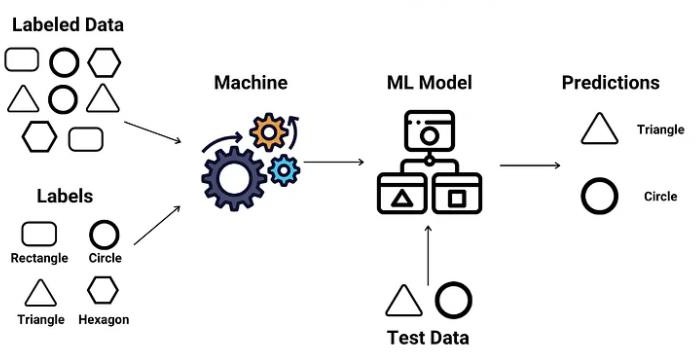
### Supervised Learning-Regression

Introduction to Regression analysis

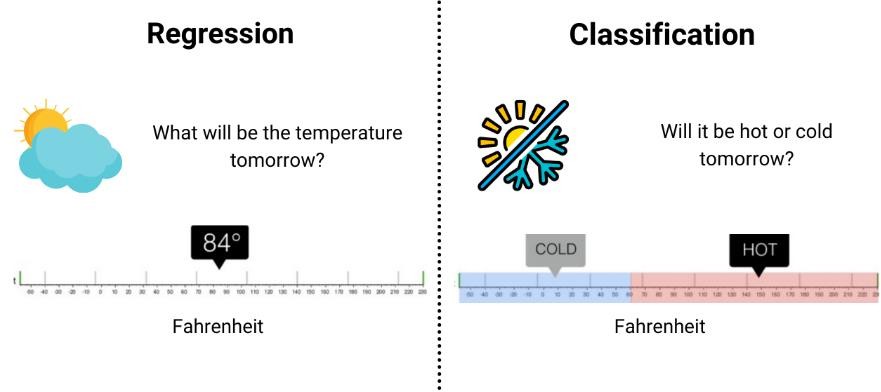
### Supervised Learning

Supervised learning is a type of machine learning where an algorithm is trained on labeled data. The model learns from input-output pairs, where the correct output is already known. The goal is to predict the output for new, unseen data.



There are two main types of supervised learning:

1. **Classification**: Predicts discrete labels (e.g., spam or not spam).
2. **Regression**: Predicts continuous values (e.g., house prices).



### Classification

Classification models classify our outputs to certain categories. If the number of categories are only two then it is specially called binary classification. For greater number of categories it is called multi-class classification.

Some examples are:

* + *Whether a patient has cancer or not*

### Regression

Regression models are for labeling outputs with continuous values.

* + Predicting house prices, or
  + How long is it gonna take you to get home

are both for regression models because the results are ever changing. The key components of supervised learning include:

1. **Input Features (X)**: These are the independent variables or attributes used by the model to make predictions. They could be anything from numerical data, such as age or income, to more complex data like images or text.
2. **Output Labels (Y)**: These are the dependent variables or target

outcomes that the model is trying to predict. In classification tasks, the outputs are discrete labels (e.g., "dog" or "cat"), while in regression

tasks, the outputs are continuous values (e.g., house prices).

1. **Training Data**: A labeled dataset that the model uses to learn. Each example in the dataset consists of an input (X) and its corresponding correct output (Y).
2. **Learning Algorithm**: This is the method the model uses to learn the relationship between inputs and outputs.

### Introduction to Regression Analysis:

Regression analysis is a key technique in supervised learning used to predict continuous outcomes. It helps in understanding the relationship between a dependent variable (the outcome) and one or more independent variables (the inputs or predictors). The primary goal of

regression is to create a model that can predict the value of the dependent variable given the independent variables.

There are various types of regression techniques, with the most common being:

1. **Linear Regression**: Models the relationship between the dependent variable (Y) and one or more independent variables (X) by fitting a straight line (Y = bX + c).
2. **Multiple Regression**: Extends linear regression to multiple input variables.
3. **Polynomial Regression**: Fits a curve to the data instead of a straight line.
4. **Logistic Regression**: Though called regression, it’s used for classification tasks, predicting the probability of an event.

### Simple Linear Regression:

Simple linear regression is a model that assesses the relationship between a dependent variable and an independent variable.

The simple linear model is expressed using the following equation:

### y=mx+b

where,

y - Dependent variable

x - Independent variable m - Slope

b - Intercept

# Y

## Y = mX + b

## m = Slope

**Change in Y**

**Change in X**

**b = Y-intercept**

# X

**Example:** Suppose there is a marketing company A, who does various advertisement every year and get sales on that. The below list shows the advertisement made by the company in the last 5 years and the

corresponding sales:

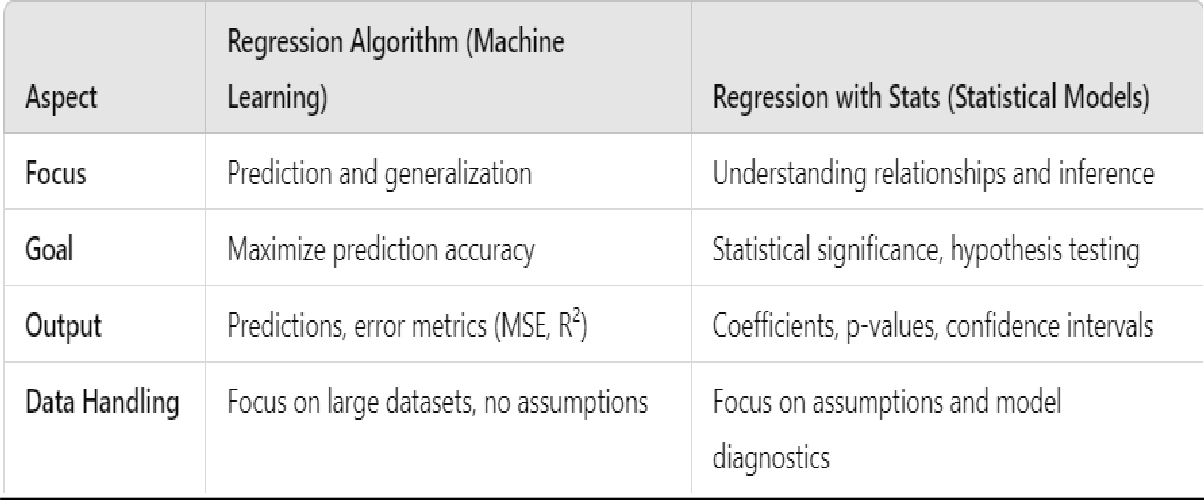


### Supervised Learning-Regression

**Measure of linear relationship, Regression with stats models**

**Regression algorithm** (uisng machine learning) and **regression with statistical models** lies in their primary goals, approaches, and outputs. Both aim to model relationships between variables, but their use cases and how they handle data can be quite distinct.

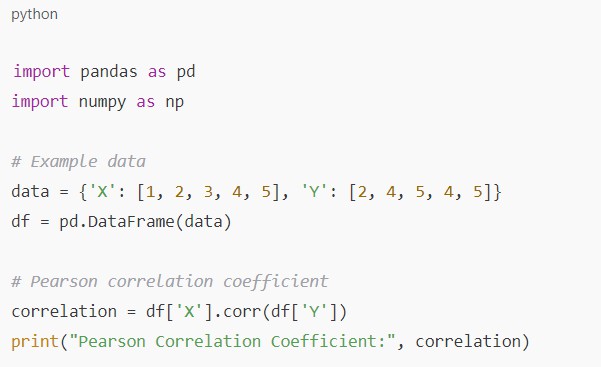
* **Regression Algorithm (Machine Learning)**: The focus is on **prediction** and **generalization** to unseen data. It aims to create a model that can predict the target (dependent variable) based on the features (independent variables.
* **Regression with Stats (Statistical Models)**: The focus is on **understanding relationships** between variables. It’s often used for explaining the nature of the relationship between independent variables (predictors) and the dependent variable (outcome).



To measure the linear relationship between two variables and perform regression analysis using **statsmodels** in Python, you can follow these steps:

### Measure Linear Relationship (Correlation)

Before performing regression, it's common to measure the linear relationship between two variables using **correlation**. You can use **Pearson's correlation coefficient** to measure this.



