1st assessment

1. Explain how to import these types of data in R using base R functions:

a. Comma Separated Values text file

To import a CSV file in R using base R functions, you can use the read.csv() function. Here's an example:

```
data <- read.csv(file_path)
```

b. Excel Data File

To import an Excel file, you need to use an additional package because base R does not support Excel files directly. A commonly used package is readxl. Here's how to do it:

```
install.packages("readxl")
# Load the readxl package
library(readxl)
# Read the Excel file
data <- read_excel(file_path)
```

c. SPSS Data File

To import an SPSS file, you need to use an additional package as well. The foreign package is commonly used for this purpose. Here's how to do it:

```
install.packages("foreign")
# Load the foreign package
library(foreign)
# Read the SPSS file
data <- read.spss(file_path, to.data.frame = TRUE)
```

2. Explain how you can do sub-setting with codes in R software:

a. Define the 6 x 5 matrix and select the last two rows:

```
mat <- matrix(1:30, nrow = 6, ncol = 5)
print(mat)
# Select the last two rows
last_two_rows <- mat[5:6, ]
print(last_two_rows)
```

b. Select third and fifth row with second and fourth column:

Select the third and fifth rows with the second and fourth columns

```
subset_matrix <- mat[c(3, 5), c(2, 4)]
print(subset_matrix)
```

c. Add 3 new rows in this matrix:

To add new rows to the matrix, we can use the rbind() function.

```
new_rows <- matrix(c(31:45), nrow = 3, ncol = 5)
# Add the new rows to the original matrix
```

```
mat_expanded <- rbind(mat, new_rows)
```

3. Explain differences of these terms with examples using R codes:

a. **Arrays and matrices:**
Arrays can have more than two dimensions.
Matrices have exactly two dimensions.

b. **List and factors:**
Lists can contain elements of different types.
Factors are used for categorical data and store both values and levels.

```
list_example <- list(Name = "Alice", Age = 25, Scores = c(90, 85, 88))
factor_example <- factor(c("High", "Medium", "Low", "Medium", "High"))
```

c. **Data frame and tibble:**
Data frames are the standard way to store tables in R.
Tibbles are a modern version of data frames with better printing, no string-to-factor conversion by default, and more consistent handling of column types.

```
df_example <- data.frame(Name = c("Alice", "Bob"), Age = c(25, 30), Gender = c("F", "M"))
tibble_example <- tibble(Name = c("Alice", "Bob"), Age = c(25, 30), Gender = c("F", "M"))
```

4. Explain the following concept of working efficiently with “big data” in R software:

a. **Sample and model:**
Use a representative sample of the data to build and test models, reducing memory and computation requirements.

```
data(iris)
set.seed(123)
sampled_data <- iris[sample(1:nrow(iris), 50), ]
model <- lm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width, data = sampled_data)
summary(model)
```

b. **Chunk and pull:**
Process large datasets in smaller chunks rather than loading the entire dataset into memory, using packages like data.table or dplyr.

```
library(data.table)
file_path <- "path/to/large_file.csv"
process_chunk <- function(chunk) {
        summary(chunk)
  }
fread(file_path, chunk.size = 1000, fun = process_chunk)
```

c. **Push compute to data:**

Perform computations closer to where the data is stored, often in a database, to minimize data movement and leverage the database's computational power.

```
library(dplyr)
library(DBI)
con <- dbConnect(RSQLite::SQLite(), ":memory:")
copy_to(con, iris, "iris")
result <- tbl(con, "iris") %>%
          group_by(Species) %>%
           summarize(mean_sepal_length = mean(Sepal.Length))
dbDisconnect(con)
```

5. **Explain different types of pipe operators with R codes and examples:**

a. **Compound assignment operator:**

The compound assignment operator (%<>%) is used to pipe a value through a series of functions and assign the result back to the original variable. This operator is provided by the magrittr package.

