CS-37 Machine Learning with Python

Unit– 02 Supervised Learning

- Regression: Pre-processing data using different techniques –

    1. mean removal
    2. scaling
    3. Normalization
    4. Binarization
    5. label encoding
    6. linear regression case study implementation using Python

- Classification:

    1. Building simple classifier
    2. logistic regression classifier
    3. Naive bayes classifier

4. training and testing dataset
5. accuracy using cross validation
6. visualizing confusion matrix
7. extracting the performance report.

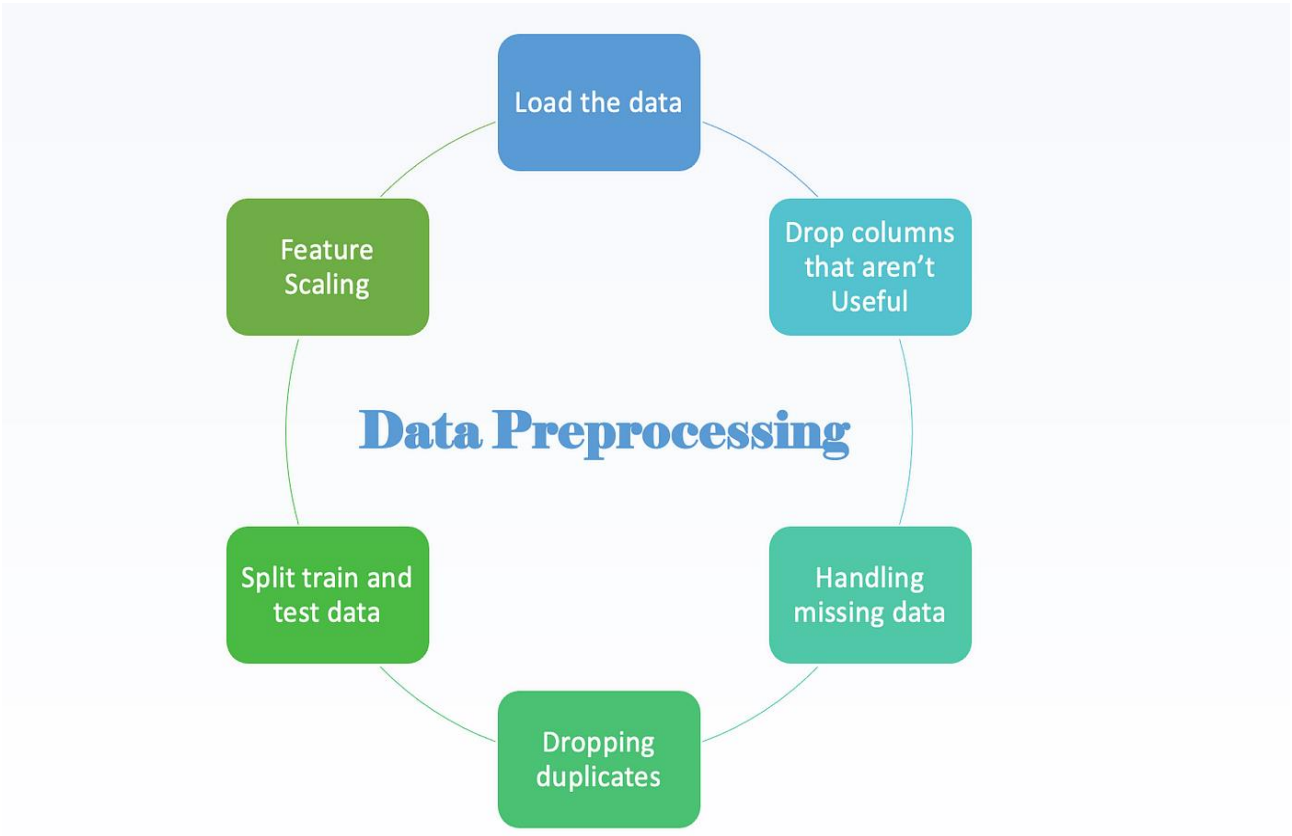
- Predictive Modeling:

1. Building linear and non-linear classifier using Support Vector Machine (SVM)
2. extracting confidence measurements
3. Case study implementation using Python.

Data Preprocessing

What is data preprocessing in machine learning?

- Data preprocessing is a important step in the data science **transforming raw data into a clean structured format for analysis.** It involves tasks like handling missing values, normalizing data and encoding variables. Mastering preprocessing in Python ensures reliable insights for accurate predictions and effective decision-making.



Steps in Data Preprocessing

Step 1: Import the necessary libraries

```
# importing libraries
import pandas as pd
import scipy
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import seaborn as sns
import matplotlib.pyplot as plt
```

Step 2: Load the dataset

You can download dataset from here.

```
# Load the dataset
df = pd.read_csv('Geeksforgeeks/Data/diabetes.csv')
print(df.head())
```

Output:

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

1. Check the data info

```
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

- As we can see from the above info that the our dataset has 9 columns and each columns has 768 values. There is no Null values in the dataset.

We can also check the null values using df.isnull()

```
df.isnull().sum()
```

Output:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Step 2: Statistical Analysis

In statistical analysis we use df.describe() which will give a descriptive overview of the dataset.

```
df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

- The above table shows the count, mean, standard deviation, min, 25%, 50%, 75% and max values for each column. When we carefully observe the table we will find that Insulin, Pregnancies, BMI, BloodPressure columns has outliers.

Preprocessing Techniques

add the following piece of code to this file –

```
import numpy as np

from sklearn import preprocessing

#We imported a couple of packages. Let's create some sample data and add the line to this file:

input_data = np.array([[3, -1.5, 3, -6.4], [0, 3, -1.3, 4.1], [1, 2.3, -2.9, -4.3]])
```

1. Mean removal :

- ✓ It involves removing the mean from each feature so that it is centered on zero. Mean removal helps in removing any bias from the features.
- ✓ You can use the following code for mean removal –

```
data_standardized = preprocessing.scale(input_data)
print ("\nMean = ", data_standardized.mean(axis = 0))
print ("Std deviation = ", data_standardized.std(axis = 0))
```

output :

```
Mean = [ 5.55111512e-17 -3.70074342e-17
0.00000000e+00 -1.85037171e-17]
Std deviation = [1. 1. 1. 1.]
```

2. Scaling :

- ✓ The values of every feature in a data point can vary between random values. So, it is important to scale them so that this matches specified rules.
- ✓ You can use the following code for scaling −

```
data_scaler =
preprocessing.MinMaxScaler(feature_range = (0, 1))
data_scaled = data_scaler.fit_transform(input_data)
print ("\nMin max scaled data = ", data_scaled)
```

output :

```
Min max scaled data = [ [ 1. 0. 1. 0. ]
                        [ 0. 1. 0.27118644 1. ]
                        [ 0.33333333 0.84444444 0. 0.2 ]
]
```

3. Normalization :

- ✓ Normalization involves adjusting the values in the feature vector so as to measure them on a common scale. Here, the values of a feature vector are adjusted so that they sum up to 1
- ✓ You can use the following code for normalization

```
data_normalized =
preprocessing.normalize(input_data, norm  = 'l1')
print ("\nL1 normalized data = ", data_normalized)
```

4. Binarization :

- ✓ Binarization is used to convert a numerical feature vector into a Boolean vector.
- ✓ You can use the following code for binarization −

```
data_binarized =
preprocessing.Binarizer(threshold=1.4).transform(inp
ut_data)
print ("\nBinarized data =", data_binarized)
```

output :

```
Binarized data = [[ 1. 0. 1. 0.]
                  [ 0. 1. 0. 1.]
                  [ 0. 1. 0. 0.]
                 ]
```

