Roll No. 21EC	**Experiment No.** 03	**Marks :**
BATCH - C		**Sign :**

Aim: Implementation of Classification algorithm (Decision Tree/Naive Bayes)

Apparatus: Google Colab

Theory:

What Is Classification?

Classification is the process of recognizing, understanding, and grouping ideas and objects into preset categories or "sub-populations." Using pre-categorized training datasets, machine learning programs use a variety of algorithms to classify future datasets into categories.

Classification algorithms in machine learning use input training data to predict the likelihood that subsequent data will fall into one of the predetermined categories. One of the most common uses of classification is filtering emails into "spam" or "non-spam."

In short, classification is a form of "pattern recognition," with classification algorithms applied to the training data to find the same pattern (similar words or sentiments, number sequences, etc.) in future sets of data.

Using classification algorithms, which we'll go into more detail about below, text analysis software can perform tasks like aspect-based sentiment analysis to categorize unstructured text by topic and polarity of opinion (positive, negative, neutral, and beyond).

Top 5 Classification Algorithms in Machine Learning

The study of classification in statistics is vast, and there are several types of classification algorithms you can use depending on the dataset you're working with. Below are five of the most common algorithms in machine learning.

1. **Logistic Regression**
2. **Naive Bayes**
3. **K-Nearest Neighbors**
4. **Decision Tree**
5. **Support Vector Machines**

Logistic Regression

Logistic regression is a calculation used to predict a binary outcome: either something happens, or does not. This can be exhibited as Yes/No, Pass/Fail, Alive/Dead, etc.

Independent variables are analyzed to determine the binary outcome with the results falling into one of two categories. The independent variables can be categorical or numeric, but the dependent variable is always categorical. Written like this:

$P(Y=1|X)$ or $P(Y=0|X)$

It calculates the probability of dependent variable Y, given independent variable X.

This can be used to calculate the probability of a word having a positive or negative connotation (0, 1, or on a scale between). Or it can be used to determine the object contained in a photo (tree, flower, grass, etc.), with each object given a probability between 0 and 1.

Naive Bayes

Naive Bayes calculates the possibility of whether a data point belongs within a certain category or does not. In text analysis, it can be used to categorize words or phrases as belonging to a preset "tag" (classification) or not. For example:

Text	Tag
"A great game"	Sports
"The election was over"	Not sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not sports

To decide whether or not a phrase should be tagged as "sports," you need to calculate:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Or… the probability of A, if B is true, is equal to the probability of B, if A is true, times the probability of A being true, divided by the probability of B being true.

