Experiment No: 01

Experiment Name: Write a Python program to import and export data using the Pandas library functions.

Objective: The objectives of this problem are to understand the basics of Pandas library import data from external sources into Pandas DataFrames, manipulate and analyze data using Pandas functions, and export processed data to different file formats.

Theory: Pandas is a powerful Python library for data manipulation and analysis. It provides data structures like Series and DataFrame and functions to manipulate large datasets efficiently. One of the key features of Pandas is its ability to import data from various sources and export processed data to different formats.

Requirements:

1. Pandas Library:
    - The program requires the Pandas library to be installed. Install it using:
      pip install pandas
2. Data Files:
    - The program expects input data in CSV and Excel formats.
    - CSV file path: path/to/your/data.csv
    - Excel file path: path/to/your/data.xlsx
3. Import Data:
    - The program should import data from the specified CSV and Excel files.
    - If the files are not found, the program should handle the FileNotFoundError gracefully and provide an error message.
4. Data Manipulation:
    - The program should demonstrate basic data manipulation using Pandas functions, such as filtering rows based on a condition, selecting specific columns, and calculating summary statistics.
5. Export Data:
    - The program should export the manipulated data to new CSV and Excel files.
    - If there is an error during export, the program should handle it gracefully and provide an error message.
6. Display or Save Results:
    - The program should display the filtered data and save it to an Excel file.
    - If there is an error during display or export, the program should handle it gracefully and provide an error message.
7. Error Handling:
    - The program should handle errors, such as file not found or export errors, gracefully and provide meaningful error messages.

Algorithm:

Step-1: Import the Pandas library
Step-2: Define file paths
Step-3: Import data from CSV and Excel files
Step-4: Data Manipulation
Step-5: Exporting data to CSV
Step-6: Display or Save Results

Input and Output:

Created data frame

	Name	Age	City
0	Alice	25	New York
1	Bob	30	San Francisco
2	Charlie	35	Los Angeles
3	David	40	Chicago

CSV file

	Name	Age	City
0	Alice	25	New York
1	Bob	30	San Francisco
2	Charlie	35	Los Angeles
3	David	40	Chicago

Excel file

	Name	Age	City
0	Alice	25	New York
1	Bob	30	San Francisco
2	Charlie	35	Los Angeles
3	David	40	Chicago

Json file

	Name	Age	City
0	Alice	25	New York
1	Bob	30	San Francisco
2	Charlie	35	Los Angeles
3	David	40	Chicago

Discussion: The Python program successfully demonstrated the use of the Pandas library to import, manipulate, and export data. It provides a foundation for more complex data analysis tasks and can be extended based on specific project requirements.

Experiment No: 02

Experiment Name: Write a Python program for pre-processing data using various techniques for a given dataset.

Objective: The objective is to clearly state the goal of the lab, specifying the dataset used and the techniques to be applied. Outline the importance of each pre-processing technique in enhancing data quality.

Theory: Data pre-processing is a crucial step in the data analysis and machine learning pipeline. It involves cleaning and transforming raw data into a format that is more suitable for analysis or for training machine learning models. The quality of the data used for analysis or model training directly impacts the results and performance.

Common Data Pre-processing Techniques:

- Handling Missing Values:
  - Techniques such as imputation or removal of missing values are used to address the issue of incomplete data.
- Data Normalization:
  - Scaling numerical features to a standard range (e.g., between 0 and 1) ensures that all features contribute equally to the analysis.
- Encoding Categorical Variables:
  - Converting categorical variables into numerical representations allows algorithms to work with these variables effectively.
- Dealing with Outliers:
  - Identification and handling of outliers prevent them from disproportionately influencing analysis or model training.

Python Libraries for Data Pre-processing:

- Pandas:
  - Pandas is a powerful library for data manipulation and analysis. It provides data structures like DataFrames, which are highly effective for handling and transforming datasets.
- NumPy:
  - NumPy is a fundamental package for scientific computing with Python. It provides support for large, multi-dimensional arrays and matrices, which is essential for numerical operations in data pre-processing.
- Scikit-learn:
  - Scikit-learn offers a wide range of tools for machine learning, including pre-processing functionalities such as scaling, encoding, and imputation.