5. Label Encoding :

- ✓ In supervised learning, we mostly come across a variety of labels which can be in the form of numbers or words. If they are numbers, then they can be used directly by the algorithm. However, many times, labels need to be in readable form. Hence, the training data is usually labelled with words.
- ✓ Label encoding refers to changing the word labels into numbers so that the algorithms can understand how to work on them. Let us understand in detail how to perform label encoding –
- ✓ Create a new Python file, and import the preprocessing package –

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
input_classes = ['suzuki', 'ford', 'suzuki', 'toyota', 'ford', 'bmw']
label_encoder.fit(input_classes)
print ("\nClass mapping:")
for i, item in enumerate(label_encoder.classes_):
    print item, '-->', i
```

output :

```
Class mapping:
bmw --> 0
ford --> 1
suzuki --> 2
toyota --> 3
```

- ✓ As shown in above output, the words have been changed into 0-indexed numbers. Now, when we deal with a set of labels, we can transform them as follows −

```
labels = ['toyota', 'ford', 'suzuki']
encoded_labels = label_encoder.transform(labels)
print ("\nLabels =", labels)
print( "Encoded labels =", list(encoded_labels))
```

output :

```
Labels = ['toyota', 'ford', 'suzuki']
Encoded labels = [3, 1, 2]
```

- ✓ This is efficient than manually maintaining mapping between words and numbers. You can check by transforming numbers back to word labels as shown in the code here −

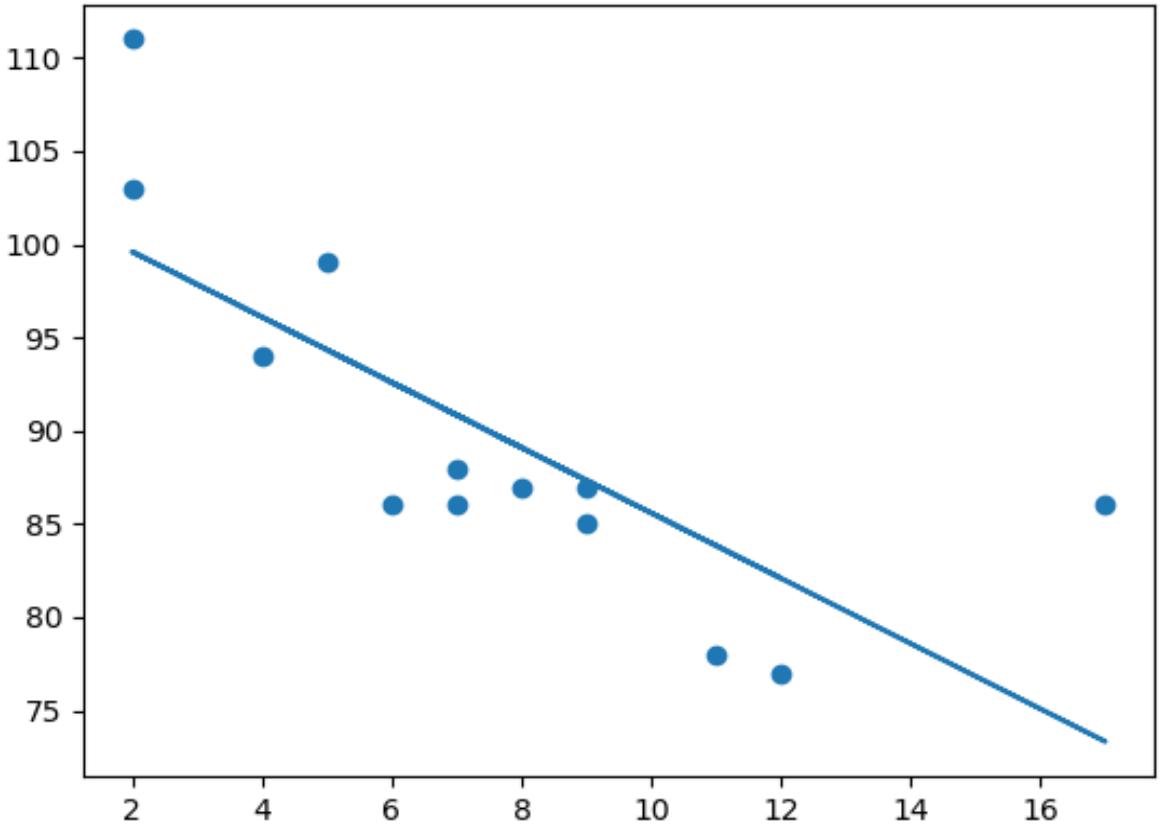
```
encoded_labels = [3, 2, 0, 2, 1]
decoded_labels =
label_encoder.inverse_transform(encoded_labels)
print( "\nEncoded labels =", encoded_labels)
print( "Decoded labels =", list(decoded_labels))
```

output :

```
Encoded labels = [3, 2, 0, 2, 1]
Decoded labels = ['toyota', 'suzuki', 'bmw', 'suzuki',
'ford']
```

6. Linear Regression:

- ✓ Linear regression uses the relationship between the data-points to draw a straight line through all them.
- ✓ This line can be used to predict future values.

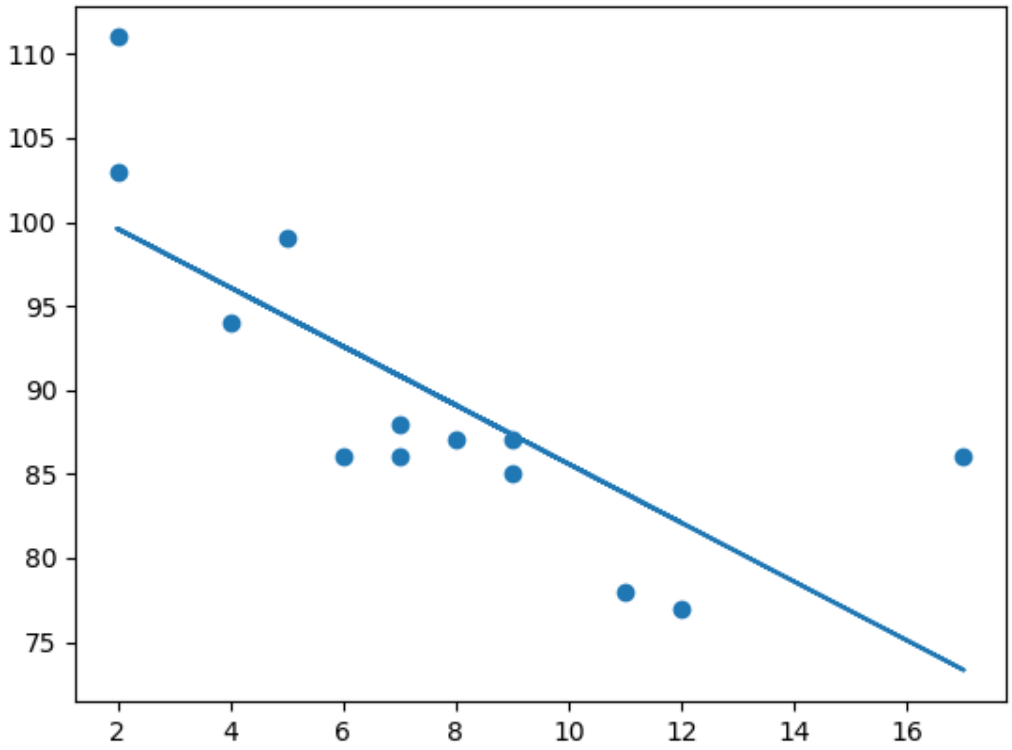


- ✓ In Machine Learning, predicting the future is very important.
- ✓ You can use the following code for scaling-

