* What is MACHINE LEARNING

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Types of machine learning

1. Supervised Machine Learning
2. Unsupervised Machine Learning

*Supervised learning

Supervised machine learning or more commonly, supervised learning, refers to algorithms that learn x to y or input to output mappings.
The key characteristic of supervised learning is that you give your learning algorithm examples to learn from.

*Unsupervised learning

uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention.
Clustering is a data mining technique which groups unlabeled data based on their similarities or differences.

Regression —

Linear regression is used for finding linear relationship between target and one or more predictors. There are two types of linear regression- Simple and Multiple.

Types:

1. **Simple Regression:**
   - Uses y = mx + b (traditional slope-intercept form).
   - m, b are learned variables; x is input data; y is prediction.
2. **Multivariable Regression:**
   - More complex, involving multiple variables.
   - Uses f(x, y, z) = w1x + w2y + w3z for prediction.
   - w1, w2, w3 are coefficients learned by the model.

*Simple Linear Regression

Simple linear regression explores the statistical relationship between two continuous variables: a predictor (independent) and a response (dependent) variable. It aims to find the best-fitting line that minimizes prediction errors for all data points. Unlike deterministic relationships, it doesn't provide precise predictions but identifies a statistical association.

Simple Regression Example:

- Dataset with Radio advertising spend and Sales.
- Developing an equation to predict Sales based on Radio spend.
- Prediction: Sales = Weight * Radio + Bias.

*Terminologies

Cost Function (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- Measures average squared difference between actual and predicted values.
- Goal: Minimize MSE for accurate predictions.

Gradient Descent:

- Minimizes MSE by calculating the gradient of the cost function.
- Updates weights (coefficients) in the direction opposite the gradient.

Vectorized Gradient Descent:

- Optimizing gradient descent by using matrix multiplication for all features and weights simultaneously.

Bias Term:

- Adding a bias term to account for base cases (e.g., non-zero sales without advertising).

-

Y(pred) = b0 + b1*x

The values b0 and b1 must be chosen so that they minimize the error. If sum of squared error is taken as a metric to evaluate the model, then goal to obtain a line that best reduces the error.

$$\text{Error} = \sum_{i=1}^{n}(actual\_output - predicted\_output) ** 2$$

If we don’t square the error, then positive and negative point will cancel out each other.

For model with one predictor,

$$b_0 = \bar{y} - b_1\bar{x}$$

$$b_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Below is the python implementation of the equation.

```
def theta_calc(x_train, y_train):
    #Initializing all variables
    n_data = x_train.shape[0]
    bias = np.ones((n_data,1))
    x_train_b = np.append(bias, x_train, axis=1)
    #
    theta_1 = np.linalg.inv(np.dot(x_train_b.T,x_train_b))
    theta_2 = np.dot(theta_1, x_train_b.T)
    theta = np.dot(theta_2,y_train)
    #
    return theta
```

***Residual Analysis**

Randomness and unpredictability are the two main components

of a regression model.
Prediction = Deterministic + Statistic

The deterministic aspect is captured by the predictor variable in the model, while the stochastic part reflects the unpredictability between expected and observed values. Residuals represent the missed information in the model and are obtained by comparing predicted and actual values. For instance, in a sales prediction model based on temperature, the difference between predicted and actual sales constitutes the residual.

Residual plots aid in model analysis by visualizing these differences, with standardized values indicating the quality of predictions. Positive residuals signify underestimation, negative residuals imply overestimation, and a zero value denotes a perfect prediction. Identifying patterns in residuals can enhance the model's performance.Non-random pattern of the residual plot indicates that the model is,

- Missing a variable which has significant contribution to the model target
- Missing to capture non-linearity (using polynomial term)
- No interaction between terms in model

