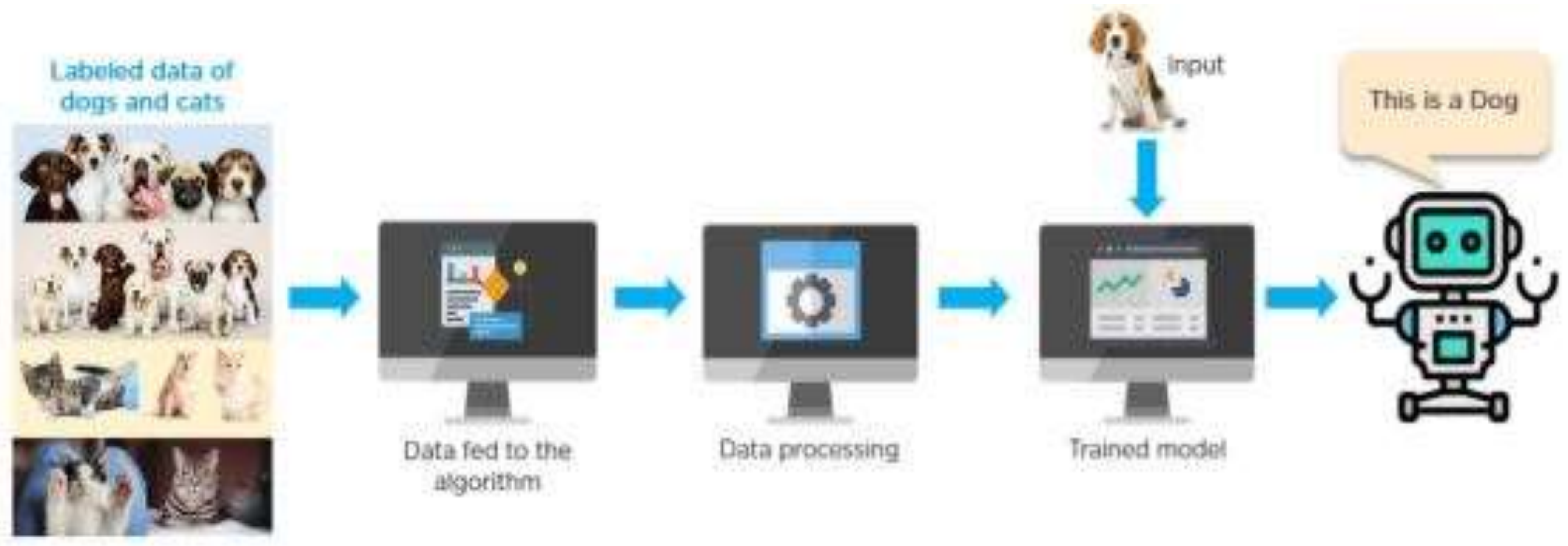
EC3357:Machine Learning

Lecture 4: Logistic Regression

Supervised learning

- Supervised machine learning algorithms derive insights, patterns, and relationships from a labeled training dataset.
- It means the dataset already contains a known value for the target variable for each record.
- Supervised learning problems can be further classified into regression and classification problems.
- Classification: In a classification problem, the output variable is a category, such as “red” or “blue,” “disease” or “no disease,” “true” or “false,” etc.
- Regression: In a regression problem, the output variable is a real continuous value, such as “dollars” or “weight.”

Example


Labeled data of dogs and cats
Data fed to the algorithm
Data processing
Input
Trained model
This is a Dog

The classification problem

- The linear regression model discussed in the previous lesson assumes that the response variable y is quantitative (metrical)
- In many situations, the response variable is instead qualitative (categorical)
- Qualitative variables take values in an unordered set $\mathcal{C}$ = "cat$_1$",…, "cat$_n$" , such as:
  - eye color { "brown", "blue", "green“}
  - email { "spam", "not spam“}