```
import matplotlib.pyplot as plt
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept
```

```
mymodel = list(map(myfunc, x))

plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
```

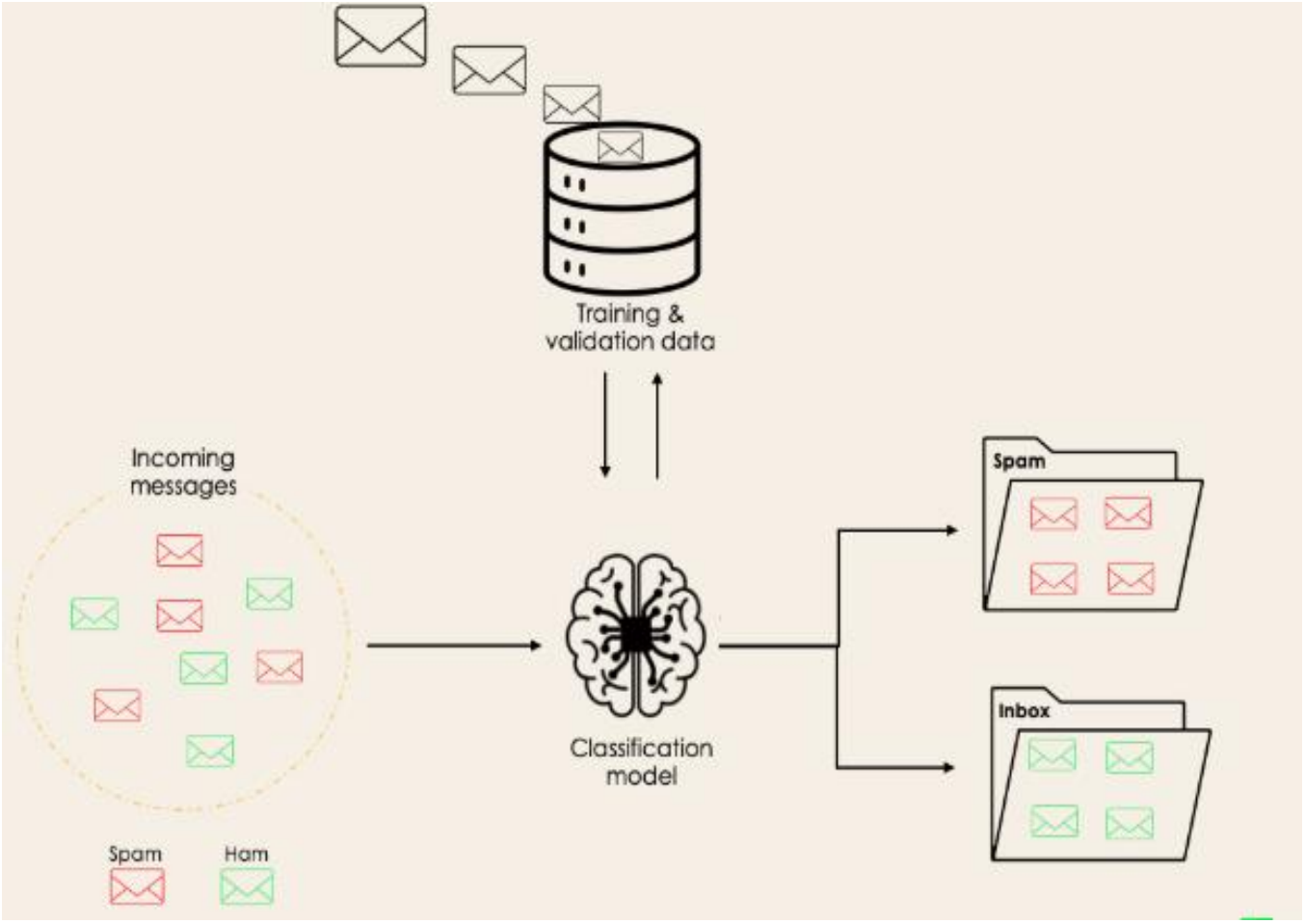
output :



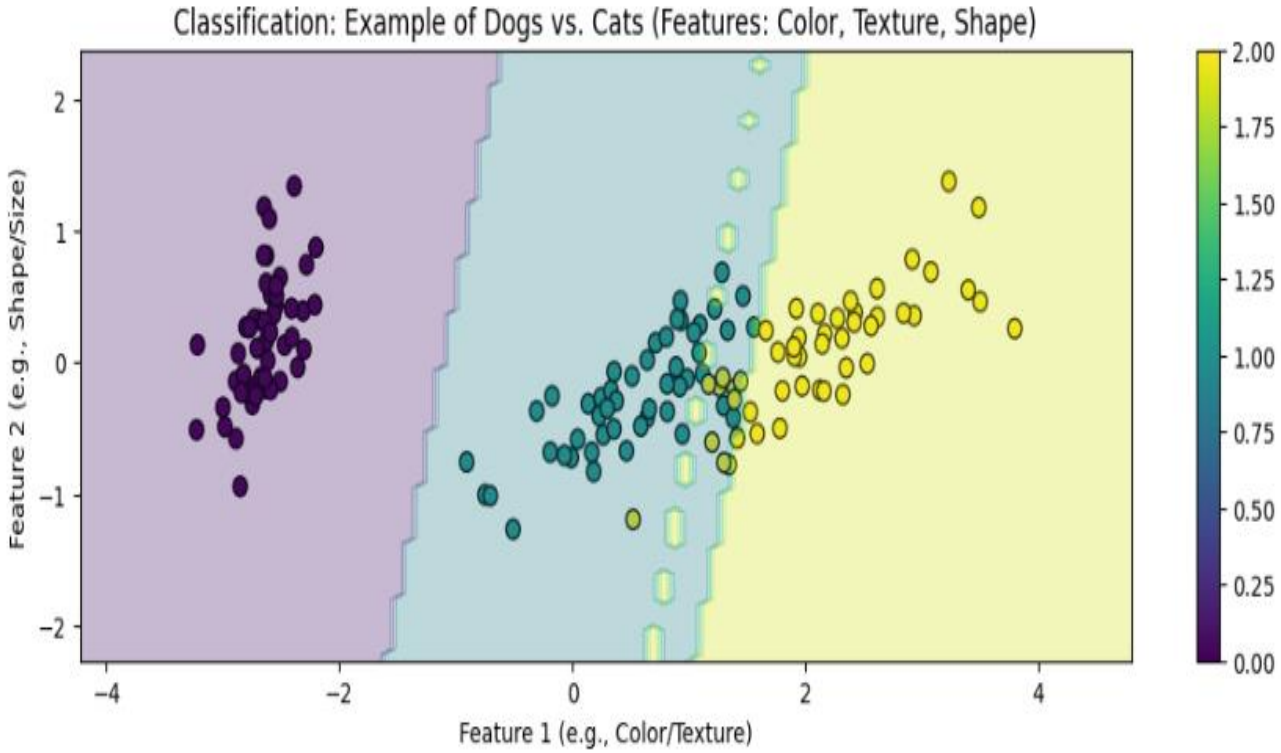
Classification

What is Classification in Machine Learning?

✓ Classification is a supervised machine learning method where the model tries to predict the correct label of a given input data. In classification, the model is fully trained using the training data, and then it is evaluated on test data before being used to perform prediction on new unseen data.



- ✓ Classification teaches a machine to sort things into categories. It learns by looking at examples with labels (like emails marked “spam” or “not spam”).
- ✓ After learning, it can decide which category new items belong to, like identifying if a new email is spam or not**.**
- ✓ For example a classification model might be trained on dataset of images labeled as either **dogs** or **cats** and it can be used to predict the class of new and unseen images as dogs or cats based on their features such as color, texture and shape.



Types of Classification

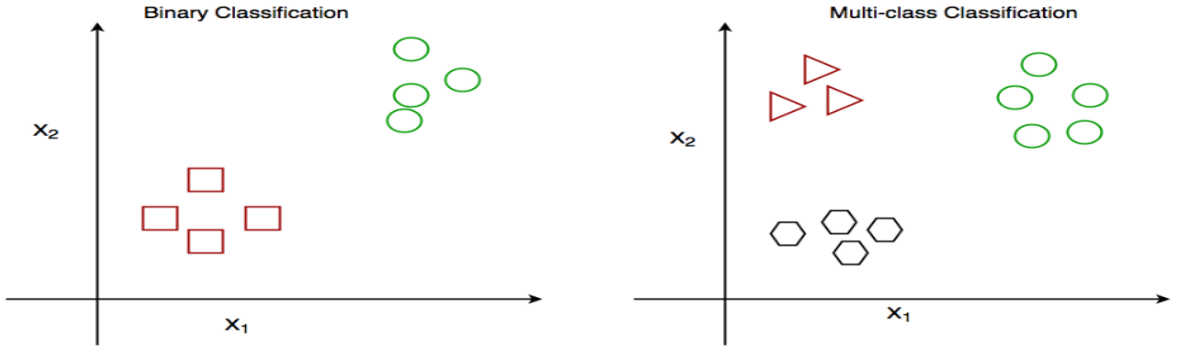
1. Binary Classification

This is the simplest kind of classification. In binary classification, the goal is to sort the data into **two distinct categories**. Think of it like a simple choice between two options. Imagine a system that sorts emails into either **spam** or **not spam**. It works by looking at **different features of the email** like certain keywords or sender details, and decides whether it's spam or not. It only chooses between these two options.

2. Multiclass Classification

Here, instead of just two categories, the data needs to be sorted into **more than two categories**. The model picks the one that best matches the input. Think of an image recognition system that sorts pictures of animals into categories like **cat**, **dog**, and **bird**.

Basically, machine looks at the **features in the image (like shape, color, or texture) and chooses which animal the picture is most likely to be based on the training it received.**



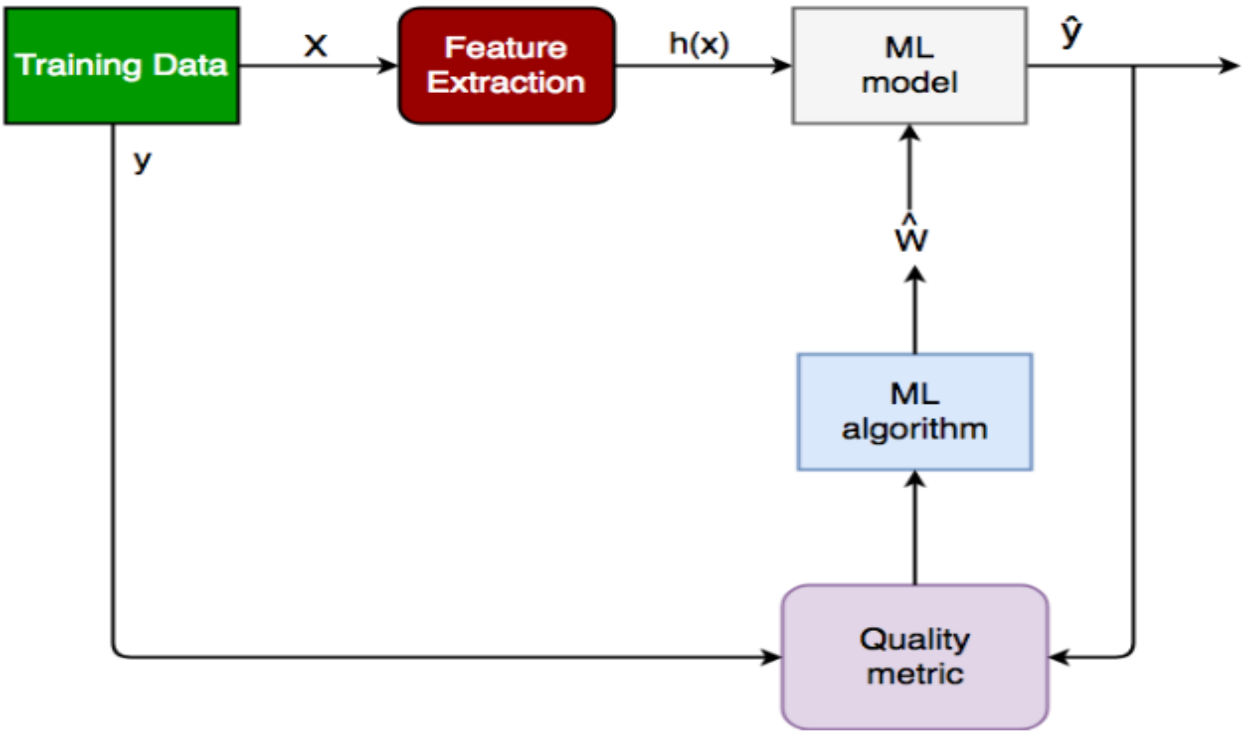
3. Multi-Label Classification

- ✓ In **multi-label classification** single piece of data can belong to **multiple categories** at once.
- ✓ Unlike multiclass classification where each data point belongs to only one class, multi-label classification allows **datapoints to belong to multiple classes.**
- ✓ A movie recommendation system could tag a movie as both **action** and **comedy**.
- ✓ The system checks various features (like movie plot, actors, or genre tags) and assigns multiple labels to a single piece of data, rather than just one.

How does Classification in Machine Learning Work?

In machine learning, classification **works by training a model to** learn patterns **from labeled data, so it can predict the category or class of new, unseen data. Here's how it works:**

1. Data Collection
2. Feature Extraction
3. Model Training
4. Model Evaluation:
5. Prediction
6. Model Evaluation



Classification Algorithms

There are various types of **classifiers algorithms**. Some of them are :

1. **Linear Classifiers**: Linear classifier models create a linear decision boundary between classes. They are simple and computationally efficient. Some of the linear **classification** models are as follows:
   - Logistic Regression