Characteristics of a residue

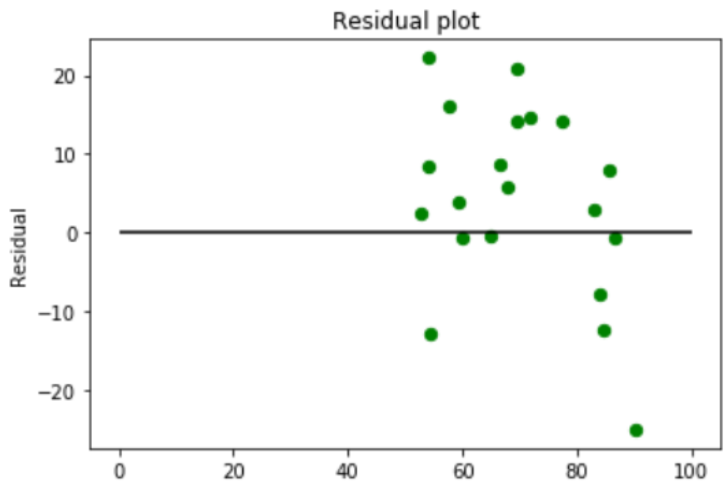
- Residuals do not exhibit any pattern

- Adjacent residuals should not be same as they indicate that there is some information missed by system.

Residual implementation and plot

```
#Residual plot
plt.scatter(prediction, prediction - y_test, c='g', s = 40)
plt.hlines(y=0, xmin=0, xmax=100)
plt.title('Residual plot')
plt.ylabel('Residual')
```

```
<matplotlib.text.Text at 0x173fafd4ac8>
```



*_**Metrics for model evaluation**_

***R-Squared Value:**

- Ranges from 0 to 1'1' indicates the predictor perfectly explains variation in Y.
- '0' indicates the predictor 'x' explains no variation in 'y'.

***Components:**

1. **Regression Sum of Squares (SSR):**
    - Measures how far the estimated regression line deviates from the horizontal 'no relationship' line (average of actual output).
2. **Sum of Squared Error (SSE):**

    - Quantifies how much the target value varies around the regression line (predicted value).
3. **Total Sum of Squares (SSTO):**
    - Represents how much the data points move around the mean.

***Interpretation:**

- Python implementation is essential for practical application.
-

***Correlation Coefficient (r):**

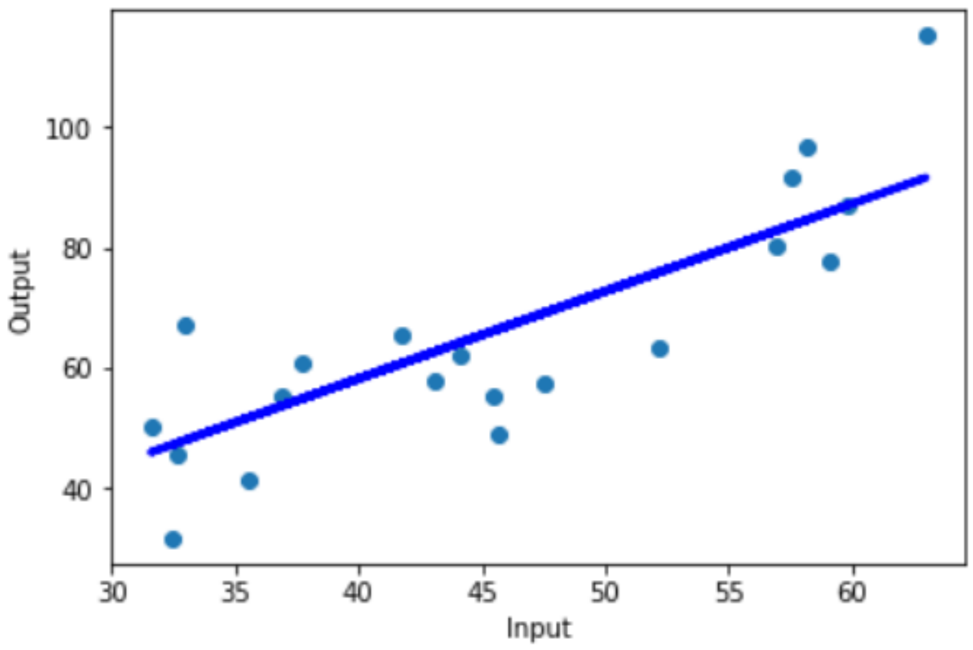
- Related to R-Squared.
- Ranges from -1 to 1.
- Unitless.