K-nearest Neighbors

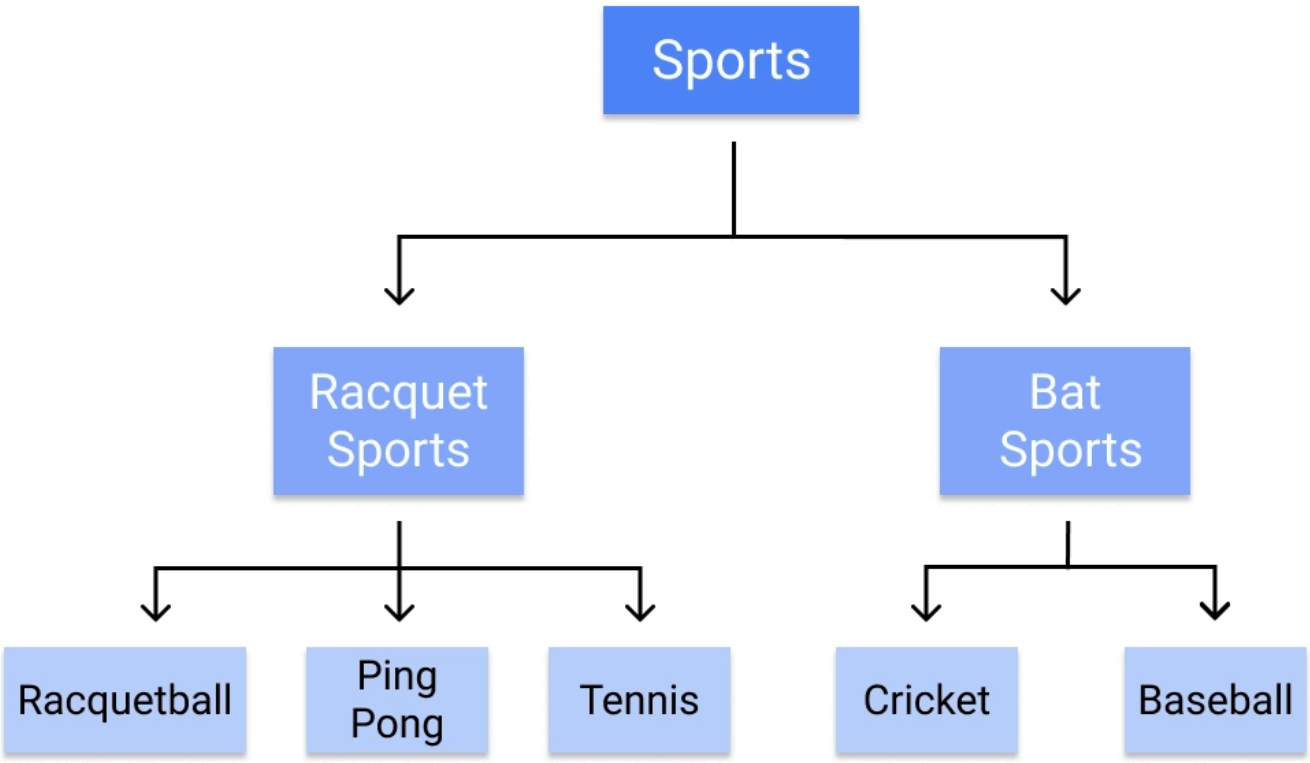
K-nearest neighbors (k-NN) is a pattern recognition algorithm that uses training datasets to find the k closest relatives in future examples.

When k-NN is used in classification, you calculate to place data within the category of its nearest neighbor. If k = 1, then it would be placed in the class nearest 1. K is classified by a plurality poll of its neighbors.

Decision Tree

A decision tree is a supervised learning algorithm that is perfect for classification problems, as it's able to order classes on a precise level. It works like a flow chart, separating data points into two similar categories at a time from the "tree trunk" to "branches," to "leaves," where the categories become more finitely similar. This creates categories within categories, allowing for organic classification with limited human supervision.

To continue with the sports example, this is how the decision tree works:



An example of a decision tree dividing different sports.

