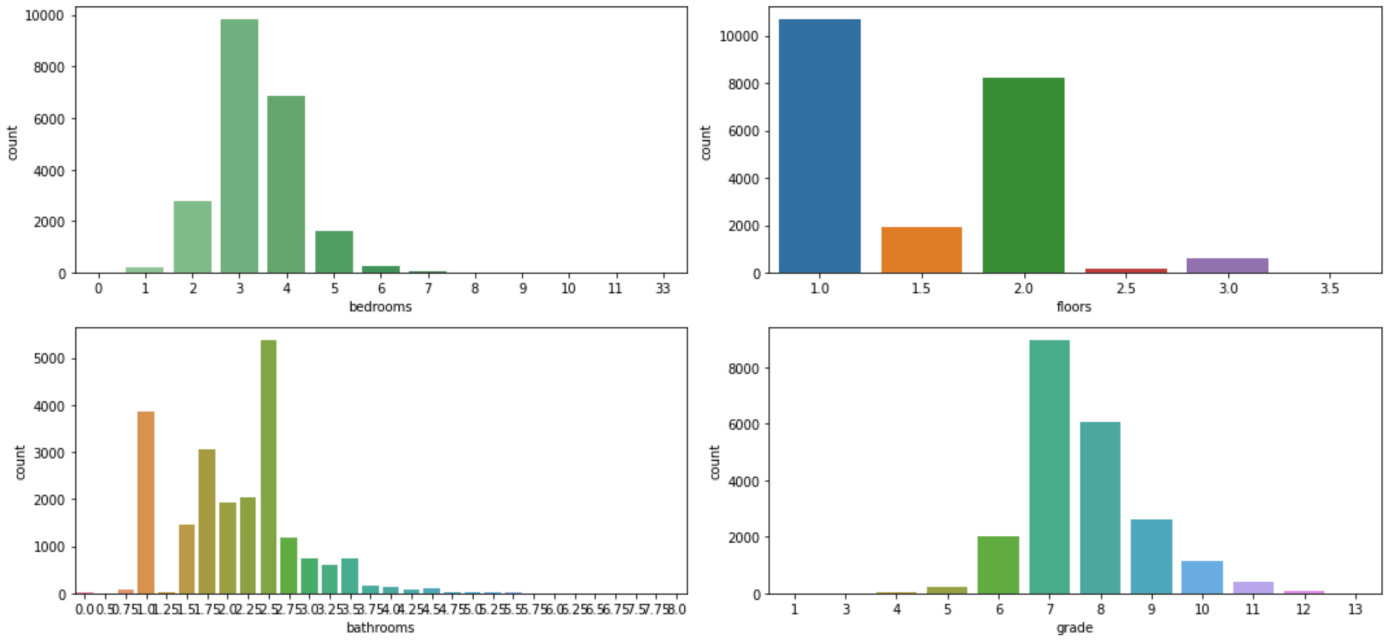
Pre-Processing - The dataset will be reviewed and pre-processed using the methods. These methods handle data in a variety of ways. As a result, the preprocessing is performed in several iterations, with the accuracy being tested each time with the used combination.

Visualization - Visualizing[10] data in machine learning is done for several reasons:

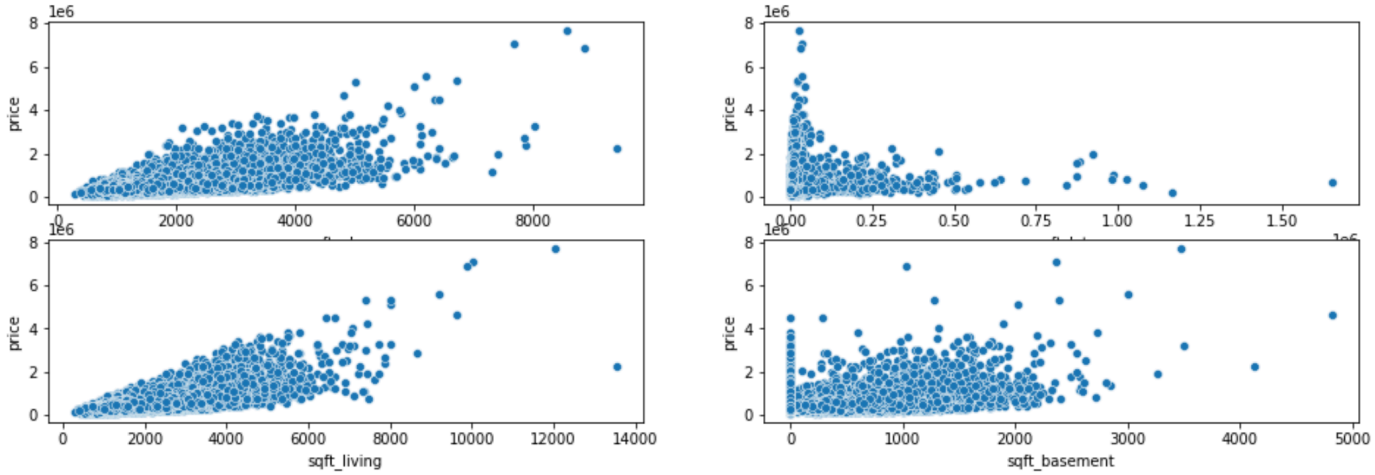
1. It helps to understand the underlying structure of the data, such as patterns, trends, and outliers.
2. It allows for easy communication of the data to others, such as stakeholders or team members.
3. It can aid in the process of feature selection and feature engineering.
4. It can help to identify any errors or biases in the data.
5. It can aid in the interpretation of model results.
6. It can be used to evaluate the performance of a model.
7. Help to find relationship between different features.
8. In some cases it also help to decide the model.