****Null-Hypothesis and P-value-***Null hypothesis is the initial claim that researcher specify using previous research or knowledge.

Low P-value: Rejects null hypothesis indicating that the predictor value is related to the response

High P-value: Changes in predictor are not associated with change in target

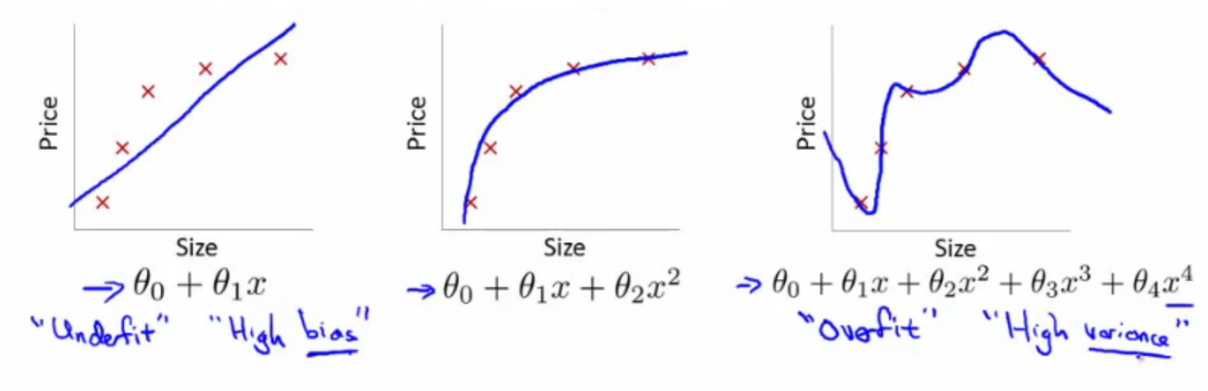
Obtained Regression line



*Overfitting with linear regression

- Using our house pricing example
    - Fit a linear function to the data - not a great model
        - This is **underfitting** - also known as **high bias**
        - Bias is a historic/technical one - if we're fitting a straight line to the data we have a strong preconception that there should be a linear fit
            - In this case, this is not correct, but a straight line can't help being straight!
    - Fit a quadratic function
        - Works well
    - Fit a 4th order polynomial
        - Now curve fit's through all five examples
            - Seems to do a good job fitting the training set
            - But, despite fitting the data we've provided very well, this is actually not such a good model
        - This is **overfitting** - also known as **high variance**
    - Algorithm has high variance

- High variance - if fitting high order polynomial then the hypothesis can basically fit any data
- Space of hypothesis is too large



*Gradient descent variants

There are three variants of gradient descent, which differ in how much data we use to compute the gradient of the objective function.

*Batch gradient descent

θ=θ-η.∇J(θ)

Batch gradient descent can be very slow and is intractable for datasets that don't fit in memory

batch gradient descent looks something like this-

```
for i in range(nb_epochs):
params_grad = evaluate_gradient(loss_function,
data, params)
params = params - learning_rate * params_grad
```

For a pre-defined number of epochs, we first compute the gradient vector **params_grad** of the loss function for the whole dataset w.r.t. our parameter vector **params**.

*Stochastic gradient descent

Stochastic gradient descent (SGD) in contrast performs a parameter update for each training example
x(I)and label y(I):

$$\theta=\theta-\eta\cdot \nabla\theta J(\theta;x(i);y(i)).$$

-Batch gradient descent: Redundant computations for large datasets, slower.