Pearson Correlation Coefficient: 0.7745966692414834 The correlation coefficient ranges from -1 to 1:

* + 1 indicates a perfect positive linear relationship.
  + -1 indicates a perfect negative linear relationship.
  + 0 indicates no linear relationship.

### Linear Regression Using Statsmodels

You can use the **statsmodels** library to perform linear regression. Here's an example:

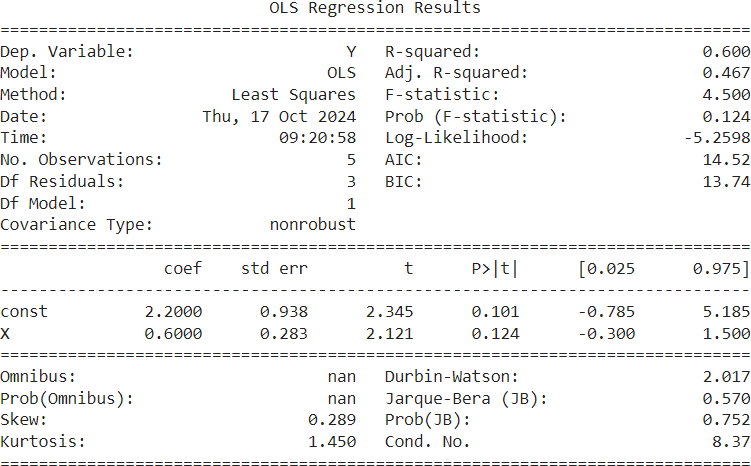
### Step-by-step guide:

1. Install **statsmodels** if not already installed.



### Perform the regression:

* + **sm.add\_constant(X)** adds a constant term to the predictor variable for the intercept.
  + **sm.OLS(Y, X)** creates the OLS (Ordinary Least Squares) model.
  + **model.fit()** fits the model to the data.
  + **model.summary()** provides a detailed statistical summary, including coefficients, R-squared value, p-values, and confidence intervals.



### Interpreting the Results

* + **R-squared**: Indicates how well the independent variable explains the variability in the dependent variable.
  + **p-value**: Tests the null hypothesis that the coefficient of a variable is zero (no relationship). A p-value less than 0.05 usually indicates statistical significance.
  + **Coefficients**: The slope of the regression line (relationship strength) and the intercept.

Studying regression with **statsmodels** offers several advantages, especially for those who want a deeper understanding of regression analysis and statistical modeling. Here are some reasons why **statsmodels** is beneficial for studying regression:

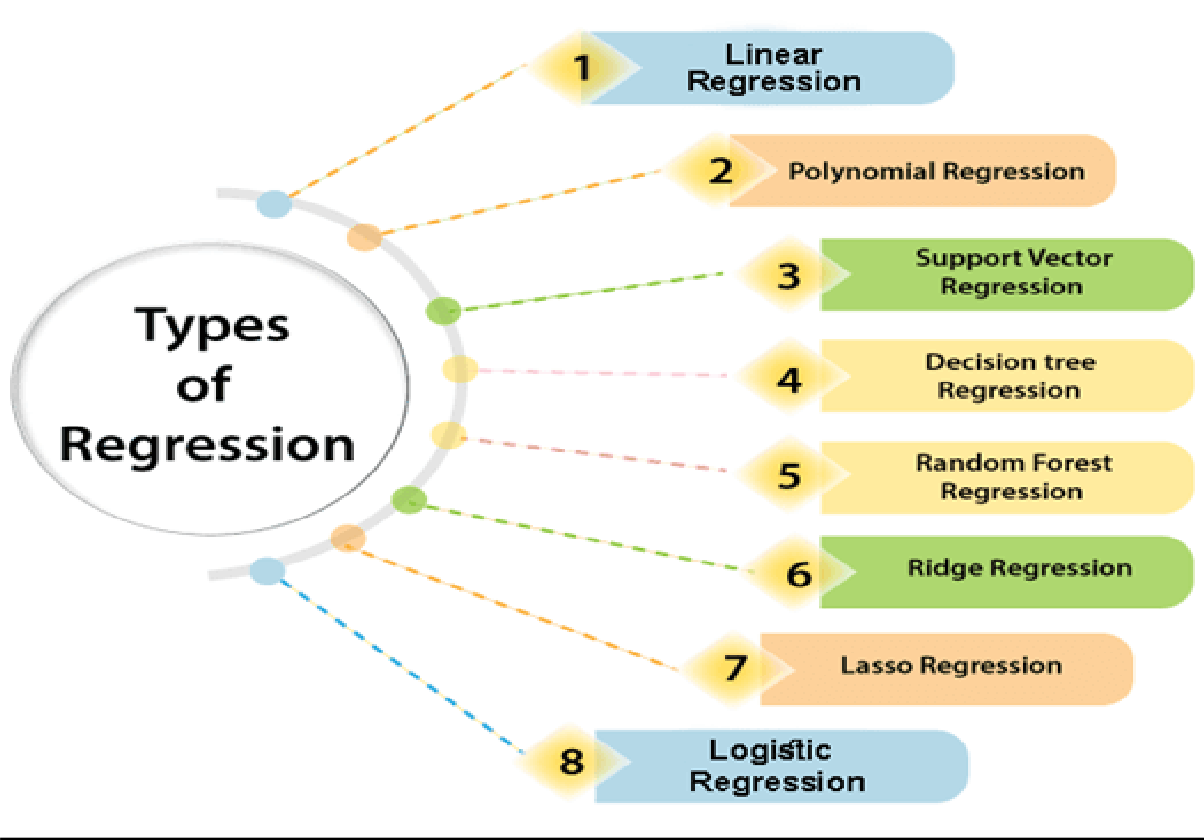
### Detailed Statistical Output

Unlike other libraries like **scikit-learn**, which focus primarily on prediction, **statsmodels** provides comprehensive statistical information about the regression model. The output includes:

* + **R-squared and Adjusted R-squared**: Measures of model fit.
  + **P-values**: Help determine the statistical significance of predictors.
  + **Confidence Intervals**: Shows the range of values for the coefficients with a certain level of confidence.

## Supervised Learning-Regression

**Types of regression**

****

### Linear Regression

The most extensively used modelling technique is linear regression, which assumes a linear connection between a dependent variable (Y) and an independent variable (X).

It employs a regression line, also known as a best-fit line. The linear connection is defined as **Y = c+m\*X + e**

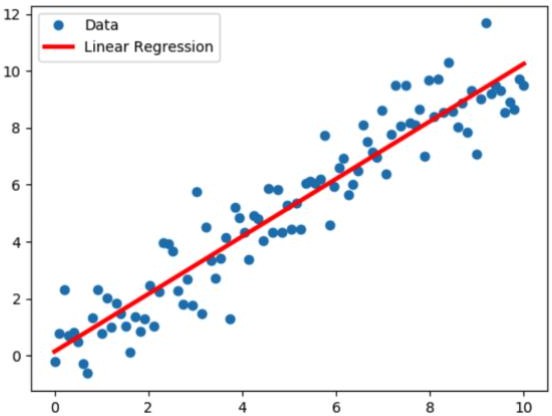
Where,

‘c’ denotes the intercept,

‘m’ denotes the slope of the line, and ‘e’ is the error term.

The linear regression model can be simple (with only one dependent and one independent variable) or complex (with numerous dependent and independent

variables) (with one dependent variable and more than one independent variable).



### Y = c+m\*X + e

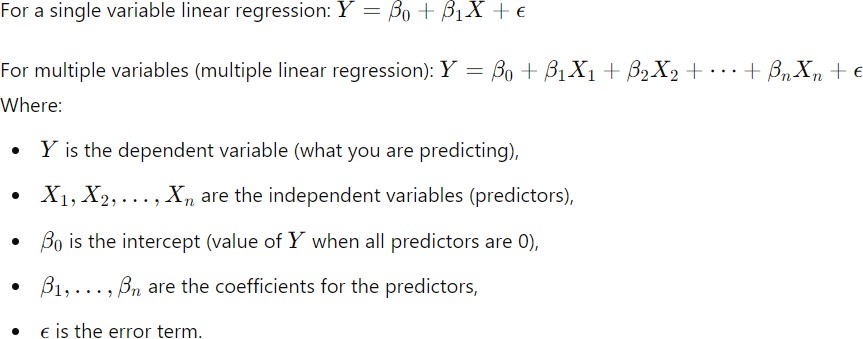
The best fit line is determined by varying the values of m and c.