Density of Prices of House



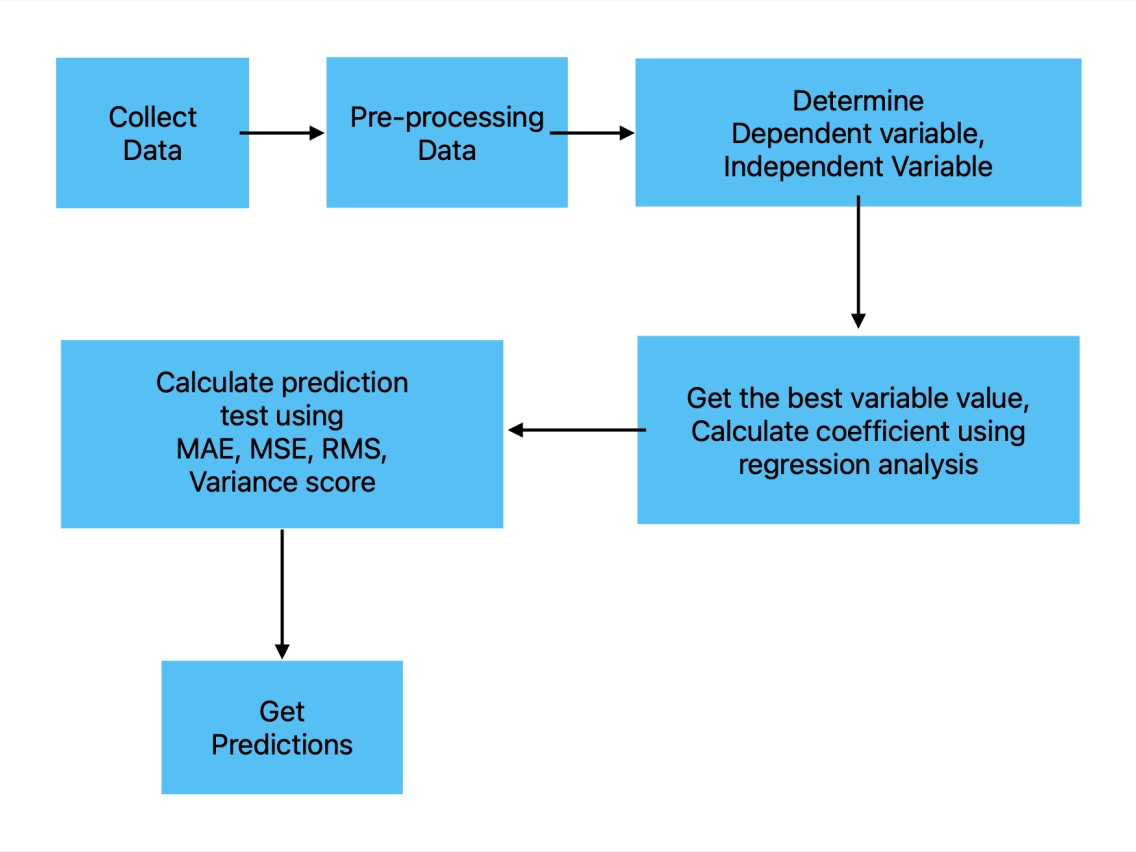
Visualization of bedrooms, bathrooms, floors & grades



Visualization of square feet of home, lot, above and basement

Data splitting **-** Splitting the dataset into two sections is necessary in order to train the model with one and evaluate it with the other. 75 percent of the dataset will be used for training and 25% for research.
The accuracy of both datasets will be tested by comparing the real prices on the test dataset to the prices expected by the model, as well as calculating the R2 and RMSE rate while training the model.