Example:

```
library(magrittr)
x <- 1:5
x %<>% sum()
```

b. **Tee Operator:**

The tee operator (%T>%) is used when you want to pipe a value through a series of functions but also return the original value. This can be useful for performing side effects (e.g., plotting) without altering the original value. This operator is also provided by the magrittr package.

Example:

```
library(magrittr)
x <- 1:5
result <- x %T>%
  print() %>%
  sum()
```

c. **Exposition Operator:**

The tee operator (%T>%) is used when you want to pipe a value through a series of functions but also return the original value. This can be useful for performing side effects (e.g., plotting) without altering the original value. This operator is also provided by the magrittr package.

Example:

```
library(magrittr)
x <- 1:5
result <- x %T>%
  print() %>%
  sum()
```

1st Reassessment

1. **Explain how can you import the following type of data into the R software with simple examples and code:**

   a. **a text file saved in computer**

   You can use the read.table() or read.csv() function to import text files. Here's an example using read.table():

   ```
   data <- read.table(file_path, header = TRUE, sep = "\t")
   ```

   b. **a table embedded in any page**

   You can use the rvest package to scrape tables from web pages. Here's an example:

   ```
   library(rvest)
   url <- "https://example.com/page-with-table"
   page <- read_html(url)
   table <- page %>%
     html_node("table") %>%  # Select the first table on the page
     html_table()
   ```

   c. **json file with web API**

   You can use the jsonlite package to read JSON data from a web API. Here's an example:

   ```
   library(jsonlite)
   url <- "https://api.example.com/data.json"
   json_data <- fromJSON(url)
   ```

2. **Explain following data types in R with the examples and R codes:**

   a. **Numeric and integer**

   Numeric: This type is used for decimal numbers. By default, numbers in R are treated as numeric (double precision floating-point).

   Examples:

   ```
   num <- 3.14
   ```

   Integer: This type is used for whole numbers. Integers are denoted by an "L" suffix.

   ```
   # Integer example
   int <- 42L
   ```

b. **Categorical and factor**

Categorical data represents discrete categories or groups. In R, categorical data is often stored as factors.

Factors are used to handle categorical data. They store both the values and the corresponding levels.

```
category <- factor(c("low", "medium", "high", "medium", "low"))
print(class(category))
print(levels(category))
```

c. **Data and Date as well as time**

Date objects store calendar dates.

```
date_example <- as.Date("2024-06-25")
```

Date-time objects store both dates and times. The POSIXct and POSIXlt classes are used for date-time data.

```
datetime_example <- as.POSIXct("2024-06-25 14:30:00")
```

3. Explain data mining in data science with focus and examples on:

a. **Descriptive mining**

Descriptive Mining: Summarizes and describes the main features of a dataset (e.g., clustering). It often involves methods such as clustering, association rules, and summarization.

Example:

```
library(datasets)
library(cluster)
data(iris)
set.seed(123)
kmeans_result <- kmeans(iris[, 1:4], centers = 3)
iris$Cluster <- kmeans_result$cluster
```

b. **Predictive mining**

Uses historical data to make predictions about future events (e.g., regression analysis).

Techniques include regression, classification, and time-series analysis.

Example:

```
library(datasets)
data(mtcars)
model <- lm(mpg ~ hp, data = mtcars)
summary(model)
predict(model, data.frame(hp = 150))
```

c. **Prescriptive mining**
Prescriptive mining provides recommendations for decision-making based on the analysis of data. e.g., optimization
Example:

```
library(lpSolve)
costs <- c(2, 3, 4)
constraints <- matrix(c(1, 1, 1, 2, 1, 0, 0, 1, 1), nrow = 3, byrow = TRUE)
rhs <- c(100, 150, 75)
directions <- c("<=", "<=", "<=")
lp_result <- lp("min", costs, constraints, directions, rhs)
lp_result$solution
```

