REPORT FOR HOUSE PRICE PRDICTION

As a project work for Course

PYTHON PROGRAMMING (INT 213)

Name : Dipanshu Singh
Registration Number : 11803562
Name : Abhishek Singh
Registration Number : 11801529
Program : CSE B.Tech (Hons.)
Semester : Third
School : School of Computer Science and Engineering
Name of the University : Lovely Professional University

Date of submission :11th NOVEMBER 2019

Lovely Professional University
Jalandhar, Punjab, India.

HOUSE PRICE PREDICTION

11th NOVEMBER 2019

ABSTRACT:-

House prices increase every year, so there is a need for a system to predict house prices in the future. House price prediction can help the developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house. There are three factors that influence the price of a house which include physical conditions, concept and location.

The relationship between house prices and the economy is an important motivating factor for predicting house prices. Housing price trends are not only the concern of buyers and sellers, but it also indicates the current economic situation. Therefore, it is important to predict housing prices without bias to help both the buyers and sellers make their decisions. This project uses an open source dataset.

ACKNOWLEDGEMENT:-

I would like to thank my mentor - Prof. Sagar Pande for his advice and inputs on this project. Many thanks to my friends and seniors as well, who spent countless hours to listen and provide feedbacks.

Table of Contents

INTRODUCTION:-

1.1 Context

This project has been done as part of my course for the CSE(H) at Lovely Professional University . Supervised by Sagar Pande, I have three months to fulfill the requirements in order to succeed the module.

1.2 Motivations

Being extremely interested in everything having a relation with the Machine Learning, the group project was a great occasion to give us the time to learn and confirm our interest for this field. The fact that we can make estimations, predictions and give the ability for machines to learn by themselves is both powerful and limitless in term of application possibilities. We can use Machine Learning in Finance, Medicine, almost everywhere. That's why I decided to conduct my project around the Machine Learning.

1.3 Idea:-

As a first experience, we wanted to make my project as much didactic as possible by approaching every different steps of the machine learning process and trying to understand them deeply. Known as " toy problem" the problems that are not immediate scientific interest but useful to illustrate and practice, we chose to take house price Prediction as approach. The goal was to predict the price of a given house according to the market prices taking into account different "features" that will be developed in the following .

TEAM MEMBERS:-

TEAM LEADER:-

Abhishek Singh:-

Contributions:-

1. Coding(joined)
2. Multivariable Regressing
3. GUI
4. Machine lerning(joined)

Dipanshu Singh:-

Contributions:-

1. Coding(joined)
2. Datasets
3. Linear regression
4. Reports
5. Machine learning(joined)

LIBRARIES:-

Numpy:-

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.It is the fundamental package for scientific computing with Python.

As the whole project is based on whole complex stats ,we will use this fast calculations and provide results.

Pandas:-

Pandas is the most popular python library that is used for data analysis. We will provide highly optimized performance with back-end source code with the use of Pandas.

Matplotlip:-

Matplotlib tries to make easy things easy and hard things possible. We will generate plots, histograms, scatterplots, etc.,to make our project more appealing and easier to understand.

Seaborn:-

We will use it forstatistical data visualization as **Seaborn** is a **Python** data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Scikit-learn:-

It is a Python library is associated with **NumPy** and **SciPy**. It is considered as one of the best libraries for working with complex data.

There are a lot of changes being made in this library. We will use it for cross-validation feature, providing the ability to use more than one metric. Lots of training methods like logistics regression will be used to provide some little improvements.

PROPOSED MODULES:-

Dataset:-

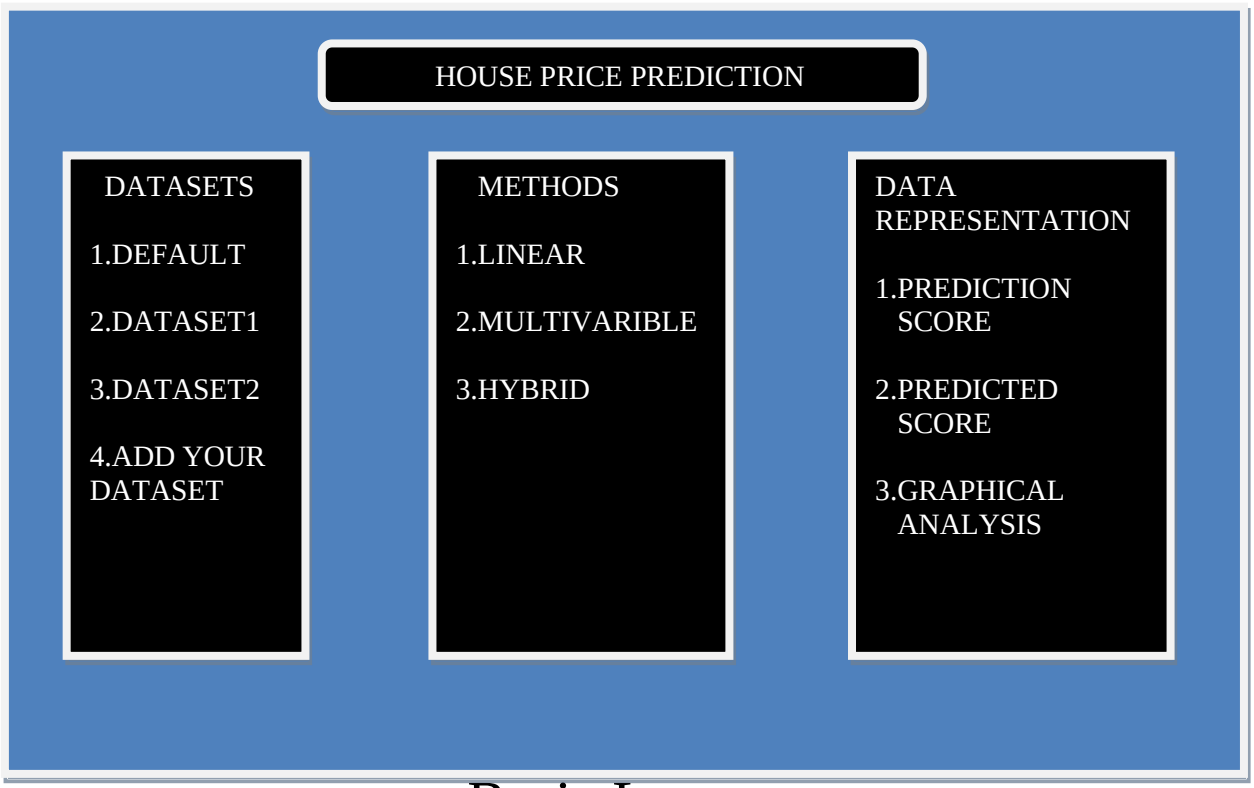
1.Enter dataset manually
2.Default dataset
3.Different datasets

