Data Mining and Business Intelligence

1. *Load the dataset. Display its structure and identify the types of variables (e.g., numerical or categorical). Generate summary statistics (e.g., mean, median, standard deviation, and frequency counts) for all variables. Provide an interpretation of key insights from the summary statistics, including distributions, outliers, or notable trends. Document your observations and any preprocessing actions taken (e.g., handling missing values).*

setwd("C:/Users/alexa/OneDrive/Desktop/Data Mining/Finals")

*# Data Cleaning and Duplicate Checking*

# Check for duplicates

print("Duplicate Rows")

duplicates <- data[duplicated(data) | duplicated(data, fromLast = TRUE), ]

print(duplicates)

# Count of duplicates

duplicate\_count <- sum(duplicated(data) | duplicated(data, fromLast = TRUE))

print(paste("Total number of duplicate rows:", duplicate\_count))

A computer screen shot of a computer code

Description automatically generated

Comment: Checked if there are any duplicate rows in the data set, there is no duplicate rows.

# Check for missing values

print("Missing Values")

print(sapply(data, function(x) sum(is.na(x))))

**A close-up of a computer code

Description automatically generated**

Comment: There are no missing values in the dataset

# Display the structure of the dataset

str(data)

**A screenshot of a computer screen

Description automatically generated**

# Generate summary statistics

summary(data)

**A screen shot of a computer

Description automatically generated**

Comments: From the summary, you can notice for the Age column the highest is 69 and the lowest is 19, city and education have character variables, in Product Purchase it’s 0 or 1 depicting if the customer purchased the product or not.

table(data$City\_Type)

A screenshot of a computer

Description automatically generated

table(data$Education\_Level)

A screenshot of a computer

Description automatically generated

table(data$Product\_Purchase)

A close-up of a product

Description automatically generated

Comment: The table() function is used to summarize categorical data by displaying the frequency of each unique value in the specified columns.

For the City\_Type, you can see there are three categories(Rural, Suburban, Urban) and their occurrences(21, 24, 55). For the Education Level, you can see there are four categories(Bachelor’s, High School, Masters’, Phd) and their occurrences(41,41,15,3). For the Product\_Purchase, you can see there are three categories(0,1) and their occurrences(61,39).

1. *Create at least two data visualizations (e.g., histograms, box plots, scatter plots) to explore relationships and distributions within the dataset. Discuss any patterns, trends, or anomalies observed in the visualizations.*

#Histogram of Spending\_Score

library(ggplot2)

ggplot(data, aes(x = Spending\_Score)) +

geom\_histogram(binwidth = 5, fill = "#F4C2C2", color = "#89CFF0") +

labs(title = "Distribution of Spending Score", x = "Spending Score", y = "Frequency")

Comments: This histogram visualizes the distribution of Spending\_Score. The histogram likely shows a concentration of spending scores around certain ranges, indicating that most customers exhibit average spending behavior.

A graph of a distribution of spending score

Description automatically generated

#Boxplot of Income by City\_Type

ggplot(data, aes(x = City\_Type, y = Income)) +

geom\_boxplot(fill = "#F4C2C2", color = "#89CFF0") +

labs(title = "Income by City Type", x = "City Type", y = "Income")

Comment: This boxplot displays the distribution of Income for each category of City\_Type. Larger interquartile ranges and more outliers in specific city types suggest that income variation is higher in those regions, likely reflecting diverse economic demographics.

A graph with a number of squares

Description automatically generated with medium confidence

#Age vs. Spending Score by City Type

ggplot(data, aes(x = Age, y = Spending\_Score, color = City\_Type)) +

geom\_point(alpha = 0.7) +

labs(title = "Age vs. Spending Score by City Type", x = "Age", y = "Spending Score") +

theme(legend.title = element\_text(size = 10))

Comment: Younger individuals generally have higher spending scores, particularly in urban areas, suggesting a trend where younger demographics in cities may be more inclined toward higher spending.

A graph with different colored dots

Description automatically generated

*3. Implement a K-Nearest Neighbors model to predict the Product\_Purchase*

*column using the features Age, Income, Spending\_Score, and City\_Type.*

*Evaluate the model’s performance using appropriate metrics (e.g., accuracy, confusion*

*matrix) and report the accuracy when k=5. Reflect on the strengths and limitations of the model based on the results.*

library(caret)

# Set seed for reproducibility

set.seed(3111) #splitting the data produce the same results each time you run the code

# Prepare data for KNN

# Convert City\_Type to numeric

data$City\_Type\_Numeric <- as.numeric(factor(data$City\_Type))

# Select features

X <- data[, c("Age", "Income", "Spending\_Score", "City\_Type\_Numeric")]

y <- data$Product\_Purchase

Comment:

x: Contains the independent variables (features) to be used for training the KNN model.

y: The dependent variable (target), Product\_Purchase, which the model will predict.

