ALGORITHM 1:

HOUSE PRICE PREDICTION USING LINEAR REGRESSION

CHAPTER 1

1.1 CASE STUDY DESCRIPTION

House price prediction using LINEAR REGRESSION

1.1.1 INTRODUCTION

A Python program that predicts the price of houses using a machine learning algorithm called Linear Regression. Linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).

1.1.2 HOW DOES LINEAR REGRESSION WORKS

In linear regression, each observation consists of two values. One value is for the dependent variable and one value is for the independent variable. In this simple model, a straight line approximates the relationship between the dependent variable and the independent variable.

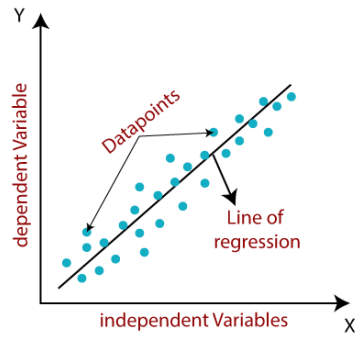
1.1.3 PROBLEM STATEMENT

To build a simple Linear Regression model for predicting the House price.

CHAPTER 2

2.1 ALGORITHM INTRODUCTION

Linear regression is a quiet and simple statistical regression method used for predictive analysis and shows the relationship between the continuous variables. Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), consequently called linear regression. If there is a single input variable (x), such linear regression is called **simple linear regression**. The linear regression model gives a sloped straight line describing the relationship within the variables.



The above graph presents the linear relationship between the dependent variable and independent variables. When the value of x (independent variable) increases, the value of y (dependent variable) is likewise increasing. The red line is referred to as the best fit straight line. Based on the given data points, we try to plot a line that models the points the best.

To calculate best-fit line linear regression uses a traditional slope-intercept form.

$$y = mx+b \quad \Longrightarrow \quad y = a_0+a_1x$$

y= Dependent Variable.

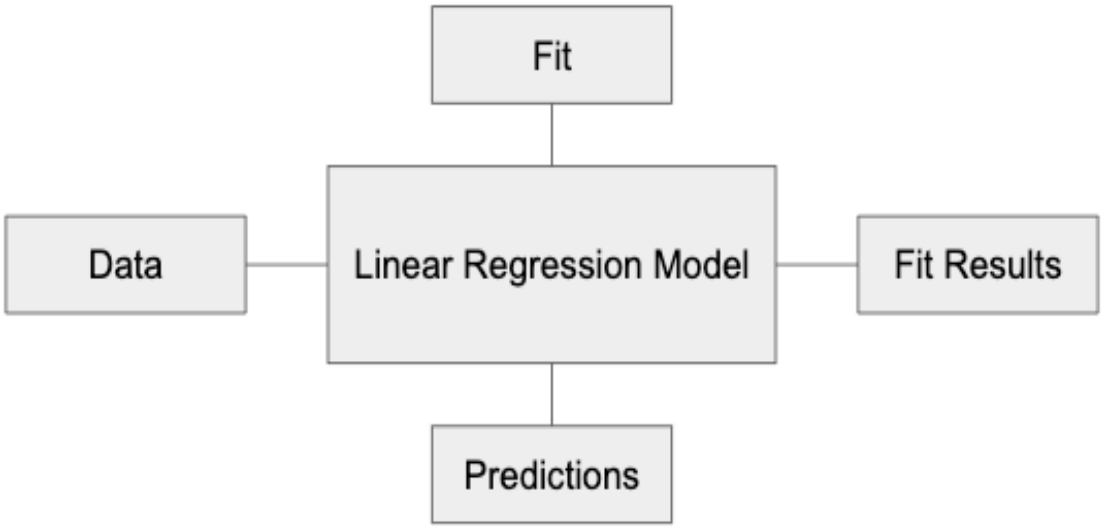
x= Independent Variable.

a0= intercept of the line.

a1 = Linear regression coefficient.

2.2 LINEAR REGRESSION WORKING

1. Import some required libraries.
2. Define the dataset.
3. Plot the data points.
4. Plot the regression line.
5. Predicting the values.