Data representation:-

1.Prediction score
2.Predicted price
3.Graphical analysis

Regeression used:-

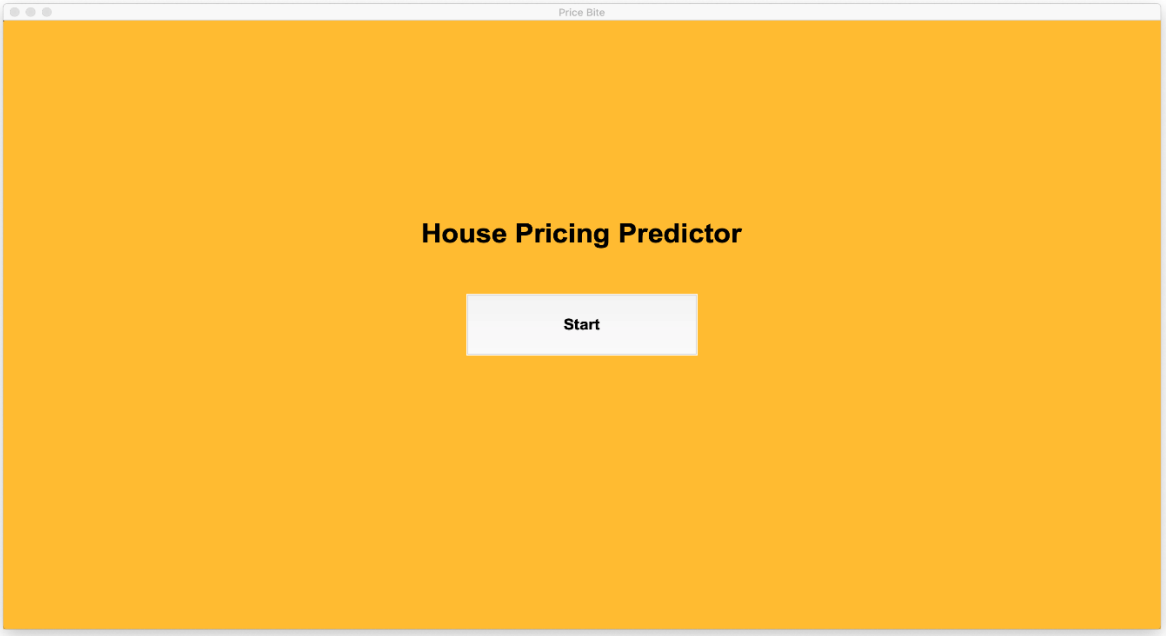
1.Linear regression
2.Multivariable regression
3.Hybrid regression