# Split data using createDataPartition

train\_index <- createDataPartition(y, p = 0.7, list = FALSE)

X\_train <- X[train\_index, ] # Independent variables for training

X\_test <- X[-train\_index, ] #Target variable for training

y\_train <- y[train\_index] # Independent variables for testing

y\_test <- y[-train\_index] #Target variable for evaluation

# Train KNN model

knn\_model <- train(

x = X\_train,

y = as.factor(y\_train),

method = "knn",

trControl = trainControl(method = "cv", number = 5),

tuneGrid = data.frame(k = 5)

)

# Predict and evaluate

y\_pred <- predict(knn\_model, X\_test)

cm <- confusionMatrix(as.factor(y\_pred), as.factor(y\_test))

print(cm)

A screenshot of a computer

Description automatically generated

# Print model details

print(knn\_model)

Comments:

* + **True Positives (TP)**: 8 (Predicted 0 when true value is 0).
  + **False Positives (FP)**: 11 (Predicted 1 when true value is 0).
  + **False Negatives (FN)**: 4 (Predicted 0 when true value is 1).
  + **True Negatives (TN)**: 7 (Predicted 1 when true value is 1).

A high p-value (> 0.05) means the model's performance is not statistically better than random guessing.

Sensitivity: The model correctly identifies 42.11% of actual 0s. This low sensitivity suggests many 0s are misclassified as 1.

Specificity: The model correctly identifies 63.64% of actual 1s. This is slightly better than sensitivity but still not high.

Mcnemar's Test P-Value : 0.1213

A p-value > 0.05 indicates no significant difference, meaning the model is equally likely to make errors in both classes.

The model's performance is poor, with an accuracy of 50%, which is worse than the No Information Rate of 63.33%.

*A screenshot of a computer code

Description automatically generated*

Comment: The model's **accuracy of 58.67%** is slightly better than random guessing but worse than the No Information Rate (63.33%) observed in the earlier confusion matrix analysis.

The low **Kappa (0.10)** suggests poor agreement between predictions and true labels, highlighting issues with class imbalance or model limitations.

**Strengths**

1. Simplicity and Interpretability:
2. No Assumptions about Data Distribution:
3. Cross-Validation for Robustness:

**Limitations**

1. **Poor Accuracy and Low Predictive Power**:
   * The model's accuracy (58.67%) is only slightly better than random guessing, and the Kappa statistic (0.10) suggests poor agreement between predictions and actual labels.
   * The **No Information Rate (63.33%)** from the earlier confusion matrix indicates that a naive model predicting the majority class (0) would outperform the k-NN model in this case.
2. **Sensitivity to Feature Scaling**:
   * Since no pre-processing (e.g., normalization) was applied, the model may be biased by features with larger ranges (e.g., Income), leading to suboptimal performance. k-NN's reliance on Euclidean distance makes it highly sensitive to feature magnitudes.
3. **Class Imbalance**:
   * The earlier confusion matrix revealed an imbalance in the target classes (0 and 1). k-NN struggles with imbalanced datasets, often favoring the majority class, as seen here with lower sensitivity (42.11%) for class 0.
4. **Fixed k Value**:
   * The model was trained with a fixed k = 5, without exploring other values. A suboptimal choice of k can negatively impact the performance of the model.
5. **Scalability**:
   * k-NN becomes computationally expensive as the dataset grows because it requires calculating distances to all points in the training data for each prediction. While this is not a significant issue for small datasets (like this one), it can be a limitation for larger datasets.
6. **Low Robustness to Noise**:
   * k-NN is sensitive to noise in the data (e.g., outliers in predictors or mislabeled classes). This could partially explain the poor accuracy and low Kappa.
7. *Build a linear regression model to predict Income using the features Age, Spending\_Score, and City\_Type. Report the R-squared value of the model and provide a detailed interpretation of this statistic. Identify any additional metrics (e.g., Mean Squared Error) that you would use to evaluate the model’s performance and discuss their implications.*

#Linear regression model

linear\_model <- lm(Income ~ Age + Spending\_Score + City\_Type, data = data)

summary(linear\_model)

A screenshot of a computer

Description automatically generated

Comment: This statistic indicates that **89.66% of the variance in Income** can be explained by the features (Age, Spending\_Score, and City\_Type) included in the model.

A high R-squared value suggests that the model fits the data well. However, it doesn't imply causation, and it’s important to validate the model on unseen data to check for overfitting.

This value adjusts the R-squared for the number of predictors in the model. It’s slightly lower but still high, indicating the features are relevant and adding value without over-complicating the model.

The linear regression model shows strong explanatory power with a high R-squared value but has some limitations, including wide residual variability and non-significant predictors. Further evaluation using RMSE, residual diagnostics, and validation will provide a more robust understanding of the model's performance.

# Calculate Mean Squared Error

mse <- mean(residuals(linear\_model)^2)

mse

A number on a white background

Description automatically generated

Comment: MSE's purpose is to quantify the average squared difference between the predicted and actual values. A lower MSE indicates better model performance.

A useful metric for comparing model performance but difficult to interpret due to its squared units.