Requirements: To perform data pre-processing in Python, we'll need to have certain tools and libraries installed. Here are the key requirements:

1. Python: Ensure that we have Python installed on your system.
2. Integrated Development Environment (IDE) or Code Editor:
   - Jupyter Notebooks
   - VSCode (Visual Studio Code)
   - PyCharm
3. Python Libraries: Install the necessary Python libraries using a package manager like pip
   - Pandas
   - NumPy
   - Scikit-learn
   - Matplotlib
   - Seaborn
5. Dataset: We'll need a dataset for practicing data pre-processing. This could be in the form of a CSV, Excel file, or any other compatible format.
6. Documentation and Learning Resources: Have access to documentation for the libraries we're using, as well as learning resources for Python and data pre-processing.

Algorithm:

Step-1: Load data in Pandas.
Step-2: Drop columns that aren't useful.
Step-3: Drop rows with missing values.
Step-4: Create dummy variables.
Step-5: Take care of missing data.
Step-6: Convert the data frame to NumPy.
Step-7: Divide the data set into training data and test data.
Step-8: Feature Scaling.

Input and Output:

status_type	num_reactions	num_comments	...	num_hahas	num_sads	num_angrys
0	3	529	512	...	1	1	0
1	1	150	0	...	0	0	0
2	3	227	236	...	1	0	0
3	1	111	0	...	0	0	0

Discussion: Data pre-processing is a critical phase that significantly influences the quality and reliability of subsequent analyses and provides a flexible and modular approach to address common challenges in real-world datasets.

Experiment No: 03

Experiment Name: Write a Python program to extract features from a colour picture to make it usable to learn machines.

Objective: The main objective is to understand the concept of feature extraction from color images, explore different techniques for extracting features from color images, and evaluate the impact of feature extraction on machine learning model performance.

Theory: Feature extraction is a part of the dimensionality reduction process, in which, an initial set of the raw data is divided and reduced to more manageable groups. So when you want to process it will be easier. The most important characteristic of these large data sets is that they have a large number of variables. These variables require a lot of computing resources to process. So Feature extraction helps to get the best feature from those big data sets by selecting and combining variables into features, thus, effectively reducing the amount of data. These features are easy to process, but still able to describe the actual data set with accuracy and originality.

Requirements: To conduct a lab on extracting features from color images for machine learning, we will need a combination of hardware, software, and data. Here's a list of requirements:

Hardware:

1. Computer

Software:

1. Programming Language
2. Integrated Development Environment (IDE)
3. Machine Learning Libraries
4. Image Processing Libraries

Data:

1. Image Dataset

Tools and Resources:

1. Documentation
2. Online Resources

Reporting and Analysis:

1. Reporting Tools:
2. Notebook or Report Template:

Algorithm:

Step-1: Import Libraries
Step-2: Load and Preprocess Images
Step-3: Feature Extraction using HOG
Step-4: Extract HOG Features for all Images
Step-5: Split Data into Training and Testing Sets
Step-6: Train SVM Model
Step-7: Make Predictions and Evaluate

Input:

Output:

Features: [0 0 0 ... 255 0 0]

Discussion: We have successfully provided a comprehensive analysis of your experiment, demonstrating a deep understanding of the impact of feature extraction on machine-learning model performance.

Experiment No:04

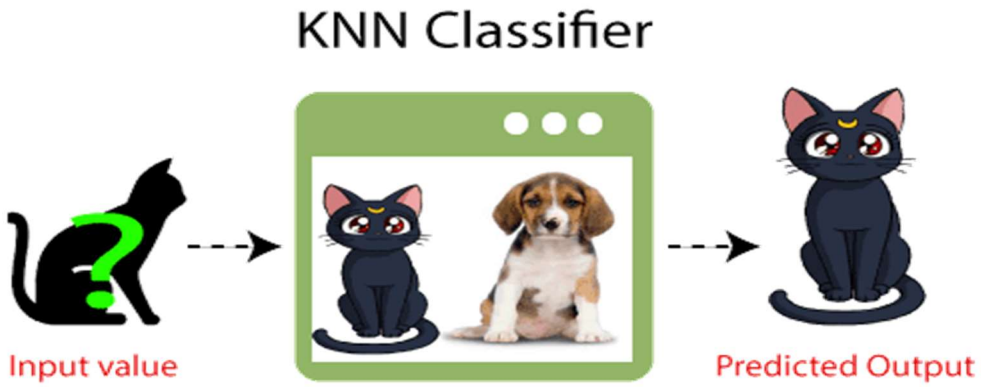
Experiment Name: Write a Python program to implement the K-Nearest Neighbour (KNN) algorithm.