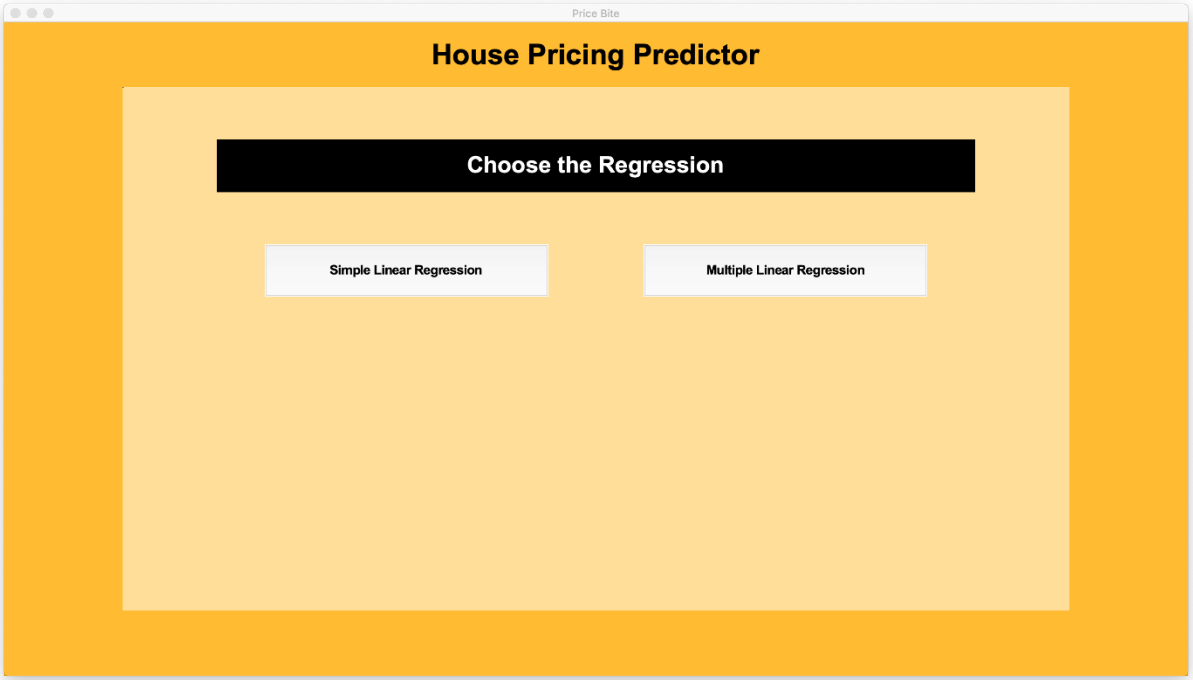
Basic Layout

SCREENSHOTS:-

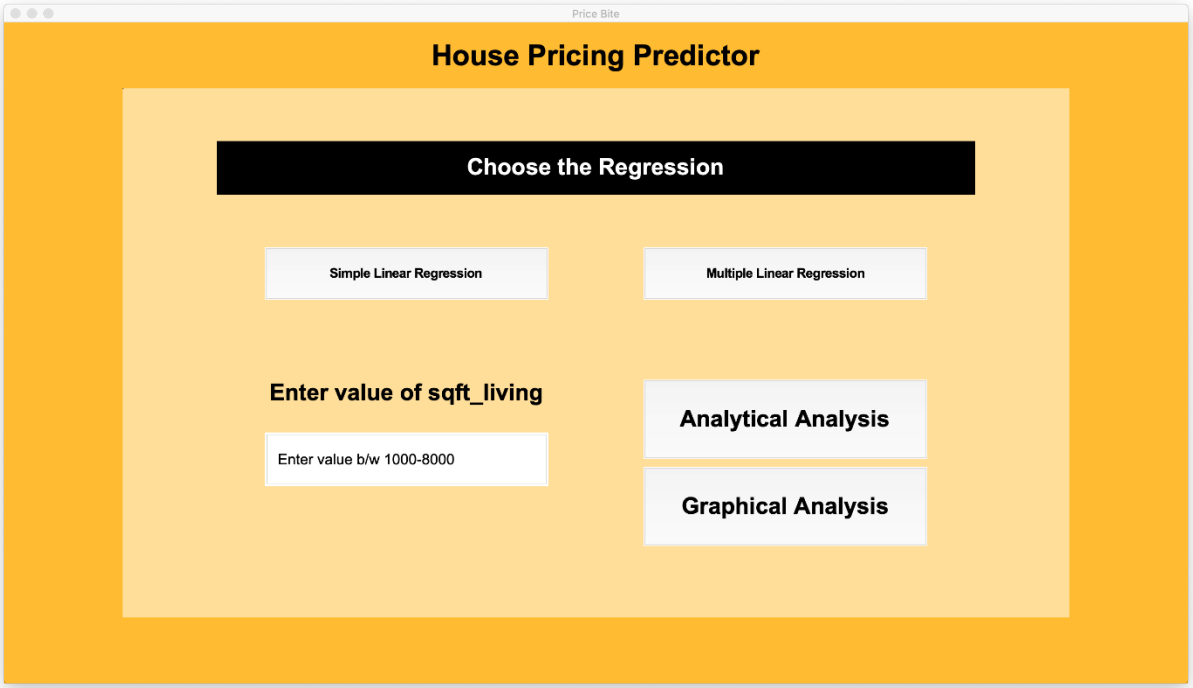
1.Main page:-



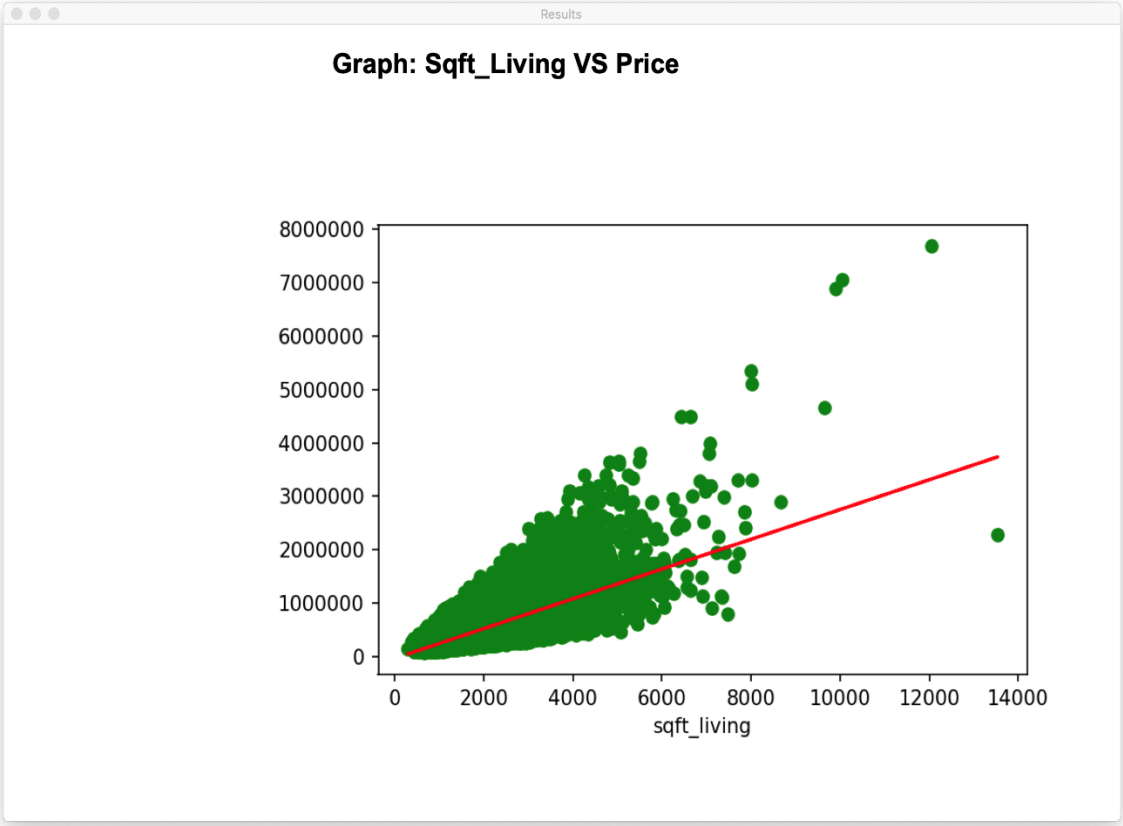
2.Home:-

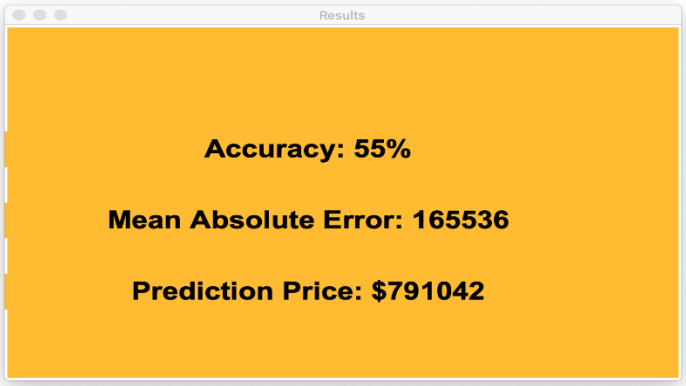


3.Linear Regression:-

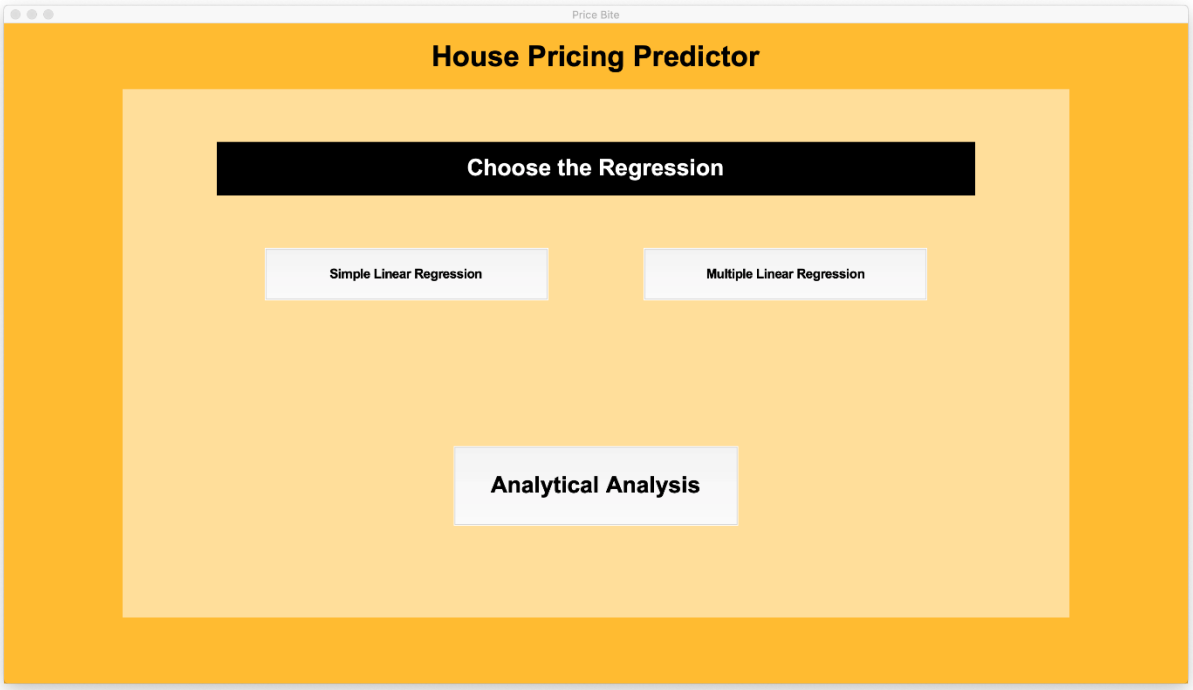


3.1.Linear Regression (Data Representation):-

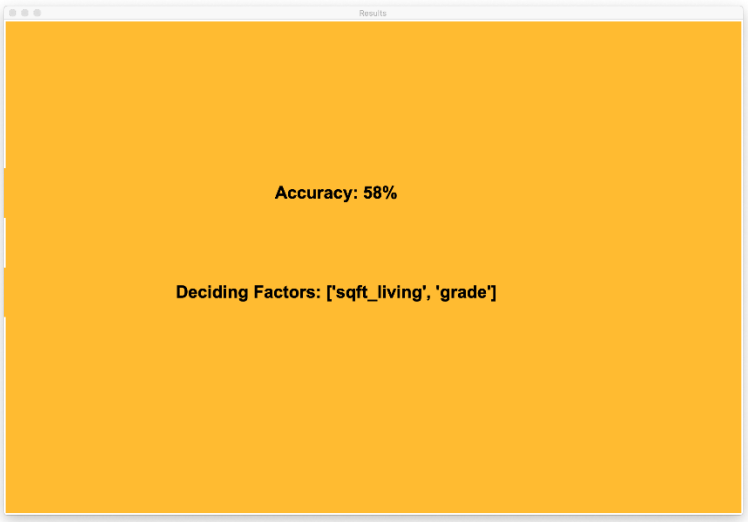




4.Multiple Regression:-



4.1.Multiple Representation:-



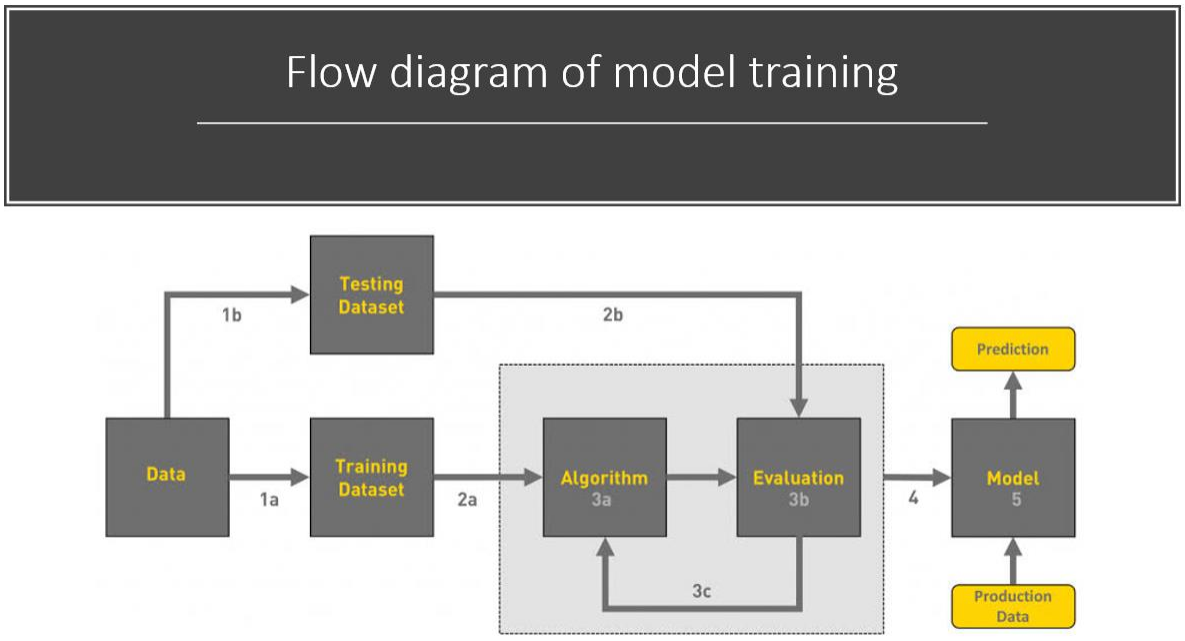
Machine Learning:-

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data.

Supervised Learning:-

- Supervised learning **is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.[1] It infers a function from *labeled training data* consisting of a set of *training examples*.**
- Supervised learning is when the model is getting trained on a labelled dataset.
- Labelled dataset is one which have both input and output parameters.

Flow Diagram:-



Regression:-

Regression analysis is one of the most important fields in statistics and machine learning. There are many regression methods available. Linear regression is one of them.

What Is Regression?

Regression searches for relationships among variables.

For example, you can observe several employees of some company and try to understand how their salaries depend on the **features**, such as experience, level of education, role, city they work in, and so on.

This is a regression problem where data related to each employee represent one **observation**. The presumption is that the experience, education, role, and city are the independent features, while the salary depends on them.

Similarly, you can try to establish a mathematical dependence of the prices of houses on their areas, numbers of bedrooms, distances to the city center, and so on.