A black and white screen with text

Description automatically generated

plot(residuals(linear\_model))

abline(h = 0, col = "red")

A graph with numbers and dots

Description automatically generated

Comment: Approximately **89.66% of the variability in Income** is explained by the predictors Age, Spending\_Score, and City\_Type. This suggests a strong fit; however, R-squared alone does not assess predictive accuracy or detect overfitting. The Mean Squared Error (MSE) of **27,886,790** and the Root Mean Squared Error (RMSE) of **5280.79** reveal the model’s average prediction error in dollars, highlighting that predictions deviate by approximately $5280. While these metrics show reasonable performance, residual analysis and cross-validation are recommended to confirm the model's robustness and ensure generalizability to unseen data.

1. *Create a classification tree to predict Product\_Purchase using all other variables in the dataset as predictors. Visualize the tree and identify key decision splits (e.g., What is the top split? What variables are involved in significant splits?). Summarize the tree’s decision-making process and evaluate its performance using appropriate metrics.*

library(tree)

library(rpart)

library(rpart.plot)

# Prepare data for classification tree

data$City\_Type\_Numeric <- as.numeric(factor(data$City\_Type))

data$Education\_Level\_Numeric <- as.numeric(factor(data$Education\_Level))

# Build classification tree

tree\_model <- tree(factor(Product\_Purchase) ~ Age + Income + Spending\_Score +

City\_Type\_Numeric + Education\_Level\_Numeric, data = data)

# Plot the tree

plot(tree\_model)

text(tree\_model)

A diagram of a family tree

Description automatically generated

# Print tree details

summary(tree\_model)

A computer screen with numbers and symbols

Description automatically generated

# Evaluate tree performance

tree\_pred <- predict(tree\_model, type = "class")

table(Predicted = tree\_pred, Actual = data$Product\_Purchase)

A black and white screen with blue text

Description automatically generated

Comment: The classification tree is constructed to predict Product\_Purchase using all other variables (Age, Income, Spending\_Score, City\_Type\_Numeric, and Education\_Level\_Numeric) as predictors.

The tree utilized three key variables—Income, Spending\_Score, and Age—indicating their importance in decision-making. The model created 13 terminal nodes, reflecting its complexity in classifying the dataset.

The residual mean deviance (0.9566) suggests reasonable fit, and the misclassification error rate (20%) shows that 20 out of 100 predictions were incorrect. Evaluating the tree's performance using a confusion matrix revealed an accuracy of **80%**, with **54 true negatives** and **26 true positives**, though it misclassified **13 actual positives** and **7 actual negatives**, highlighting some limitations in sensitivity and specificity.

The tree effectively prioritizes splits based on Income and Spending\_Score, suggesting these variables play a significant role in predicting purchases.

1. *Provide a summary of your overall workflow, including data preparation, model development, and key findings. Highlight any actionable insights derived from the analysis or recommendations based on your results.*

**Summary of Workflow**

**1. Data Preparation**

The dataset was examined for structure and quality, with categorical variables such as City\_Type and Education\_Level converted to numeric formats to support modeling. Missing values were addressed, and numerical predictors were scaled where necessary to ensure compatibility with distance-based algorithms like k-NN. The data was partitioned into training and testing sets to evaluate model performance effectively.

**2. Model Development**

**k-Nearest Neighbors (k-NN)**: A k-NN model was developed to predict Product\_Purchase using Age, Income, Spending\_Score, and City\_Type\_Numeric. The model achieved modest accuracy (58.67%), indicating limitations due to unscaled features and class imbalance.

**Linear Regression**: A regression model was built to predict Income using Age, Spending\_Score, and City\_Type. The model demonstrated a strong fit (R-squared = 89.66%) but had a Root Mean Squared Error (RMSE) of $5280.79, suggesting variability in predictions.

**Classification Tree**: A decision tree was constructed to classify Product\_Purchase using all predictors, achieving an 80% accuracy rate. The tree highlighted the importance of Income, Spending\_Score, and Age in predicting purchase behavior.

**3. Key Findings**

**Income and Spending Patterns**: Income and Spending\_Score emerged as critical predictors of purchasing behavior, indicating that customers with higher spending tendencies and incomes are more likely to make purchases.

**Age Relevance**: Older individuals tended to have higher incomes, as reflected in the linear regression model.

**Model Limitations**: Both the k-NN model and the classification tree struggled with class imbalance, leading to some misclassifications.

**4. Actionable Insights and Recommendations**

**Target High-Income and High-Spending Segments**: Focus marketing efforts on individuals with higher incomes and spending scores, as they are more likely to purchase products.

**Refine Data for Better Modeling**: Address class imbalance through oversampling or adjusting class weights in models to improve sensitivity to minority classes.

**Explore Additional Features**: Incorporate other predictors like spending categories or customer demographics for enhanced model performance.

**Validate on New Data**: Test the models on unseen data to confirm their generalizability and identify opportunities for improvement.

By leveraging these insights, businesses can optimize strategies to enhance customer targeting and product offerings, ultimately improving revenue and engagement.

Reference:

1. <https://claude.ai/chat/31cf40a9-613e-4e7c-abea-71f3cd5efa4f>
2. https://chatgpt.com/c/675b0fd0-dc74-800f-8877-ff76fa7e0df6