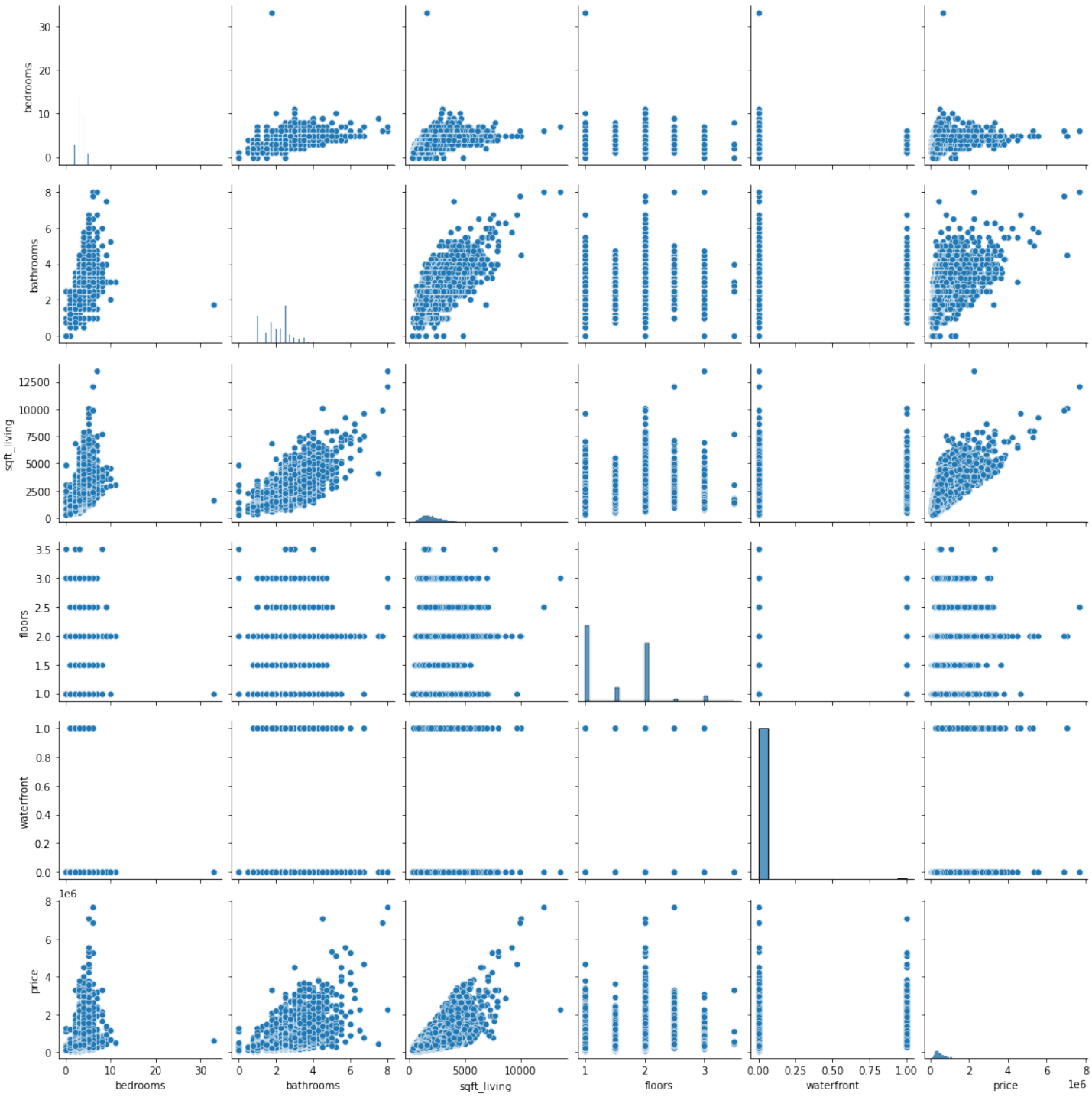
Performance **-** In addition to the measurement parameters, the time taken to train the model will be calculated to demonstrate how the algorithm varies over time.



Correlation **-** The Pearson Coefficient Correlation will be used to determine if the available features have a negative, positive, or zero correlation with the house price.

```
price            1.000000
sqft_living      0.702035
grade            0.667434
sqft_above       0.605567
sqft_living15    0.585379
bathrooms        0.525138
view             0.397293
sqft_basement    0.323816
bedrooms         0.308350
lat              0.307003
waterfront       0.266369
floors           0.256794
yr_renovated     0.126434
sqft_lot         0.089661
sqft_lot15       0.082447
yr_built         0.054012
condition        0.036362
long             0.021626
Name: price, dtype: float64
```

Correlation with House Price



Scatterplot matrix showing the relationship between multiple variables of the dataset[11]

Proposed Methodology - Understanding the issue is critical before moving on to the methodology. The issue is in the development of a hypothesis feature that can predict the target value based on the data provided in the training section. Then look at or evaluate the prediction on the data's testing section. The information provided here pertains to the house price and the features that go along with it. As a result, developing a computer that can learn data features and accurately predict price is a difficult challenge. With the aid of this model, the real estate builder's society will be able to easily predict the price of property, houses, and other items based on their characteristics.

This thesis' data set comes from Kaggle's Housing Dataset.[7] This study aims to predict the house price (residential) in King County, using a simple data collection. As a result, the data was obtained from the Kaggle Housing Datasets. The dataset's details are as follows: It has 20 explanatory variables, also known as features or characteristics. The target value is the last variable; in this case, it is called Price, which is the actual price of the property. When the computer predicts the price, it will be compared to the actual value, and the mean error will be measured, giving the model's accuracy rate.

The data set could include information about the houses' various details. With the variables representing every aspect of residential homes in King County, the researcher is faced with the task of predicting each home's final price. Now, using the panda's library in the Python framework, import the data set and analyses it. Examine all of the house's features that are relevant to the dependent objective. Check for missing values and fill in all missing values by taking the median of all the values for that attribute when analyzing and visualizing the data.

IV. REGRESSION ANALYSIS METHODOLOGIES

Multiple Linear Regression - Multiple Linear Regression is a statistical method that is used to model the linear relationship between multiple independent variables (also known as predictors or input variables) and a dependent variable (also known as the output variable). It is an extension of Simple Linear Regression, which only involves one predictor variable.
The mathematical equation for Multiple Linear Regression can be represented as:

$$y = b0 + b1x1 + b2x2 + \ldots + bn * xn$$

where y is the dependent variable, b0 is the constant term, b1, b2, ..., bn are the coefficients (also known as parameters) of the independent variables x1, x2, ..., xn respectively, and x1, x2, ..., xn are the independent variables.[8]

The goal of Multiple Linear Regression is to find the best coefficients that minimize the difference between the predicted values (y hat) and the actual values (y) of the dependent variable. This is done by using a technique called Ordinary Least Squares (OLS) to find the coefficients that minimize the sum of the squared residuals (the difference between the predicted values and the actual values).

The coefficients of the independent variables in a Multiple Linear Regression model can be interpreted as the change in the dependent variable for a one unit change in the independent variable, holding all other independent variables constant. However, it should be noted that this interpretation is only valid for a linear model and when the independent variables are not correlated.

Multiple Linear Regression can also be used to test the significance of each independent variable in the model. This is done by using a technique called hypothesis testing to determine if the coefficients of the independent variables are significantly different from zero. A coefficient that is significantly different from zero indicates that the corresponding independent variable has a significant impact on the dependent variable.