   - Support Vector Machines having kernel = 'linear'
   - Single-layer Perceptron
   - Stochastic Gradient Descent (SGD) Classifier

2. **Non-linear Classifiers**: Non-linear models create a non-linear decision boundary between classes. They can capture more complex relationships between input features and target variable. Some of the non-linear **classification** models are as follows:

- K-Nearest Neighbours
- Kernel SVM
- Naive Bayes
- Decision Tree Classification
- Ensemble learning classifiers:
- Random Forests,
- AdaBoost,
- Bagging Classifier,
- Voting Classifier,
- Extra Trees Classifier
- Multi-layer Artificial Neural Networks

What is Logistic Regression?

- ✓ **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.
- ✓ Logistic regression is a statistical algorithm which analyze the relationship between two data factors.
- ✓ The article explores the fundamentals of logistic regression, it's types and implementations.

- ✓ Logistic regression is used for binary classification where we use sigmoid function, that takes input as independent variables and produces a probability value between 0 and 1.

Types of Logistic Regression

Logistic Regression can be classified into three types:

1. **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

2. **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as “cat”, “dogs”, or “sheep”

3. **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as “low”, “Medium”, or “High”.

Differences Between Linear and Logistic Regression

Linear Regression	Logistic Regression
Linear regression is used to predict the continuous dependent variable using a given set of independent variables.	Logistic regression is used to predict the categorical dependent variable using a given set of independent variables.
Linear regression is used for solving regression problem.	It is used for solving classification problems.
In this we predict the value of continuous variables	In this we predict values of categorical variables
In this we find best fit line.	In this we find S-Curve.
Least square estimation method is used for estimation of accuracy.	Maximum likelihood estimation method is used for Estimation of accuracy.
The output must be continuous value, such as price, age, etc.	Output must be categorical value such as 0 or 1, Yes or no, etc.
It required linear relationship between dependent and independent variables.	It not required linear relationship.
There may be collinearity between the independent variables.	There should be little to no collinearity between independent variables.

Text Classification using Logistic Regression

✓ **Text classification** is the process of automatically assigning labels or categories to pieces of text. This has tons of applications, like sorting emails into spam or not-spam, figuring out if a product review is positive or negative, or even identifying the topic of a news article.

• Logistic Regression Text Classification with Scikit-Learn

✓ We'll use the popular SMS Collection Dataset, consists of a collection of SMS (Short Message Service) messages, which are labeled as either "ham" (non-spam) or "spam" based on their content.

✓ The implementation is designed to classify text messages into two categories: spam (unwanted messages) and ham (legitimate messages), using a logistic regression model.

✓ The process is broken down into several key steps:

Step 1. Import Libraries

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,
confusion_matrix
from sklearn import metrics
```