4. Explain how to work efficiently with "big data" in R software in relation to the:

a. **Sub setting with base R and dplyr packages.**
Base R provides basic subsetting functions that can be used on large datasets.

```
large_df <- data.frame(
  ID = 1:1000000,
  Value = rnorm(1000000)
)
subset_base <- large_df [ large_df$Value > 0, ]
```

The dplyr package provides a more efficient and readable way to subset and manipulate large datasets.

```
library(dplyr)
subset_dplyr <- large_df %>%
  filter(Value > 0) %>%
  select(ID, Value)
```

b. **ff, ffbase and ffbase2 packages**
These packages provide tools to handle datasets larger than memory by storing data on disk but allowing access and manipulation as if they were in memory.

```
large_ff <- ff(rnorm(1000000))
subset_ff <- large_ff[large_ff > 0]
head(as.vector(subset_ff))
```

ffbase and ffbase2:

```
library(ffbase)
large_ffdf <- as.ffdf(large_df)
```

```
subset_ffdf <- large_ffdf[large_ffdf$Value > 0, ]
```

c. **data.table package**

The data.table package provides a high-performance version of data frames with syntax and functionality similar to dplyr, but optimized for speed and memory efficiency.

```
library(data.table)
large_dt <- as.data.table(large_df)
subset_dt <- large_dt[Value > 0]
```

5. Explain social network analysis and describe its use in real-life situation with:

a. **Nodes**

Nodes (also called vertices) represent the entities in the network. In a social network, nodes could be individuals, organizations, or any other discrete entities.

Example:

```
nodes <- data.frame(
  id = 1:5,
  name = c("Alice", "Bob", "Charlie", "David", "Eve")
)
```

b. **Links**

Links (also called edges) represent the relationships or interactions between the nodes. Links can be directed (indicating a direction of relationship) or undirected (indicating a mutual relationship).

Example:

```
links <- data.frame(
  source = c(1, 1, 2, 3, 4, 5),
  target = c(2, 3, 4, 5, 5, 1),
  type = c("friend", "colleague", "family", "friend", "colleague", "friend")
)
```

c. **Attributes**

Attributes are properties or characteristics of nodes (or sometimes links). For nodes, attributes could be things like age, gender, job position, etc. For links, attributes could include the type of relationship, the strength of the connection, and so on.

Example:

In a social network of friends, attributes for nodes could include age and gender, while attributes for links could include the type of relationship.

```
nodes <- data.frame(
  id = 1:5,
  name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  age = c(25, 30, 35, 28, 22),
  gender = c("F", "M", "M", "M", "F")
)
```

2nd Assessment

1. **Describe Data Visualization with focus on :**
    a. **Concept of grammer of graphics with Wilkinson's approach**
    A framework for creating data visualizations by breaking down graphs into semantic components, as proposed by Leland Wilkinson.

    b. **Layers in grammer of graphics with ggplot packages approach**
    ggplot2 uses a layered approach to build plots, including data, aesthetics, geometries, statistics, scales, coordinates, facets, and themes.
    Example with ggplot2:
```
library(ggplot2)
data(mtcars)
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  geom_smooth(method = "lm") +
  scale_x_continuous(name = "Weight") +
  scale_y_continuous(name = "Miles per Gallon") +
  coord_cartesian() +
  theme_minimal()
```

    c. **Statistical transformations in grammar of graphics**
    Involve summarizing or transforming data in meaningful ways before plotting, such as smoothing, counting, and aggregating.
```
library(ggplot2)
data(diamonds)
ggplot(diamonds, aes(x = carat)) +
  geom_histogram(binwidth = 0.2) +
  scale_x_continuous(name = "Carat") +
  scale_y_continuous(name = "Count") +
  theme_minimal()
```