*Logistic Regression

Logistic regression, also known as the logit model, is a statistical model commonly used for classification and predictive analytics.

- it estimates the probability of an event occurring based on independent variables.
- The dependent variable is bounded between 0 and 1, representing probabilities.
- The logistic function transforms the odds of success to log odds
- The logistic regression equation relates the log odds to the independent variables through coefficients estimated via maximum likelihood estimation (MLE).

*Types of logistic regression

Binomial Logistic Regression:
Binary outcome: Pass/Fail, Yes/No, 0/1.
Multinomial Logistic Regression:
Multiple unordered categories: Cat/Dog/Sheep, Red/Blue/Green.
Ordinal Logistic Regression:
Multiple ordered categories: Low/Medium/High, Small/Medium/Large

*Sigmoid function

Now we use the sigmoid function where the input will be z and we find the probability between 0 and 1. i.e. predicted y.

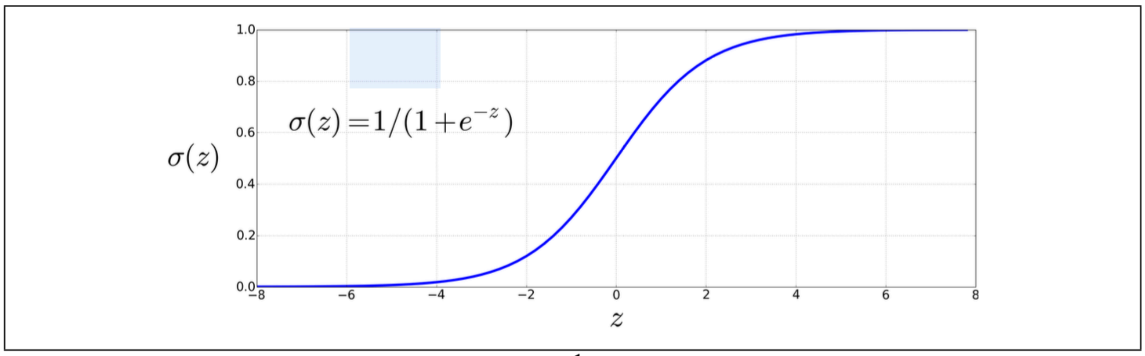


Figure 5.1 The sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ takes a real value and maps it to the range $(0,1)$. It is nearly linear around 0 but outlier values get squashed toward 0 or 1.

Logit: The input to the sigmoid function, denoted as 'z', is often referred to as the logit. The logit function is the inverse of the sigmoid and is represented as:

logit(p) = ln(p / (1 - p))

It takes a probability 'p' and returns the corresponding log odds.

Interpretation of Logit: Using the term "logit" for 'z' reminds us that when applying the sigmoid function, we are interpreting 'z' not just as any real-valued number but specifically as the log odds.

Probability Calculation: To calculate the probability of an event (e.g., y = 1 or y = 0), the sigmoid function is applied to the sum of the weighted features (z = Xw + b).

Relationship between Probabilities: The probabilities of y = 1 and y = 0 should sum up to 1. This is achieved by using the properties of the sigmoid function, such as 1 - σ(x) = σ(-x).

*why Cost function cannot be used for logistic regression

Output Compatibility: MSE is designed for continuous outputs, while logistic regression predicts probabilities between 0 and 1, making MSE incompatible.

Assumption of Linearity: Logistic regression involves modeling the log-odds of the outcome, resulting in a non-linear relationship with predictors, contrary to the linear assumption of MSE.

*classification with logistic regression

Decision Boundary: After calculating the probability
P(y=1∣x)
P(y=1∣x) using the sigmoid function, a decision is made whether to assign the class "1" (positive sentiment) or "0" (negative sentiment) based on whether the probability exceeds 0.5 (the decision boundary).

Sentiment Classification
Suppose we are doing binary sentiment classification on movie review text, and we would like to know whether to assign the sentiment class + or – to a review document doc.