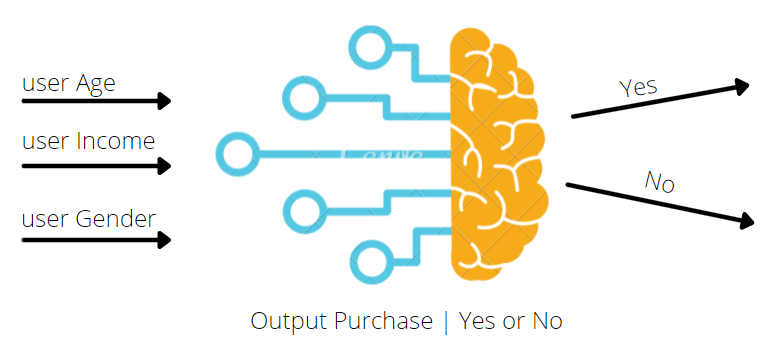
Logistic Regression

**Introduction**

**Logistic regression** is a **supervised machine learning algorithm**used for **classification tasks** where the goal is to predict the probability that an instance belongs to a given class or not. For example, age, income, and gender are three independent variables that can be used to determine whether a person will make a purchase or not.



**When to use logistic regression?**

Logistic regression is used when we need to solve a binary classification problem, when the relationship between the independent variables and the log-odds of the dependent variable is linear, and when model interpretability and probability estimates are important.

Example use cases:

**1) Medical diagnosis:** Predicting whether a patient has a particular disease (e.g., cancer, diabetes) based on clinical and demographic variables.

**2) Email classification:** Determining whether an email is spam or not based on the content of the email.

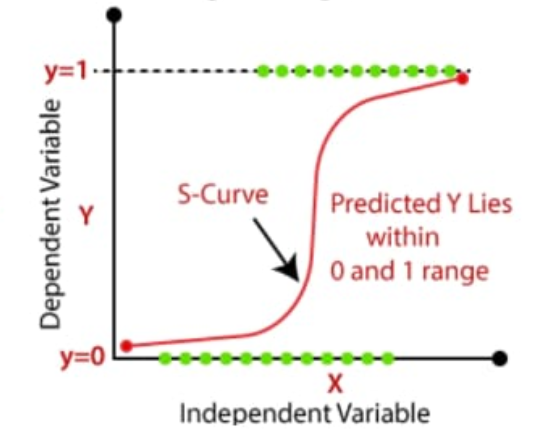
**3) Marketing:** Predicting whether a customer will buy a product based on their age, income, and past purchase behaviour.

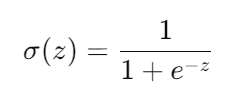
**4) Political election outcomes:** Estimating the probability that a candidate will win an election based on polling data, demographic information, and historical voting patterns.

**5) Admissions decisions:** Predicting whether a student will be admitted to a college based on their GPA, standardized test scores, extracurricular activities, and recommendation letters.

**Sigmoid function**

The sigmoid function is an S-shaped curve that can take any real valued number and map into a value between the range of 0 to 1, but never exactly as those limits.



The sigmoid function 𝜎(𝑧) is defined as: 

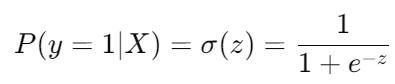
How it works in logistic regression?

1) First, a linear combination of the input features is computed. This is often represented as:

*z*=*c*​+*m*1​*x*1​+*m*2​*x*2​+…+*mn*​*xn*

where c is the intercept and m1,m2,….mn are the coeffecients associated with each feature x1, x2,….xn.

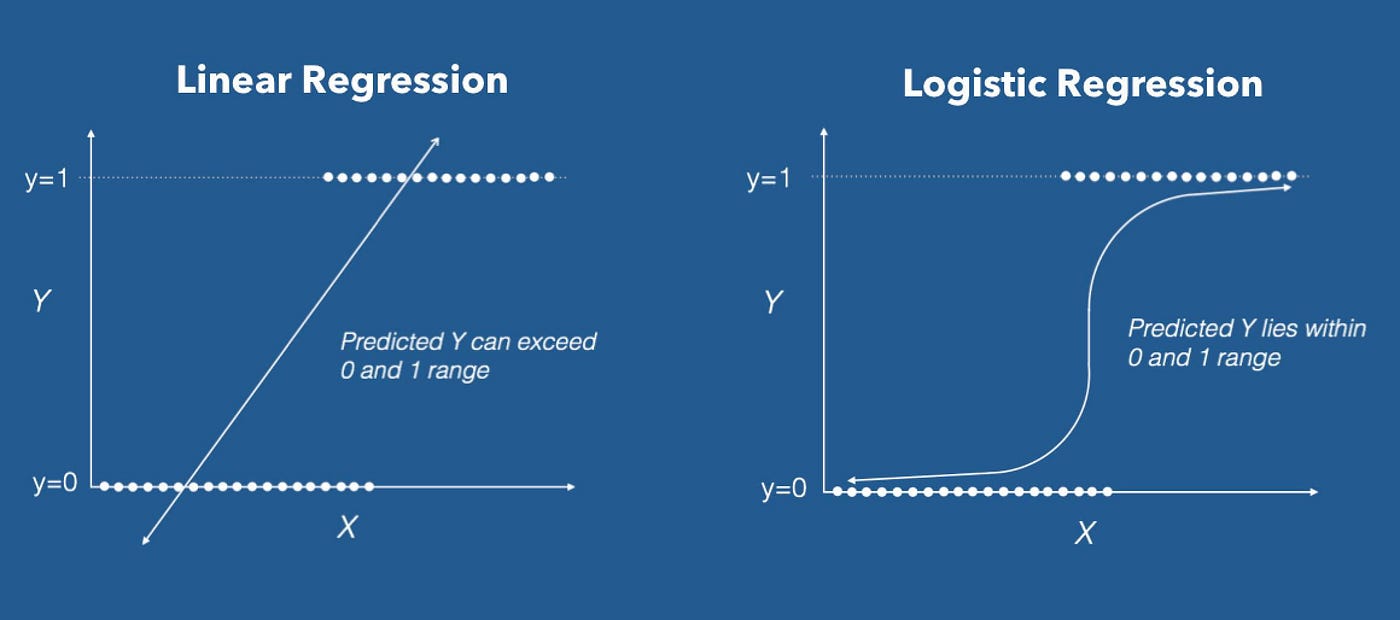
2) The sigmoid function is then applied to the linear combination *z* to produce a probability:



This probability 𝑃(𝑦=1∣X) represents the likelihood that the given input X belongs to the positive class (e.g., spam, disease present).

**Linear regression vs Logistic regression**

|  |  |  |
| --- | --- | --- |
| Parameter | Linear Regression | Logistic Regression |
| **Outcome variable type** | Continuous variable  Eg: Price, Temperature | Categorical variable (binary)  Eg: 0/1, Yes/No, T/F |
| **Model Purpose** | Regression (predicting numerical values) | Classification (categorizing into discrete classes) |
| **Equation/Function** | Linear equation:  Y = β0 + β1X + ε | Logistic (Sigmoid) function: p(X) = 1 / (1 + e^-(β0 + β1X)) |
| **Estimation Method** | Ordinary Least Squares (OLS) | Maximum Likelihood Estimation (MLE) |
| **Application Scope** | Suitable for forecasting, effect analysis of independent variables | Ideal for binary classification in various fields |



The choice between logistic regression and linear regression hinges on the nature of the problem.

**Example: Email classification**

**Problem statement:**   
The problem is to classify whether a given email is spam or not based on its content and characteristics.

The dataset contains 2 columns v1 and v2. v1 represents whether the email is spam or ham and v2 represents the content of the email. Here, v2 is the independent variable and v1 is the dependent variable. So, now the problem is to classify whether an email is spam or ham.

Python implementation:

**Step-1:** In order to implement the logistic regression expressions in python, we need to import all the necessary libraries using the keyword *import*.

#importing libraries

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

from sklearn.compose import ColumnTransformer

import seaborn as sns

import matplotlib.pyplot as plt

**Step-2:** The dataset is imported using UTF-8 encoding, a common standard for text files. If a Unicode Decode Error occurs, the except block retries with latin1 encoding (ISO-8859-1), which supports a different character set. After importing, first 5 rows are printed.

#importing dataset

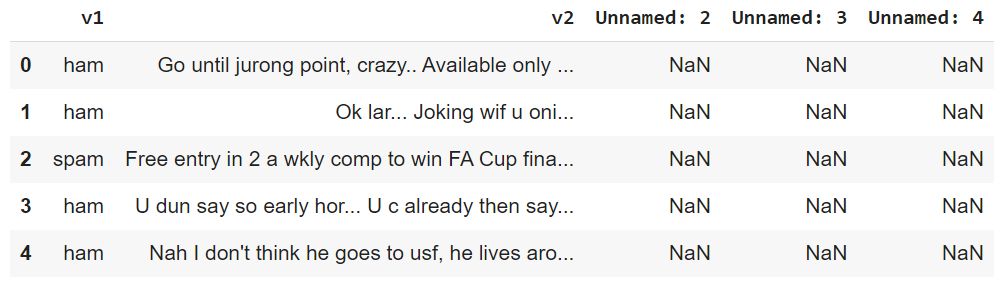
try:

    df = pd.read\_csv('/content/spam.csv', encoding='utf-8')

except UnicodeDecodeError:

    df = pd.read\_csv('/content/spam.csv', encoding='latin1')

df.head()



#generating the confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(4,4))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Ham', 'Spam'], yticklabels=['Ham', 'Spam'])

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

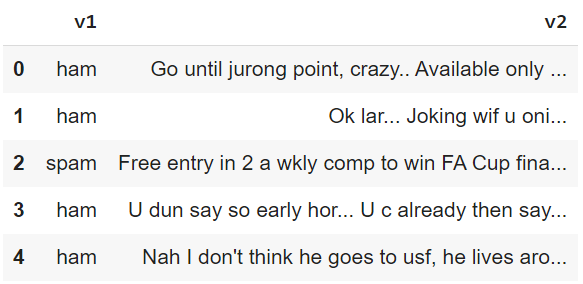
plt.show()

**Step-3:** Data preprocessing: In the dataset, Unnamed: 2, Unnamed: 3, Unnamed: 4 are the 3 unnecessary columns for our analysis. So, we can drop those 3 columns.