2. **Describe following for checking fit of the multiple linear regression model**
   a. **Outliers**
   Points with unusually large residuals; they can be identified by standardized residuals.
```
library(ggplot2)
model <- lm(mpg ~ wt + hp + qsec, data = mtcars)
residuals <- rstandard(model)
outliers <- which(abs(residuals) > 3)
```

   b. **Cook's distance**
   Combines residuals and leverage to measure the influence of each observation on the regression coefficients; values greater than 1 or 4/n indicate influential points.
```
cooks_dist <- cooks.distance(model)
influential_points <- which(cooks_dist > (4 / nrow(mtcars)))
```

   c. **Levarage**
   Measures how far an observation's predictor values are from the average; high leverage points have values greater than 2(p+1)/n.
   Example:
```
leverage_values <- hatvalues(model)
high_leverage_points <- which(leverage_values > (2 * (length(coef(model)) - 1) /
nrow(mtcars)))
```

3. **Describe supervised lesrning classification regression model with focus on:**
   a. **Model fit indices**
   Metrics to evaluate model performance. For classification models: accuracy, precision, recall, and F1 score. For regression models: R-squared, MAE, MSE, and RMSE.

   b. **Confusion matrix with an example**
   A table showing actual vs. predicted classifications. It helps calculate accuracy, precision, recall, and F1 score.
```
library(caret)
set.seed(123)
data(iris)
iris <- iris[iris$Species != "setosa",]
iris$Species <- factor(iris$Species)
trainIndex <- createDataPartition(iris$Species, p = 0.8, list = FALSE)
trainData <- iris[trainIndex,]
testData <- iris[-trainIndex,]
```

```
model <- train(Species ~ ., data = trainData, method = "glm", family = "binomial")
predictions <- predict(model, testData)
confMatrix <- confusionMatrix(predictions, testData$Species)
print(confMatrix)
```

c. **Prediction accuracy with ROC curve**
   A plot of the true positive rate against the false positive rate. The AUC (area under the curve) measures the model's discriminatory ability.

```
library(pROC)
probabilities <- predict(model, testData, type = "prob")[,2]
rocCurve <- roc(testData$Species, probabilities)
plot(rocCurve, main = "ROC Curve")
auc <- auc(rocCurve)
print(auc)
```

4. **Describe following with example on it use:**

a. **Poisson regression**
   Used for modeling count data assuming the data follows a Poisson distribution.
   Example: Number of calls received by a call center per hour.

b. **Zero-inflated Poisson regression**
   Used for count data with excess zeros, combining a Poisson count model with a logit model for predicting excess zeros.
   Example: Number of insurance claims with many zero claims.

c. **Negative binomial regression**
   Used for count data with overdispersion, generalizing the Poisson regression by adding a parameter for overdispersion.
   Example: Number of hospital visits with overdispersed data.
   These models help in dealing with different types of count data scenarios, ensuring more accurate and appropriate modeling.

5. **Describe supervised linear regression model with focus on:**

a. **Cross validation**
   Used to estimate model performance by dividing the data into training and testing sets multiple times. LOOCV is a specific form where each observation is used as a test set exactly once.

```
library(boot)
model <- lm(mpg ~ wt + hp, data = mtcars)
loocv_result <- cv.glm(mtcars, model, K = nrow(mtcars))
```

b. **K-fold cross validation**
The dataset is split into k folds, and the model is trained on k−1 folds and tested on the remaining fold. This process is repeated k times.
library(caret)
train_control <- trainControl(method = "cv", number = 5)
model_cv <- train(mpg ~ wt + hp, data = mtcars, method = "lm", trControl = train_control)

c. **Repeated k-fold cross validation**
Extends K-fold cross-validation by repeating the process multiple times, providing a more robust estimate of model performance.
These techniques help in assessing the reliability and generalizability of supervised linear regression models, ensuring they perform well on unseen data.
library(caret)
train_control_repeated <- trainControl(method = "repeatedcv", number = 5, repeats = 10)
model_repeated_cv <- train(mpg ~ wt + hp, data = mtcars, method = "lm", trControl = train_control_repeated)