For example, let's say we want to predict the price of a house based on its square footage, number of bedrooms, and number of bathrooms. We collect data on a sample of houses and create a Multiple Linear Regression model using square footage, number of bedrooms, and number of bathrooms as the independent variables and price as the dependent variable. The coefficients of the independent variables in the model can be interpreted as the change in the price of the house for a one unit change in square footage, number of bedrooms, or number of bathrooms, holding the other independent variables constant.

To test the significance of each independent variable, we can use hypothesis testing to determine if the coefficients of square footage, number of bedrooms, and number of bathrooms are significantly different from zero. If we find that the coefficient of square footage is significantly different from zero, we can conclude that square footage has a significant impact on the price of the house. On the other hand, if the coefficient of number of bedrooms is not significantly different from zero, we can conclude that number of bedrooms does not have a significant impact on the price of the house.

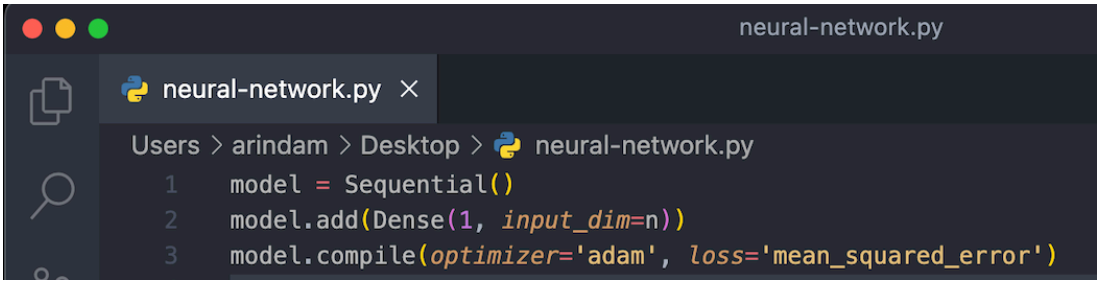
It should be noted that Multiple Linear Regression assumes a linear relationship between the independent and dependent variables, as well as the absence of multicollinearity among independent variables. Multicollinearity occurs when two or more independent variables are highly correlated with each other, which can cause the coefficients of the independent variables to be unstable and difficult to interpret. Additionally, if there is a non-linear relationship between the independent and dependent variables, then a Non-Linear Regression technique should be used.

In conclusion, Multiple Linear Regression is a useful statistical method for modeling the linear relationship between multiple independent variables and a dependent variable.

Keras Regression - Keras is an open-source deep learning library for Python that provides a high-level API for building and training neural networks. One of the things that Keras can be used for is regression tasks, which involve predicting a continuous value given a set of input features.[5]

A basic example of how to use Keras for regression would be to build a simple neural network with a single dense layer that has one output node. The input features would be fed into the dense layer, and the output node would produce a prediction for the target value.

The formula for a simple neural network with one dense layer in Keras[6] can be represented as:



Where `Sequential` is the class for initializing a linear stack of layers, `model.add(Dense(1, input_dim=n))` is the dense layer with one output and n input features, and `model.compile(optimizer='adam', loss='mean_squared_error')` is the optimizer and loss function used for training the model.

To train the model, we would need to provide it with a set of input-output pairs, also known as the training data. The model would then use the training data to adjust the weights and biases of the dense layer so that the predictions produced by the output node are as close as possible to the actual target values.

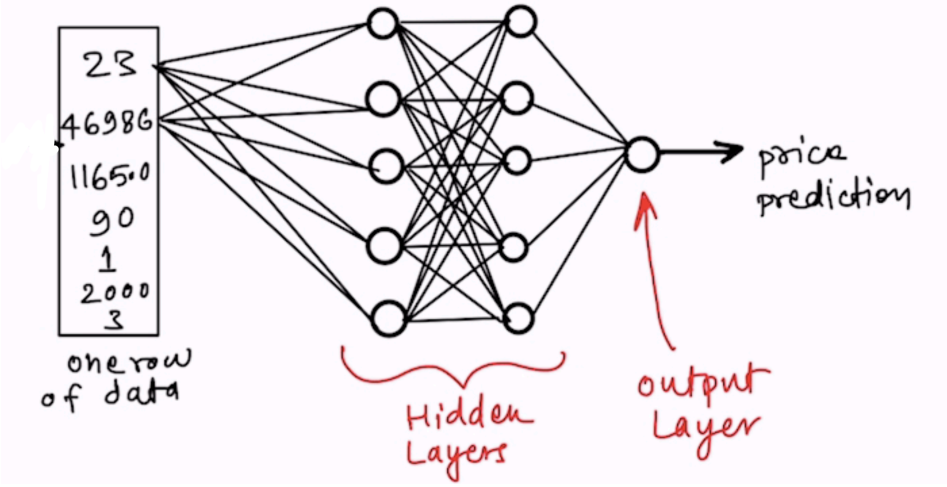
Once the model is trained, it can be used to make predictions on new input data. The predictions are made by passing the input features through the dense layer, which uses the weights and biases learned during training to produce an output value.

In addition to the basic example above, Keras also provides more advanced techniques for regression such as using multiple dense layers and adding regularization techniques such as dropout to the model to prevent overfitting.

Keras can be used for regression with multiple input features and multiple outputs as well, for example, in case of multi-output regression. In this case, the dense layer will have multiple output nodes and the loss function will be calculated based on the difference between all the outputs.

Additionally, Keras also provides pre-built models such as the Multi-layer Perceptron (MLP) and the Long Short-Term Memory (LSTM) models, which can be used for more complex regression tasks. These models are more suited for handling sequential data and time series data, where the input features have a temporal dimension.