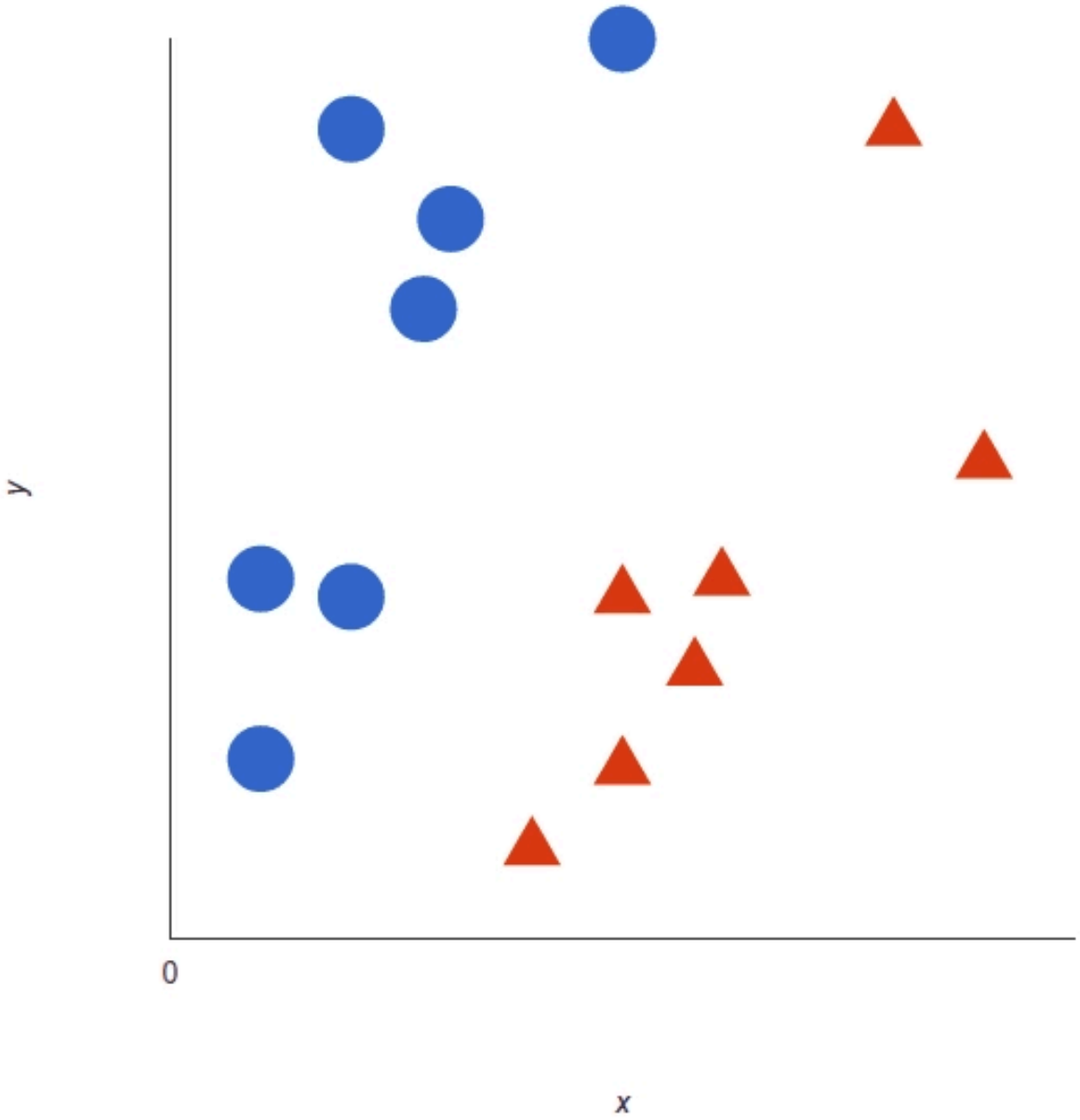
Random Forest

The random forest algorithm is an expansion of the decision tree, in that you first construct a multitude of decision trees with training data, then fit your new data within one of the trees as a "random forest." It, essentially, averages your data to connect it to the nearest tree on the data scale. Random forest models are helpful as they remedy for the decision tree's problem of "forcing" data points within a category unnecessarily.