The predictor error is the difference between the observed values and the predicted value.

The values of m and c get selected in such a way that it gives the minimum predictor error.

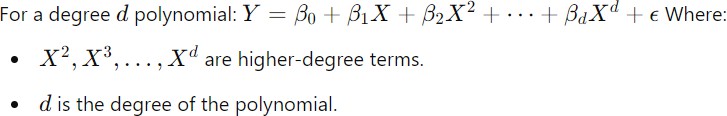
There are different types of linear regression. The two major types of linear regression are

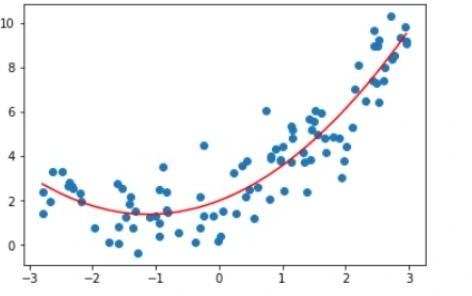
* simple linear regression and
* multiple linear regression.

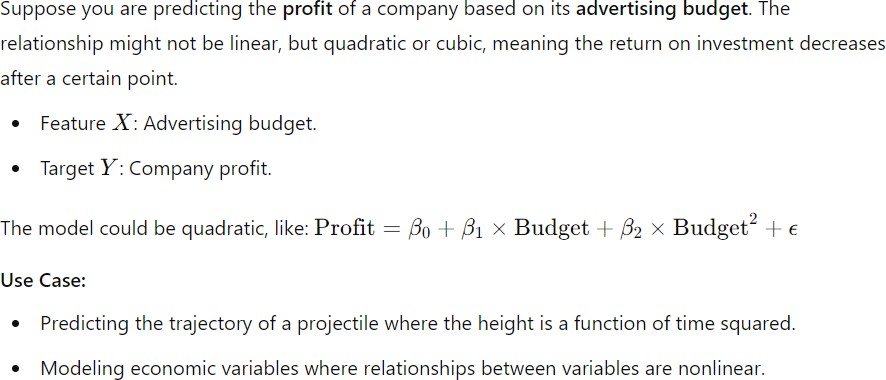


### Polynomial Regression

Polynomial regression is an extension of linear regression where the relationship between the dependent variable and the independent variables is modeled as a polynomial. This allows for **non-linear relationships** between the variables but still uses the framework of linear regression to estimate the model coefficients.







## Supervised Learning-Regression Simple Linear Regression using sample dataset.

Simple Linear Regression is a statistical technique that models the relationship between two variables: a dependent variable (Y) and an independent variable

(X). It assumes that the relationship between the variables is linear, which means that we can represent it with a straight line of the form:



Where:

* Y is the dependent variable (what you are trying to predict).
* X is the independent variable (the predictor).
* β0 is the intercept of the regression line (the value of Y when X = 0).
* β1 is the slope of the regression line (how much Y changes for a unit change in X).
* ϵ is the error term (captures any deviations from the exact linear relationship).

### Methodology

1. **Hypothesis Setup**: Simple linear regression aims to find the line that best fits the data points in a two-dimensional plane.
2. **Data Collection**: Collect data where you have observations for both the dependent and independent variables.
3. **Fitting the Model**: The goal is to estimate the parameters β0 & β1 such that the sum of squared residuals (the difference between the actual value and the predicted value) is minimized.
4. **Model Evaluation**: After fitting the model, evaluate how well the model predicts by using metrics such as **R-squared** and the **mean squared error (MSE)**.
5. **Prediction**: Once the model is trained, we can use the estimated coefficients to predict new data.

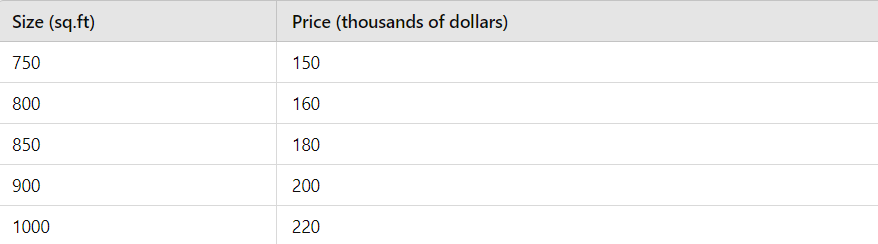
### Use Case: Predicting House Prices

Consider a real-world scenario where you want to predict house prices

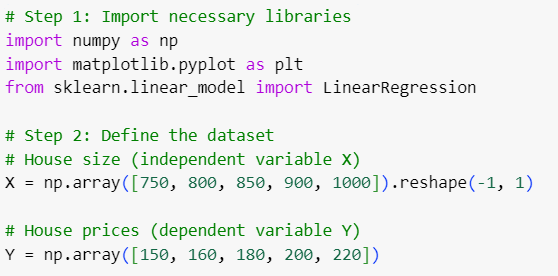
(Y) based on the size of the house (X). This is a simple linear regression problem because house price can be predicted using one feature: house size.

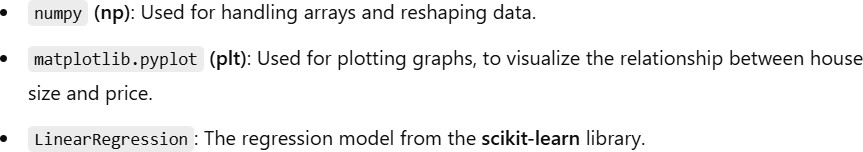
### Dataset

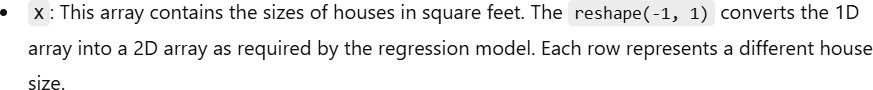
For simplicity, let's use a sample dataset of house sizes (in square feet) and house prices (in thousands of dollars):

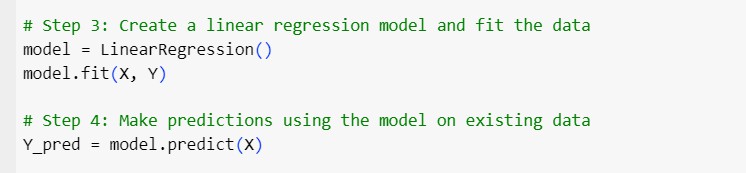


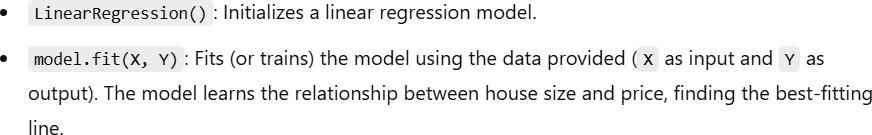
* **Import Libraries**: The necessary libraries to be imported (numpy, matplotlib, and sklearn).
* **Define the Dataset**: The initial dataset of house sizes and prices
* **Model Creation**: A LinearRegression object is created, and the model is trained on the existing dataset using the .fit() method.
* **New Data for Prediction**: A new dataset (new\_house\_sizes) is created with house sizes of 1100, 1200, and 1300 square feet. This data will be used to make new price predictions using the .predict() method.
* **Plotting**: The plot includes the original data points, the regression line, and the predicted prices for the new house sizes (displayed in green).