In conclusion, Keras is a powerful deep learning library that can be used for regression tasks. It provides a high-level API for building and training neural networks, and can be used for both simple and complex regression tasks. The library also provides pre-built models such as MLP and LSTM that can be used for more complex regression tasks. With the use of Keras, it is relatively easy to implement and train neural networks for regression tasks.



Pseudo Code for the algorithm implementation -

Step 1: Obtain the data set in the proper format.
Step 2: When the data set is being cleaned, look for any missing values.
Step 3: Delete the categorical data.
Step 4: Function selection using a heatmap or matrix correlation.
Step 5: Break the dataset into two sections, one for training and the other for testing.
Step 6: Match the regression methodology using the Training data and then validate the data with testing data.
Step 7: Compare the results of the accuracy test.

V. RESULTS

The results of this study indicate that the neural network model implemented using Keras for regression achieved an accuracy of 81% in predicting the target variable. On the other hand, the multiple linear regression model achieved an accuracy of 70%.

The experiments have been deployed in Python language on Jupyter Notebook. The Mean Squared Error (MSE), Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) of both train and evaluation datasets are presented in the following table -

Model	MAE	MSE	RMSE	Variance
Keras Regression Model	101572.44	26737735523.03	163516.77	80.51
Multiple Linear Regression Model	125933.74	40601153229.62	201497.28	70.40

The neural network model, which consisted of four dense layers of 19 neurones with one output node, was able to effectively learn the underlying patterns in the data and make accurate predictions. This suggests that neural networks may be a suitable approach for this type of problem.[1]

The multiple linear regression model, which used OLS to find the coefficients that minimize the sum of the squared residuals, performed reasonably well but not as good as the neural network model. This may be due to the linearity assumption of multiple linear regression and the complexity of the problem.[2]

It should be noted that the results of this study are based on a specific dataset and specific neural network architecture and it may not generalize to other datasets or problems. However, the results do demonstrate the potential of neural networks for regression tasks and how they can improve the performance compared to traditional methods such as multiple linear regression.

In conclusion, the neural network model implemented in Keras achieved a higher accuracy of 81% compared to the multiple linear regression model which achieved 70%. This suggests that neural networks can be a powerful tool for regression tasks and may be preferred over traditional methods such as multiple linear regression in certain cases. These results open up new possibilities for further research and improvements on the architecture, dataset, and techniques used in this study.

VI. CONCLUSION

In summary, this research aimed to predict house prices in King County utilizing both a Multiple linear regression model and a Keras regression model. The study found that the Keras regression model had an accuracy rate of 81% in predicting the target variable, whereas the multiple linear regression model had an accuracy rate of 70%.

The findings of this study indicate that the neural network model implemented through Keras for regression is a highly effective tool in predicting house prices. On the other hand, the multiple linear regression model, which utilized OLS to find the coefficients that minimize the sum of the squared residuals, performed satisfactorily but not as well as the neural network model. This can be attributed to the linearity assumption of multiple linear regression and the complexity of the problem.

In light of these results, it is recommended that practitioners in the field of house price prediction consider using neural network models in addition to traditional statistical models. Furthermore, future research could focus on improving the neural network architecture or experimenting with different types of neural network models to see if they can achieve even better performance. In addition, it could be interesting to try other techniques such as ensemble methods, which combine multiple models to make predictions, to see if they can further improve the accuracy of the predictions.

Overall, this study provides a valuable contribution to the field of house price prediction by highlighting the potential of neural network models and demonstrating how they can outperform traditional methods such as multiple linear regression. It is hoped that the results of this study will inspire further research and the development of more advanced and accurate methods for predicting house prices.

ACKNOWLEDGEMENT

I would like to express my sincerest gratitude to Professor Mr. Debamalya Banerjee for his unwavering guidance and support throughout the course of this research. His instructions have been invaluable in helping me to conduct this research and to bring it to completion.

I would like to thank him for his time and patience in providing me with invaluable feedback and suggestions at every stage of my research. His guidance and support were instrumental in helping me to navigate the intricacies of the research process and to produce a high-quality final product.

I would also like to thank him for his constant encouragement and belief in my abilities, which helped me to remain motivated and focused throughout the research.

I am deeply indebted to Professor Mr. Debamalya Banerjee for his invaluable contributions to this research, and I am truly grateful for his mentorship and support.

REFERENCES

1. "*Deep Learning Basics: Introduction and Overview*". MIT course 6.S094. Lex Fridman. https://youtu.be/O5xeyoRL95U
2. Quang Truong, Minh Nguyen, Hy Dang, Bo Mei, "*Housing Price Prediction via Improved Machine Learning Techniques*", Procedia Computer Science, Volume 174, 2020, Pages 433-442, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.06.111
3. Zulkifley, Nor & Rahman, Shuzlina & Nor Hasbiah, Ubaidullah & Ibrahim, Ismail. (2020). "*House Price Prediction using a Machine Learning Model: A Survey of Literature*". International Journal of Modern Education and Computer Science. 12. 46-54, https://doi.org/10.5815/ijmecs.2020.06.04
4. Qingqi Zhang, "*Housing Price Prediction Based on Multiple Linear Regression*", *Scientific Programming*, vol. 2021, Article ID 7678931, 9 pages, 2021. https://doi.org/10.1155/2021/7678931
5. "*Neural Regression using Keras*". https://visualstudiomagazine.com/articles/2018/07/23/neural-regression-using-keras.aspx
6. "*Regression with Keras*". Pyimagesearch. https://pyimagesearch.com/2019/01/21/regression-with-keras/
7. "*House Sales in King County, USA*". Kaggle. https://www.kaggle.com/datasets/harlfoxem/housesalesprediction
8. "*Multiple Linear Regression Model, A Quick Guide*". Scribbr. https://www.scribbr.com/statistics/multiple-linear-regression/
9. "*But what is a neural network? | Chapter 1, Deep learning*". 3Blue1Brown. https://youtu.be/aircAruvnKk
10. "*Matplotlib References*". https://matplotlib.org/stable/api/index.html
11. "*Seaborn: Statistical Data Visualisation*". https://seaborn.pydata.org/