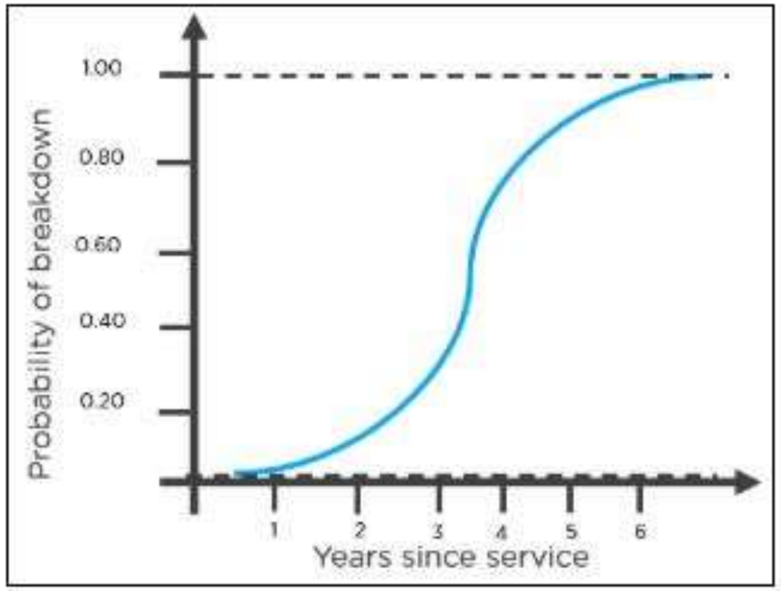
- Metric data
  - Describe a quantity
  - An ordering is defined
  - A distance is defined
- Categorical data
  - Describe membership categories
  - It is not meaningful to apply an ordering
  - It is not meaningful to compute distances

What is Logistic Regression?

- Logistic regression is a statistical method that is used for building machine learning models where the dependent variable is dichotomous: i.e. binary.
- Logistic regression is used to describe data and the relationship between one dependent variable and one or more independent variables.
- The independent variables can be nominal, ordinal, or of interval type.
- The name “logistic regression” is derived from the concept of the logistic function that it uses. The logistic function is also known as the sigmoid function. The value of this logistic function lies between zero and one.

