**1.Linear Regression:** Several modifications were made to enhance its functionality. The dataset was pre-processed to handle missing values and ensure data consistency. The specific changes made to the original algorithm include handling missing values using mean imputation, adding histograms and scatter plots for better data visualization, implementing feature scaling to normalize input variables, and optimizing model training by tuning the train-test split ratio. A new feature selection step was implemented to improve model performance by analysing feature importance. Additional performance evaluation metrics, such as R-squared and adjusted R-squared, were included to provide a more comprehensive analysis of the model’s effectiveness. A new section was added to compare the actual and predicted values using a residual plot to assess model accuracy visually.

**2.Decision tree:** I implemented a Decision Tree model using DecisionTreeClassifier from sklearn.tree and assessed its performance through accuracy, a classification report, and a confusion matrix. The tree structure was visualized using plot\_tree, and I explicitly set criterion='entropy' to examine its impact on decision-making, aiming to enhance classification effectiveness. To manage model complexity, I restricted the tree depth to five (max\_depth=5) and set a minimum sample requirement per node (min\_samples\_split=10). Additionally, I applied pruning techniques to remove unnecessary branches, improving efficiency. Visualization was enhanced using Matplotlib, and feature importance was analyzed through feature\_importances\_ to improve model interpretability. To provide a more in-depth performance evaluation, I incorporated additional metrics such as precision, recall, and F1-score alongside accuracy, ensuring a more comprehensive assessment.

**3.Logistic Regression:** I applied LogisticRegression from sklearn.linear\_model and assessed the model's performance using metrics like accuracy, confusion matrix, and classification report. The dataset was divided into training and testing subsets using train\_test\_split. For visualization, I utilized seaborn and matplotlib, and standardized the data with StandardScaler from sklearn.preprocessing to enhance numerical stability and possibly improve classification results.The dataset underwent pre-processing to address missing values and ensure consistency. The notebook also incorporated libraries like pandas, numpy, matplotlib, and seaborn, along with additional performance evaluation tools such as accuracy\_score and confusion\_matrix. Furthermore, I ensured the dataset was properly cleaned and pre-processed to handle any inconsistencies, and I utilized various evaluation metrics to analyze model performance more thoroughly.

**4.KNN**:My notebook uses a blood donation dataset, while the example notebook uses heart disease dataset and included standard data preprocessing steps, such as loading the dataset, checking for missing values, and performing exploratory data analysis. My notebook employs GridSearchCV for hyperparameter tuning, while example notebookevaluates different k values using cross\_val\_score in a loop. Model evaluation in my notebook relies on accuracy and a classification report, while the otheruses cross-validation scores and tests predictions with specific patient examples. Visualization included in my notebook for better understanding of data.

**5.Support vector Machine:** My code likely works with a milk quality dataset and included standard data preprocessing steps, such as loading the dataset, checking for missing values, and performing exploratory data analysis. In terms of data preprocessing, the example code manually splits the dataset using a custom function (df\_split()), while I use a standard approach like train\_test\_split(). For model evaluation, the example code employs cross-validation (cross\_val\_score()) and computes accuracy both with and without normalization using make\_scorer(). In contrast, my notebook likely evaluates performance using accuracy, precision, recall, and F1-score.

**6. K-Means:** My notebook follows standard data preprocessing steps, including dataset loading, handling missing values, and exploratory data analysis. While both my implementation and the example K-Means script follow similar steps, certain differences could influence the final clustering results. In my approach, cluster centers are determined using the mean of assigned points, aligning with the traditional K-Means method. Various visualization techniques are used to enhance result interpretation. Unlike the example script, which seems to execute a single iteration, my notebook runs multiple iterations, potentially leading to more refined clusters.

**7.Bayes Algorithm:** The Bayes notebook includes standard data preprocessing steps, such as loading the dataset, checking for missing values, and performing exploratory data analysis. Additionally, while my notebook evaluates model performance using accuracy and classification reports, the example notebook includes more advanced evaluation metrics, such as ROC curves, AUC scores, and confusion matrices. Finally I added a some visualization methods for better understanding of data.

**8.Random Forest:** In my notebook, I have imported some additional libraries like matplotlib and seaborn for data visualization, as well as sklearn. metrics for evaluating model performance using accuracy score, classification reports, and confusion matrices. My RandomForestClassifier setup includes custom parameters, such as max\_depth=10 and n\_estimators=200, while the original code relied on default settings.My notebook also enhances data exploration by including .describe() for statistical summaries and generating correlation heatmaps, which were not part of the provided code.My notebook visualized feature importance using a horizontal bar plot with matplotlib. For model evaluation, I included accuracy\_score from sklearn.metrics, whereas the example code relied solely on pd.crosstab to compare predictions with actual labels.