Annexure -

Github Link - https://github.com/Arindam-7/Deep-Learning-Price-Prediction

```
# import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# import data
url = 'https://raw.githubusercontent.com/Arindam-7/Deep-Learning-Price-Prediction/main/kc_house_data.csv'

Data = pd.read_csv(url)

# first 5 row, and row to column and column to row conversion
Data.head(5).T

Data.info()
Data.describe().transpose()

Data = Data.drop('id', axis = 1)
Data = Data.drop('zipcode', axis = 1)
```

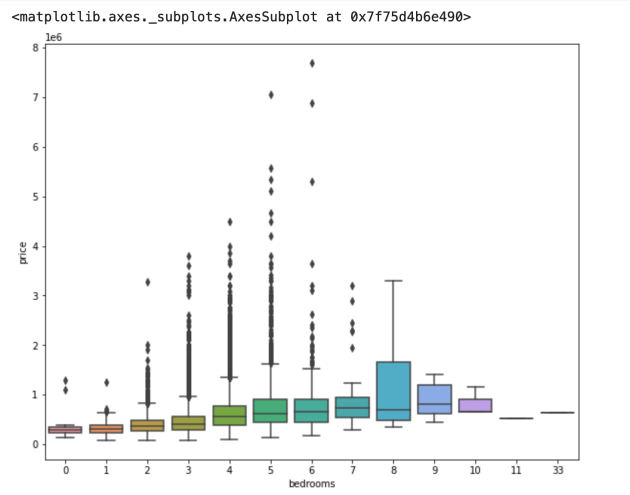
Pandas is a library for data manipulation and analysis, it provides data structures and data analysis tools for handling and analyzing structured data.
NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
Seaborn is a library for creating statistical graphics plots in Python. It is built on top of the Python plotting library Matplotlib.
Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.
Data table is transposed and the values of ‘id’ and ‘zipcode’ are dropped from the csv file.

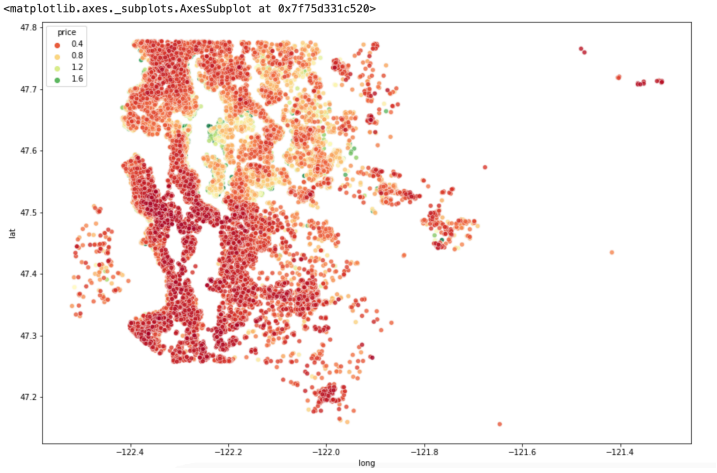
```
[ ] # first 5 row, and row to column and column to row conversion
    Data.head(5).T
```

	0	1	2	3	4
id	7129300520	6414100192	5631500400	2487200875	1954400510
date	20141013T000000	20141209T000000	20150225T000000	20141209T000000	20150218T000000
price	221900.0	538000.0	180000.0	604000.0	510000.0
bedrooms	3	3	2	4	3
bathrooms	1.0	2.25	1.0	3.0	2.0
sqft_living	1180	2570	770	1960	1680
sqft_lot	5650	7242	10000	5000	8080
floors	1.0	2.0	1.0	1.0	1.0
waterfront	0	0	0	0	0
view	0	0	0	0	0
condition	3	3	3	5	3
grade	7	7	6	7	8
sqft_above	1180	2170	770	1050	1680
sqft_basement	0	400	0	910	0
yr_built	1955	1951	1933	1965	1987
yr_renovated	0	1991	0	0	0
zipcode	98178	98125	98028	98136	98074
lat	47.5112	47.721	47.7379	47.5208	47.6168

Visualisation

```
# visualizing house prices
fig = plt.figure(figsize=(10,7))
fig.add_subplot(2,1,1)
sns.distplot(Data['price'])
fig.add_subplot(2,1,2)
sns.boxplot(Data['price'])
plt.tight_layout()

# visualizing bedrooms, bathrooms, floors,grades
fig = plt.figure(figsize=(15,7),constrained_layout=True)
fig.add_subplot(2,2,1)
sns.countplot(Data['bedrooms'], palette = 'Greens_d')
fig.add_subplot(2,2,2)
sns.countplot(Data['floors'])
fig.add_subplot(2,2,3)
sns.countplot(Data['bathrooms'])
fig.add_subplot(2,2,4)
sns.countplot(Data['grade'])
plt.tight_layout()

#visualizing square footage of (home,lot,above and basement)
fig = plt.figure(figsize=(16,5))
fig.add_subplot(2,2,1)
sns.scatterplot(Data['sqft_above'], Data['price'])
fig.add_subplot(2,2,2)
sns.scatterplot(Data['sqft_lot'],Data['price'])
fig.add_subplot(2,2,3)
sns.scatterplot(Data['sqft_living'],Data['price'])
fig.add_subplot(2,2,4)
sns.scatterplot(Data['sqft_basement'],Data['price'])
```