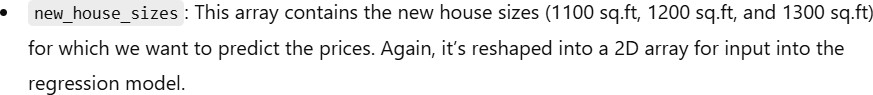
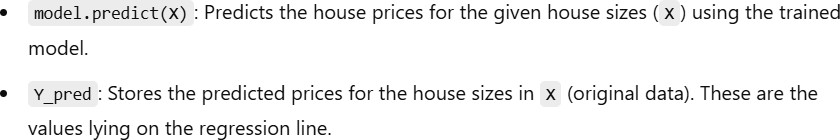


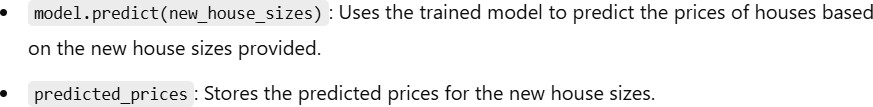


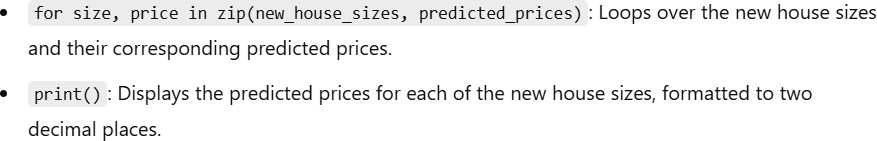


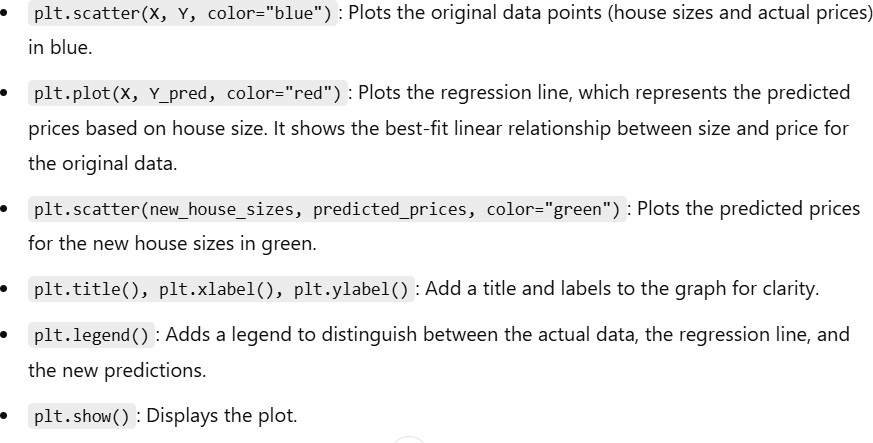








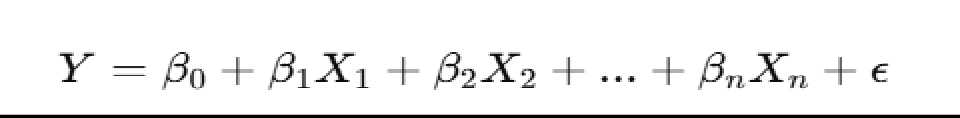






## Supervised Learning-Regression Multiple Linear Regression using sample dataset

Multiple Linear Regression is an extension of Simple Linear Regression where we model the relationship between a dependent variable (Y) and multiple independent variables (X1, X2, ..., Xn). The goal is to predict the value of the dependent variable based on several features.



Where:

* Y is the dependent variable (what you are trying to predict).
* X1,X2,...,Xn are the independent variables (predictors).
* β0 is the intercept.
* β1,β2,...,βn are the coefficients for the independent variables.
* ϵ is the error term (captures deviations from the predicted linear relationship).

### Use Case: Predicting House Prices with Multiple Features

Consider a real-world example where we want to predict house prices (Y) based on multiple factors such as:

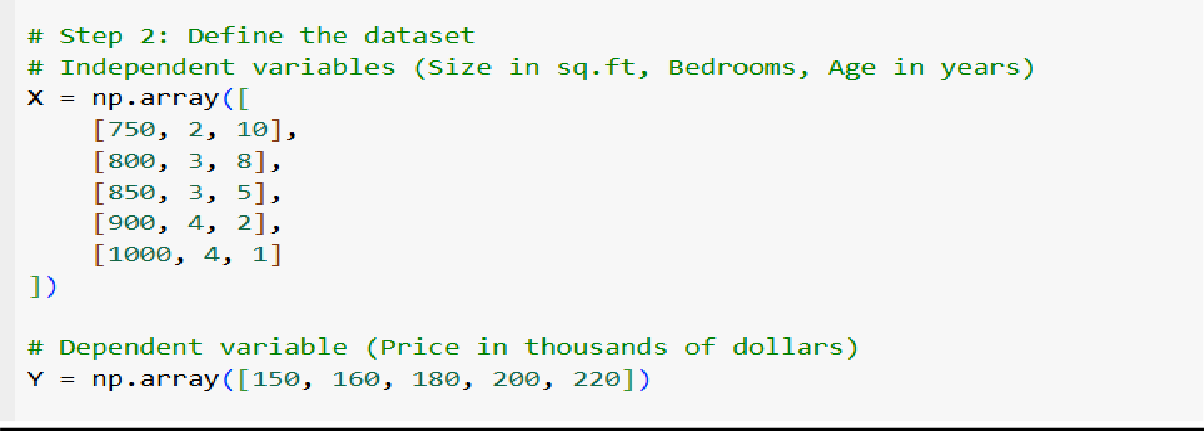
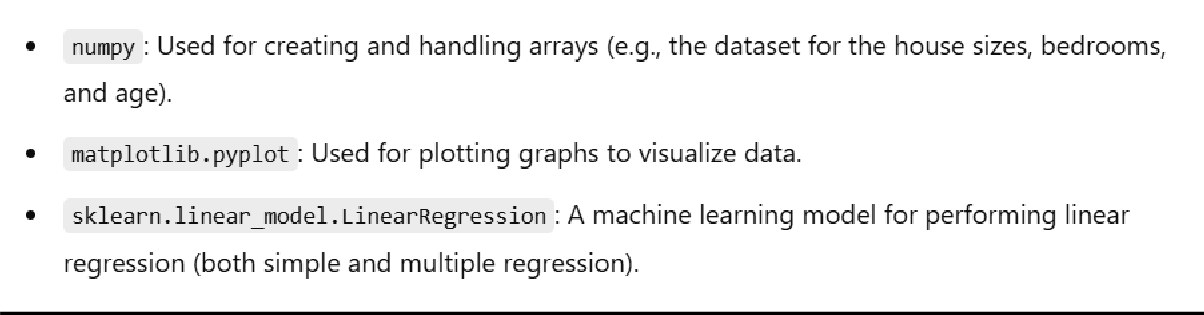
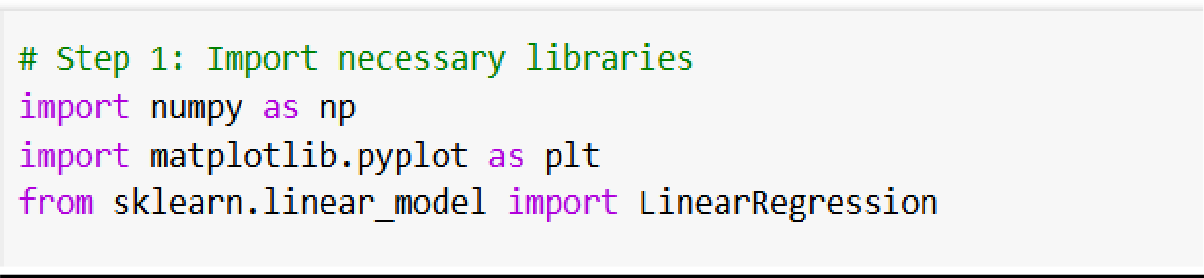
* Size of the house (X1)
* Number of bedrooms (X2)
* Age of the house (X3)

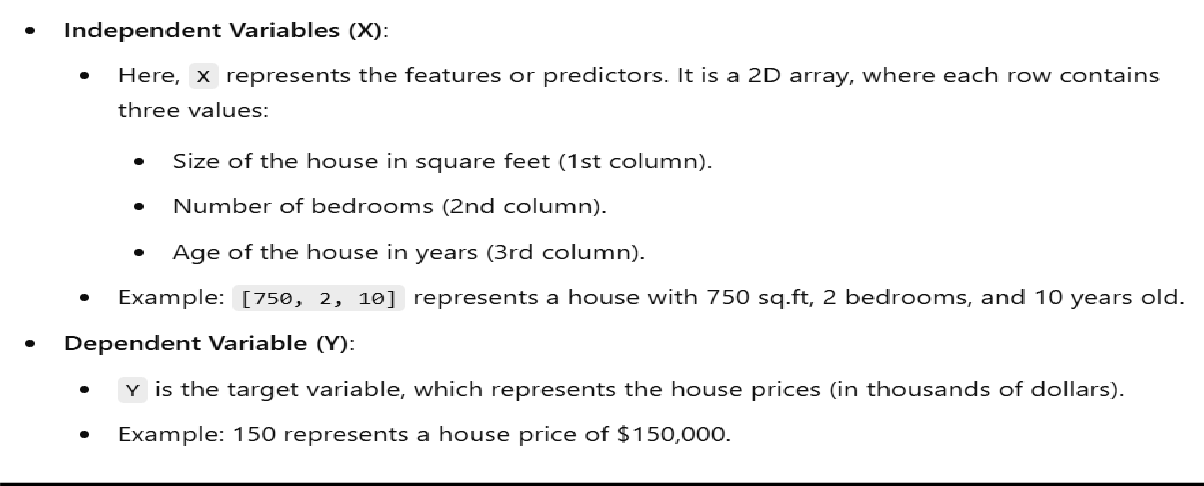
These factors influence the price of the house, and by using multiple linear regression, we can predict prices more accurately than using just one factor.