2nd Reassessment

1. Compare these two concept of grammer of graphics with example for:

I. Wilkinson's approach and ggplot2 package's approach

Wilkinson's approach provides a theoretical framework for constructing visualizations by breaking them down into their fundamental components.

ggplot2 implements these principles in a practical, user-friendly way, allowing for the creation of complex plots through a layering system.

```
data <- data.frame(x = rnorm(100), y = rnorm(100))
ggplot(data, aes(x = x, y = y)) +
  geom_point() +              # Geometry layer
  theme_minimal()
```

II. Cartesian and polar coordinates in ggplot2 package

Cartesian Coordinates use perpendicular axes and are the default in ggplot2.
library(ggplot2)
data <- data.frame(x = rnorm(100), y = rnorm(100))

```
ggplot(data, aes(x = x, y = y)) +
  geom_point() +
  theme_minimal()
```

Polar Coordinates transform the plot into a circular system, useful for pie charts and radial plots.

```
data <- data.frame(
  category = factor(c("A", "B", "C", "D")),
  value = c(3, 7, 9, 1)
)
p <- ggplot(data, aes(x = category, y = value)) +
  geom_bar(stat = "identity") +
  theme_minimal()
```

2. Describe the followings residual siagnosis for linear regression model:

- **Hat values**

  Measure the influence of each observation on the fitted values. High leverage points have hat values much larger than the average, indicating they have a significant impact on the model. In R, hat values can be calculated using the hatvalues() function.

  ```
  library(ggplot2)
  model <- lm(mpg ~ wt + hp, data = mtcars)
  hat_values <- hatvalues(model)
  ```

- **Cook's distance**

  Combines the information from residuals and leverage to measure the influence of each observation on the regression coefficients. High Cook's distance values (typically greater than 1) indicate influential points that could unduly influence the model. In R, Cook's distance can be calculated using the cooks.distance() function.

  ```
  cooks_dist <- cooks.distance(model)
  ```

3. Describe supervised learning classification regression model with:

- **Confusion matrix, accuracy and misclassification error with example**

  Confusion Matrix is a table that shows the actual vs. predicted classifications and helps calculate metrics like accuracy and misclassification error.
  Accuracy is the proportion of correctly classified instances.
  Misclassification Error is the proportion of incorrectly classified instances.

- **Area under curve as accuracy with ROC curve with the same example**
  ROC curve plot of the true positive rate against the false positive rate at various thresholds.
  AUC (Area Under the Curve) is a scalar value representing the model's ability to discriminate between positive and negative classes, with higher values indicating better performance

4. Compare the two concepts following with the clear example:

a. Simple linear correlation and simple linear regression

Simple linear correlation is the correlation measures the strength and direction of the linear relationship between two variables. The correlation coefficient r ranges from -1 to 1.

```
data <- data.frame(
  x = rnorm(100),
  y = rnorm(100)
)
correlation <- cor(data$x, data$y)
```

Simple Linear Regression models the relationship between two variables by fitting a linear equation to the observed data. The equation is of the form $y=\beta0 + \beta1x + \epsilon$, where: y is the dependent variable.
x is the independent variable.

```
model <- lm(y ~ x, data = data)
summary(model)
ggplot(data, aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Simple Linear Regression")
```

b. BLUE and LINE test

BLUE (Best Linear Unbiased Estimator) refers to the properties of the Ordinary Least Squares (OLS) estimator in linear regression.
Best: Minimum variance among all unbiased estimators.
Linear: Linear function of the dependent variable.
Unbiased: Expected value of the estimator equals the true parameter value.

```
model <- lm(y ~ x, data = data)
```

LINE Test (Linear, Independent, Normally Distributed, Equal Variance) refers to the assumptions that need to be satisfied for the OLS estimators to be BLUE:

Linearity: The relationship between the predictors and the outcome is linear.
Independence: Observations are independent of each other.
Normality: The residuals of the model are normally distributed.
Equal Variance (Homoscedasticity): The variance of the residuals is constant across all levels of the independent variable.

```
plot(model, which = 1)
plot(model, which = 2)
library(car)
durbinWatsonTest(model)
```

5. Compare supervised linear regression models with focus on:

a. K-nearest neighbourhood and decision tree models

K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm. It makes predictions based on the k closest training examples in the feature space.

```
library(class)
library(ggplot2)
data(iris)
set.seed(123)
trainIndex <- sample(1:nrow(iris), 0.8*nrow(iris))
trainData <- iris[trainIndex, ]
testData <- iris[-trainIndex, ]
knn_model <- knn(train = trainData[, -5], test = testData[, -5], cl = trainData$Species, k = 3)
```

Decision Trees are tree-structured models that split the data into subsets based on the value of input features. Each internal node represents a decision based on a feature, and each leaf node represents the output.

```
library(rpart)
library(caret)
tree_model <- rpart(Species ~ ., data = trainData)
tree_predictions <- predict(tree_model, testData, type = "class")
```

b. Support vector machine and ANN MLP models

Support Vector Machines (SVM) are powerful classifiers that find the hyperplane that best separates the classes in the feature space. SVMs can also be used for regression (SVR).

```
library(e1071)
library(caret)
svm_model <- svm(Species ~ ., data = trainData, kernel = "linear")
svm_predictions <- predict(svm_model, testData)
```

Artificial Neural Networks (ANN) with Multilayer Perceptrons (MLP) are computational models inspired by biological neural networks. They consist of input, hidden, and output layers of interconnected nodes (neurons).

```
library(nnet)
library(caret)
ann_model <- nnet(Species ~ ., data = trainData, size = 5, decay = 0.1, maxit = 200)
ann_predictions <- predict(ann_model, testData, type = "class")
```

#2080

1. Explain these terms with examples for R:

a. Getting multi-way table with array

A multi-way table can be created using the array function in R. This is useful for representing data that have more than two dimensions.

Example in R:

```
data <- array(1:24, dim = c(3, 4, 2))
```

b. Creating class intervals of continuous variable

Class intervals can be created using the cut function, which converts a continuous variable into a factor by dividing the range of the variable into intervals.

Example in R:

```
continuous_var <- rnorm(100, mean = 50, sd = 10)
class_intervals <- cut(continuous_var, breaks = 5)
table(class_intervals)
```

c. Missingness vs. nothingness

Missingness refers to data that is missing or not recorded.

Nothingness could be interpreted as data that is intentionally left blank or zero, which is different from missing data.

Example in R:

```
data <- data.frame(
  x = c(1, 2, NA, 4, 5),
  y = c(NA, 2, 3, NA, 5)
)
missing_values <- is.na(data)
```

2. Explain the following concepts with focus on R software:

a. Raw data

Raw data is unprocessed data directly collected from a source. It often needs cleaning before analysis.

Example in R:# Load raw data (example)

```
raw_data <- data.frame(
  id = 1:5,
  name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  score = c(85, 90, 78, 88, 95)
)
```

b. Data wrangling

Data wrangling is the process of cleaning and transforming raw data into a format suitable for analysis.

Example in R:

```
library(dplyr)
cleaned_data <- raw_data %>%
  filter(score > 80) %>%
  select(id, name, score) %>%
  mutate(grade = ifelse(score > 90, "A", "B"))
```

c. Tidy data

Tidy data is a standard way of organizing data that makes it easier to analyze. Each variable is a column, each observation is a row, and each type of observational unit is a table.

Example in R:

```
tidy_data <- data.frame(
  id = 1:3,
  name = c("Alice", "Bob", "Charlie"),
  age = c(25, 30, 35),
  score = c(85, 90, 78)
)
```

3. Explain the followings with examples for R:

a. Reference range based on mean

A reference range can be calculated using the mean and standard deviation.