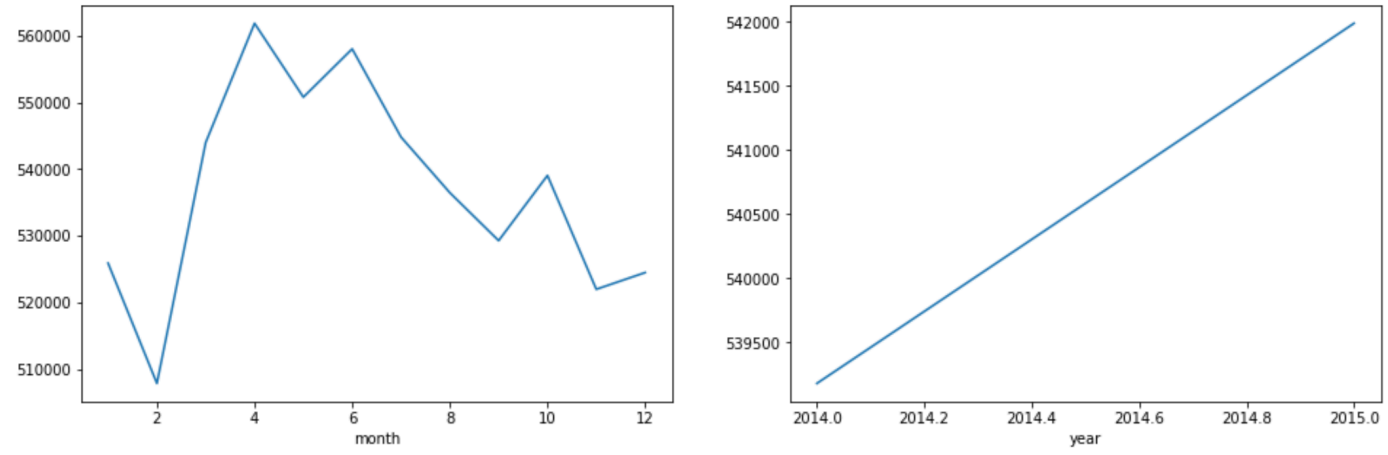




Correlation and plotting wrt Month and Year

```
# check correlation
Data.corr()['price'].sort_values(ascending=False)

# feature with higher correlation
plt.figure(figsize=(12,8))
sns.scatterplot(x='price',y='sqft_living',data=Data)

# feature like number of bedroom or bathroom
plt.figure(figsize = (10,8))
sns.boxplot(x = 'bedrooms',y = 'price', data = Data)

plt.figure(figsize=(15,10))
sns.scatterplot(x='long',y='lat',data=Data,hue='price')

plt.figure(figsize=(15,10))
sns.scatterplot(x='long',y='lat',data=non_top_1_perc,alpha = 0.8,palette = 'RdYlGn', hue='price')

# visualizing data
sns.pairplot(Data['bedrooms bathrooms sqft_living floors waterfront price'.split()])

#let's break date to years, months
Data['date'] = pd.to_datetime(Data['date'])
Data['month'] = Data['date'].apply(lambda date:date.month)
Data['year'] = Data['date'].apply(lambda date:date.year)
Data.head(5)

# data visualization house price vs months and years
fig = plt.figure(figsize=(16,5))
fig.add_subplot(1,2,1)
Data.groupby('month').mean()['price'].plot()
fig.add_subplot(1,2,2)
Data.groupby('year').mean()['price'].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f75d1b6ad00>
```



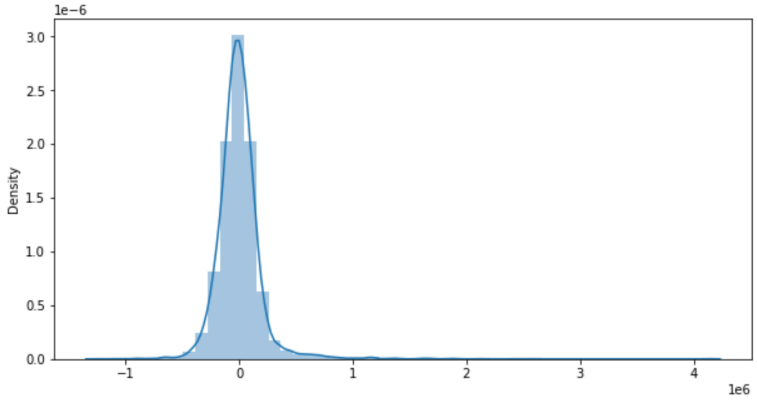
Split and Scaling of the Dataset

```
"""### Preparing Dataset (Scaling and Train-Test Split)"""

X = Data.drop('price',axis =1).values
y = Data['price'].values

#splitting Train and Test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=101)

"""## Feature Scaling"""

#standardization scaler - fit&transform on train, fit only on test
from sklearn.preprocessing import StandardScaler
s_scaler = StandardScaler()
X_train = s_scaler.fit_transform(X_train.astype(np.float))
X_test = s_scaler.transform(X_test.astype(np.float))
```