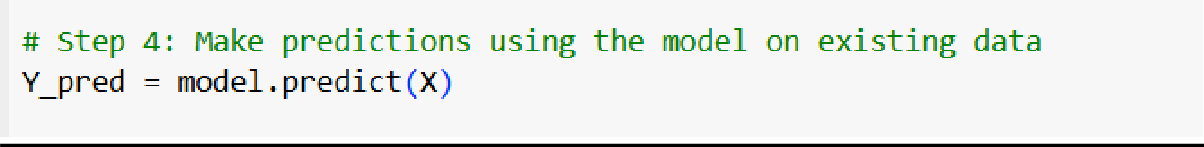
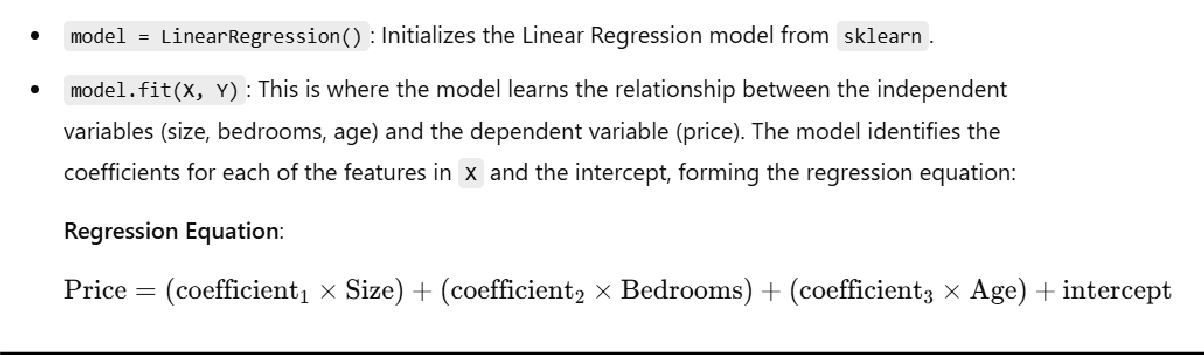
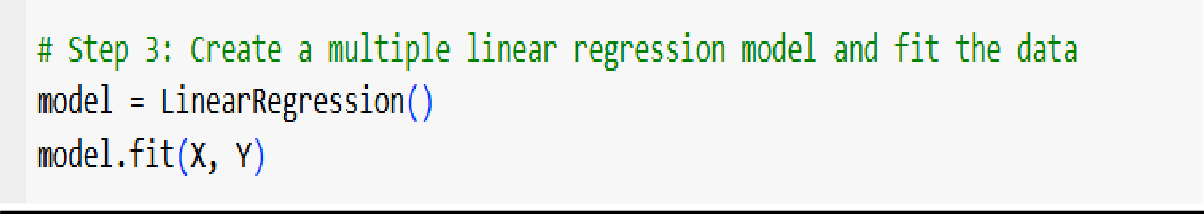
### Sample Dataset

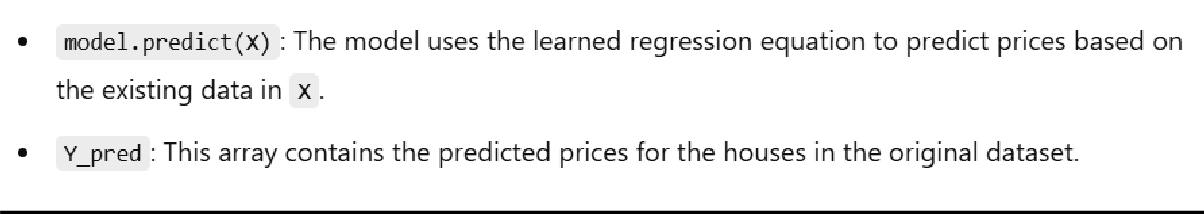
Let’s assume the following dataset with three independent variables:

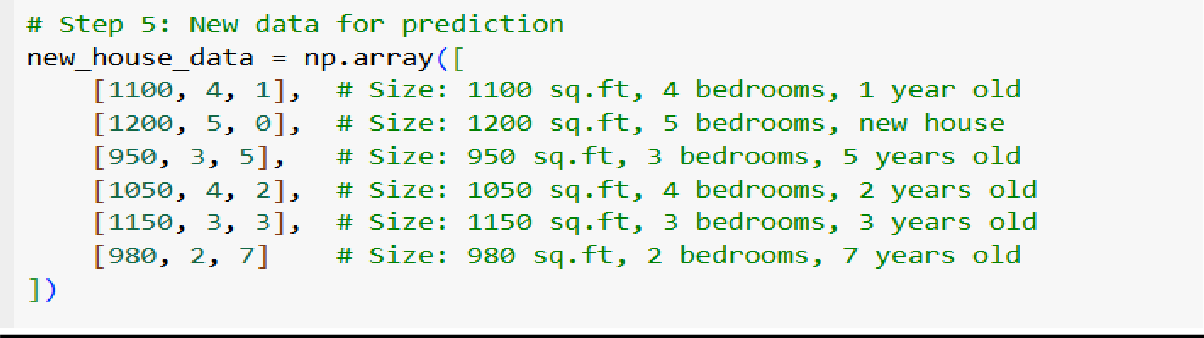


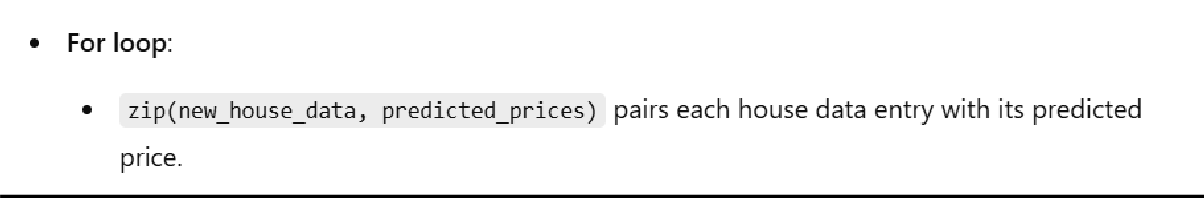
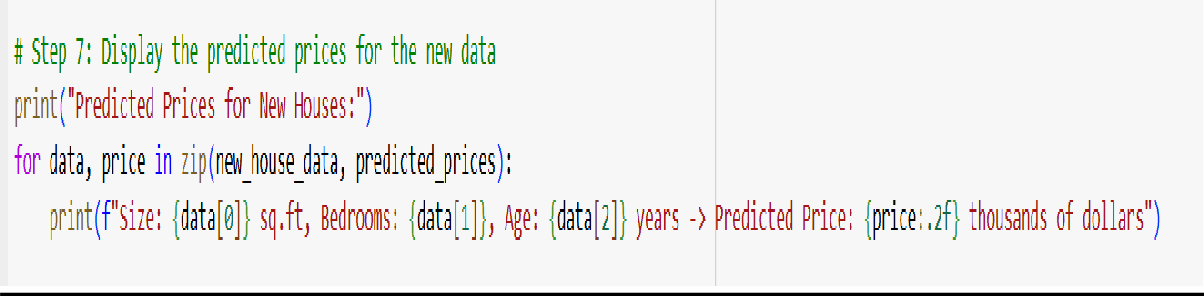
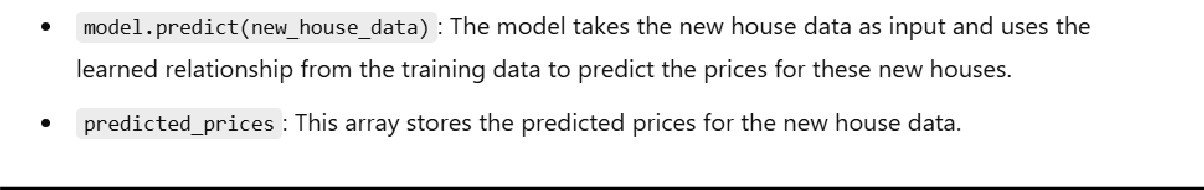
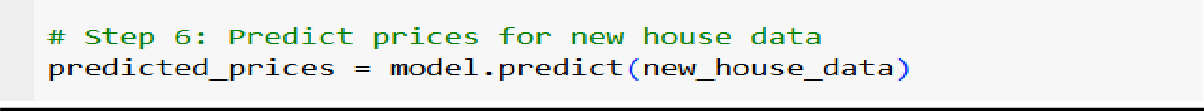
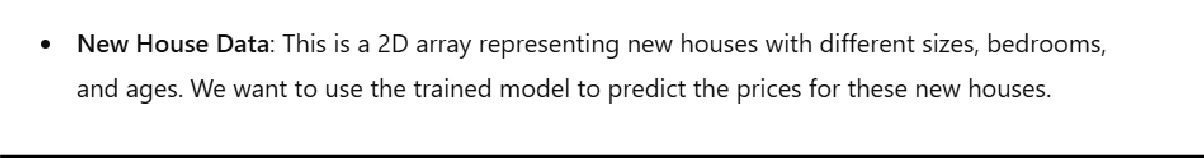