Objective: The main objective is to implement the K-Nearest Neighbour(KNN) algorithm using the Python language.

Theory: The K-Nearest Neighbors (KNN) algorithm is a simple yet powerful classification and regression technique widely used in machine learning. It belongs to the family of supervised learning algorithms and is based on the principle that similar instances should have similar class labels. This lab aims to implement the KNN algorithm and evaluate its performance on a given dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



Requirements:

1. Dataset
2. Programming Language
3. Libraries
4. Data Preprocessing
5. Feature Scaling
6. KNN Implementation
7. Model Evaluation
8. Documentation.

Algorithm:

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready.

Input and Output:

[1, 2, 2, 0, 1, 0, 0, 0, 1, 2, 1, 0, 2, 1, 0, 1, 2, 0, 2, 1, 1, 1, 1, 1, 2, 0, 2, 1, 2, 0]
0.9666666666666667

Discussion: We successfully implement the KNN algorithm which offering a foundation for understanding its behavior, strengths, and limitations in real-world applications

Experiment No: 05

Experiment Name: Write a Python program to implement the K-Means Clustering Algorithm and test it using the appropriate dataset.

Objective: The main objective is to implement the K-Means Clustering Algorithm and test it using the appropriate dataset using Python language.

Theory: K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on. It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training. It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

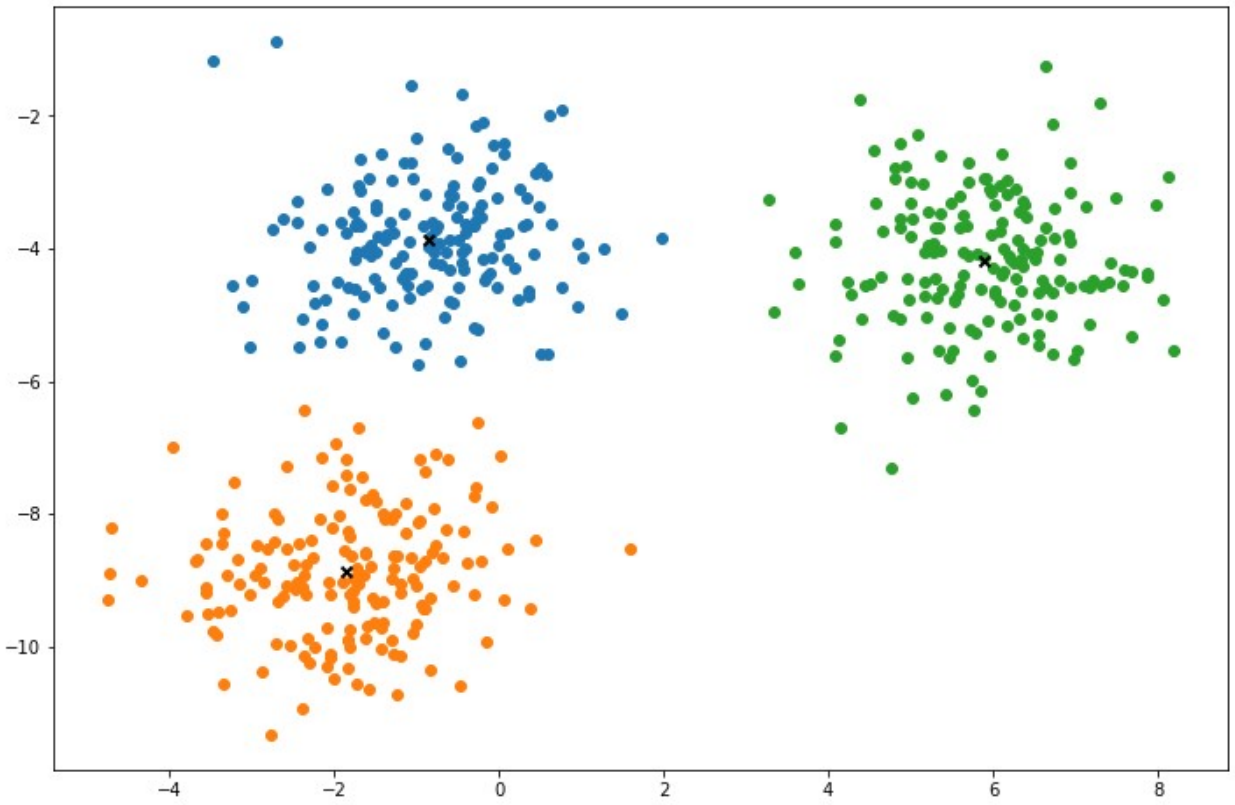
Requirements:

1. Dataset
2. Programming Language
3. Libraries
4. Data Preprocessing
5. Feature Scaling
6. KNN Implementation
7. Model Evaluation
8. Documentation.

Algorithm:

Step-1: Select the number K to decide the number of clusters.
Step-2: Select random K points or centroids. (It can be other from the input dataset).
Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.
Step-4: Calculate the variance and place a new centroid of each cluster.
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.
Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.
Step-7: The model is ready.

Input and Output:



Discussion: We successfully implement the K-Means Clustering algorithm which offering a foundation for understanding its behavior, strengths, and limitations in real-world applications

Experiment No:06

Experiment Name: Write a Python program to design KNN classification model for a given dataset (like Iris dataset).

Objective: The main objective is to design KNN classification model for a given dataset (like Iris dataset).

Theory: In machine learning, K-Nearest Neighbours or KNN is the simplest of all machine learning algorithms. It is a non-parametric algorithm used for classification and regression tasks. Non-parametric means there is no assumption required for data distribution. So, KNN does not require any underlying assumption to be made. In both classification and regression tasks, the input consists of the k-closest training examples in the feature space. The output depends upon whether KNN is used for classification or regression purposes.

In KNN classification, the output is a class membership. The given data point is classified based on the majority of types of its neighbors. The data point is assigned to the most frequent class among its k nearest neighbors. Usually, k is a small positive integer. If k=1, then the data point is simply assigned to the class of that single nearest neighbor.

In KNN regression, the output is simply some property value for the object. This value is the average of the values of k nearest neighbors.

KNN is a type of instance-based learning or lazy learning. Lazy learning means it does not require any training data points for model generation. All training data will be used in the testing phase. This makes training faster and testing slower and costlier. So, the testing phase requires more time and memory resources.

In KNN, the neighbors are taken from a set of objects for which the class or the object property value is known. This can be thought of as the training set for the KNN algorithm, though no explicit training step is required. In both the classification and regression KNN algorithms, we can assign weight to the contributions of the neighbors. So, nearest neighbors contribute more to the average than the more distant ones.

Requirements:

1. Dataset
2. Programming Language
3. Libraries
4. Data Preprocessing
5. Feature Scaling
6. KNN Implementation
7. Model Evaluation
8. Documentation.

Algorithm:

The K-NN working can be explained on the basis of the below algorithm:

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready.

Input:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Output:

Accuracy: 96.66666666666667

Discussion: We successfully implement the KNN algorithm which offering a foundation for understanding its behavior, strengths, and limitations in real-world applications

Experiment No:07

Experiment Name: Write a Python program to build an Artificial Neural Network (ANN) model with back propagation on a given dataset.

Objective: The main objective is to build an Artificial Neural Network (ANN) model with back propagation on a given dataset.

