Data Frames and Basic Data Pre-processing

```
# Read data from CSV file

students_df <- read.csv("students.csv")

# View the structure of the data frame

str(students_df)

# Load necessary package for reading JSON files

library(jsonlite)

# Read data from JSON file

students_df_json <- fromJSON("students.json")

# Convert to data frame

students_df_json <- as.data.frame(students_df_json)

# View the structure of the data frame

str(students_df_json)
```

```
# Handling missing values (replace missing values with NA)

students_df[students_df == ""] <- NA


# Handling outliers (remove outliers in a specific column, for example, 'age')

students_df <- students_df[!students_df$age > 100, ]
```

```
# Filtering data (filter students who have passed the exam)

passed_students <- subset(students_df, exam_score >= 60)
```

```
# Sorting data (sort students by their age in descending order)

sorted_students <- students_df[order(students_df$age, decreasing = TRUE), ]


# Grouping data (calculate average exam score by gender)

avg_exam_score_by_gender <- aggregate(exam_score ~ gender, data = students_df, FUN = mean)
```

Feature Scaling and Dummification

```
# Load necessary libraries

library(dplyr)
```

```
# Sample dataset with numerical and categorical features
data <- data.frame(
  numerical1 = c(10, 20, 30, 40),
  numerical2 = c(0.5, 0.7, 0.9, 1.2),
  category = c("A", "B", "A", "C")
)
# Display the dataset
print(data)
# Standardization (z-score normalization)
scaled_data <- data %>%
  mutate(
    numerical1_scaled = scale(numerical1),
    numerical2_scaled = scale(numerical2)
  )
# Display the scaled data
print(scaled_data)
```

```
# Normalization (min-max scaling)
normalized_data <- data %>%
  mutate(
    numerical1_normalized = (numerical1 - min(numerical1)) / (max(numerical1) - min(numerical1)),
    numerical2_normalized = (numerical2 - min(numerical2)) / (max(numerical2) - min(numerical2))
  )
# Display the normalized data
print(normalized_data)


# Perform feature dummification using one-hot encoding
dummy_data <- data %>%
  mutate(
    category_A = ifelse(category == "A", 1, 0),
    category_B = ifelse(category == "B", 1, 0),
    category_C = ifelse(category == "C", 1, 0)
  )
# Display the dummy data
print(dummy_data)
```

Hypothesis Testing

```
# Example data (replace with your actual data)
group_A <- c(80, 85, 90, 95, 100)
group_B <- c(75, 80, 85, 90, 95)
# Perform independent samples t-test
t_test_result <- t.test(group_A, group_B)
# Print the test result
print(t_test_result)
# Extract p-value from the test result
p_value <- t_test_result$p.value
# Set significance level
alpha <- 0.05
# Compare p-value with significance level
if (p_value < alpha) {
  print("Reject Null Hypothesis: There is a significant difference in the mean exam scores between Group A and Group B.")
} else {
  print("Fail to Reject Null Hypothesis: There is no significant difference in the mean exam scores between Group A and Group B.")
}
```

ANOVA (Analysis of Variance)

```
# Example data (replace with your actual data)
data <- data.frame(
  group = rep(c("A", "B", "C"), each = 20),
  measurement = c(rnorm(20, mean = 10, sd = 2),
                  rnorm(20, mean = 12, sd = 2),
                  rnorm(20, mean = 15, sd = 2))
)
# Perform one-way ANOVA
anova_result <- aov(measurement ~ group, data = data)
# Print ANOVA summary
print(summary(anova_result))
# Perform Tukey's HSD test
tukey_result <- TukeyHSD(anova_result)
print(tukey_result)
```

Regression and Its Types

```
# Load necessary libraries
library(ggplot2)
# Example dataset (replace with your actual data)
data <- data.frame(
  X = c(1, 2, 3, 4, 5),
  Y = c(2, 4, 5, 4, 5)
)
# Perform simple linear regression
lm_model <- lm(Y ~ X, data = data)
# Print regression summary
summary(lm_model)
```

```
# Example dataset for multiple linear regression (replace with your actual data)
data <- data.frame(
  X1 = c(1, 2, 3, 4, 5),
  X2 = c(2, 3, 4, 5, 6),
  Y = c(2, 4, 5, 4, 5)
)
# Perform multiple linear regression
mlm_model <- lm(Y ~ X1 + X2, data = data)
# Print regression summary
summary(mlm_model)
```

Logistic Regression and Decision Tree

```
# Load necessary libraries

library(caret)

# Example dataset (replace with your actual data)

data <- read.csv("your_dataset.csv")

# Split data into training and testing sets

set.seed(123)

train_index <- createDataPartition(data$target_variable, p = 0.7, list = FALSE)

train_data <- data[train_index, ]

test_data <- data[-train_index, ]

# Build logistic regression model

logistic_model <- glm(target_variable ~ ., data = train_data, family = binomial)

# Print model summary

summary(logistic_model)

# Make predictions on the test data

predictions <- predict(logistic_model, newdata = test_data, type = "response")

predicted_classes <- ifelse(predictions > 0.5, 1, 0)


# Calculate classification metrics

confusion_matrix <- confusionMatrix(table(predicted_classes, test_data$target_variable))

print(confusion_matrix)


# Build decision tree model

tree_model <- rpart(target_variable ~ ., data = train_data, method = "class")

# Print decision tree

print(tree_model)


# Visualize decision tree

plot(tree_model)

text(tree_model)
```