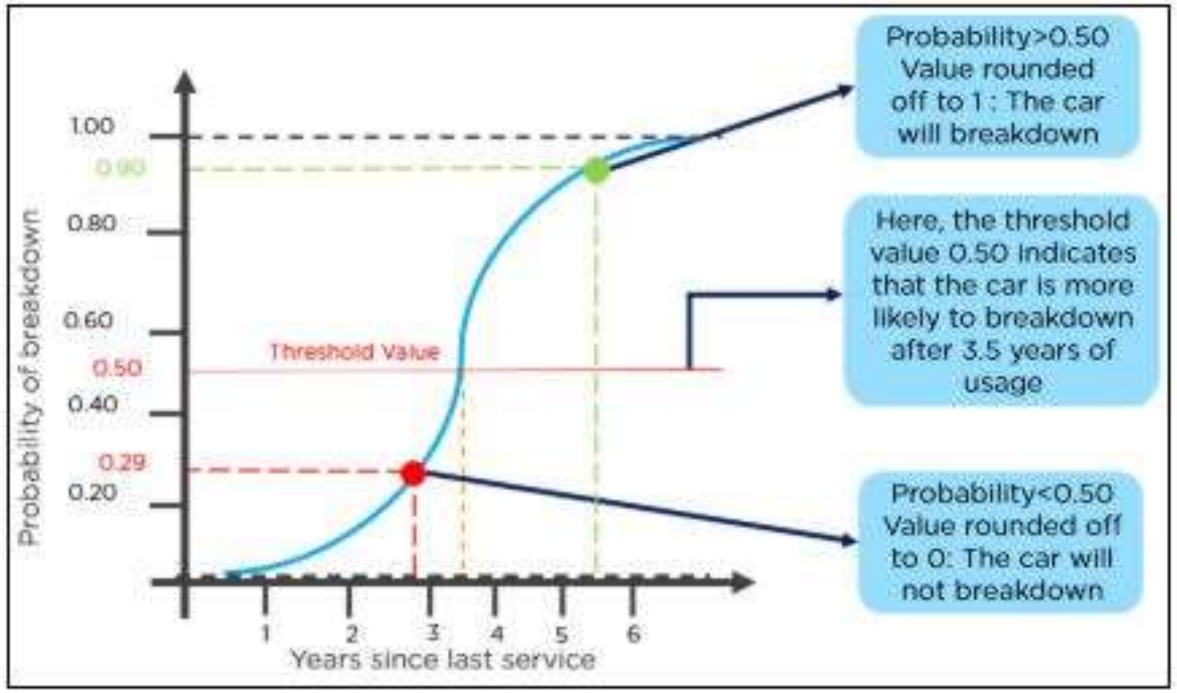
Example

- **The following is an example of a logistic function we can use to find the probability of a vehicle breaking down, depending on how many years it has been since it was serviced last.**



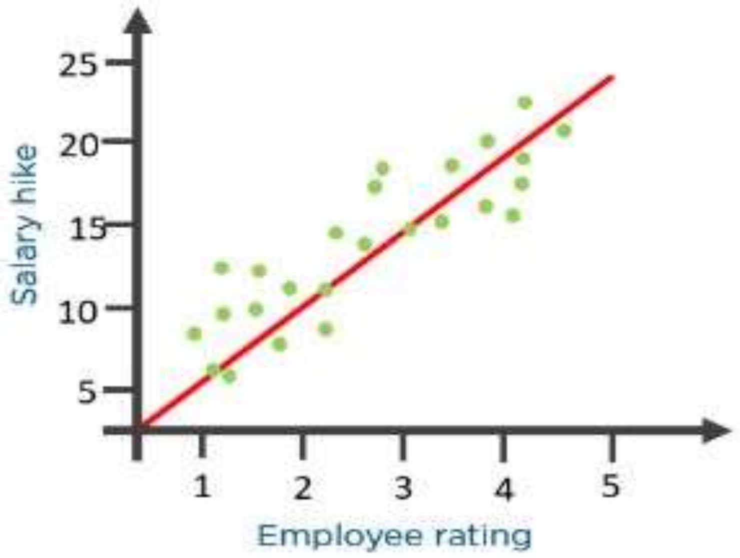
Example-Cont..

- **Here is how you can interpret the results from the graph to decide whether the vehicle will break down or not.**

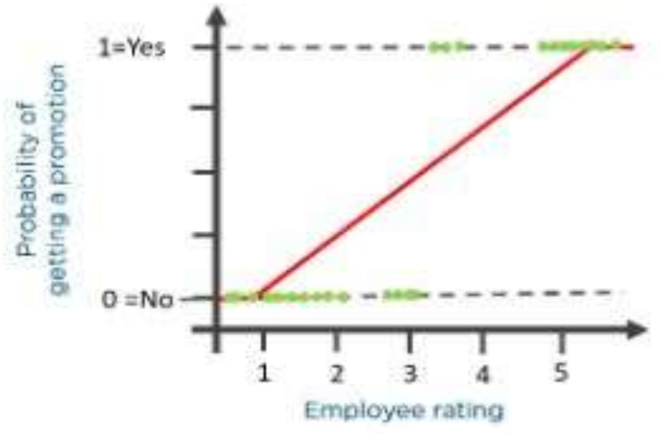


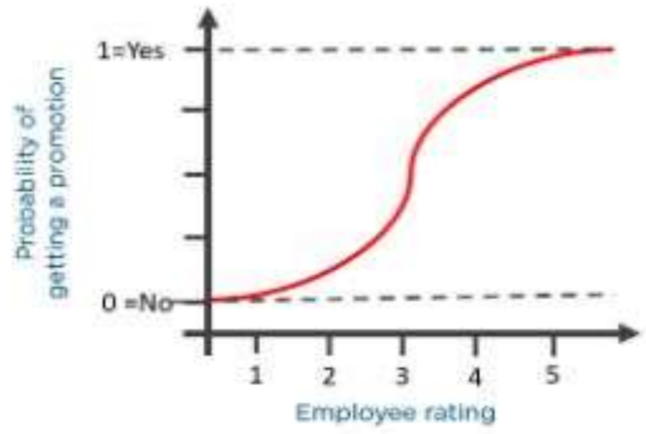
How Does the Logistic Regression Algorithm Work?