Example in R:

```
mean_score <- mean(raw_data$score)
sd_score <- sd(raw_data$score)
```

```
reference_range <- c(mean_score - 2 * sd_score, mean_score + 2 * sd_score)
```

b. Reference range based on median

A reference range can be calculated using the median and the interquartile range (IQR).

Example in R:

```
median_score <- median(raw_data$score)
iqr_score <- IQR(raw_data$score)
reference_range_median <- c(median_score - 1.5 * iqr_score, median_score + 1.5 * iqr_score)
```

c. Outliers and extreme values

Outliers are observations that fall far from the central value of a dataset. They can be identified using statistical methods such as the IQR method.

Example in R:

```
q1 <- quantile(raw_data$score, 0.25)
q3 <- quantile(raw_data$score, 0.75)
iqr <- q3 - q1
outliers <- raw_data$score[raw_data$score < (q1 - 1.5 * iqr) | raw_data$score > (q3 + 1.5 * iqr)]
```

4. Explain the following concepts with focus on R software:

a. Test of normality

Tests of normality, such as the Shapiro-Wilk test, can determine whether data follows a normal distribution.

Example in R:

```
shapiro_test <- shapiro.test(raw_data$score)
```

b. Parametric tests

Parametric tests assume that data follows a certain distribution (usually normal). Examples include t-tests and ANOVA.

Example in R:

```
t_test <- t.test(score ~ as.factor(name), data = raw_data)
```

c. Residual analysis

Residual analysis involves checking the residuals of a regression model to ensure they meet assumptions such as normality and homoscedasticity.

Example in R:

```
model <- lm(score ~ age, data = tidy_data)
```

5. **Describe decision tree classification model with focus on:**
   a. **Bagging**
   Bagging (Bootstrap Aggregating) involves training multiple models on different subsets of the data and averaging their predictions.
   Example in R:
   library(randomForest)
   bagging_model <- randomForest(Species ~ ., data = iris, mtry = 4)

   b. **Improved bagging**
   Improved bagging could involve techniques such as Random Forest, which improves on bagging by selecting a random subset of features for each split.
   Example in R:
   random_forest_model <- randomForest(Species ~ ., data = iris)

   c. **Boosting**
   Boosting involves training multiple models sequentially, where each model tries to correct the errors of the previous one.
   Example in R:
   library(gbm)
   boosting_model <- gbm(Species ~ ., data = iris, distribution = "multinomial", n.trees = 100)

#2078

1. **Describe the following concepts with focus on R software:**
   a. **Loops**
   Loops are used to iterate over a sequence of elements or to repeat a block of code multiple times.
   Example in R:

```
for(i in 1:5) {
  print(i)
}
i <- 1
while(i <= 5) {
  print(i)
  i <- i + 1
}
```

b. Function

A function is a block of code designed to perform a particular task, which can be reused.

Example in R:

```
my_function <- function(x, y) {
  return(x + y)
}
result <- my_function(3, 5)
```

c. Pipe

The pipe operator %>% from the dplyr package is used to chain together multiple operations.

Example in R:

```
library(dplyr)
data <- data.frame(x = 1:5, y = c(3, 2, 5, 1, 4))
result <- data %>%
  filter(x > 2) %>%
  mutate(z = x * y)
```

2. Explain the following concepts with examples focusing on R software:

a. Big data

Big data refers to large and complex datasets that are difficult to process using traditional data processing techniques. In R, packages like data.table and sparklyr are used to handle big data.

Example in R:

```
library(data.table)
big_data <- data.table(id = 1:1e6, value = rnorm(1e6))
```

b. Data wrangling

Data wrangling involves cleaning and transforming raw data into a format suitable for analysis.

