PREDICTIVE ANALYSIS PROJECT REPORT

Implementation of ML Algorithms

Using

R Programming

Submitted by

Hardik

Registration No.: - 12109777

Roll No.- 14

Course Code: - **INT234**

Under the Guidance of

Prof. Tanima Thakur (28373)

Discipline of CSE/IT

School of Computer Science and Engineering(P-132)

Lovely Professional University, Phagwara

Student Declaration

I, Hardik, Registration Number 12109777, a Bachelor of Technology (B. Tech) student in the Computer Science and Engineering/Information Technology discipline at Lovely Professional University, Punjab, hereby formally declare that the work presented in this project report is my original work and has not been submitted previously at this or any other institution for academic qualification or certification. I affirm that all data, results, and analyses contained in this report are genuine and based on my own independent research and effort.

Name of student: Hardik

Registration Number: 12109777

Dated: 10/11/2024

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my professor, **Tanima Thakur**, for their invaluable support, guidance, and encouragement throughout the course of this project. Their expertise and constructive feedback played a significant role in enhancing the quality and depth of my work.

I also wish to extend my heartfelt thanks to **Lovely Professional University** for providing me with the necessary resources and environment to pursue this project. The opportunity to work under the guidance of such esteemed faculty has been a remarkable learning experience.

Furthermore, I am deeply grateful to my friends for their constant support and insightful feedback. Their suggestions were essential in helping me refine my ideas and approach.

Thank you all for your unwavering support and encouragement.

Name of student: Hardik

Registration Number: 121009777

Dated: 10/11/2024

1. 📌 Introduction

The purpose of this project is to perform customer segmentation analysis using a dataset containing demographic and behavioural information. Customer segmentation helps companies understand the unique needs of different customer groups, which can optimize marketing strategies, improve customer retention, and increase revenue by providing more personalized experiences.

2. Objective/Scope of Analysis:

This project aims to categorize customers based on their purchasing behaviours and demographics, enabling the identification of high-spending segments. By categorizing spending scores into "Low," "Medium," and "High," we can draw actionable insights to improve customer engagement. Our analysis will employ machine learning algorithms to predict spending behaviour, evaluate their performance, and choose the most effective model.

3. 🗄 Dataset

The dataset includes the following features:

1. **CustomerID**: Unique identifier for each customer.

2. **Genre**: Customer gender.

3. **Age**: Age of the customer, used to observe spending trends across age groups.

4. **Annual Income**: Customer's annual income, which serves as an indicator of purchasing power.

5. **Spending Score**: Based on customer behavior and purchasing patterns, categorized into "Low," "Medium," and "High." This is the target variable for predictive analysis.

The dataset offers a diverse set of demographic and behavioral characteristics that provide a robust foundation for customer segmentation.

4. 🧠 Algorithms Applied

The following machine learning algorithms were used to predict and categorize customer spending behavior:

1. **K-Nearest Neighbors (KNN)**
   - **Description**: A distance-based algorithm that assigns each test observation to the most common category among its k-nearest neighbors. This approach is effective in datasets with well-defined separations between categories.
   - **Implementation**: With k=5k = 5k=5, "Age" and "Annual Income" were used as features. KNN assigns spending categories based on the majority label among the five closest neighbors.
   - **Results**: KNN achieved moderate accuracy and precision but may struggle with overlapping or densely packed clusters.
2. **Naive Bayes**
   - **Description**: A probabilistic classifier based on Bayes' theorem that assumes feature independence, making it suitable for categorical variables and well-separated classes.
   - **Implementation**: Using "Spending Score" as a function of "Age" and "Annual Income," Naive Bayes calculates the probability of each class and assigns the highest-probability label to each observation.
   - **Results**: Naive Bayes showed solid recall and precision, though feature independence may limit its applicability in real-world settings.
3. **Decision Tree**
   - **Description**: A tree-like model that splits data into branches based on feature values, ultimately leading to a decision or category, making it interpretable and insightful for identifying key features.
   - **Implementation**: The Decision Tree splits data using "Age" and "Annual Income" until pure nodes or a stopping criterion is reached.
   - **Results**: High accuracy and interpretability, though the model may overfit with limited data or numerous categories.
4. **Logistic Regression**
   - **Description**: Predicts binary outcomes, which is useful in binary classification problems or when the target variable can be binarized.
   - **Implementation**: Binarized "Spending Score" into "High" vs. "Low/Medium" categories. The logistic function maps predictions to a probability score, classifying scores above 0.5 as "High" spending.