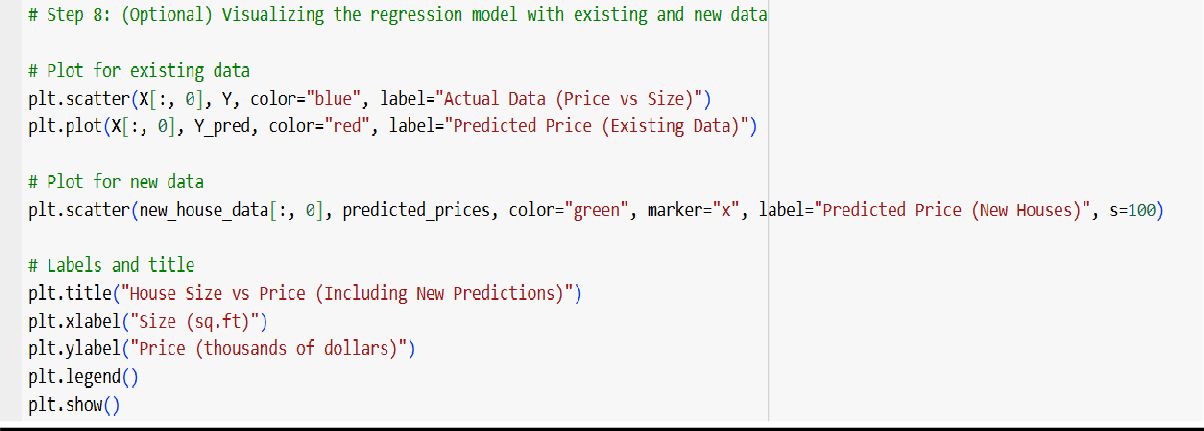


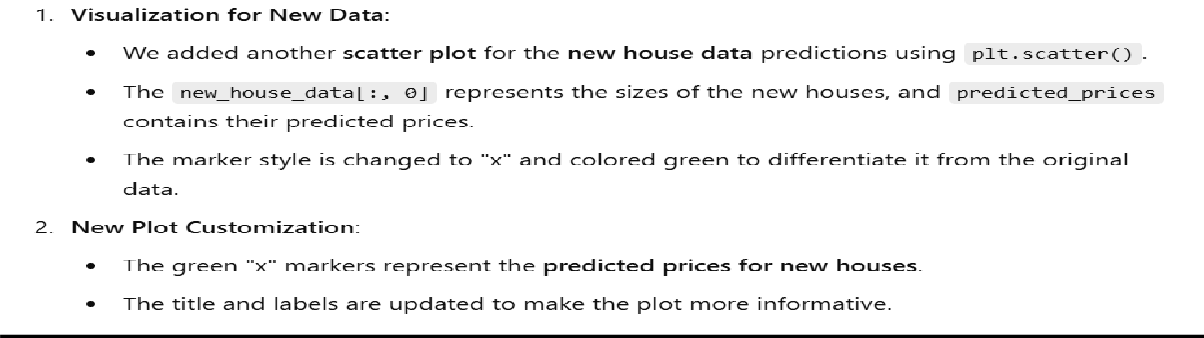








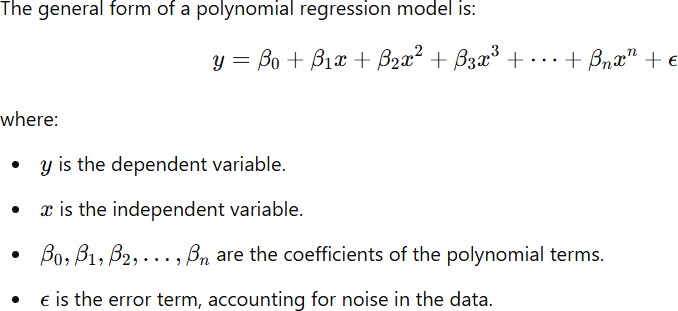




## Supervised Learning-Regression

### Polynomial Regression using sample dataset

Polynomial Regression is an extension of linear regression where the relationship between the independent and dependent variables is modeled as an n-degree polynomial. It's useful when the data shows a non-linear relationship, but we still want to model it with linear techniques by transforming the input features.



The degree of the polynomial n, determines the complexity of the model. For example:

* n=1: Linear regression (a straight line).
* n=2: Quadratic regression (a parabola).
* n=3: Cubic regression (a more flexible curve), and so on.

### Why Use Polynomial Regression?

Polynomial regression is useful when data shows a non-linear trend that cannot be captured by a simple linear model. Examples include:

* Modeling the growth of populations.
* Predicting stock prices with seasonal trends.
* Tracking temperature fluctuations over time.

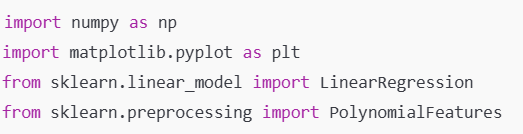
It provides a simple way to extend linear regression for capturing such non-linear patterns without resorting to more complex models like neural networks or non- parametric techniques.

### Steps for Polynomial Regression

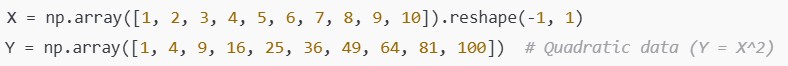
1. Import necessary libraries
2. Define the dataset
3. Transform the features into polynomial features
4. Fit the polynomial regression model
5. Predict new values
6. Visualize the result

### Step 1: Import necessary libraries

* + **numpy**: For handling array operations.
  + **matplotlib.pyplot**: For plotting the graphs.
  + **LinearRegression**: The linear regression model from sklearn.
  + **PolynomialFeatures**: This will be used to transform the input features into polynomial features.

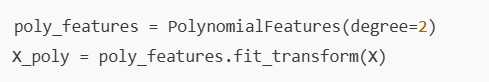


### Step 2: Define the dataset

* + **X**: The independent variable (input), representing numbers 1 through 10.
  + **Y**: The dependent variable (output), following a quadratic pattern (i.e., Y=X^2)

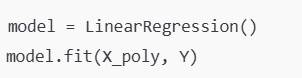
### Step 3: Transform the features into polynomial features

* + **PolynomialFeatures(degree=2)**: Transforms the input features into polynomial features of degree 2. For example, if X = [x1], it becomes X\_poly = [1, x1, x1^2].
  + **fit\_transform(X)**: Transforms the input array X into polynomial features.



### Step 4: Fit the polynomial regression model

* + **LinearRegression()**: Initializes the linear regression model.
  + **model.fit(X\_poly, Y)**: Trains the linear regression model using the polynomial features of X and the dependent variable Y.



### Step 5: Make predictions using the model on the training data

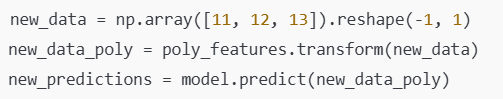
* + **model.predict(X\_poly)**: Uses the trained polynomial regression model to make predictions based on the original X values transformed into

polynomial features.



