1. Predict the price of a house by applying linear regression using a real estate dataset:

- **Explanation:**

- Linear regression is used to model the relationship between a dependent variable (house price) and one or more independent variables (features like bedrooms, square footage, location).

- The goal is to find the best-fitting line (regression line) that minimizes the difference between the actual house prices and the predicted prices based on the features.

**Packages used in the code:**

mlbench: For accessing the BostonHousing dataset, which is one of the machine learning benchmark problems.

dplyr: For data manipulation tasks.

ggplot2: For creating powerful graphics and plots.

reshape2: For transforming data between wide and long formats.

caret: For splitting the data into training and testing sets.

**Step-by-step process in 5 steps:**

Load the BostonHousing dataset and perform basic data exploration using packages mlbench, dplyr, and ggplot2.

Split the data into training and testing sets (75% training, 25% testing) using the caret package.

Fit a linear regression model to predict the "medv" (Median Value) house price based on selected variables like "crim," "rm," "tax," and "lstat" in the training set.

Evaluate the model's performance by obtaining the R-squared value and checking the assumption that the mean of residuals is zero.

Visualize the model's predictions against the original values using ggplot2, comparing the predicted versus original house prices.

2. Apply k-nearest neighbor algorithm to classify and analyze the ionosphere data:

- **Explanation:**

- The k-nearest neighbor (k-NN) algorithm is a simple classification algorithm based on the distance between data points.

- For each new data point, k-NN calculates the distance to all existing data points and assigns the majority class among its k-nearest neighbors as its predicted class.

**Packages used in the code:**

KernelKnn: For implementing the Kernel k-nearest neighbor algorithm.

dplyr: For data manipulation tasks.

ggplot2: For creating powerful graphics and plots.

**Step-by-step process in 5 steps:**

Load the "ionosphere" dataset from the "KernelKnn" package and perform basic data manipulation using packages mlbench and dplyr.

Scale the data to ensure output depends on distance calculations.

Split the data into training and testing sets using random sampling (75% training, 25% testing).

Implement Kernel k-nearest neighbor algorithm using the "KernelKnn" package with different kernel functions (tricube and self-defined normal distribution) and various parameters (k and distance metric).

Evaluate the accuracy of the models using 5-fold cross-validation and calculate the mean accuracy for each set of parameters.

3. Classify messages as spam and ham using the naïve Bayes algorithm on an SMS dataset:

**- Explanation:**

- Naïve Bayes is a probabilistic algorithm based on Bayes' theorem, which calculates the probability of an event based on prior knowledge.

- Naïve Bayes assumes that the presence of each word in the SMS is independent of the presence of other words, making the calculations simpler.

**Packages:-**

quanteda: For text analysis, creating a document-feature matrix, and building the Naive Bayes classifier.

RColorBrewer: For generating color palettes for word clouds.

**Step-by-step process in 5 steps:**

Load the "spam.csv" dataset, read it into R, and prepare the data by renaming columns and shuffling the dataset.

Create a corpus of text messages using the "quanteda" package and attach class labels (spam/ham) to the corpus.

Preprocess the text data by removing punctuation, numbers, and stopwords, and convert it into a document-feature matrix (dfm).

Visualize the most frequent words in spam and ham messages using word clouds with different color palettes.

Split the data into training and testing sets, create a document-feature matrix for both sets, and train the Naive Bayes classifier on the training data. Evaluate the classifier's performance on the test data and calculate the accuracy of the predictions.

4. Implement logistic regression algorithm on the iris flower dataset to classify the flower into different types:

- **Explanation**:

- Logistic regression is used for binary or multi-class classification problems.

- In this example, we have multiple classes (types of iris flowers), so it's a multi-class classification problem.

**Packages:**

The code installs the required packages "caret" and "tidyr" using the install.packages() function.

**Step-by-step process in 5 steps:**

Load and View Data: The code loads the built-in Iris dataset into the R environment and views its structure and top few rows using help(iris) and View(iris\_dataset) respectively.

Rename Columns: The code renames the columns of the dataset to meaningful names using the colnames() function.

Data Partition: The code uses the "caret" package to create a partition of the data into training and testing sets. It randomly selects 80% of the data for training and 20% for testing.

Data Analysis and Visualization: The code performs various data analysis and visualization tasks using ggplot2 and ggthemes packages. It creates histograms, box plots, scatter plots, and facet plots to understand the distribution and relationships between different features of the Iris flowers based on their species.