K-Means Clustering

```
# Load necessary libraries

library(cluster)

library(ggplot2)

# Example dataset (replace with your actual data)

data <- read.csv("your_dataset.csv")

# Remove any rows with missing values if necessary

data <- na.omit(data)

# Select relevant columns for clustering

# For example, if you have numerical features columns 2 to 4:

selected_data <- data[, 2:4]


# Calculate total within-cluster sum of squares for different values of k

wss <- numeric(10)

for (i in 1:10) {

  wss[i] <- sum(kmeans(selected_data, centers = i)$withinss)

}

# Plot the elbow curve
```

```
plot(1:10, wss, type = "b", xlab = "Number of Clusters (k)", ylab = "Within-cluster sum of squares")


# Perform silhouette analysis for different values of k

sil_width <- c(NA)

for (i in 2:10) {

  kmeans_fit <- kmeans(selected_data, centers = i)

  sil_width[i] <- silhouette(kmeans_fit$cluster, dist(selected_data))$avg.width

}

# Plot silhouette widths

plot(2:10, sil_width[-1], type = "b", xlab = "Number of Clusters (k)", ylab = "Average Silhouette Width")


# Set the optimal number of clusters based on the elbow method or silhouette analysis

k_optimal <- 3  # Update with the chosen value of k


# Apply K-Means algorithm

kmeans_result <- kmeans(selected_data, centers = k_optimal)


# Add cluster assignment to the dataset

data$cluster <- as.factor(kmeans_result$cluster)


# Visualize clustering results (for example, if you have 2-dimensional data)

ggplot(data, aes(x = feature1, y = feature2, color = cluster)) +

  geom_point() +

  labs(title = "K-Means Clustering Results",

       x = "Feature 1",

       y = "Feature 2") +

  theme_minimal()


# Analyze cluster characteristics

aggregate(selected_data, by = list(data$cluster), FUN = mean)
```

Principal Component Analysis (PCA)

```
# Load necessary libraries

library(FactoMineR)

library(FactoMineR)

# Example dataset (replace with your actual data)

data <- read.csv("your_dataset.csv")

# Remove any rows with missing values if necessary

data <- na.omit(data)

# Select relevant columns for PCA

# For example, if you have numerical features columns 2 to 4:

selected_data <- data[, 2:4]


# Perform PCA

pca_result <- PCA(selected_data, graph = FALSE)
```

Plot the scree plot to visualize explained variance

plot(pca_result$eig, type = "b")

Select the appropriate number of principal components based on the plot

For example, you can visually inspect the scree plot and choose the number of components

Get the coordinates of individuals in the reduced-dimensional space

individuals <- as.data.frame(pca_resultindcoord)

Plot the data in the reduced-dimensional space

plot(individuals[, 1], individuals[, 2], pch = 19, col = "blue", xlab = "Principal Component 1", ylab = "Principal Component 2")

Creating meaningful visualizations and storytelling with data involves selecting appropriate visualizations to represent the data effectively, combining multiple visualizations to convey insights, and presenting findings in a clear and concise manner. Let's walk through each step:

Step 1: Select Appropriate Visualizations

Identify the type of data you have (e.g., categorical, numerical, time-series) and choose visualizations that best represent the relationships and patterns in the data. Some common types of visualizations include:

- Scatter plots
- Bar charts
- Line charts
- Histograms
- Pie charts
- Box plots

Step 2: Combine Multiple Visualizations

Combine multiple visualizations to tell a compelling data story and provide a comprehensive view of the data. You can use a combination of different types of visualizations to highlight different aspects of the data and uncover insights. For example, you can use a line chart to show trends over time and a scatter plot to explore relationships between variables.

Step 3: Present Findings and Insights

Present the findings and insights in a clear and concise manner, focusing on the key takeaways from the data analysis. Use descriptive titles and labels to explain the visualizations, and provide context and interpretation for the audience to understand the significance of the findings. Summarize the main insights and conclusions drawn from the data analysis.

Example:

Let's say you have a dataset containing sales data for different products over time. You can create visualizations such as:

1. A line chart showing the trend in total sales over time.

2. A bar chart comparing sales of different products.

3. A scatter plot exploring the relationship between sales and marketing expenditure.

You can then combine these visualizations to tell a data story, starting with an overview of total sales trends, diving into specific product sales, and examining the impact of marketing expenditure on sales. Finally, present the insights gained from the analysis, such as identifying top-selling products and assessing the effectiveness of marketing strategies.

Conclusion:

Creating meaningful visualizations and storytelling with data involves selecting appropriate visualizations, combining multiple visualizations to tell a compelling story, and presenting findings and insights in a clear and concise manner. By following these steps, you can effectively communicate the findings from your data analysis and drive informed decision-making.