CHAPTER 3

3.1 IMPLEMENTATION

3.1.1 Steps for implementing Linear Regression algorithm

Importing Libraries

First import the dependencies, that will make this program a little easier to write. Importing the machine learning library numpy , pandas , matplotlib and seaborn.

```
In [1]: import pandas as pd
        import numpy as np
```

```
In [8]: import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline
```

Importing the Dataset

To import the dataset and load it into our pandas dataframe, execute the following code and to see what the dataset actually looks like, execute the following command:

```
In [2]: data=pd.read_csv("train.csv")
```

```
In [3]: data.head()
```

Out[3]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	F
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	

5 rows × 81 columns

To get some statistics from the data set like the count or the number of rows the data contains for each column, the minimum value for each column, the maximum value for each column, and the mean for each column.

```
In [6]: data.describe()
```

Out[6]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnr
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1452.00
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.267808	1984.865753	103.68
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.202904	20.645407	181.06
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.00
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.000000	1967.000000	0.00
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.000000	1994.000000	0.00
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.000000	2004.000000	166.00
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.00

8 rows × 38 columns

Scatter plots are the graphs that present the relationship between two variables in a data-set. It represents data points on a two-dimensional plane or on a **Cartesian system**. The independent variable or attribute is plotted on the X-axis, while the dependent variable is plotted on the Y-axis. These plots are often called **scatter graphs** or **scatter diagrams**.

```
In [14]: plt.scatter(data['GrLivArea'],data['SalePrice'])
         plt.xlabel('Ground Living Area(in sqaure feet)')
         plt.ylabel('SalePrice')
         plt.show()
```

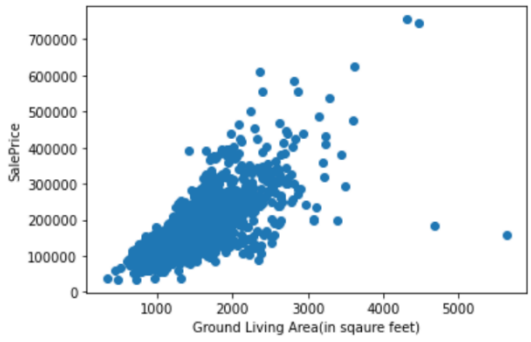


Fig : Scatter plot between Ground living area and saleprice

```
In [15]: plt.scatter(data['GarageArea'],data['SalePrice'])
         plt.xlabel('Garage Area')
         plt.ylabel('SalePrice')
         plt.show()
```

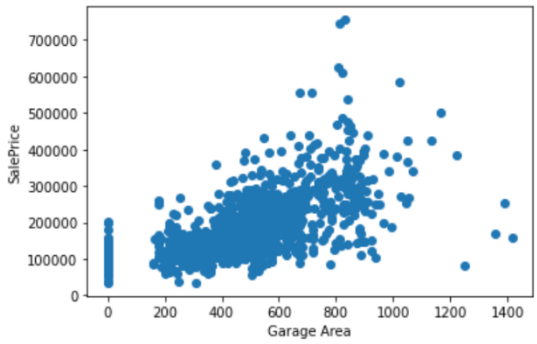


Fig : Scatter plot between Garage area and saleprice

Initialize the Linear Regression model, split the data into 80% training and 20% testing data, and then train the model with the training data set that contains the independent variables.

```
In [23]: from sklearn.model_selection import train_test_split
         X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=42)
```

```
In [24]: from sklearn.linear_model import LinearRegression
         linear_reg=LinearRegression()
         linear_reg.fit(X_train,y_train)
```

Now we are done training the linear regression model that describe the linear function, so print the model's predictions (what it thinks the values will be for houses) on the test data.

CHAPTER 4

4.1 OUTCOME

```
In [25]: print(linear_reg.score(X_test,y_test))

         0.7986350795736639
```

```
In [66]: y_pred = linear_reg.predict(X_test)
         print(y_pred)

[149575.3367402  158230.46370932 108496.2982573  141537.83886374
 157079.77923371 266339.39714974 160676.92370332 161327.45788819
 223054.45004596 134569.62405405 190235.28420267 313752.04777606
 235532.31134564 171356.85682057 116443.12817273 122963.66464875
 181032.5513217  203871.3381106  201252.0846765  194611.14198792
 222067.6687562  122053.00429987  97944.52355641 220639.43213173
 144038.26842503 109284.61162159 468973.51695547 133395.15027051
 170580.14794245 140716.44980235 209365.18080575 235226.24568971
 199572.1022882  278863.99644345 309983.05964858 214820.94568093
 195909.45138059 160542.56676926 188232.37184063  83366.16524593
 178077.39329344 258455.68544767 124788.34147503 107059.75995948
 288105.67944994 239599.07357772 126509.28401483 199670.36952673
 148970.4806716  179254.63797342 200771.96972668 263018.07538204
 170863.54285273 180350.52778316 215129.20038659 243827.64999693
 159155.96095527 176393.12581747 206624.73470181  74371.55919139
 204209.78064052 143984.36591997 110440.2732703  106970.12118131
 305317.11912672 109941.08605643  96966.04521749 188792.17645875
 159265.42849905 415845.58462644 261163.8360221  196380.31923455
 141565.83074738 321070.93769255 163502.53091932 120199.7414433
 193127.86462027 326629.27437008 136371.77182833  80594.56305263
 222354.54494346 124163.52090755 115038.5213363  170585.38016946
 180048.58860577 237454.46997884 186731.64411489 205773.41495385
 222503.4651849  204650.18709144 159615.56057591 171297.47601245
```