Theory: A neural network with a single layer is called a perceptron. A multi-layer perceptron is called Artificial Neural Networks. A Neural network can possess any number of layers. Each layer can have one or more neurons or units. Each of the neurons is interconnected with each and every other neuron. Each layer could have different activation functions as well.
ANN consists of two phases Forward propagation and Backpropagation. The forward propagation involves multiplying weights, adding bias, and applying activation function to the inputs and propagating it forward.
The backpropagation step is the most important step which usually involves finding optimal parameters for the model by propagating in the backward direction of the Neural network layers. The backpropagation requires optimization function to find the optimal weights for the model.
ANN can be applied to both Regression and Classification tasks by changing the activation functions of the output layers accordingly. (Sigmoid activation function for binary classification, Softmax activation function for multi-class classification and Linear activation function for Regression).

Requirements:

Hardware Requirements:

1. CPU/GPU
2. RAM (Memory)

Software Requirements:

1. Programming Language
2. Machine Learning Libraries
3. Development Environment
4. Data Processing Libraries
5. Visualization Libraries

Algorithm:

Step-1: Randomly initialize the weights and biases for each neuron in the network. This is typically done using small random values.

Step-2: Input data is passed through the network to generate predictions (output) using the current weights and biases.

Step-3: Compare the network's predictions to the actual target values using a loss function. The loss function measures the difference between the predicted and actual values.

Step-4: Compute the gradient of the loss with respect to the weights and biases using the chain rule of calculus. Update weights and biases in the opposite direction of the gradient to minimize the loss.

Step-5: Repeat steps 2-4 for a specified number of epochs or until convergence.

Step-6: Use the trained model to make predictions on new, unseen data. Evaluate the model's

Input:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Output:

Sample predictions:

```
[[0.28386715 0.31935492 0.3967779 ]
 [0.5104412  0.1897936  0.29976526]
 [0.20244241 0.29193234 0.50562525]
 [0.30980957 0.32851517 0.3616752 ]
 [0.45372778 0.20556031 0.34071186]]
```

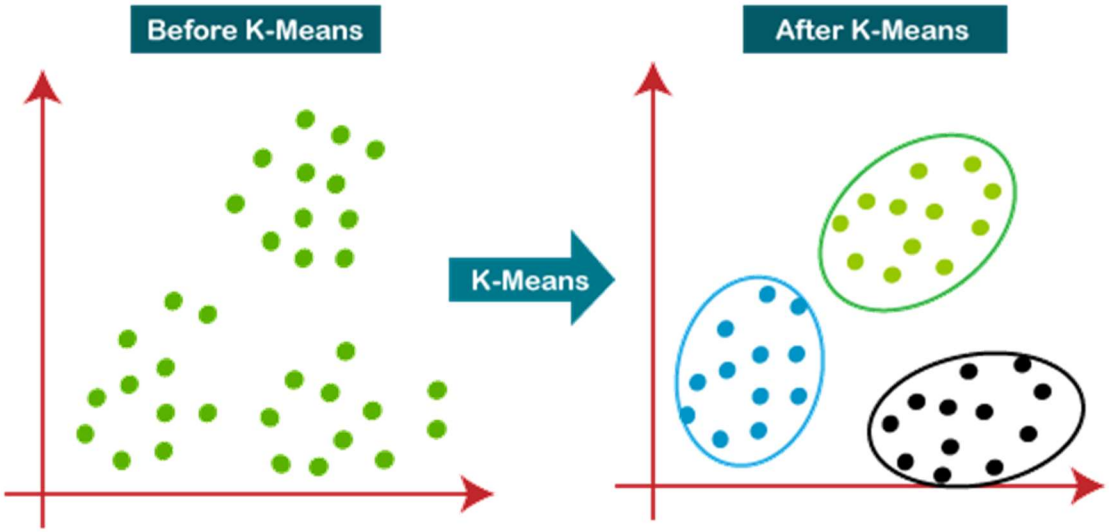
Discussion: We have successfully run the program and gained insights that contribute to a deeper understanding of neural networks and lay the foundation for continued exploration and refinement in the field of artificial intelligence.

Experiment No:08

Experiment Name: Write a Python program to develop a k-means clustering model with 3 means and test it with a dataset.

Objective: The main objective is to develop a k-means clustering model with 3 means and test it with a dataset using Python language.