Step 2. Load and Prepare the Data

```
data = pd.read_csv("spam.csv", encoding='latin-1')
data.rename(columns={'v1': 'label', 'v2': 'text'}, inplace=True)
data['label'] = data['label'].map({'ham': 0, 'spam': 1})
```

Step 3. Text Vectorization

```
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(data['text'])
y = data['label']
```

Step 4. Split Data into Training and Testing Sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=42)
```

Step 5. Train the Logistic Regression Model

```
model = LogisticRegression(random_state=42)
model.fit(X_train, y_train)

acc = accuracy_score(y_test, model.predict(X_test)) * 100
print("Logistic Regression model accuracy",acc)


y_pred = model.predict(X_test)
print(y_pred)

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(cnf_matrix)
```

- **Manual Testing : Function to Classify Text Messages**

```
def classify_message(model, vectorizer, message):
   message_vect = vectorizer.transform([message])
   prediction = model.predict(message_vect)
   return "ham" if prediction[0] == 0 else "spam"

message = "Is that seriously how you spell his name?"
print(classify_message(model, vectorizer, message))
```

Output

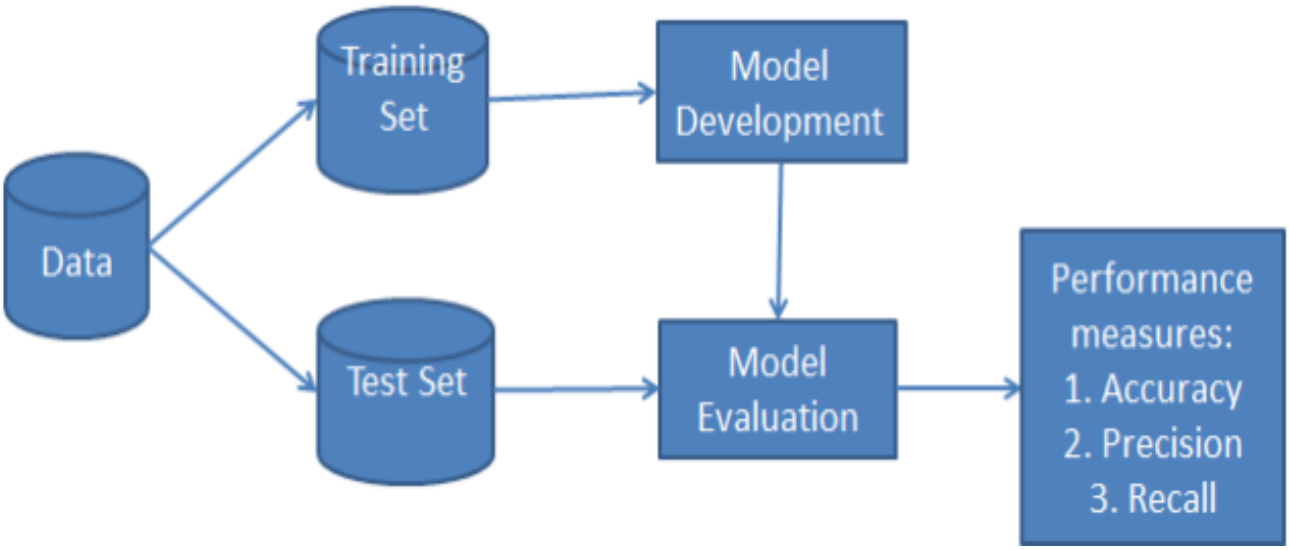
```
Logistic Regression model accuracy 97.48743718592965
[0 0 1 ... 0 0 0]
[[1199    3]
   [  32  159]]
ham
```

- The model is 97.6% correct on unseen data.
  The **Confusion Matrix** stated:
- 1201 messages correctly classified as 'ham'.
- 159 messages correctly classified as 'spam'.
- 32 'ham' messages wrongly labeled as 'spam'
- and 1 'spam' wrongly labeled as 'ham'.

What is Naive Bayes Classifier?

- ✓ Naive Bayes is a statistical classification technique based on Bayes Theorem.
- ✓ It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm.
- ✓ Naive Bayes classifiers have high accuracy and speed on large datasets.
- ✓ The classification has two phases, a learning phase and the evaluation phase.
- ✓ In the learning phase, the classifier trains its model on a given dataset, and in the evaluation phase, it tests the classifier's performance.

- ✓ Performance is evaluated on the basis of various parameters such as accuracy, error, precision, and recall.



- ✓ Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location.
- ✓ Even if these features are interdependent, these features are still considered independently.
- ✓ This assumption simplifies computation, and that's why it is considered as naive.
- ✓ This assumption is called class conditional independence.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- ✓ P(h): the probability of hypothesis h being true (regardless of the data).
- ✓ This is known as the prior probability of h.
- ✓ P(D): the probability of the data (regardless of the hypothesis). This is known as the prior probability.
- ✓ P(h|D): the probability of hypothesis h given the data D. This is known as posterior probability.
- ✓ P(D|h): the probability of data d given that the hypothesis h was true. This is known as posterior probability.

Why it is Called Naive Bayes?

- ✓ It is named as "Naive" because it assumes the presence of one feature does not affect other features.
- ✓ The "Bayes" part of the name refers to for the basis in Bayes' Theorem.
- ✓ Consider a fictional dataset that describes the weather conditions for playing a game of golf.
- ✓ Given the weather conditions, each tuple classifies the conditions as fit("Yes") or unfit("No") for playing golf.
- ✓ Here is a tabular representation of our dataset.

	Outlook	Temperature	Humidity	Windy	Play Golf
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes

How Naive Bayes Classifier Works?

Naive Bayes classifier calculates the probability of an event in the following steps:

- ✓ **Step 1**: Calculate the prior probability for given class labels
- ✓ **Step 2**: Find Likelihood probability with each attribute for each class
- ✓ **Step 3**: Put these value in Bayes Formula and calculate posterior probability.

✓ **Step 4**: See which class has a higher probability, given the input belongs to the higher probability class.

Whether	Play
Sunny	No
Sunny	No
Overcast	Yes
Rainy	Yes
Rainy	Yes
Rainy	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rainy	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table

Whether	No	Yes
Overcast		4
Sunny	2	3
Rainy	3	2
Total	5	9

Likelihood Table 1

Whether	No	Yes		
Overcast		4	=4/14	0.29
Sunny	2	3	=5/14	0.36
Rainy	3	2	=5/14	0.36
Total	5	9		
	=5/14	=9/14		
	0.36	0.64		

Likelihood Table 2

Whether	No	Yes	Posterior Probability for No	Posterior Probability for Yes
Overcast		4	0/5=0	4/9=0.44
Sunny	2	3	2/5=0.4	3/9=0.33
Rainy	3	2	3/5=0.6	2/9=0.22
Total	5	9		

Types of Naive Bayes Model

There are three types of Naive Bayes Model :

1.Gaussian Naive Bayes

In **Gaussian Naive Bayes**, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called Normal distribution When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below

2.Multinomial Naive Bayes

Multinomial Naive Bayes is used when features represent the frequency of terms (such as word counts) in a document. It is commonly applied in text classification, where term frequencies are important.

3.Bernoulli Naive Bayes

Bernoulli Naive Bayes deals with binary features, where each feature indicates whether a word appears or not in a document. It is suited for scenarios where the presence or absence of terms is more relevant than their frequency. Both models are widely used in document classification tasks

- **Naive Bayes Classifier with Loan Dataset**

❖ **Data Loading**

In this example, we will be loading **Loan Data** from DataLab using the pandas '**read_csv**` function.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

#Data Loading
df = pd.read_csv('loan_data.csv')
```

```
df.head()
```