Example in R:

```
library(dplyr)
data <- data.frame(name = c("Alice", "Bob", "Charlie"), score = c(85, NA, 78))
cleaned_data <- data %>%
  mutate(score = ifelse(is.na(score), mean(score, na.rm = TRUE), score)) %>%
  filter(score > 80)
```

c. Tidy data

Tidy data is a standardized way of organizing data where each variable is a column, each observation is a row, and each type of observational unit is a table.
Example in R:
```
library(tidyr)
data <- data.frame(
  id = 1:2,
  name_score = c("Alice:85", "Bob:78")
)
tidy_data <- separate(data, name_score, into = c("name", "score"), sep = ":")
```

3. Explain the following concept with examples focusing on R software:

a. Measures of central tendency

Measures of central tendency include the mean, median, and mode.
```
data <- c(1, 2, 2, 3, 4, 5, 5, 5, 6)
mean_value <- mean(data)
median_value <- median(data)
mode_value <- as.numeric(names(sort(table(data), decreasing = TRUE)[1]))
```

b. Measures of dispersion

Measures of dispersion include range, variance, and standard deviation.
Example in R:
```
range_value <- range(data)
variance_value <- var(data)
sd_value <- sd(data)
```

c. Measures of relative position

Measures of relative position include percentiles, quartiles, and z-scores.
Example in R:
```
quartiles <- quantile(data)
z_scores <- scale(data)
```

4. Explain the following concepts with examples focusing on R software:

a. Correlation

Correlation measures the strength and direction of the linear relationship between two variables.
Example in R:
```
data <- data.frame(x = rnorm(100), y = rnorm(100))
correlation <- cor(data$x, data$y)
```

b. Parametric tests

Parametric tests assume the data follows a certain distribution, usually normal. Examples include t-tests and ANOVA.
Example in R:
```
t_test <- t.test(data$x, data$y)
```

c. **Non-parametric tests**
Non-parametric tests do not assume a specific distribution for the data. Examples include the Wilcoxon rank-sum test and the Kruskal-Wallis test.
Example in R:
```
wilcox_test <- wilcox.test(data$x, data$y)
```

5. Compare the following models with focus on R software:

a. **Naive Bayes and Support Vector Machine**
Naive Bayes is a probabilistic classifier based on Bayes' theorem, assuming independence between predictors.
Example in R:
```
library(e1071)
library(caret)
naive_bayes_model <- naiveBayes(Species ~ ., data = iris)
nb_predictions <- predict(naive_bayes_model, iris)
```

Support Vector Machine (SVM) is a classifier that finds the hyperplane that best separates the classes in the feature space.
```
svm_model <- svm(Species ~ ., data = iris)
svm_predictions <- predict(svm_model, iris)
print(confusionMatrix(svm_predictions, iris$Species))
```

b. **Decision Tree and Random Forest**
Decision Tree is a model that splits the data into subsets based on the value of input features.
Random Forest is an ensemble method that uses multiple decision trees to improve predictive performance.
Example in R:
```
library(rpart)
library(randomForest)

# Decision Tree
tree_model <- rpart(Species ~ ., data = iris)
tree_predictions <- predict(tree_model, iris, type = "class")
print(confusionMatrix(tree_predictions, iris$Species))
```

```
# Random Forest
rf_model <- randomForest(Species ~ ., data = iris)
rf_predictions <- predict(rf_model, iris)
print(confusionMatrix(rf_predictions, iris$Species))
```

c. **Feed-forward and feed-backward neural network**

Feed-forward Neural Network is a type of ANN where connections between nodes do not form cycles. Data moves in one direction, from input to output.

Feed-backward Neural Network is also known as Recurrent Neural Network (RNN), where connections between nodes can form cycles, allowing data to flow in both directions.

Example in R:

```
library(nnet)

# Feed-forward Neural Network
ffnn_model <- nnet(Species ~ ., data = iris, size = 5, decay = 0.1, maxit = 200)
ffnn_predictions <- predict(ffnn_model, iris, type = "class")
```