Aim -

Implement a Linear Regression Model to predict house prices for regions in the USA using the provided dataset.

Objective - Develop a model to estimate house prices based on relevant features using Linear Regression.

Software Requirements –

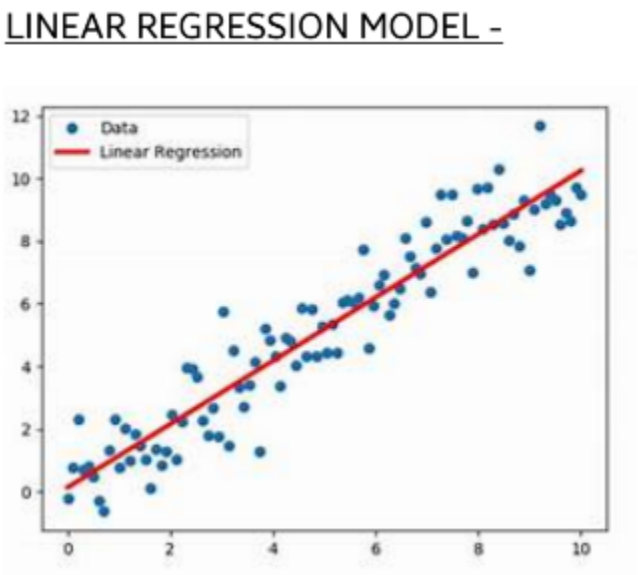
1) Python (3.x recommended) .

2) Jupyter Notebook or any Python IDE

Hardware Requirements - A machine with sufficient RAM and processing power for model training (8GB RAM recommended)

Dataset - https://github.com/huzaifsayed/Linear-Regression-Model-for-House-Price-

LINEAR REGRESSION MODEL –



Libraries or Modules Used –

1)NumPy

2) pandas

3)scikit-learn

4) Matplotlib

5) Seaborn

Theory -

When we see a relationship in a scatterplot, we can use a line to summarize the relationship in the data. We can also use that line to make predictions in the data. This process is called linear regression.Linear Regression is a supervised learning algorithm used for predicting a continuous outcome, typically represented by the target variable. In the context of this assignment, we aim to predict house prices based on various features such as area income, house age, number of rooms, and others. Linear regression is used to study the linear relationship between a dependent variable Y (blood pressure) and one or more independent variables X (age, weight, sex). The dependent variable Y must be continuous, while the independent variables may be either continuous (age), binary (sex), or categorical (social status). The initial judgment of a possible relationship between two continuous variables should always be made on the basis of a scatter plot (scatter graph). This type of plot will show whether the relationship is linear (figure 1) or nonlinear (figure 2).

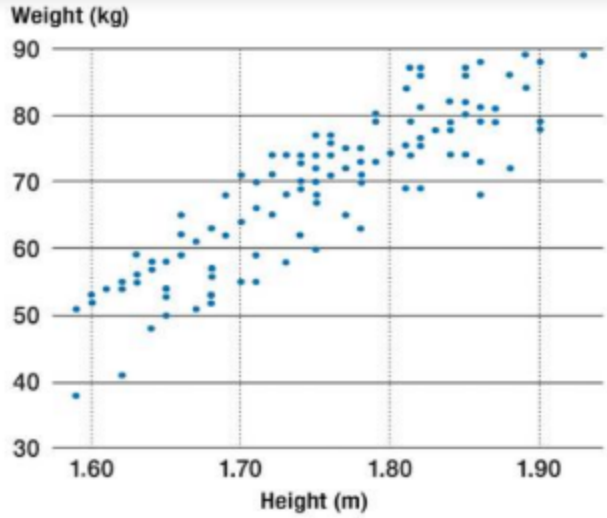
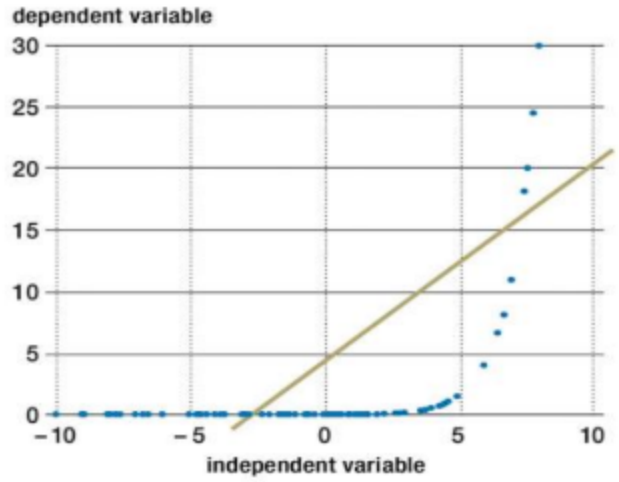


Figure 1
A scatter plot showing a linear relationship



Simple linear regression formula- The formula for a simple linear regression is:

$$y = \beta_0 + \beta_1 X + \epsilon$$

- y is the predicted value of the dependent variable (y) for any given value of the independent variable (x).