- **Consider the following example: An organization wants to determine an employee's salary increase based on their performance.**
- **For this purpose, a linear regression algorithm will help them decide. Plotting a regression line by considering the employee's performance as the independent variable, and the salary increase as the dependent variable will make their task easier.**



- Now, what if the organization wants to know whether an employee would get a promotion or not based on their performance? The above linear graph won't be suitable in this case. As such, we clip the line at zero and one, and convert it into a sigmoid curve (S curve).





- Based on the threshold values, the organization can decide whether an employee will get a salary increase or not.

Logistic Function (Sigmoid Function)

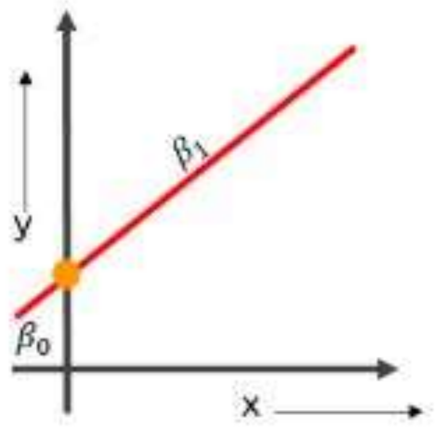
- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

sigmoid function

- To understand logistic regression, let’s go over the odds of success.
- Odds (θ) = Probability of an event happening / Probability of an event not happening
  - θ = p / 1 – p
- The values of odds range from zero to ∞ and the values of probability lies between zero and one.

- Consider the equation of a straight line:
  - y = β0 + β1* x

- Here, β0 is the y-intercept
- β1 is the slope of the line
- x is the value of the x coordinate
- y is the value of the prediction



- Now to predict the odds of success, we use the following formula:

$$\log\left(\frac{p(x)}{1-P(x)}\right) = \beta_0 + \beta_1 x$$

- Exponentiating both the sides, we have:

$$e^{ln}\left(\frac{p(x)}{1-p(x)}\right) = e^{\beta_0+\beta_1 x}$$

$$\left(\frac{p(x)}{1-p(x)}\right) = e^{\beta_0+\beta_1 x}$$

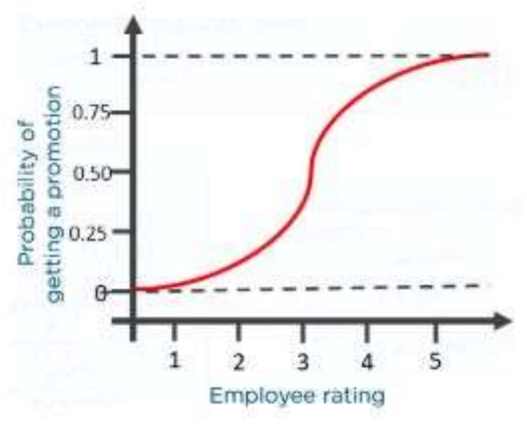
- Let Y = $e^{\beta 0+\beta 1 * x}$
- Then p(x) (/ 1 - p(x)) = Y
- p(x) = Y(1 - p(x))
- p(x) = Y - Y(p(x))
- p(x) + Y(p(x)) = Y
- p(x)(1+Y) = Y
- p(x) = Y / 1+Y

$$p(x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}$$

- The equation of the sigmoid function is:

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

- The sigmoid curve obtained from the above equation is as follows:



Types of Logistic Regression

- There are three main types of logistic regression: binary, multinomial and ordinal. They differ in execution and theory.
- Binary regression deals with two possible values, essentially: yes or no.
- Multinomial logistic regression deals with three or more values.
- And ordinal logistic regression deals with three or more classes in a predetermined order.

