# **Task 2: Understand and explain what you analyzed in the code. Make a detailed analysis.**

Initially, NumPy and matplotlib libraries are imported. Then, datasets and train\_test\_split are imported from the sklearn library.

After the datasets are imported, data is classified using the below methods.

# Ridge Regression:

Ridge Regression is a method used to analyze multicollinear data.

In the code, initially “make.regression” function is used to generate a random regression problem. Then, train\_test\_split is used to split the data into random training and test set.

Then, scatter() is used to plot a scatter plot of training data and the plot is displayed using the show() function.

Chart, scatter chart

Description automatically generated

Here, we train the model and predict outputs. The RidgeRegressor function is imported from the ridge\_regressor file. The fit function is called using the training set data. Then the output is predicted using the predict() function. Finally, the scatter plot is marked using test data and predicted data and displayed using the show() function.

Graphical user interface, chart

Description automatically generated

Below is the logic for RegressionRegressor Function.

A picture containing text

Description automatically generated

# Gaussian process:

The Gaussian regression calculates the probability distribution over all admissible functions that fit the data

Initially, time steps are initialized that are later used for creating training and test datasets using the train\_test\_split function, which splits the training and test data randomly. Then, a Scatter plot is displayed when plotted with training data.

Graphical user interface

Description automatically generated

Here, we train the model and predict outputs. The GaussianRegressor function is imported from the gaussian\_regressor file. The fit function is called using the training set data. Then the output is predicted using the predict() function. Finally, the scatter plot is marked using test data and predicted data and displayed using the show () function.

Graphical user interface

Description automatically generated

Below is the logic for GaussianRegressor Function.

Text

Description automatically generated

# SVM:

Note: Support vector machines (SVMs) are supervised machine learning algorithms for outlier detection, regression, and classification that are both powerful and adaptable.

Initially, data set load\_wine() is loaded into the variable and later used for creating training and test datasets using the train\_test\_split function, which splits the training and test data randomly.

Now, we train the model and predict outputs. The SVM function is imported from the svm file. The fit function is called using the training set data. Then the output is predicted using the predict() function. Finally, test data is evaluated using test data to generate accuracy.

Graphical user interface

Description automatically generated with medium confidence

Below is the logic for SVM Function.

Graphical user interface, text, application, email

Description automatically generated

# Decision tree:

**Note: Decision Trees** are a non-parametric supervised learning method used for [classification](https://scikit-learn.org/stable/modules/tree.html#tree-classification) and [regression](https://scikit-learn.org/stable/modules/tree.html#tree-regression). The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Initially, data set load\_diabetes() is loaded into the variable and later used for creating training and test datasets using the train\_test\_split function, which splits the training and test data randomly.

Now, we train the model and predict outputs. The DecisionTree function is imported from the decision\_tree file. The fit function is called using the training set data. Then the output is predicted using the predict() function. Finally, test data is evaluated using test data to generate accuracy.

Graphical user interface, text, application, email

Description automatically generated

Below is the logic for Decision Tree.

# Naive Bayes:

Note:  It is useful for making predictions and forecasting data based on historical results

Initially, data set fetch\_20newsgroups with categories 'comp.graphics' and 'sci.med'

are loaded into the dataset variable and later used for creating training and test datasets using the train\_test\_split function, which splits the training and test data randomly.

Now, we train the model and predict outputs. The NaiveBayes function is imported from the naïve\_bayes file. The fit function is called using the training set data. Then the output is predicted using the predict() function. Finally, test data is evaluated using test data to generate accuracy.

Graphical user interface, text, application

Description automatically generated

Below is the code for naive Bayes.

Text

Description automatically generated

# Gaussian Bayes:

Note: Gaussian Naïve Bayes is used when we assume all the continuous variables associated with each feature to be distributed according to Gaussian Distribution.

Initially, data set load\_wine() is loaded into the variable and later used for creating training and test datasets using the train\_test\_split function, which splits the training and test data randomly.

Now, we train the model and predict outputs. The GaussianBayes function is imported from the gaussian\_bayes file. The fit function is called using the training set data. Then the output is predicted using the predict() function. Finally, test data is evaluated using test data to generate accuracy.

Graphical user interface, text, application, email

Description automatically generated

Below is the code for the Gaussian Bayes function.

Graphical user interface, text

Description automatically generated

# k-mean

Note: The K-means clustering algorithm is used to find groups which have not been explicitly labeled in the data.

Initially, make\_blobs method in datasets is used to generate sample data. A scatter plot is marked using the data.

Graphical user interface, application

Description automatically generated

Now, we train the model and predict outputs. The KMEAN() function is imported from the k\_mean file. The fit function is called using the data with cluster number as 3.

Graphical user interface, application, Word

Description automatically generated

Below is the KMEAN class with methods in it.

Text

Description automatically generated

# Gaussian mixture model:

Note: A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters

Initially, make\_blobs method in datasets is used to generate sample data. A scatter plot is marked using the data,

Chart, scatter chart

Description automatically generated

Now, we train the model and predict outputs. The GMM() function is imported from the k\_mean file. The fit function is called using the data with cluster number as 3.

A picture containing scatter chart

Description automatically generated

Below is the code for GMM function.

Text

Description automatically generated

# Basic Artificial neural network:

An artificial neural network is divided into layers, different layers perform different transformations on their inputs. Signals travel from the input layer to the output layer after traversing the layers multiple times.

Here, data set load\_diabetes() is loaded into the variable and later used for creating training and test datasets using the train\_test\_split function, which splits the training and test data randomly.

Now, we train the model and predict outputs. The MLPClassifier(Multi-layer Perceptron classifier) function is imported from the Sklearn.neural\_networks library . The fit function is called using the training set data. Then the training and test values are predicted using the predict() function. Finally, test and training data are evaluated using the accuracy\_score function.

Note: Accuracy score calculates the accuracy score for a set of predicted labels against the true labels

Text

Description automatically generated

Now a confusion matrix and classification report are generated using the training set.

Confusion matrix shows a number of correct and incorrect predictions made by a classifier.

Graphical user interface

Description automatically generated

Table

Description automatically generated with medium confidence