- B0 is the intercept, the predicted value of y when the x is 0.

-B1 is the regression coefficient – how much we expect y to change as x increases. x is the independent variable (the variable we expect is influencing y).

- e is the error of the estimate, or how much variation there is in our estimate of theregression coefficient.

Linear regression finds the line of best fit line through your data by searching for the regression coefficient (B1) that minimizes the total error (e) of the model. While you can perform a linear regression by hand, this is a tedious process, so most people use statistical programs to help them quickly analyze the data.

Model Training – Training a regression model involves teaching the model to predict continuous values based on input features. Here's a brief explanation of the process, along with some images to illustrate key concepts.

Regression Model Training:

1. Data Collection: Gather a dataset with input features (independent variables) andcorresponding target values (dependent variable).

2. Data Splitting: Split the dataset into training and testing sets. The training set is used to train the model, and the testing set is used to evaluate its performance.

3. Model Selection: Choose a regression model architecture. Common choices include linear regression, decision trees, or more complex models like neural networks.

4. Feature Scaling: Normalize or standardize the input features to ensure that they are on asimilar scale. This helps the model converge faster during training.

5. Model Training: Feed the training data into the chosen model and adjust the model'sparameters to minimize the difference between predicted and actual target values.

6. Loss Function: Use a loss function to measure the difference between predicted and actualvalues. The goal is to minimize this loss during training.

7. Gradient Descent: Use optimization algorithms like gradient descent to iteratively update themodel parameters and reduce the loss.

8. Model Evaluation: Evaluate the trained model on the testing set to assess its performance onunseen data.

9. Prediction: Once satisfied with the model's performance, use it to make predictions on new, unseen data.

10 Model Deployment: If the model performs well, deploy it to production for making real-world predictions.

Evaluation Metrics - The performance of the model will be assessed using various evaluation metrics, such as Mean Squared Error (MSE) and R-squared. These metrics provide insights into how well the model

generalizes to unseen data.

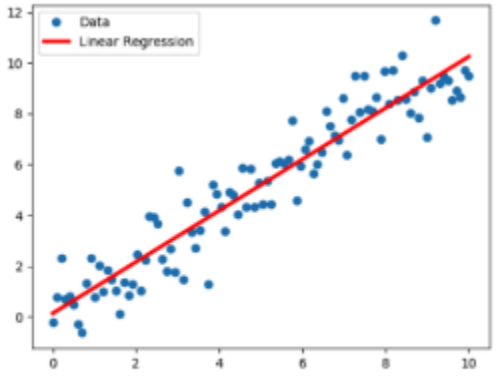


Figure: Linear Regression Model

Algorithm –

Simple Linear Regression Algorithm –

You ll start with the simplest case, which is simple linear regression. There are five basic steps when you re-implementing linear regression:

1. Import the packages and classes that you need.

2. Provide data to work with, and eventually do appropriate transformations.

3. Create a regression model and fit it with existing data.

4. Check the results of model fitting to know whether the model is satisfactory.

5. Apply the model for predictions.

Application –

1. Real estate pricing strategies

2. Assisting clients in making informed decisions

3. Market analysis for regions in the USA

4. Predictive tool for estimating house prices

Code -

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read_csv('USA_Housing.csv')

# Explore the dataset

print(df.head())

print(df.info())

# Handle missing values if any

df = df.dropna()

# Select relevant features and target variable

X = df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms', 'Avg. Area

Number of Bedrooms', 'Area Population']]
```

```
y = df['Price']

# Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Linear Regression model

model = LinearRegression()

# Train the model

model.fit(X_train, y_train)

# Make predictions on the test set

y_pred = model.predict(X_test)

# Evaluate the model performance

mse = mean_squared_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')

print(f'R-squared: {r2}')
```