Binary logistic regression

- Binary logistic regression used for an either/or solution. There are just two possible outcome answers. This concept is typically represented as a 0 or a 1 in coding. Examples include:
  - Whether or not to lend to a bank customer (outcomes are yes or no).
  - Assessing cancer risk (outcomes are high or low).
  - Will a team win tomorrow’s game (outcomes are yes or no).

Multinomial logistic regression

- Multinomial logistic regression is a model where there are multiple classes that an item can be classified as. There is a set of three or more predefined classes set up prior to running the model. Examples include:
    - Classifying texts into what language they come from.
    - Predicting whether a student will go to college, trade school or into the workforce.
    - Does your cat prefer wet food, dry food or human food?

Ordinal logistic regression

- Ordinal logistic regression is also a model where there are multiple classes that an item can be classified as; however, in this case an ordering of classes is required. Classes do not need to be proportionate. The distance between each class can vary. Examples include:
    - Ranking restaurants on a scale of 0 to 5 stars.
    - Predicting the podium results of an Olympic event.
    - Assessing a choice of candidates, specifically in places that institute ranked-choice voting.

Linear Regression vs. Logistic Regression

Linear Regression	Logistic Regression
Used to solve regression problems	Used to solve classification problems
The response variables are continuous in nature	The response variable is categorical in nature
It helps estimate the dependent variable when there is a change in the independent variable	It helps to calculate the possibility of a particular event taking place
It is a straight line	It is an S-curve (S = Sigmoid)

Applications of Logistic Regression

- Using the logistic regression algorithm, banks can predict whether a customer would default on loans or not
- To predict the weather conditions of a certain place (sunny, windy, rainy, humid, etc.)
- Ecommerce companies can identify buyers if they are likely to purchase a certain product
- Companies can predict whether they will gain or lose money in the next quarter, year, or month based on their current performance
- To classify objects based on their features and attributes

Logistic Regression

- Logistic regression is used for classification, not regression!
- Logistic regression has some commonalities with linear regression, but you should think of it as classification, not regression!
- In many ways, logistic regression is a more advanced version of the perceptron classifier.
- The Perceptron is a linear machine learning algorithm for binary classification tasks.

Example

- In general, a binary logistic regression describes the relationship between the dependent binary variable and one or more independent variable/s.
- The binary dependent variable has two possible outcomes:

  - '1' for true/success; or

  - '0' for false/failure

- To start with a simple example, let's say that your goal is to build a logistic regression model in Python in order to determine whether candidates would get admitted to a prestigious university.

Step 1: Gather your data

- To start with a simple example, let’s say that your goal is to build a logistic regression model in Python in order to determine whether candidates would get admitted to a prestigious university.
- Here, there are two possible outcomes: Admitted (represented by the value of ‘1’) vs. Rejected (represented by the value of ‘0’).
- The dependent variable represents whether a person gets admitted; and
- The 3 independent variables are the GMAT score, GPA and Years of work experience

Step 2: Import the needed Python packages

- pandas – used to create the DataFrame to capture the dataset in Python
- sklearn – used to build the logistic regression model in Python
- seaborn – used to create the Confusion Matrix
- matplotlib – used to display charts
- import all the packages as follows:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import seaborn as sn
import matplotlib.pyplot as plt
```

Step 3: Build a dataframe

```
import pandas as pd
candidates = {'gmat':
[780,750,690,710,680,730,690,720,740,690,610,690,710,680,770,610,580,650,540,590,620,600,55
0,550,570,670,660,580,650,660,640,620,660,660,680,650,670,580,590,690],
            'gpa':
[4,3.9,3.3,3.7,3.9,3.7,2.3,3.3,3.3,1.7,2.7,3.7,3.7,3.3,3.3,3,2.7,3.7,2.7,2.3,3.3,2,2.3,2.7,3,3.3,3.7,2.3,3.
7,3.3,3,2.7,4,3.3,3.3,2.3,2.7,3.3,1.7,3.7],
            'work_experience':
[3,4,3,5,4,6,1,4,5,1,3,5,6,4,3,1,4,6,2,3,2,1,4,1,2,6,4,2,6,5,1,2,4,6,5,1,2,1,4,5],
            'admitted': [1,1,0,1,0,1,0,1,1,0,0,1,1,0,1,0,0,1,0,0,1,0,0,0,0,1,1,0,1,1,0,0,1,1,1,0,0,0,0,1]
            }