❖ Data Exploration

To understand more about the dataset we will use `.info()`.

- The dataset consists of 14 columns and 9578 rows.
- Apart from “purpose”, columns are either floats or integers.
- Our target column is “not.fully.paid”.

```
#Data Exploration
df.info()

sns.countplot(data=df,x='purpose',hue='not.fully.paid')
plt.xticks(rotation=45, ha='right');
```

Output:

```
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   credit.policy     9578 non-null   int64
 1   purpose          9578 non-null   object
 2   int.rate         9578 non-null   float64
 3   installment       9578 non-null   float64
 4   log.annual.inc    9578 non-null   float64
 5   dti             9578 non-null   float64
```

```
 6   fico               9578 non-null   int64
 7   days.with.cr.line  9578 non-null   float64
 8   revol.bal          9578 non-null   int64
 9   revol.util         9578 non-null   float64
 10  inq.last.6mths     9578 non-null   int64
 11  delinq.2yrs        9578 non-null   int64
 12  pub.rec            9578 non-null   int64
 13  not.fully.paid     9578 non-null   int64
dtypes: float64(6), int64(7), object(1)
memory usage: 1.0+ MB
```

❖ Data Processing

```
pre_df = pd.get_dummies(df,columns=['purpose'],drop_first=True)
pre_df.head()

X = pre_df.drop('not.fully.paid', axis=1)
y = pre_df['not.fully.paid']

X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.33, random_state=125
)
```

❖ Model Building and Training

```
model = GaussianNB()

model.fit(X_train, y_train);
```

❖ Model Evaluation

```
from sklearn.metrics import (
   accuracy_score,
   confusion_matrix,
   ConfusionMatrixDisplay,
   f1_score,
   classification_report,
)

y_pred = model.predict(X_test)

accuray = accuracy_score(y_pred, y_test)
f1 = f1_score(y_pred, y_test, average="weighted")

print("Accuracy:", accuray)
print("F1 Score:", f1)
```

Output:

```
Accuracy: 0.8206263840556786
F1 Score: 0.8686606980013266
```

- **Advantages of Naive Bayes Classifier**

1) Easy to implement and computationally efficient.
2) Effective in cases with a large number of features.
3) Performs well even with limited training data.
4) It performs well in the presence of categorical features.
5) For numerical features data is assumed to come from normal distributions

- **Disadvantages of Naive Bayes Classifier**

1) Assumes that features are independent, which may not always hold in real-world data.
2) Can be influenced by irrelevant attributes.
3) May assign zero probability to unseen events, leading to poor generalization.

- **Applications of Naive Bayes Classifier**

1) **Spam Email Filtering**: Classifies emails as spam or non-spam based on features.
2) **Text Classification**: Used in sentiment analysis, document categorization, and topic classification.
3) **Medical Diagnosis:** Helps in predicting the likelihood of a disease based on symptoms.
4) **Credit Scoring:** Evaluates creditworthiness of individuals for loan approval.
5) **Weather Prediction**: Classifies weather conditions based on various factors.

Predictive Modeling

What is Predictive Modeling ?

✓ **Predictive modelling** is a process used in **data science** to create a mathematical model that predicts an outcome based on input data. It involves using statistical algorithms and machine learning techniques to analyze historical data and make predictions about future or unknown events.

Importance of Predictive Modeling

1. **Decision Making**: It helps businesses and organizations make informed decisions by providing insights into future trends and outcomes based on historical data.
2. **Risk Management**: It helps in assessing and managing risks by predicting potential outcomes and allowing organizations to take proactive measures.
3. **Resource Optimization**: It helps in optimizing resources such as time, money, and manpower by providing forecasts and insights that can be used to allocate resources more efficiently.
4. **Customer Insights**: It helps in understanding customer behavior and preferences, which can be used to personalize products, services, and marketing strategies.
5. **Competitive Advantage**: It can provide a competitive advantage by enabling organizations to anticipate market trends and customer needs ahead of competitors.

6. **Cost Reduction**: By predicting future outcomes, organizations can reduce costs associated with errors, inefficiencies, and unnecessary expenditures.
7. **Improved Outcomes**: In fields like healthcare, predictive modeling can help in improving patient outcomes by predicting diseases, identifying high-risk patients, and recommending personalized treatments

Applications of Predictive Modeling

1. Finance

- **Risk Assessment** – Evaluates creditworthiness to reduce default risk.
- **Fraud Detection** – Identifies fraudulent activities in transactions.

2. Healthcare

- **Disease Prediction** – Predicts diseases for early intervention.
- **Resource Allocation** – Optimizes hospital resources and staffing.

3. Marketing & CRM

- **Customer Segmentation** – Groups customers for targeted marketing.
- **Churn Prediction** – Identifies customers likely to leave a service.

4. Supply Chain Management

- **Demand Forecasting** – Predicts product demand for inventory optimization.
- **Logistics Optimization** – Improves routing and transportation efficiency.

5. Human Resources

- **Talent Acquisition** – Identifies top candidates for job openings.
- **Employee Retention** – Predicts turnover and improves retention strategies.

Types of Predictive Models

- **Linear Regression**: Linear regression is used when the relationship between the dependent variable and the independent variables is linear. It is often used for predicting continuous outcomes.
- **Logistic Regression**: Logistic regression is used when the dependent variable is binary (i.e., has two possible outcomes). It is commonly used for classification problems.
- **Decision Trees**: Decision trees are used to create a model that predicts the value of a target variable based on several input variables. They are easy to interpret and can handle both numerical and categorical data.
- **Random Forests**: Random forests are an ensemble learning method that uses multiple decision trees to improve the accuracy of the predictions. They are robust against overfitting and can handle large datasets with high dimensionality.