Theory: K means is one of the most popular Unsupervised Machine Learning Algorithms Used for Solving Classification Problems in data science and is very important if you are aiming for a data scientist role. K Means segregates the unlabeled data into various groups, called clusters, based on having similar features and common patterns. This tutorial will teach you the definition and applications of clustering, focusing on the K means clustering algorithm and its implementation in Python. It will also tell you how to choose the optimum number of clusters for a dataset.



Requirements:

1. Programming Language
2. Machine Learning Library
3. Dataset
4. Data Exploration
5. Data Preprocessing
6. K-Means Clustering Algorithm
7. Model Training
8. Cluster Assignment
9. Results Visualization

Algorithm:

Step-1: Choose the number of clusters (k = 3). Randomly initialize three cluster centroids in the feature space.
Step-2: For each data point in the dataset, calculate the Euclidean distance to each of the three centroids. Assign each data point to the cluster associated with the nearest centroid.
Step-3: Recalculate the centroids of the three clusters as the mean of the data points assigned to each cluster.
Step-4: Repeat steps 2 and 3 until convergence. Convergence can be determined by checking if the centroids no longer change significantly or after a predefined number of iterations.
Step-5: Once convergence is achieved, each data point is associated with one of the three clusters.
Step-6: The model is ready.

Input:

	status_type	num_reactions	...	num_hahas	num_sads	num_angrys
0	video	529	...	1	1	0
1	photo	150	...	0	0	0
2	video	227	...	1	0	0
3	photo	111	...	0	0	0
4	photo	213	...	0	0	0

Output:
Accuracy score: 0.43

Discussion: We have successfully run the program and assume that it has functions for random selection, finding the nearest centroid, and calculating the mean of a set of points.

Experiment No: 09

Experiment Name: Write a Python program to implement naïve baye's theorem to classify the English text whether it exist a positive or negative message.

Objective: The main objective is to implement naïve baye's theorem to classify the English text whether it exist a positive or negative message.

Theory: The Naive Bayes Algorithm is a classification technique based on Bayes' Theorem with an independence assumption among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. The Naïve Bayes classifier is a popular supervised machine learning algorithm used for classification tasks such as text classification. It belongs to the family of generative learning algorithms, which means that it models the distribution of inputs for a given class or category. This approach is based on the assumption that the features of the input data are conditionally independent given the class, allowing the algorithm to make predictions quickly and accurately.
In statistics, naive Bayes classifiers are considered as simple probabilistic classifiers that apply Bayes' theorem. This theorem is based on the probability of a hypothesis, given the data and some prior knowledge. The naive Bayes classifier assumes that all features in the input data are independent of each other, which is often not true in real-world scenarios. However, despite this simplifying assumption, the naive Bayes classifier is widely used because of its efficiency and good performance in many real-world applications.

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.The formula for Bayes' theorem is given as:

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

Here,
P(A|B) is Posterior probability
P(B|A) is Likelihood probability
P(A) is Prior Probability
P(B) is Marginal Probability

Requirements:

1. Problem Statement
2. Dataset
3. Data Preprocessing
4. Feature Extraction
5. Naive Bayes' Theorem
6. Model Training
7. Model Evaluation

Algorithm:

Step-1: Importing the Libraries
Step-2: Loading the Dataset
Step-3: Splitting the dataset into the Training set and Test set
Step-4: Feature Scaling
Step-5: Training the Naive Bayes Classification model on the Training Set
Step-6: Predicting the Test set results
Step-7: Confusion Matrix and Accuracy.

Input:

	headline		is_sarcastic
0	former versace store clerk sues over secret 'b	...	0
1	the 'roseanne' revival catches up to our thorn	...	0
2	mom starting to fear son's web series closest	...	1
3	boehner just wants wife to listen, not come up	...	1
4	j.k. rowling wishes snape happy birthday in th...		0

Output:

Accuracy score: 0.8448637316561844

Discussion: We have successfully run the program and assume the Naïve Bayes' approach to sentiment analysis offers a reliable and interpretable solution. It demonstrates effectiveness in distinguishing positive from negative messages in English text.