#data preprocessing

df = df.drop(columns=['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'])

df.head()

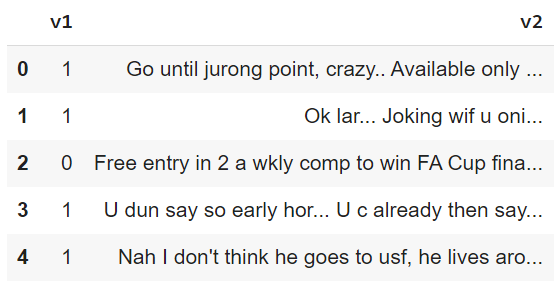


**Step-4**: Since the v1 column indicates the categorical data of whether an email is spam or ham, we encode the data for better analysis: spam is encoded as 0, and ham is encoded as 1 using map function.

#encoding spam as 0 and ham as 1

df['v1'] = df['v1'].map({'spam': 0, 'ham': 1})

df.head()



**Step-5:** Now the data is ready for applying linear regression. The features are copied into the variable 'X' and labels are copied into variable 'y'.

#extracting features and target

X=df['v2']

y=df['v1']

**Step-6:** The data set is divided into two parts training data and testing data. The training data is used to train the regression algorithm. While the testing data is used for validation of the model. In general for any machine learning algorithm, 20-30 data samples are used for training where as the remaining 70-80 data samples are used for testing. In the code test\_size=0.2 indicates 20% data is used for testing.

#splitting training and testing data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Step-7:** Vectorization is a step in feature extraction. The idea is to get some distinct features out of the text for the model to train on, by converting text to numerical vectors. Here, we use td-idf vectorizer that converts text documents to a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features.

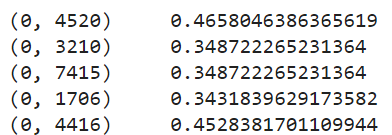
feature\_extraction = TfidfVectorizer(min\_df=1, stop\_words='english', lowercase=True)

# Fit and transform the training data, transform the test data

X\_train\_features = feature\_extraction.fit\_transform(X\_train)

X\_test\_features = feature\_extraction.transform(X\_test)

print(X\_train\_features)



**Step-8:** The logistic regression function is named as model. The regression algorithm is fitted to the training data using model.fit function on the training data.

model=LogisticRegression()

model.fit(X\_train\_features, y\_train)

**Step-9:** To evaluate the prediction accuracy accuracy\_score function is used which is imported from sklearn.metrics.

#accuracy on training data

prediction\_on\_training\_data=model.predict(X\_train\_features)

train\_accuracy=accuracy\_score(y\_train, prediction\_on\_training\_data)

print("Accuracy on training data: ", train\_accuracy)

Accuracy on training data: 0.9694862014808167

**Step-10:** Prediction accuracy on testing data

#accuracy on testing data

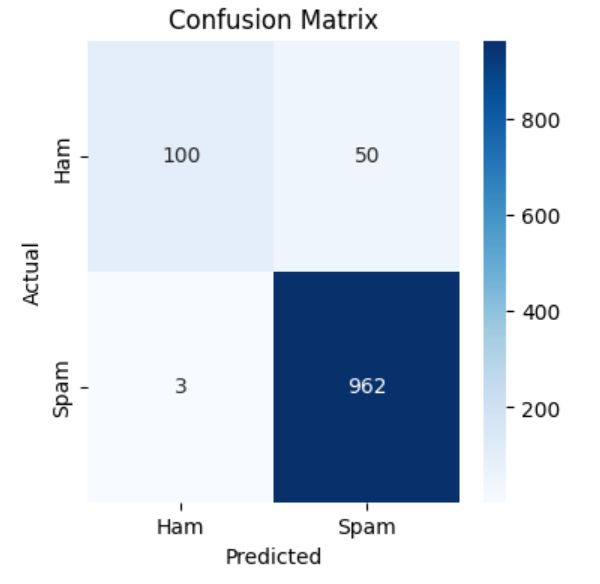
y\_pred=model.predict(X\_test\_features)

test\_accuracy=accuracy\_score(y\_test, y\_pred)

print("Accuracy on test data: ", test\_accuracy)

Accuracy on test data: 0.9524663677130045

**Step-11:** The confusion matrix function is imported from sklearn.metrics. It evaluates classification model performance by displaying true positives, true negatives, false positives, and false negatives. This helps in calculating metrics like accuracy, precision, and recall to diagnose and improve model performance.



**Step-12:** Checking the model

input\_mail=["You got a free offer of 10 lakh rupees. click on the link below..."]

input\_mail\_features=feature\_extraction.transform(input\_mail)

prediction=model.predict(input\_mail\_features)

print(prediction)

if(prediction[0]==1):

  print("Spam mail")

else:

  print("Not spam")

Output: [1]

Spam mail