Generally, in regression analysis, you usually consider some phenomenon of interest and have a number of observations. Each observation has two or more features. Following the assumption that (at least) one of the features depends on the others, you try to establish a relation among them.

In other words, **you need to find a function that maps some features or variables to others sufficiently well**.

The dependent features are called the **dependent variables**, **outputs**, or **responses**.

The independent features are called the **independent variables**, **inputs**, or **predictors**.

Regression problems usually have one continuous and unbounded dependent variable. The inputs, however, can be continuous, discrete, or even categorical data such as gender, nationality, brand, and so on.

It is a common practice to denote the outputs with y and inputs with x. If there are two or more independent variables, they can be represented as the vector $\mathbf{x} = (x_1, \ldots, x_r)$, where r is the number of inputs.

When Do You Need Regression?

Typically, you need regression to answer whether and how some phenomenon influences the other or **how several variables are related**. For example, you can use it to determine *if* and *to what extent* the experience or gender impact salaries.

Regression is also useful when you want **to forecast a response** using a new set of predictors. For example, you could try to predict electricity consumption of a household for the next hour given the outdoor temperature, time of day, and number of residents in that household.

Regression is used in many different fields: economy, computer science, social sciences, and so on. Its importance rises every day with the availability of large amounts of data and increased awareness of the practical value of data.

Linear Regression

Linear regression is probably one of the most important and widely used regression techniques. It's among the simplest regression methods. One of its main advantages is the ease of interpreting results.

Problem Formulation

When implementing linear regression of some dependent variable y on the set of independent variables $\mathbf{x} = (x_1, \ldots, x_r)$, where r is the number of predictors, you assume a linear relationship between y and $\mathbf{x}$: $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_r x_r + \varepsilon$. This equation is the **regression equation**. β_0, β_1, …, β_r are the **regression coefficients**, and ε is the **random error**.

Regression Performance

The variation of actual responses y_i, $i = 1, \ldots, n$, occurs partly due to the dependence on the predictors $\mathbf{x}_i$. However, there is also an additional inherent variance of the output.

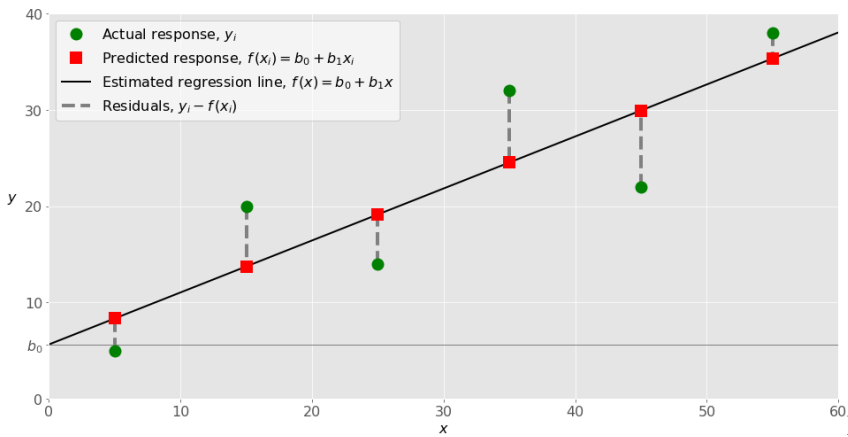
The **coefficient of determination**, denoted as R^2, tells you which amount of variation in y can be explained by the dependence on $\mathbf{x}$ using the particular regression model. Larger R^2 indicates a better fit and means that the model can better explain the variation of the output with different inputs.

The value $R^2 = 1$ corresponds to SSR = 0, that is to the **perfect fit** since the values of predicted and actual responses fit completely to each other.

Simple Linear Regression

Simple or single-variate linear regression is the simplest case of linear regression with a single independent variable, $\mathbf{x} = x$.

The following figure illustrates simple linear regression:



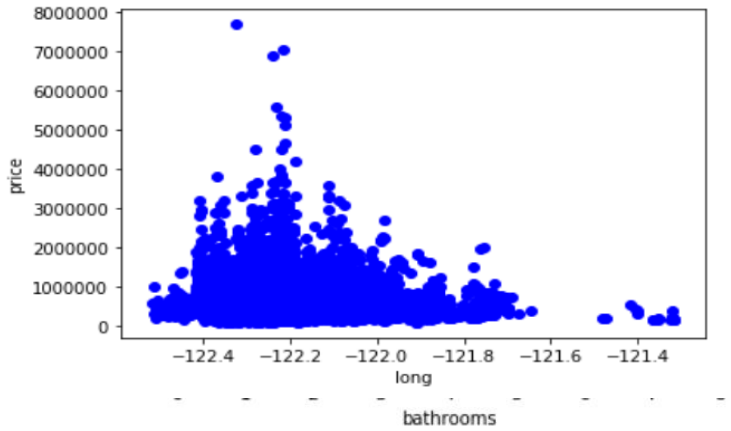
Example of simple linear regression

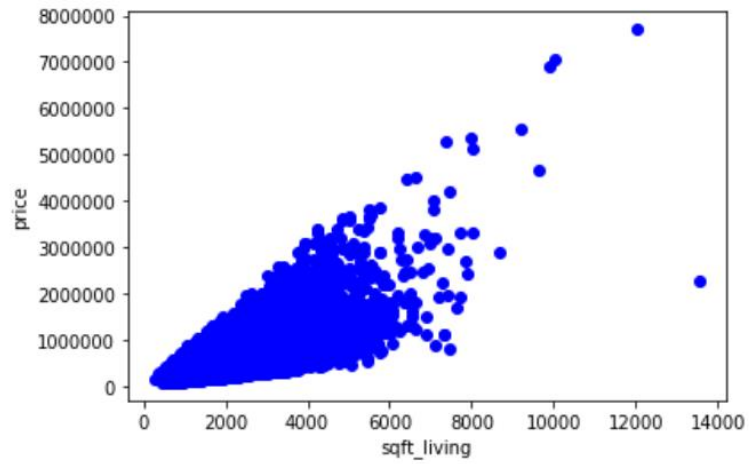
When implementing simple linear regression, you typically start with a given set of input-output (x-y) pairs (green circles). These pairs are your observations. For example, the leftmost observation (green circle) has the input $x = 5$ and the actual output (response) $y = 5$. The next one has $x = 15$ and $y = 20$, and so on.

The estimated regression function (black line) has the equation $f(x) = b_0 + b_1x$. Your goal is to calculate the optimal values of the predicted weights b_0 and b_1 that minimize SSR and determine the estimated regression function. The value of b_0, also called the **intercept**, shows the point where the estimated regression line crosses the y axis. It is the value of the estimated response $f(x)$ for $x = 0$. The value of b_1 determines the **slope** of the estimated regression line.