df = pd.DataFrame(candidates,columns= ['gmat', 'gpa','work_experience','admitted'])
print (df)
```

	gmat	gpa	work_ experience	admitted
0	780	4.0	3	1
1	750	3.9	4	1
2	690	3.3	3	0
3	710	3.7	5	1
4	680	3.9	4	0
5	730	3.7	6	1
6	690	2.3	1	0
7	720	3.3	4	1
8	740	3.3	5	1
9	690	1.7	1	0
10	610	2.7	3	0
11	690	3.7	5	1
12	710	3.7	6	1
13	680	3.3	4	0
14	770	3.3	3	1
15	610	3.0	1	0
16	580	2.7	4	0
17	650	3.7	6	1
18	540	2.7	2	0
19	590	2.3	3	0
20	620	3.3	2	1
21	600	2.0	1	0
22	550	2.3	4	0
23	550	2.7	1	0
24	570	3.0	2	0
25	670	3.3	6	1
26	660	3.7	4	1
27	580	2.3	2	0
28	650	3.7	6	1
29	660	3.3	5	1
30	640	3.0	1	0
31	620	2.7	2	0
32	660	4.0	4	1
33	660	3.3	6	1
34	680	3.3	5	1
35	650	2.3	1	0
36	670	2.7	2	0
37	580	3.3	1	0
38	590	1.7	4	0
39	690	3.7	5	1

Step 4: Create the logistic regression in Python

- Set the independent variables (represented as X) and the dependent variable (represented as y):

  X = df[['gmat', 'gpa','work_experience’]]

  y = df['admitted']

- Apply train_test_split.
  - X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=0)
- Apply the logistic regression as follows:
  - logistic_regression= LogisticRegression()
  - logistic_regression.fit(X_train,y_train)
  - y_pred=logistic_regression.predict(X_test)

- To evaluate the model, use the code below to get the Confusion Matrix:

```
confusion_matrix = pd.crosstab(y_test, y_pred, rownames=['Actual’], colnames=['Predicted’])
sn.heatmap(confusion_matrix, annot=True)
```

- Print the Accuracy and plot the Confusion Matrix:

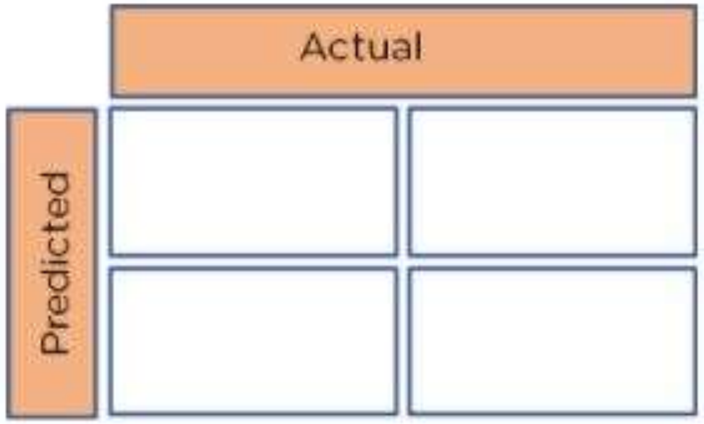
```
print('Accuracy: ',metrics.accuracy_score(y_test, y_pred))
plt.show()
```

Confusion Matrix in Machine Learning

- In machine learning, Classification is used to split data into categories.
- But after cleaning and preprocessing the data and training our model, how do we know if our classification model performs well?
- That is where a confusion matrix comes into the picture.
- A confusion matrix is used to measure the performance of a classifier in depth.
- A confusion matrix presents a table layout of the different outcomes of the prediction and results of a classification problem and helps visualize its outcomes.

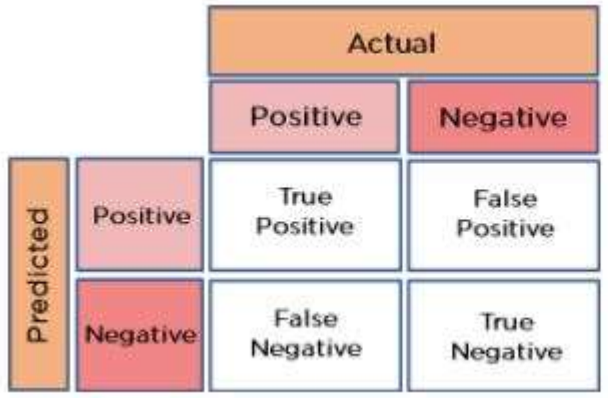
Basic layout of a Confusion Matrix

- **It plots a table of all the predicted and actual values of a classifier.**



Create a 2x2 Confusion Matrix

- **We can obtain four different combinations from the predicted and actual values of a classifier:**

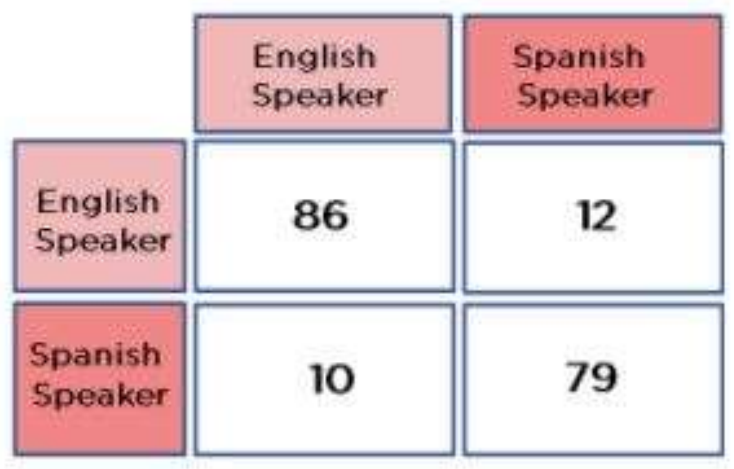


		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

- True Positive: The number of times our actual positive values are equal to the predicted positive. You predicted a positive value, and it is correct.
- False Positive: The number of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.
- True Negative: The number of times our actual negative values are equal to predicted negative values. You predicted a negative value, and it is actually negative.
- False Negative: The number of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.

Example

- Consider a confusion matrix made for a classifier that classifies people based on whether they speak English or Spanish.
- From the above diagram, we can see that:
  - True Positives (TP) = 86
  - True Negatives (TN) = 79
  - False Positives (FP) = 12
  - False Negatives (FN) = 10



	English Speaker	Spanish Speaker
English Speaker	86	12
Spanish Speaker	10	79

Evaluation Matrices

- Accuracy: The accuracy is used to find the portion of correctly classified values. It tells us how often our classifier is right. It is the sum of all true values divided by total values.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- In this case: Accuracy = (86 +79) / (86 + 79 + 12 + 10) = 0.8823 = 88.23%
- Phyton code for accuracy:

```
from sklearn.metrics import accuracy_score
y_true = [0, 1, 1, 0, 1, 0]
y_pred = [0, 1, 0, 0, 1, 1]
accuracy = accuracy_score(y_true, y_pred)
print("Accuracy: ", accuracy)
```

- Precision: Precision is used to calculate the model's ability to classify positive values correctly. It is the true positives divided by the total number of predicted positive values.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- In this case, Precision = 86 / (86 + 12) = 0.8775 = 87.75%
- Phyton code for Precision:

```
from sklearn.metrics import precision_score
y_true = [0, 1, 1, 0, 1, 0]
y_pred = [0, 1, 0, 0, 1, 1]
precision = precision_score(y_true, y_pred)
print("Precision: ", precision)
```

- Recall: It is used to calculate the model's ability to predict positive values. "How often does the model predict the correct positive values?". It is the true positives divided by the total number of actual positive values.

$$\boxed{\text{Recall} = \frac{TP}{TP + FN}}$$

- In this case, Recall = 86 / (86 + 10) = 0.8983 = 89.83%

- Phyton code:

```
from sklearn.metrics import recall_score
y_true = [0, 1, 1, 0, 1, 0]
y_pred = [0, 1, 0, 0, 1, 1]
recall = recall_score(y_true, y_pred)
print("Recall: ", recall)
```

- F1-Score: It is the harmonic mean of Recall and Precision. It is useful when you need to take both Precision and Recall into account.

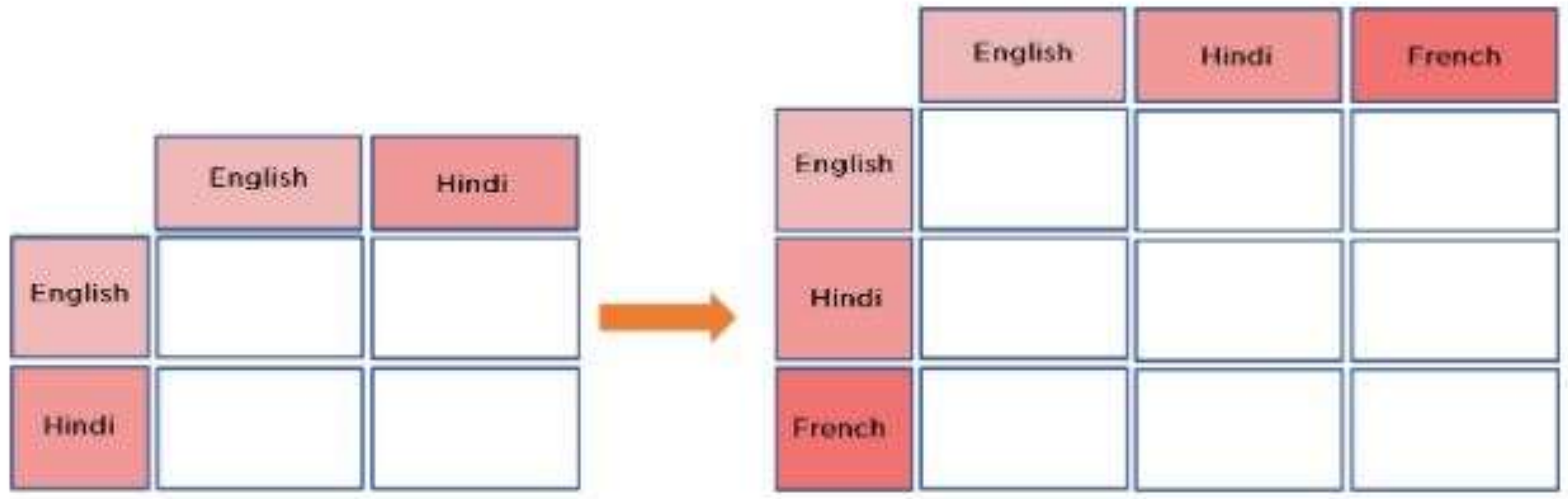
$$\boxed{\text{F1-Score} = \frac{2*\text{Precision}*\text{Recall}}{\text{Precision} + \text{Recall}}}$$

- In this case, F1-Score = (2* 0.8775 * 0.8983) / (0.8775 + 0.8983) = 0.8877 = 88.77%
- Phyton code:

```
from sklearn.metrics import f1_score
y_true = [0, 1, 1, 0, 1, 0]
y_pred = [0, 1, 0, 0, 1, 1]
f1 = f1_score(y_true, y_pred)
print("F1-Score: ", f1)
```

Scaling a Confusion Matrix

- To scale a confusion matrix, increase the number of rows and columns. All the True Positives will be along the diagonal. The other values will be False Positives or False Negatives.



Practical Example

- https://github.com/diandiaye/Institut-des-Algorithmes-du-S-n-gal/blob/main/tutorialsram/tuto17.md
- https://machinelearningmastery.com/multinomial-logistic-regression-with-python/