- **Support Vector Machines (SVM)**: SVMs are used for both regression and classification tasks. They work well for complex, high-dimensional datasets and can handle non-linear relationships between variables.
- **Neural Networks**: Neural networks are a class of deep learning models inspired by the structure of the human brain. They are used for complex problems such as image recognition, natural language processing, and speech recognition.
- **Gradient Boosting Machines**: Gradient boosting machines are another ensemble learning method that builds models sequentially, each new model correcting errors made by the previous ones. They are often used for regression and classification tasks.
- **Time Series Models**: Time series models are used for predicting future values based on past observations. They are commonly used in finance, economics, and weather forecasting.

• Support Vector Machine (SVM) Algorithm

✓ Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. While it can handle regression problems, SVM is particularly well-suited for classification tasks.

✓ SVM aims to find the optimal hyperplane in an N-dimensional space to separate data points into different classes. The algorithm maximizes the margin between the closest points of different classes.

- **Support Vector Machine (SVM) Terminology**

- **Hyperplane**: A decision boundary separating different classes in feature space, represented by the equation **wx + b = 0** in linear classification.

- **Support Vectors**: The closest data points to the hyperplane, crucial for determining the hyperplane and margin in SVM.

- **Margin**: The distance between the hyperplane and the support vectors. SVM aims to maximize this margin for better classification performance.

- **Kernel**: A function that maps data to a higher-dimensional space, enabling SVM to handle non-linearly separable data.

- **Hard Margin**: A maximum-margin hyperplane that perfectly separates the data without misclassifications.

- **Soft Margin**: Allows some misclassifications by introducing slack variables, balancing margin maximization and misclassification penalties when data is not perfectly separable.

- **C**: A regularization term balancing margin maximization and misclassification penalties. A higher C value enforces a stricter penalty for misclassifications.

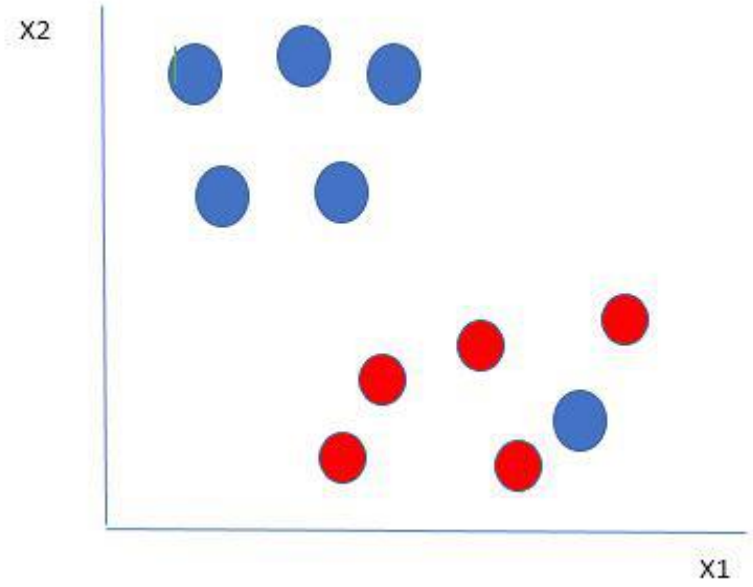
- **Hinge Loss**: A loss function penalizing misclassified points or margin violations, combined with regularization in SVM.

- **Dual Problem**: Involves solving for Lagrange multipliers associated with support vectors, facilitating the kernel trick and efficient computation.

How does Support Vector Machine Algorithm Work?

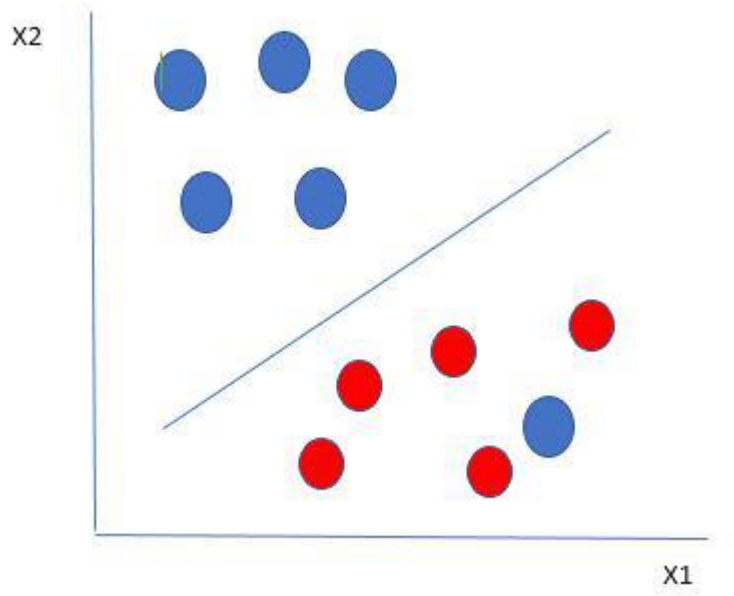
✓ The key idea behind the SVM algorithm is to find the hyperplane that best separates two classes by maximizing the margin between them. This margin is the distance from the hyperplane to the nearest data points (**support vectors**) on each side.

Let's consider a scenario like shown below:



How does SVM classify the data?

✓ It's simple! The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.



- **<u>Types of Support Vector Machine</u>**

Based on the nature of the decision boundary, Support Vector Machines (SVM) can be divided into two main parts:

- **Linear SVM:** Linear SVMs use a linear decision boundary to separate the data points of different classes. When the data can be precisely linearly separated, linear SVMs are very suitable. This means that a single straight line (in 2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes. A hyperplane

that maximizes the margin between the classes is the decision boundary.

- **Non-Linear SVM:** Non-Linear SVM can be used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data. The original input data is transformed by these kernel functions into a higher-dimensional feature space, where the data points can be linearly separated. A linear SVM is used to locate a nonlinear decision boundary in this modified space.