The predicted responses (red squares) are the points on the regression line that correspond to the input values. For example, for the input $x = 5$, the predicted response is $f(5) = 8.33$ (represented with the leftmost red square).

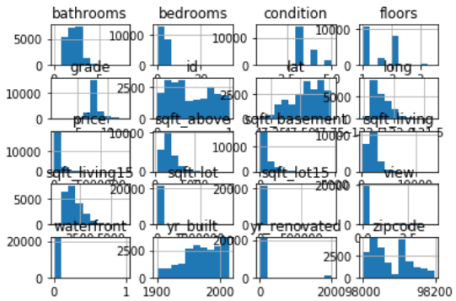
The residuals (vertical dashed gray lines) can be calculated as $y_i - f(\mathbf{x}_i) = y_i - b_0 - b_1x_i$ for $i = 1, \ldots, n$. They are the distances between the green circles and red squares. When you implement linear regression, you are actually trying to minimize these distances and make the red squares as close to the predefined green circles as possible.





In [6]:
```
plt.scatter(cdf.id,cdf.price,color='blue')
plt.xlabel('id')
plt.ylabel('price')
plt.show()

plt.scatter(cdf.date,cdf.price,color='blue')
plt.xlabel('date')
plt.ylabel('price')
plt.show()

plt.scatter(cdf.bedrooms,cdf.price,color='blue')
plt.xlabel('bedrooms')
plt.ylabel('price')
plt.show()

plt.scatter(cdf.bathrooms,cdf.price,color='blue')
plt.xlabel('bathrooms')
plt.ylabel('price')
plt.show()

plt.scatter(cdf.sqft_living,cdf.price,color='blue')
plt.xlabel('sqft_living')
plt.ylabel('price')
plt.show()

plt.scatter(cdf.sqft_lot,cdf.price,color='blue')
plt.xlabel('sqft_lot')
plt.ylabel('price')
plt.show()
```

In [5]:
```
viz = cdf[['id','date','price','bedrooms','bathrooms','sqft_living','sqft_lot','floors','waterfront','view','condition','grade',
viz.hist()
plt.show()
```



In [4]:
```
cdf = df[['id','date','price','bedrooms','bathrooms','sqft_living','sqft_lot','floors','waterfront','view','condition','grade','
cdf.head(9)
```

Out[4]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_bu
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	19
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	19
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	19
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	19
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	19
5	7237550310	20140512T000000	1225000.0	4	4.50	5420	101930	1.0	0	0	...	11	3890	1530	20
6	1321400060	20140627T000000	257500.0	3	2.25	1715	6819	2.0	0	0	...	7	1715	0	19
7	2008000270	20150115T000000	291850.0	3	1.50	1060	9711	1.0	0	0	...	7	1060	0	19
8	2414600126	20150415T000000	229500.0	3	1.00	1780	7470	1.0	0	0	...	7	1050	730	19

9 rows × 21 columns

In [3]:
```
df.describe()
```

Out[3]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	21613.0
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.409430	7.6
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.650743	1.1
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	1.0
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.000000	7.0
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.000000	7.0
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	4.000000	8.0
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	13.0

In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import pylab as pl
import numpy as np
%matplotlib inline
```

In [2]:
```
df = pd.read_csv("house_data.csv")
df.head()
```

Out[2]:

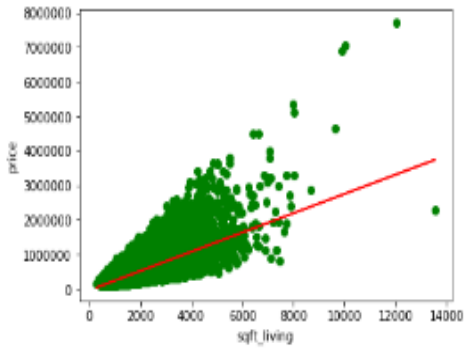
	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_buil
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	195
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	195
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	193
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	196
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	198

5 rows × 21 columns

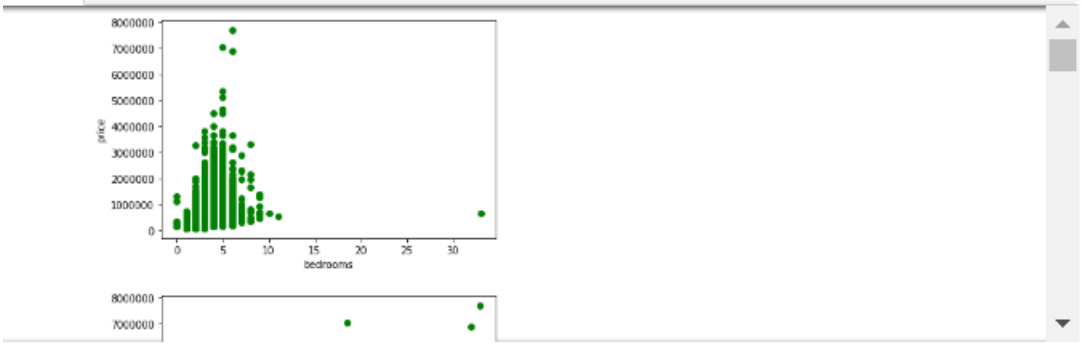
In [11]:
```
from sklearn import linear_model
regr = linear_model.LinearRegression()
#train_x = np.asanyarray(train[['bedrooms']])  ---9%
#train_x = np.asanyarray(train[['bathrooms']]) ---38%
train_x = np.asanyarray(train[['sqft_living']])
#train_x = np.asanyarray(train[['grade']]) ---46%
#train_x = np.asanyarray(train[['sqft_above']]) ---39%
#train_x = np.asanyarray(train[['sqft_basement']]) ---11%
#train_x = np.asanyarray(train[['lat']]) ---10%
#train_x = np.asanyarray(train[['long']]) ---0%
#train_x = np.asanyarray(train[['sqft_living15']]) ---35%
train_y = np.asanyarray(train[['price']])
regr.fit (train_x, train_y)
# The coefficients
print ('Coefficients: ', regr.coef_)
print ('Intercept: ',regr.intercept_)
```

