**1. K-Nearest Neighbors (KNN) Algorithm**

**Steps:**

1. **Start:** Import necessary libraries (e.g., numpy, pandas, sklearn).
2. Load the dataset and preprocess it (e.g., handle missing values, normalization).
3. Split the dataset into training and test sets.
4. Choose the number of neighbors kkk.
5. For each test instance:
   * Calculate the distance between the test instance and all training instances (e.g., Euclidean distance).
   * Identify the kkk closest neighbors.
   * Assign the label based on the majority label among the kkk neighbors.
6. Evaluate the model using a suitable metric (e.g., accuracy, precision).
7. **Stop:** Return the evaluation results.

**2. K-Nearest Neighbors for Diabetes Prediction**

**Steps:**

1. **Start:** Import necessary libraries and the diabetes dataset.
2. Clean and preprocess data (e.g., fill missing values, standardize features).
3. Divide the data into training and test sets.
4. Choose an appropriate kkk value for the model.
5. For each test sample:
   * Compute the distance to each training sample.
   * Select the kkk nearest training samples.
   * Assign the most common label among the kkk samples as the prediction.
6. Assess model accuracy and adjust kkk if necessary.
7. **Stop:** Return model accuracy and predictions.

**3. Decision Tree Algorithm**

**Steps:**

1. **Start:** Import necessary libraries (e.g., sklearn.tree, pandas).
2. Load the dataset and preprocess (e.g., convert categorical variables, handle missing data).
3. Split the data into training and testing sets.
4. Initialize the decision tree classifier and configure hyperparameters.
5. Train the decision tree model on the training data.
6. Use the trained model to predict labels for the test set.
7. Evaluate model performance using metrics such as accuracy or F1-score.
8. **Stop:** Output performance metrics and model structure if desired.

**4. Naive Bayes Classifier**

**Steps:**

1. **Start:** Import necessary libraries (e.g., sklearn.naive\_bayes, pandas).
2. Load and preprocess the dataset (e.g., handling missing values, encoding categorical variables).
3. Divide data into training and test sets.
4. Initialize the Naive Bayes classifier (e.g., GaussianNB for continuous data).
5. Train the classifier with the training data.
6. Make predictions on the test data.
7. Evaluate the classifier using performance metrics (e.g., accuracy, recall).
8. **Stop:** Display results and model insights.

**5. Linear Regression Algorithm**

**Steps:**

1. **Start:** Import necessary libraries (e.g., numpy, pandas, sklearn.linear\_model).
2. Load and preprocess the dataset (e.g., handle missing values, feature scaling).
3. Split the data into training and testing sets.
4. Initialize the linear regression model.
5. Train the model on the training set.
6. Use the trained model to make predictions on the test set.
7. Evaluate the model using regression metrics (e.g., RMSE, R2R^2R2).
8. **Stop:** Output evaluation results and coefficients.

**6. K-Means Clustering**

**Steps:**

1. **Start:** Import required libraries (e.g., sklearn.cluster, numpy, pandas).
2. Load the dataset and preprocess it (e.g., scale features).
3. Select the number of clusters kkk.
4. Initialize kkk cluster centroids randomly.
5. For each instance:
   * Calculate the distance to each centroid.
   * Assign the instance to the nearest cluster.
6. Recalculate the centroids for each cluster.
7. Repeat steps 5–6 until centroids stabilize (convergence).
8. **Stop:** Output the cluster assignments and centroids.