Fig : A small sample of the predicted values

To know what was the actual values for that test data set, print those values to the screen, but first print at least one row from the model's prediction, just to make it a little easier to compare data.

```
In [67]: #print the actual values
         print(y_test)

1297    140000
132     150750
300     157000
1307    138000
778     144000
         ...
401     164990
543     133000
1062     90000
1026    167500
246     137000
Name: SalePrice, Length: 225, dtype: int64
```

Fig : The printed actual values of the testing data set

By looking at the predicted values that the model came up with, and the actual values of the testing data set, it looks like the model is pretty good at making predictions. It’s not exact, but it is pretty close and much better than guessing. But to check the model performance using a more mathematical approach.

ALGORITHM 2: SIZE PREDICTION USING KNN

CHAPTER 1

1.2 CASE STUDY DESCRIPTION

size of a person using KNN

1.1.1INTRODUCTION

K Nearest Neighbour Classification Algorithm popularly known by the name KNN classifiers. We will mainly focus on learning to build your first KNN model.

Classification Machine Learning is a technique of learning where a particular instance is mapped against one of the many labels. The labels are prespecified to train your model . The machine learns the pattern from the data in such a way that the learned representation successfully maps the original dimension to the suggested label/class without any more intervention from a human expert.

1.1.2 HOW DOES KNN WORKS

The KNN algorithm is belongs to the family of instance-based, competitive learning and lazy learning algorithms.

Instance-based algorithms are those algorithms that model the problem using data instances (or rows) in order to make predictive decisions. The KNN algorithm is an extreme form of instance-based methods because all training observations are retained as part of the model. KNN is powerful because it does not assume anything about the data, other than a distance measure can be calculated consistently between any two instances. As such, it is called non-parametric or non-linear as it does not assume a functional form.

1.1.3 PROBLEM STATEMENT

To build a simple KNN classification model for predicting the size of a person given few of other person attributes.

CHAPTER 2

2.1 ALGORITHM INTRODUCTION

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

2.2 KNN WORKING

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of K number of neighbors
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

CHAPTER 3

3.1 IMPLEMENTATION

3.1.1 Steps for implementing KNN algorithm

Step 1- Import pandas,and read CSV

```
In [1]: import pandas as pd
        data_set=pd.read_csv("KNN1.csv")
```

Step 2 - Separate Features and Target

```
In [5]: X=data_set.drop('size',axis=1).values
        y=data_set['size'].values
```

Step 3 - Split features and target into test and train

```
In [3]: from sklearn.model_selection import train_test_split
        X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

Step 4 - Import KNN create object,train model

```
In [4]: from sklearn.neighbors import KNeighborsClassifier
        knn=KNeighborsClassifier(n_neighbors=3)

In [5]: knn.fit(X_train,y_train)
```

Step 5 – check accuracy

```
In [12]: accuracy=knn.score(X_test,y_test)
         print(accuracy)

         1.0
```

Step 6 – Do prediction

```
In [10]: prediction=knn.predict([[150,45]])
         print(prediction)

         ['M']
```

Few other person attributes(dataset) stored in csv file:

KNN1.csv

height,weight,size

158,58,M

158,59,M

158,63,M

160,59,M

163,60,M

163,64,L

165,61,L

165,62,L

165,65,L

168,65,L

CHAPTER 4

4.1 OUTCOMES

```
In [14]: prediction=knn.predict([[160,55]])
         print(prediction)

         ['M']
```

```
In [15]: prediction=knn.predict([[170,56]])
         print(prediction)

         ['L']
```