```
Coefficients:  [[278.99677771]]
Intercept:  [-39963.02049679]
```

In [16]:
```
#plt.scatter(train.bedrooms,train.price,color='green')
#plt.scatter(train.bathrooms,train.price,color='green')
plt.scatter(train.sqft_living,train.price,color='green')
#plt.scatter(train.grade,train.price,color='green')
#plt.scatter(train.sqft_above,train.price,color='green')
#plt.scatter(train.sqft_basement,train.price,color='green')
#plt.scatter(train.lat,train.price,color='green')
#plt.scatter(train.long,train.price,color='green')
#plt.scatter(train.sqft_living15,train.price,color='green')
plt.plot(train_x, regr.coef_[0][0]*train_x + regr.intercept_[0], '-r')
#plt.xlabel('bedrooms')
#plt.xlabel('bathrooms')
plt.xlabel('sqft_living')
plt.savefig('graph1.png', dpi=150)
#plt.xlabel('grade')
#plt.xlabel('sqft_above')
#plt.xlabel('sqft_basement')
#plt.xlabel('lat')
#plt.xlabel('long')
#plt.xlabel('sqft_living15')
plt.ylabel('price')
plt.show()
```



In [8]:
```
plt.scatter(train.bedrooms,train.price,color='green')
plt.xlabel('bedrooms')
plt.ylabel('price')
plt.show()

plt.scatter(train.bathrooms,train.price,color='green')
plt.xlabel('bathrooms')
plt.ylabel('price')
plt.show()

plt.scatter(train.sqft_living,train.price,color='green')
plt.xlabel('sqft_living')
plt.ylabel('price')
plt.show()

plt.scatter(train.sqft_lot,train.price,color='green')
plt.xlabel('sqft_lot')
plt.ylabel('price')
plt.show()

plt.scatter(train.grade,train.price,color='green')
plt.xlabel('grade')
plt.ylabel('price')
plt.show()

plt.scatter(train.sqft_above,train.price,color='green')
plt.xlabel('sqft_above')
plt.ylabel('price')
plt.show()

plt.scatter(train.sqft_basement,train.price,color='green')
plt.xlabel('sqft_basement')
plt.ylabel('price')
plt.show()

plt.scatter(train.lat,train.price,color='green')
plt.xlabel('lat')
plt.ylabel('price')
plt.show()

plt.scatter(train.long,train.price,color='green')
plt.xlabel('long')
plt.ylabel('price')
plt.show()

plt.scatter(train.sqft_living15,train.price,color='green')
plt.xlabel('sqft_living15')
plt.ylabel('price')
plt.show()
```



In [7]:
```
msk = np.random.rand(len(df)) < 0.8
train = cdf[msk]
test = cdf[~msk]
```

In [8]:
```
plt.scatter(train.bedrooms,train.price,color='green')
plt.xlabel('bedrooms')
plt.ylabel('price')
plt.show()

plt.scatter(train.bathrooms,train.price,color='green')
plt.xlabel('bathrooms')
plt.ylabel('price')
plt.show()

plt.scatter(train.sqft_living,train.price,color='green')
plt.xlabel('sqft_living')
plt.ylabel('price')
plt.show()

plt.scatter(train.sqft_lot,train.price,color='green')
plt.xlabel('sqft_lot')
plt.ylabel('price')
plt.show()

plt.scatter(train.grade,train.price,color='green')
plt.xlabel('grade')
plt.ylabel('price')
plt.show()

plt.scatter(train.sqft_above,train.price,color='green')
plt.xlabel('sqft_above')
```

Multiple Linear Regression

Multiple or multivariate linear regression is a case of linear regression with two or more independent variables.

If there are just two independent variables, the estimated regression function is $f(x_1, x_2) = b_0 + b_1x_1 + b_2x_2$. It represents a regression plane in a three-dimensional space. The goal of regression is to determine the values of the weights b_0, b_1, and b_2 such that this plane is as close as possible to the actual responses and yield the minimal SSR.

The case of more than two independent variables is similar, but more general. The estimated regression function is $f(x_1, \ldots, x_r) = b_0 + b_1x_1 + \cdots + b_rx_r$, and there are $r + 1$ weights to be determined when the number of inputs is r.

Polynomial Regression

You can regard polynomial regression as a generalized case of linear regression. You assume the polynomial dependence between the output and inputs and, consequently, the polynomial estimated regression function.

In other words, in addition to linear terms like b_1x_1, your regression function f can include non-linear terms such as $b_2x_1^2$, $b_3x_1^3$, or even $b_4x_1x_2$, $b_5x_1^2x_2$, and so on.

The simplest example of polynomial regression has a single independent variable, and the estimated regression function is a polynomial of degree 2: $f(x) = b_0 + b_1x + b_2x^2$.

Now, remember that you want to calculate b_0, b_1, and b_2, which minimize SSR. These are your unknowns!

Keeping this in mind, compare the previous regression function with the function $f(x_1, x_2) = b_0 + b_1x_1 + b_2x_2$ used for linear regression. They look very similar and are both linear functions of the unknowns b_0, b_1, and b_2. This is why you can **solve the polynomial regression problem as a linear problem** with the term x^2 regarded as an input variable.

In the case of two variables and the polynomial of degree 2, the regression function has this form: $f(x_1, x_2) = b_0 + b_1x_1 + b_2x_2 + b_3x_1^2 + b_4x_1x_2 + b_5x_2^2$. The procedure for solving the problem is identical to the previous case. You apply linear regression for five inputs: x_1, x_2, x_1^2, x_1x_2, and x_2^2. What you get as the result of regression are the values of six weights which minimize SSR: b_0, b_1, b_2, b_3, b_4, and b_5.

Of course, there are more general problems, but this should be enough to illustrate the point.

In [8]:
```
from sklearn import linear_model
regr = linear_model.LinearRegression()
x = np.asanyarray(train[['grade','sqft_living','sqft_above','bathrooms']])
y = np.asanyarray(train[['price']])
regr.fit (x, y)
# The coefficients
print ('Coefficients: ', regr.coef_)
print ('Intercept: ',regr.intercept_)
```

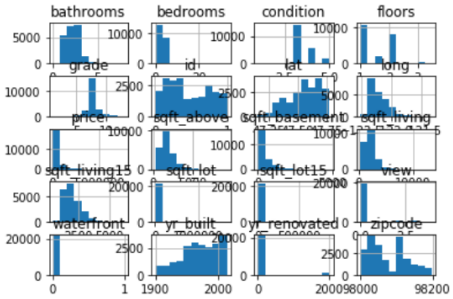
```
Coefficients:  [[ 1.15394510e+05  2.53523700e+02 -7.71761205e+01 -3.78484614e+04]]
Intercept:  [-652632.60286174]
```

In [56]:
```
y_hat= regr.predict(test[['grade','sqft_living','sqft_above','bathrooms']])
x = np.asanyarray(test[['grade','sqft_living','sqft_above','bathrooms']])
y = np.asanyarray(test[['price']])
print("Residual sum of squares: %.2f"
      % np.mean((y_hat - y) ** 2))

# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % regr.score(x, y))
regr.score(x,y)
```

```
Residual sum of squares: 53732804077.37
Variance score: 0.49
```

Out[56]: 0.4940094929167949

In [5]:
```
viz = cdf[['id','date','price','bedrooms','bathrooms','sqft_living','sqft_lot','floors','waterfront','view','condition','grade',
viz.hist()
plt.show()
```



In [6]:
```
msk = np.random.rand(len(df)) < 0.99
train = cdf[msk]
test = cdf[~msk]
```

In [4]:
```
cdf = df[['id','date','price','bedrooms','bathrooms','sqft_living','sqft_lot','floors','waterfront','view','condition','grade','
cdf.head(9)
```

Out[4]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_bu
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	195
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	195
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	193
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	196
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	198
5	7237550310	20140512T000000	1225000.0	4	4.50	5420	101930	1.0	0	0	...	11	3890	1530	200
6	1321400060	20140627T000000	257500.0	3	2.25	1715	6819	2.0	0	0	...	7	1715	0	195
7	2008000270	20150115T000000	291850.0	3	1.50	1060	9711	1.0	0	0	...	7	1060	0	196
8	2414600126	20150415T000000	229500.0	3	1.00	1780	7470	1.0	0	0	...	7	1050	730	196

9 rows × 21 columns

```
In [1]: import matplotlib.pyplot as plt
        import pandas as pd
        import pylab as pl
        import numpy as np
        %matplotlib inline
```

```
In [2]: df = pd.read_csv("house_data.csv")
        df.head()
```

Out[2]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_buil
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	195
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	195
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	193
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	196
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	198

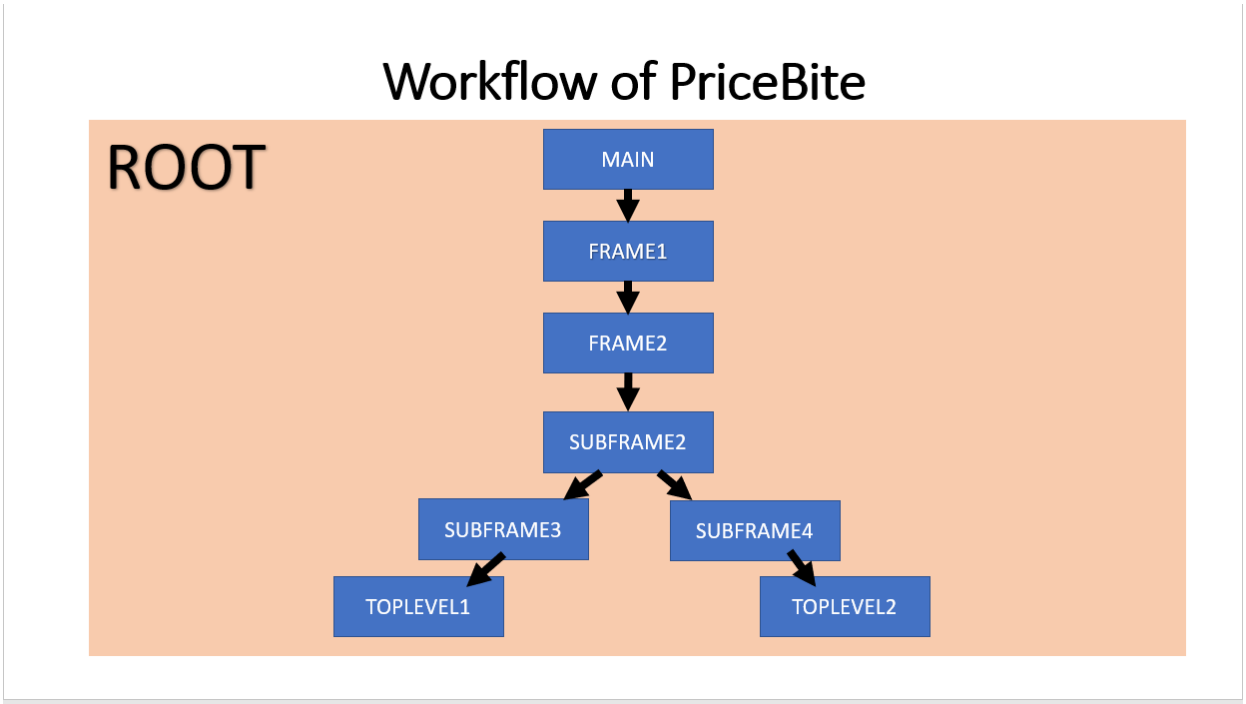
5 rows × 21 columns

```
In [3]: df.describe()
```

Out[3]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	21613.0
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.409430	7.6
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.650743	1.1

Workflow of pricebite:-



TEST-CASES:-

A test case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. The process of developing test cases can also help find problems in the requirements or design of an application.

Test cases:

s.no	Testing name	description	Expected result	Actual result	remarks	output
1.	Dataset	Uploadng and Describe	Display Atributes	Display Atrributes	Details and fields are correct	passed
2.	Program Initiating	Runing the whole algo through the environment	Program run	failed	Failed because technical error	failed
2.	Program Initiating	Runing the whole algo through the environment	Program run	failed	Failed because System issue	failed
2.	Program Initiating	Runing the whole algo through the environment	Program run	passed	NO error or System issue	Passed
3.	Lr and mlr	Runnig of both regression	Worked successfully	Worked successfully	Collection of dataset and the algo run perfectly with good accuracy.	passed

Conclusions:-

It is our team's hope that this document will be of huge help with understanding of our little project as we have used a different approach which has proved beneficial for us and easy for us to understand the vast ocean that is Machine Leaning.We have reached the maximum accuracy of 95% after data cleaning but we will work forward to increase this accuracy little by little.

REFRENCES:-

To conduct this project the following tools have been used :

- Jupyter notebook and spyder
- Pandas (Library) : http://pandas.pydata.org/
- Numpy (Library) : http://www.numpy.org/

1.1 Coursera:-

We have used this side for our basis knowledge gain of the methods that will be used in the project

https://www.coursera.org/learn/machine-learning-with-python

1.2_Kaggle:-

We have used this site for our datasets gathering and asample case project.

https://www.kaggle.com/search?q=house+price+prediction

https://towardsdatascience.com/create-a-model-to-predict-house-prices-using-python-d34fe8fad88f (sample case for study)

1.3 Stackoverflow:-

We have used this site for solving our different problems which has occurred during this project.

https://stackoverflow.com/questions/2620343/what-is-machine-learning