### Step 6: New data for prediction

* + **new\_data**: Represents new data points (11, 12, 13) that were not part of the original dataset.
  + **new\_data\_poly**: Transforms the new data into polynomial features.
  + **new\_predictions**: Predicts the Y values for the new data using the polynomial regression model.

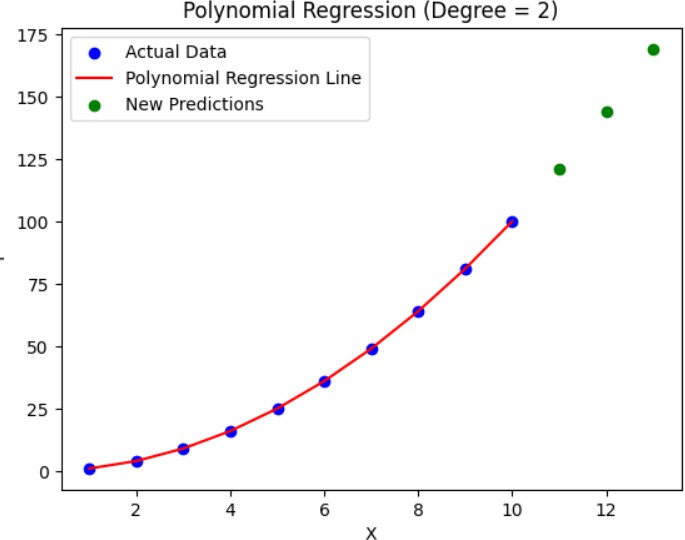


### Step 7: Display the predicted prices for new data

* + **zip(new\_data, new\_predictions)**: Loops through each pair of new data points and their corresponding predicted values.
  + **print()**: Displays the predictions for the new data.

### Step 8: Visualization

* + **plt.scatter(X, Y, color="blue")**: Plots the original data points.
  + **plt.plot(X, Y\_pred, color="red")**: Plots the regression line (polynomial curve) based on the predicted values.
  + **plt.scatter(new\_data, new\_predictions, color="green")**: Plots the predicted values for the new data points.
  + **plt.show()**: Displays the plot.



## Supervised Learning-Regression

### Metrics for Regression

Regression is a type of Machine learning algorithm which helps in finding the relationship between independent and dependent variables.

Before learning about precise metrics, let’s familiarize ourselves with a few essential concepts related to regression metrics:

1. **True Values and Predicted Values:**

In regression, we’ve got two units of values to compare: the actual target values (authentic values) and the values expected by the model (predicted values). The performance of the model is assessed by means of measuring the similarity among these sets.

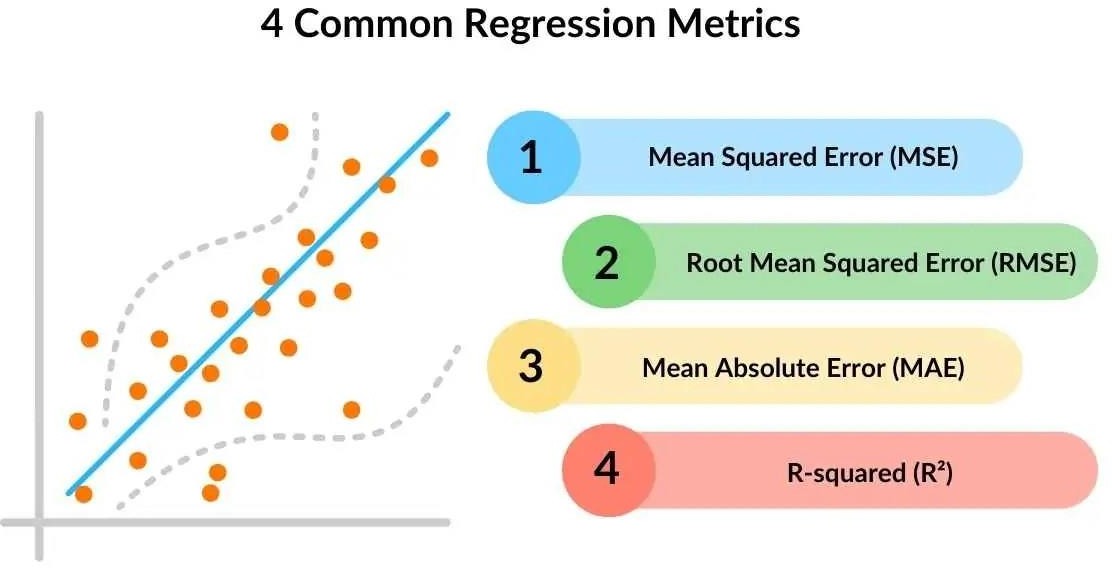
1. **Evaluation Metrics:**

Regression metrics are used to evaluate the performance of regression models by quantifying how well a model's predictions match actual values. Evaluation Metrics for regression are essential for assessing the performance of regression models specifically. These metrics help in measuring how well a regression model is able to predict continuous outcomes.

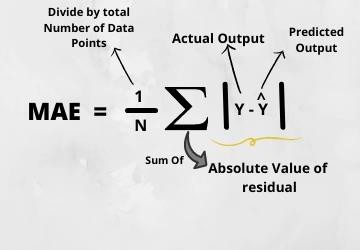
**Types of Regression Metrics**

Some common regression metrics in scikit-learn with examples

* + Mean Absolute Error (MAE)
  + Mean Squared Error (MSE)
  + Root Mean Squared Error (RMSE)
  + R-squared (R²) Score
  + Adjusted R-Squared

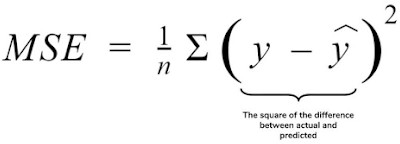


1. **Mean Absolute Error (MAE)**

****

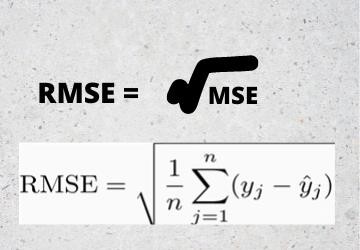
* + **Description**: MAE is the average of the absolute differences between the predicted and actual values. It measures how close predictions are to the actual outcomes.
  + **Interpretation**: Lower MAE values indicate better model performance, as it implies smaller errors on average.
  + **Sensitivity**: It is less sensitive to outliers than other metrics because it doesn’t square the errors. However, it may undervalue large errors.

1. **Mean Squared Error (MSE)**

****

* + **Description**: MSE is the average of the squared differences between actual and predicted values. Squaring the errors penalizes larger errors more than smaller ones.
  + **Interpretation**: Lower MSE values are preferred. It emphasizes larger errors, which can be useful if large errors are particularly undesirable in your application.
  + **Sensitivity**: Highly sensitive to outliers because errors are squared, causing models with high variance to have higher MSE values.

1. **Root Mean Squared Error (RMSE)**

****

* + **Description**: RMSE is the square root of MSE. It provides the error in the same units as the dependent variable y, making it more interpretable.
  + **Interpretation**: Lower RMSE indicates better model performance. It is especially useful when large errors should be heavily penalized.
  + **Sensitivity**: Like MSE, it is sensitive to outliers. However, RMSE is more interpretable than MSE because it is in the original units of the outcome variable.

1. **R-Squared (Coefficient of Determination)**

****

* + **Description**: R2 represents the proportion of variance in the dependent variable explained by the model. It ranges from 0 to 1.
  + **Interpretation**: Higher values (closer to 1) indicate that the model explains a large proportion of the variance. An R2 of 0 means the model does not explain any variance, while 1 indicates perfect prediction.
  + **Sensitivity**: Sensitive to data variability. While useful, it does not provide a measure of error magnitude, and its value can be misleading, especially with high-dimensional or overfitted models.

1. **Adjusted R-Squared Formula**:



* + **Description**: Adjusted R2 modifies R2 by adjusting for the number of predictors in the model. This avoids overestimating the model’s performance when additional predictors are added.
  + **Interpretation**: Higher adjusted R2 values suggest a better model fit while penalizing for unnecessary predictors.
  + **Sensitivity**: Adjusted R2 is particularly useful when comparing models with different numbers of predictors, as it accounts for added complexity.