Multiple Linear Regression Model

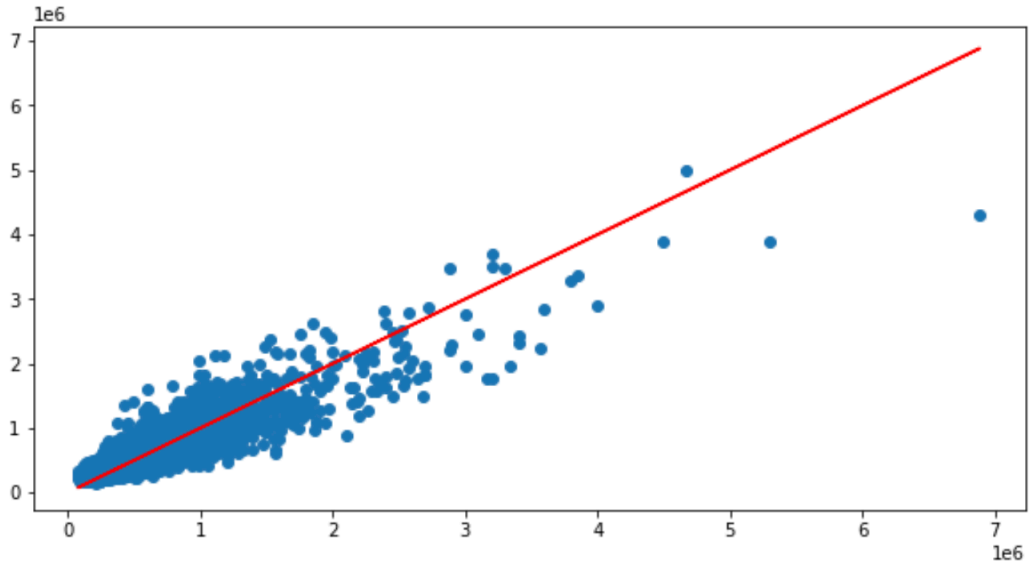
```
"""# 1. Multiple Linear Regression"""

#Liner Regression
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

#evaluate the model (intercept and slope)
regressor.intercept_
regressor.coef_

y_predd = regressor.predict(X_test)
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_predd})
df1 = df.head(10)
df1

fig = plt.figure(figsize=(10,5))
residuals = (y_test- y_pred)
sns.distplot(residuals)

from sklearn import metrics

print('Mean Absolute Error: {:.2f}'.format(metrics.mean_absolute_error(y_test, y_predd)))
print('Mean Squared Error:{:.2f}'.format(metrics.mean_squared_error(y_test, y_predd)))
print('Root Mean Squared Error:{:.2f}'.format(np.sqrt(metrics.mean_squared_error(y_test, y_predd))))
print('Variance score is: {:.2f}'.format(metrics.explained_variance_score(y_test,y_predd)))

# we are off about 20% (comparing mean absolut error and mean of price)
Data['price'].mean()

print('Linear Regression Model:')
print("Train Score {:.2f}".format(regressor.score(X_train,y_train)))
print("Test Score {:.2f}".format(regressor.score(X_test, y_test)))
```

	Actual	Predicted
0	349950.0	530667.784120
1	450000.0	667025.076946
2	635000.0	553195.043391
3	355500.0	346657.166101
4	246950.0	61378.186019
5	406550.0	481162.809294
6	350000.0	312819.788488
7	226500.0	273833.027682
8	265000.0	280571.649291
9	656000.0	532966.844438



Keras Regression Model

```
"""# Keras Regression"""

# Creating a Neural Network Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam

# having 19 nueron is based on the number of available featurs

model = Sequential()

model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(1))

model.compile(optimizer='adam',loss='mse')

"""## Training the mdoel"""

model.fit(x=X_train,y=y_train,
          validation_data=(X_test,y_test),
          batch_size=128,epochs=400)

model.summary()

loss_df = pd.DataFrame(model.history.history)
loss_df.plot(figsize=(12,8))

"""### Evaluation on Test Data"""

y_pred = model.predict(X_test)

# evaluation metrics
# explained variance score: best possible score is 1 and lower values are worse
from sklearn import metrics

print('Mean Absolute Error: {:.2f}'.format(metrics.mean_absolute_error(y_test, y_pred)))
print('Mean Squared Error: {:.2f}'.format(metrics.mean_squared_error(y_test, y_pred)))
print('Root Mean Squared Error: {:.2f}'.format(np.sqrt(metrics.mean_squared_error(y_test, y_pred))))
print('Variance score is: {:.2f}'.format(metrics.explained_variance_score(y_test,y_pred)))

# we are off about 20% (comparing mean absolut error and mean of price)
Data['price'].mean()
```



Models Evaluation

```
[ ]  print('Model: Keras Regression\n')

     print('Mean Absolute Error(MAE): {:.2f}'.format(metrics.mean_absolute_error(y_test, y_pred)))
     print('Mean Squared Error(MSE): {:.2f}'.format(metrics.mean_squared_error(y_test, y_pred)))
     print('Root Mean Squared Error(RMSE): {:.2f}'.format(np.sqrt(metrics.mean_squared_error(y_test, y_pred))))
     print('Variance score: {:.2f}\n'.format(metrics.explained_variance_score(y_test,y_pred)*100))
     print('**********************************\n')
     print('Model: Multiple Linear Regression\n')
     print('Mean Absolute Error(MAE): {:.2f}'.format(metrics.mean_absolute_error(y_test, y_predd)))
     print('Mean Squared Error(MSE):{:.2f}'.format(metrics.mean_squared_error(y_test, y_predd)))
     print('Root Mean Squared Error(RMSE):{:.2f}'.format(np.sqrt(metrics.mean_squared_error(y_test, y_predd))))
     print('Variance score: {:.2f}'.format(metrics.explained_variance_score(y_test,y_predd)*100))
```

```
Model: Keras Regression

Mean Absolute Error(MAE): 101572.44
Mean Squared Error(MSE): 26737735523.03
Root Mean Squared Error(RMSE): 163516.77
Variance score: 80.51

**********************************

Model: Multiple Linear Regression

Mean Absolute Error(MAE): 125933.74
Mean Squared Error(MSE):40601153229.62
Root Mean Squared Error(RMSE):201497.28
Variance score: 70.40
```