Support Vector Machines

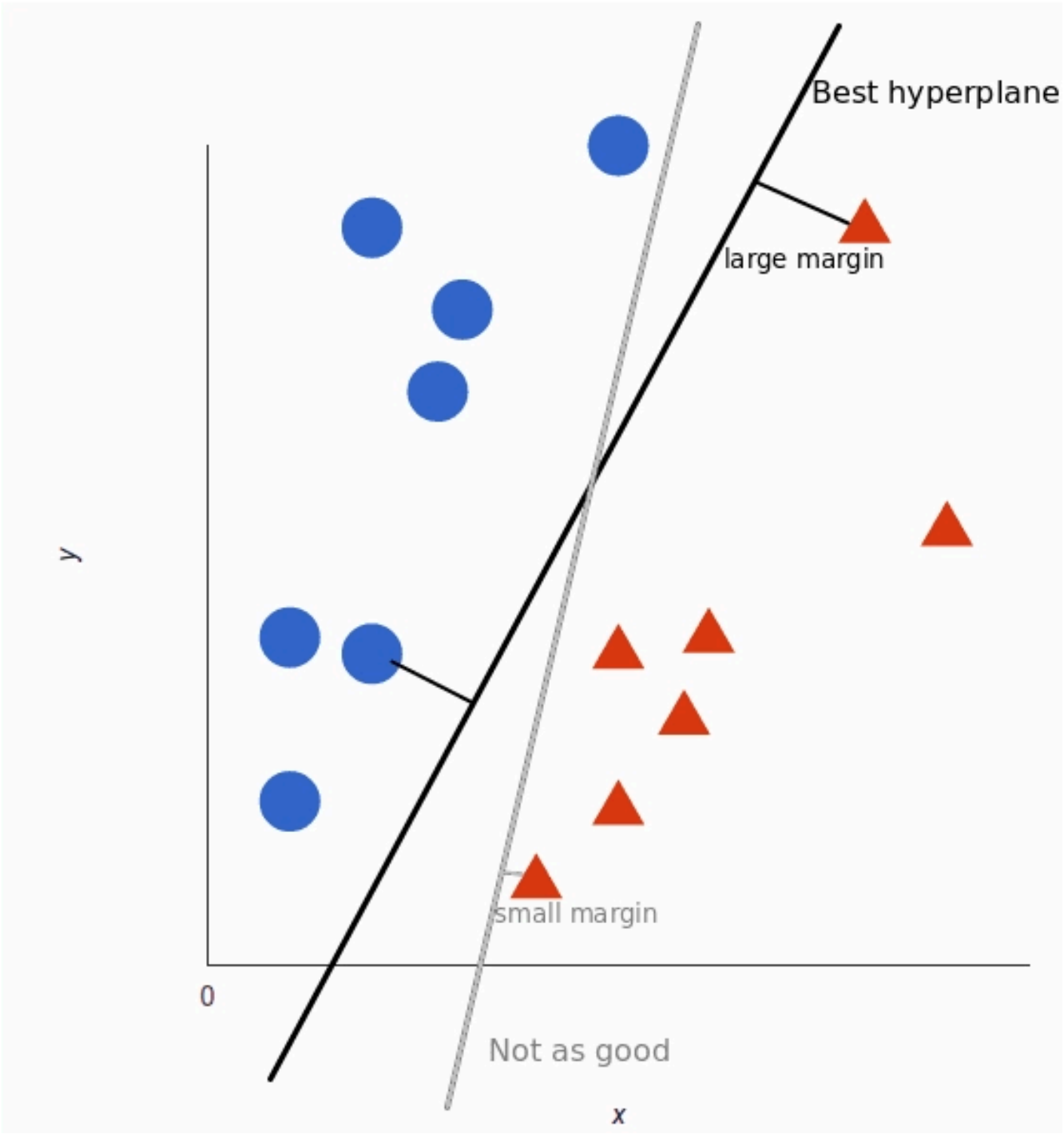
A support vector machine (SVM) uses algorithms to train and classify data within degrees of polarity, taking it to a degree beyond X/Y prediction.

For a simple visual explanation, we'll use two tags: red and blue, with two data features: X and Y, then train our classifier to output an X/Y coordinate as either red or blue.