**Choosing the Right Metric**

* + **Interpretability**: Metrics like RMSE and MAE provide errors in actual units, making them interpretable.
  + **Outliers**: If the data has outliers, MAE or MSLE may be preferable as they are less sensitive than MSE and RMSE.
  + **Percentage-Based Accuracy**: Use MAPE or SMAPE for models where percentage accuracy is meaningful and the dataset doesn’t contain zero or near-zero values.
  + **Model Comparison**: Adjusted R2R^2R2 and Explained Variance Score are valuable when comparing models, especially with different numbers of predictors.

Selecting the best metric will depend on your specific goals, data properties, and the application context.

### Supervised Learning-Regression

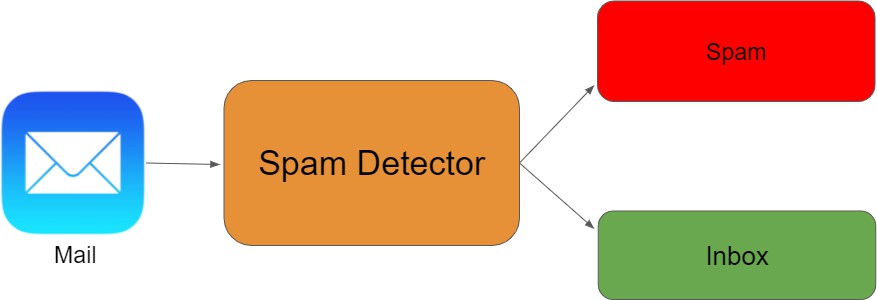
**Generalization, Overfitting and Underfitting**

Real-world data is inherently complex, encompassing variations, noise, and unpredictable factors. In the realm of machine learning and data science, the ultimate objective is to develop models capable of delivering accurate predictions and valuable insights when confronted with new and unseen data.

To achieve this objective, the concept of generalization plays a pivotal role. Generalization is a widely recognized technique in the world of machine learning and artificial intelligence

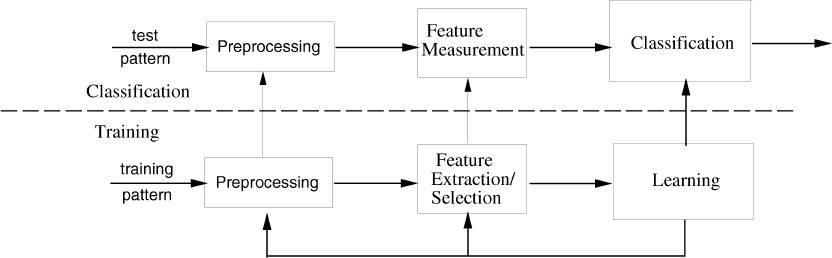
**Generalization**

* **Definition**: Generalization is the model's ability to perform well on new, unseen data after being trained on a given dataset. A well-generalized model learns patterns in the training data that are relevant to the larger data distribution, allowing it to make accurate predictions on data it hasn’t seen.
* **Importance**: Achieving good generalization is essential in machine learning because it enables models to handle real-world scenarios accurately.
* **Influence Factors**: Generalization is influenced by the complexity of the model, the amount and quality of data, and the regularization techniques used.
* A spam email classifier is a great example of generalization in machine learning. Suppose you have a training dataset containing emails labeled as either *spam* or *not spam* and your goal is to build a model that can accurately classify incoming emails as spam or legitimate based on their content.

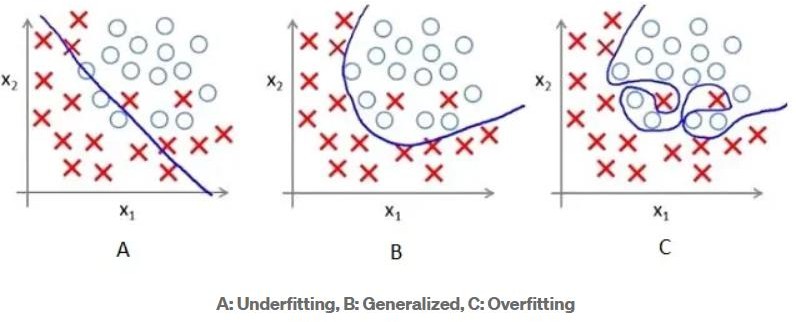


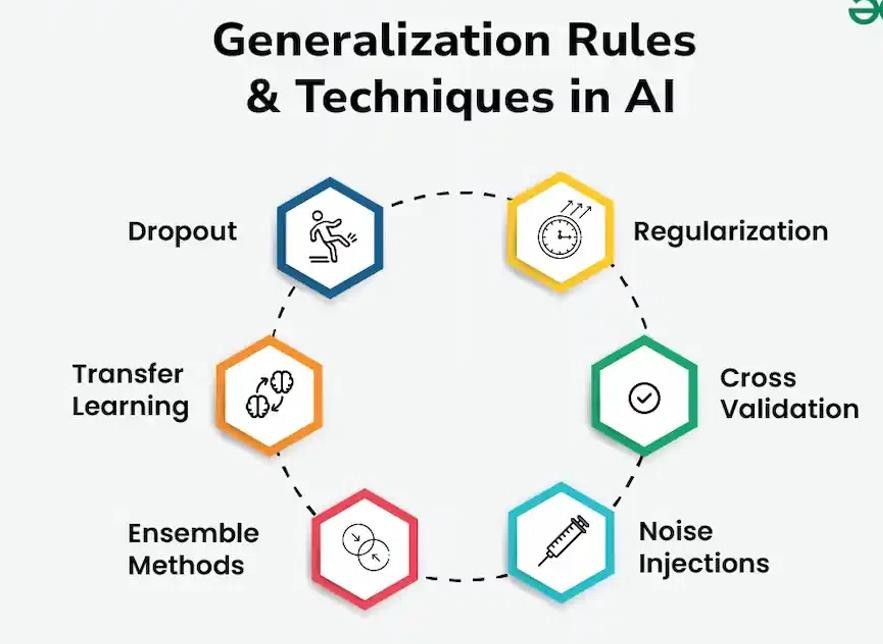
During the training phase, the machine learning algorithm learns from the set of labeled emails, extracting relevant features and patterns to make predictions. The model optimizes its parameters to minimize the training error and achieve high accuracy on the training data.

Now, the true test of the model's effectiveness lies in its ability to generalize to new, unseen emails. When new emails arrive, the model needs to accurately classify them as spam or legitimate without prior exposure to their content. This is where generalization comes in.



**Generalization** is a measure of how your model performs on predicting unseen data. So, it is important to come up with the best-generalized model to give better performance against future data. Let us first understand what is underfitting and overfitting, and then see what are the best practices to train a generalized model.





**Overfitting**

**Definition**: Overfitting occurs when a model that performs well on the training data but poorly on unseen data because it has memorized specifics rather than generalizing the patterns.

**Indicators**:

* High accuracy on training data but low accuracy on validation/test data.

**Causes**:

* Complex models with too many parameters (e.g., deep neural networks).
* Small training datasets with insufficient diversity.

**Solutions**:

* **Regularization**: Techniques like L1, L2 regularization penalize large weights, encouraging simpler models.
* **Pruning**: Removing redundant parts of the model architecture, particularly for decision trees or neural networks.
* **Cross-validation**: Using techniques like k-fold cross-validation to better evaluate model performance.
* **Early Stopping**: Ending the training process when validation error starts increasing.
* **Ensembling**: Combining multiple models

**Underfitting**

**Definition**: Underfitting occurs when a model is too simplistic to capture the underlying structure of the data. It fails to fit the training data and, as a result, also performs poorly on new data.

* **Indicators**:
  + Low accuracy on both training and validation/test datasets.
* **Causes**:
  + Models that are too simple (e.g., linear regression for non-linear data).
  + Insufficient training time.
  + Lack of relevant features in the data.

**Solutions**:

* **Model Complexity**: Use more complex models that can capture patterns (e.g., neural networks, ensemble methods).
* **Feature Engineering**: Add or transform features to make patterns more accessible.
* **Increasing Training Time**: Train the model longer to allow it to learn complex patterns.
* **Hyperparameter Tuning**: Adjust parameters to allow for a more flexible fit to the data.
* **Increasing Data Quality/Quantity**: More or better-quality data can help the model learn more about the true data distribution.