- **Implementing SVM Algorithm in Python**

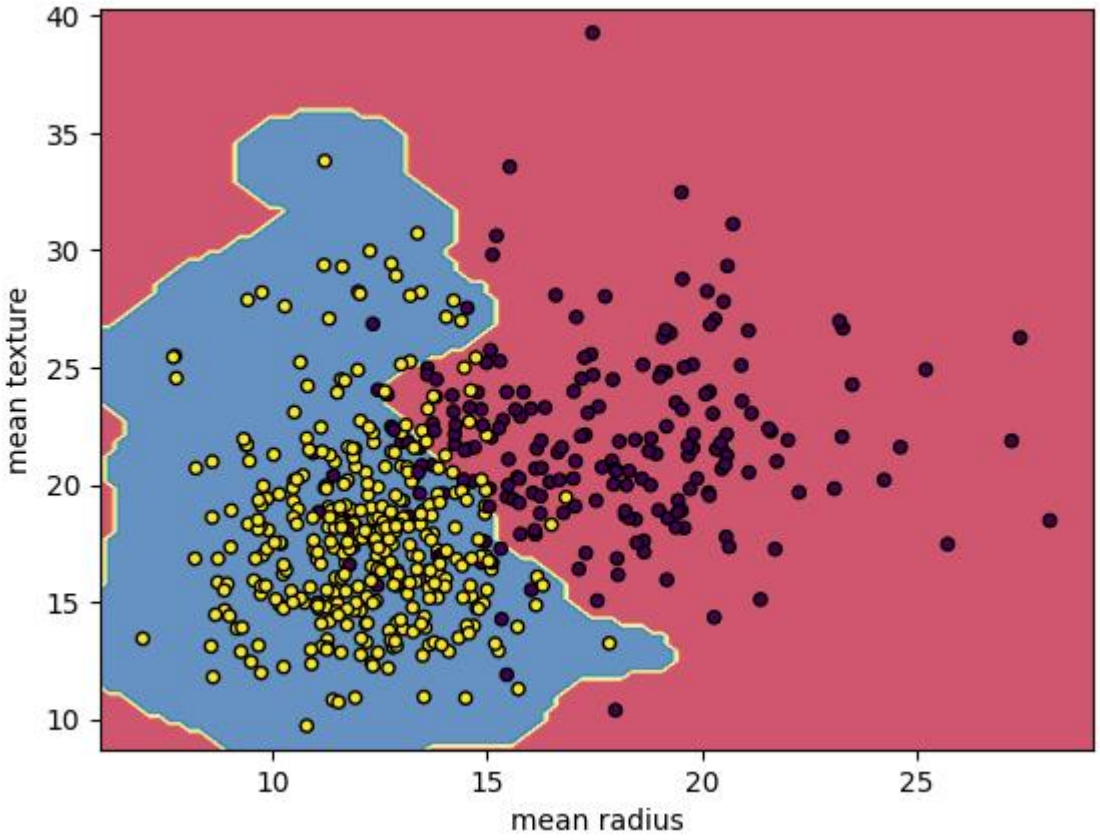
Predict if cancer is Benign or malignant. Using historical data about patients diagnosed with cancer enables doctors to differentiate malignant cases and benign ones are given independent attributes.

- Load the breast cancer dataset from sklearn.datasets
- Separate input features and target variables.
- Build and train the SVM classifiers using RBF kernel.
- Plot the scatter plot of the input features.

```
# Load the important packages
from sklearn.datasets import load_breast_cancer
import matplotlib.pyplot as plt
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import SVC
```

```
# Load the datasets
cancer = load_breast_cancer()
X = cancer.data[:, :2]
y = cancer.target

#Build the model
svm = SVC(kernel="rbf", gamma=0.5, C=1.0)
# Trained the model
svm.fit(X, y)

# Plot Decision Boundary
DecisionBoundaryDisplay.from_estimator(
      svm,
      X,
      response_method="predict",
      cmap=plt.cm.Spectral,
      alpha=0.8,
      xlabel=cancer.feature_names[0],
      ylabel=cancer.feature_names[1],
   )

# Scatter plot
plt.scatter(X[:, 0], X[:, 1],
          c=y,
          s=20, edgecolors="k")
plt.show()
```

Output:



- **Advantages of Support Vector Machine (SVM)**

1. High-Dimensional Performance
2. Nonlinear Capability
3. Outlier Resilience
4. Binary and Multiclass Support
5. Memory Efficiency

- **Disadvantages of Support Vector Machine (SVM)**

1. Slow Training
2. Parameter Tuning Difficulty
3. Noise Sensitivity
4. Limited Interpretability
5. Feature Scaling Sensitivity

EXAM IMP QUESTIONS (UNIT 2)

One Mark Questions

Fill in the Blanks (5 Questions)

1. In supervised learning, the model learns from _____ data.
   Ans: labeled

2. The equation of simple linear regression is _____.
   Ans: Y=mX+CY = mX + CY=mX+C

3. _____ algorithm is used when the output variable is categorical.
   Ans: Classification

4. The process of splitting data into training and testing sets is called _____.
   Ans: Data Splitting

5. The technique used to reduce overfitting in Decision Trees is _____.
   Ans: Pruning

Full Form Questions (5 Questions)

1) What is the full form of SVM?
   Ans: Support Vector Machine

2) What is the full form of MSE in regression models?
   Ans: Mean Squared Error

3) What is the full form of KNN?
   Ans: K-Nearest Neighbors

4) What is the full form of ROC in classification evaluation?
   Ans: Receiver Operating Characteristic

5) What is the full form of RMSE?
   Ans: Root Mean Squared Error

True/False Questions (5 Questions)

1) In supervised learning, the model is trained without labeled data. **(True/False)**
   Ans: False

2) Logistic regression is used for regression problems. **(True/False)**
   Ans: False

3) A higher accuracy score always means a better classification model. **(True/False)**
   Ans: False

4) Overfitting occurs when the model performs well on training data but poorly on test data. **(True/False)**
   Ans: True

5) Decision Trees can be used for both classification and regression tasks. **(True/False)**
   Ans: True

Definitions (5 Questions)

1) **Define Supervised Learning.**
   Ans: Supervised learning is a type of machine learning where the model is trained on labeled data to make predictions.

2) **Define Classification.**
   Ans: Classification is a supervised learning task where the output variable is categorical, such as "Spam" or "Not Spam".

3) **Define Regression.**
   Ans: Regression is a supervised learning task where the output variable is continuous, such as predicting house prices.

4) **Define Overfitting.**
   Ans: Overfitting occurs when a model learns the training data too well, leading to poor performance on new data.

5) **Define Confusion Matrix.**
   Ans: A confusion matrix is a table used to evaluate the performance of a classification model by displaying true positives, false positives, true negatives, and false negatives.

Fill in the Blanks (5 Questions)

1) The target variable in regression is always _____.
   Ans: Continuous

2) _____ is a supervised learning algorithm used for both classification and regression tasks.
   Ans: Decision Tree

3) The method used to prevent overfitting in machine learning models is called _____.
   Ans: Regularization

4) In a confusion matrix, False Positives occur when the model incorrectly predicts the _____ class as the _____ class.
   Ans: Negative, Positive

5) The performance of a regression model is measured using metrics like _____ and _____.
Ans: Mean Squared Error (MSE), R-squared

Two Mark Questions

1) What is the difference between linear and non-linear classifiers?
2) What are the key assumptions of Naive Bayes classification?
3) Give Advantages and Disadvantages of Supervised Learning

Three Mark Questions

1) Compare logistic regression and linear regression.
2) Describe how the mean removal technique is applied in data preprocessing.
3) What are the advantages and disadvantages of Naive Bayes?
4) What are the types of Naive Bayes classifiers?
5) How does the Support Vector Machine (SVM) work?
6) What is Data Pre-Processing in Machine learning?Explain with different techniques.

Five Mark Questions

1) Explain in detail the working of Naive Bayes classification with an example.
2) What is logistic regression? Explain its types and applications with Python implementation.

3) Explain the SVM algorithm with an example and implementation in Python.
4) Explain how text classification is performed using logistic regression.
5) Explain how Naive Bayes classifier is used for Spam Filtering.
6) Explain KNN algorithm with suitable example.
7) Explain Supervised Learning in detail with proper figure.
8) Explain Regration in detail.
9) Explain types of Supervised Learning in detail.
10) Explain term mean removal, scaling, normalization, banalization, label encoding with example.
11) Explain Linear Regression with supervised machine learning.
12) Explain term Classification. Explain types of Classification.
13) Explain Logistic Regression classifier with Implementation Text Classification with Scikit-learn using Python.
14) Explain Naïve Bayes Classifier with training and testing dataset using example.
15) Implementation of Naïve Bayes Classifier with any data set example using Python. (Use loan data set).
16) Explain linear and non-linear classifier using Support Vector Machine with example.