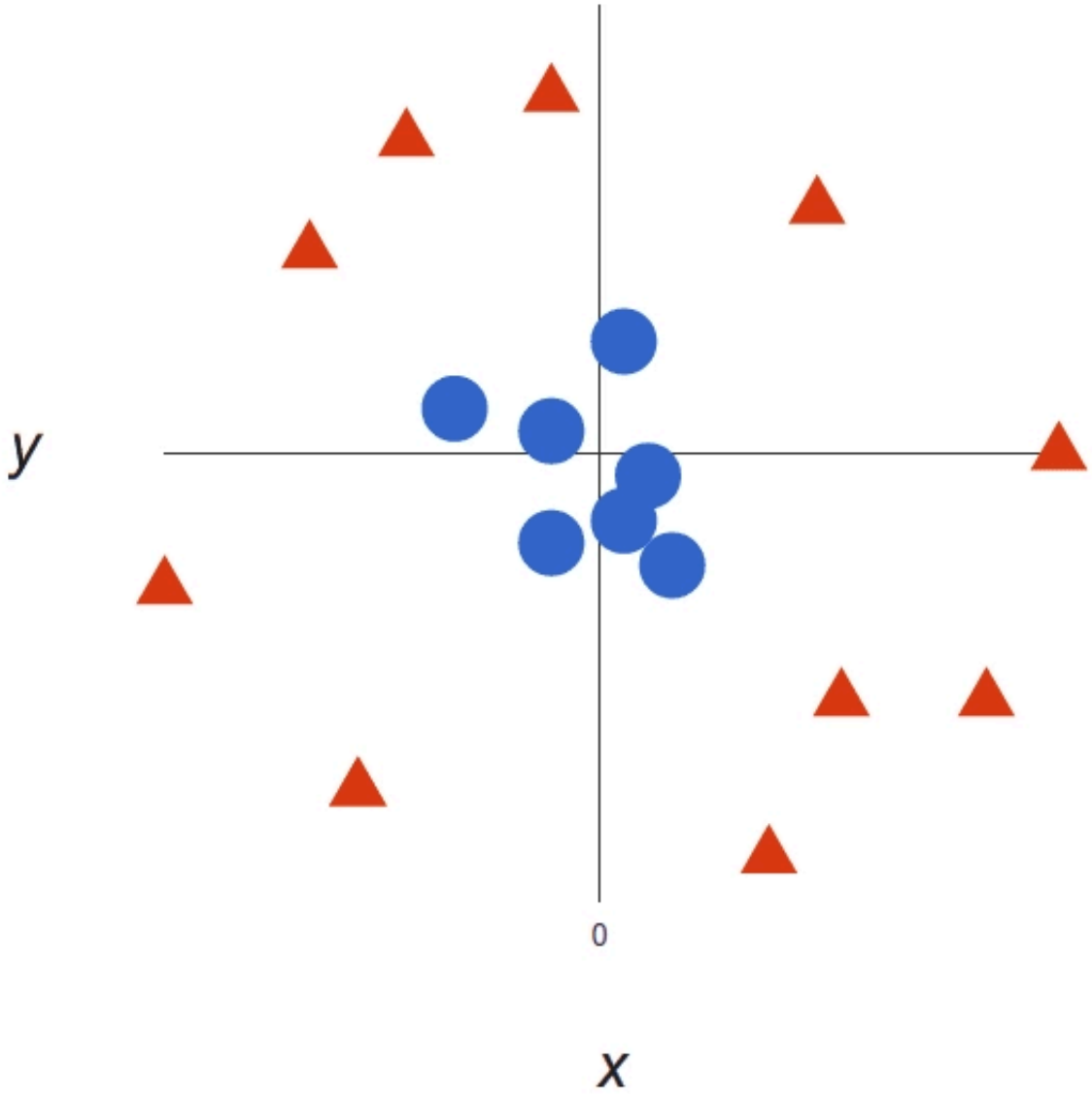


The SVM then assigns a hyperplane that best separates the tags. In two dimensions this is simply a line. Anything on one side of the line is red and anything on the other side is blue. In sentiment analysis, for example, this would be positive and negative.

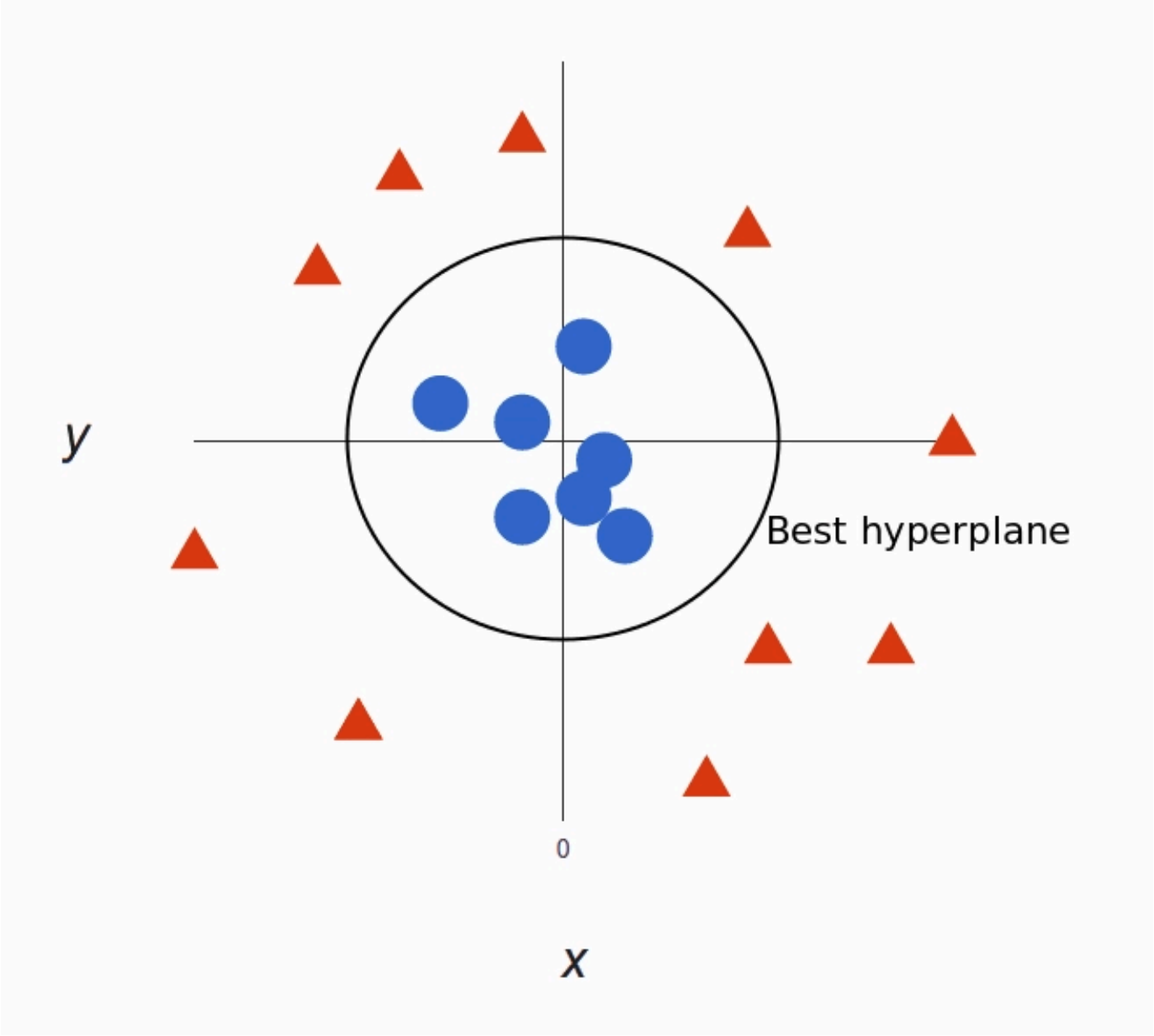
In order to maximize machine learning, the best hyperplane is the one with the largest distance between each tag:



However, as data sets become more complex, it may not be possible to draw a single line to classify the data into two camps:



Using SVM, the more complex the data, the more accurate the predictor will become. Imagine the above in three dimensions, with a Z-axis added, so it becomes a circle.
Mapped back to two dimensions with the best hyperplane, it looks like this:



SVM allows for more accurate machine learning because it's multidimensional.

Implementation:

1. **Naive bayes :**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder

# Load the CSV data into a DataFrame
data = pd.read_csv("/content/drive/MyDrive/DWM/bread basket.csv")
# Drop any rows with missing values
data.dropna(inplace=True)
# Convert date_time to datetime format
data['date_time'] = pd.to_datetime(data['date_time'])
data['hour_of_day'] = data['date_time'].dt.hour
# Encode categorical features using LabelEncoder
label_encoders = {}
for column in ['Item', 'period_day', 'weekday_weekend']:
    label_encoders[column] = LabelEncoder()
    data[column] = label_encoders[column].fit_transform(data[column])

X = data[['Item', 'hour_of_day', 'period_day', 'weekday_weekend']]
y = data['Transaction']
# Split the data 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 and train the Naive Bayes classifier
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
# Evaluate the classifier
accuracy = nb_classifier.score(X_test, y_test)
print("Accuracy:", accuracy)
```

OUTPUT:

```
Accuracy: 0.00048756704046806434
```

2. Decision Tree:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# 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 and train the Decision Tree classifier
dt_classifier = DecisionTreeClassifier()
dt_classifier.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = dt_classifier.predict(X_test)

# Evaluate model performance
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

OUTPUT:

```
Accuracy: 1.0
```

Conclusion:

Implementing Naive Bayes or Decision Tree classifiers in Google Colab offers a seamless environment for machine learning experimentation. With access to powerful hardware and pre-installed libraries like scikit-learn, developers can quickly prototype, train, and evaluate these classifiers on datasets, making rapid progress in model development and evaluation.