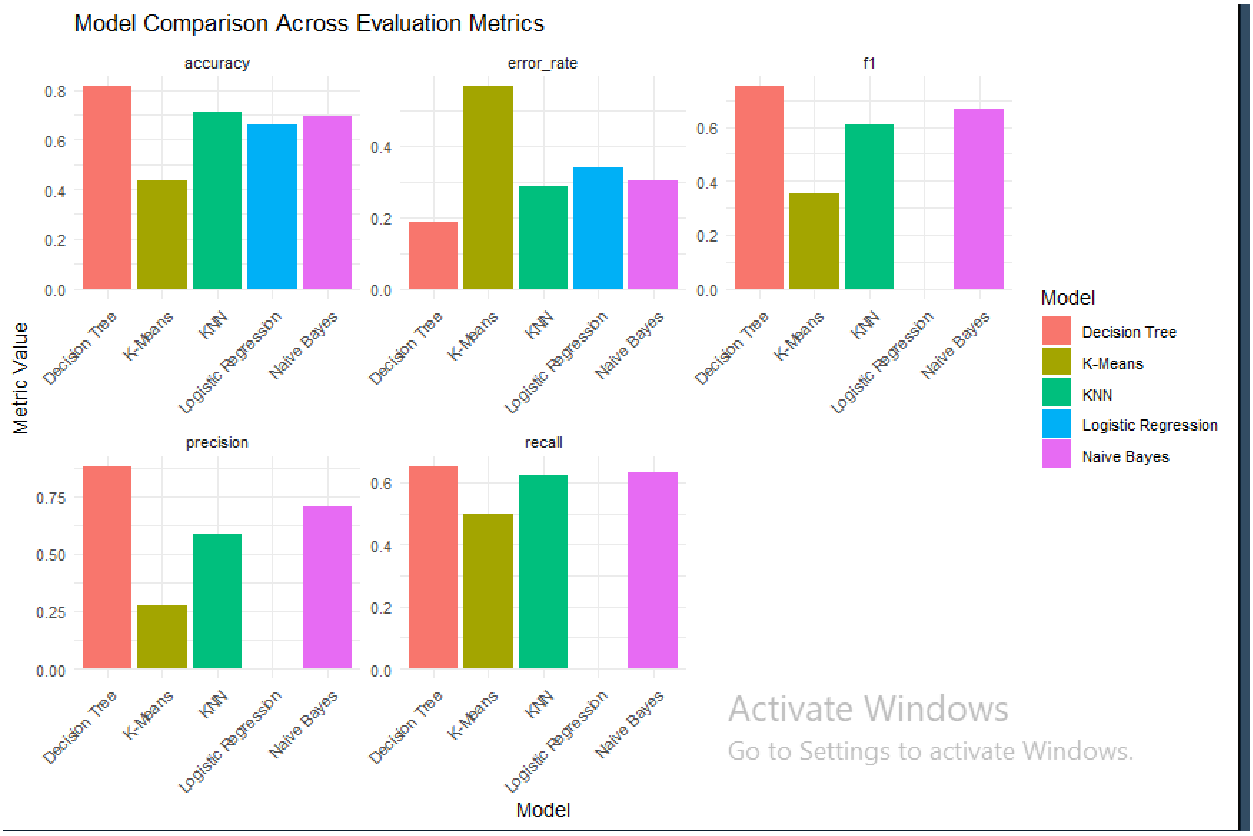
- **Results**: Strong performance on binarized spending predictions, though less effective for multi-category segmentation.

5. **K-Means Clustering**

   - **Description**: An unsupervised algorithm for segmenting data into k clusters based on feature similarity.
   - **Implementation**: Setting k=3k = 3k=3, K-Means clustered customers by similarity in "Age" and "Annual Income," roughly mapping clusters to "Low," "Medium," and "High" spending categories.
   - **Results**: While not intended for supervised prediction, K-Means revealed natural clusters within customer demographics.

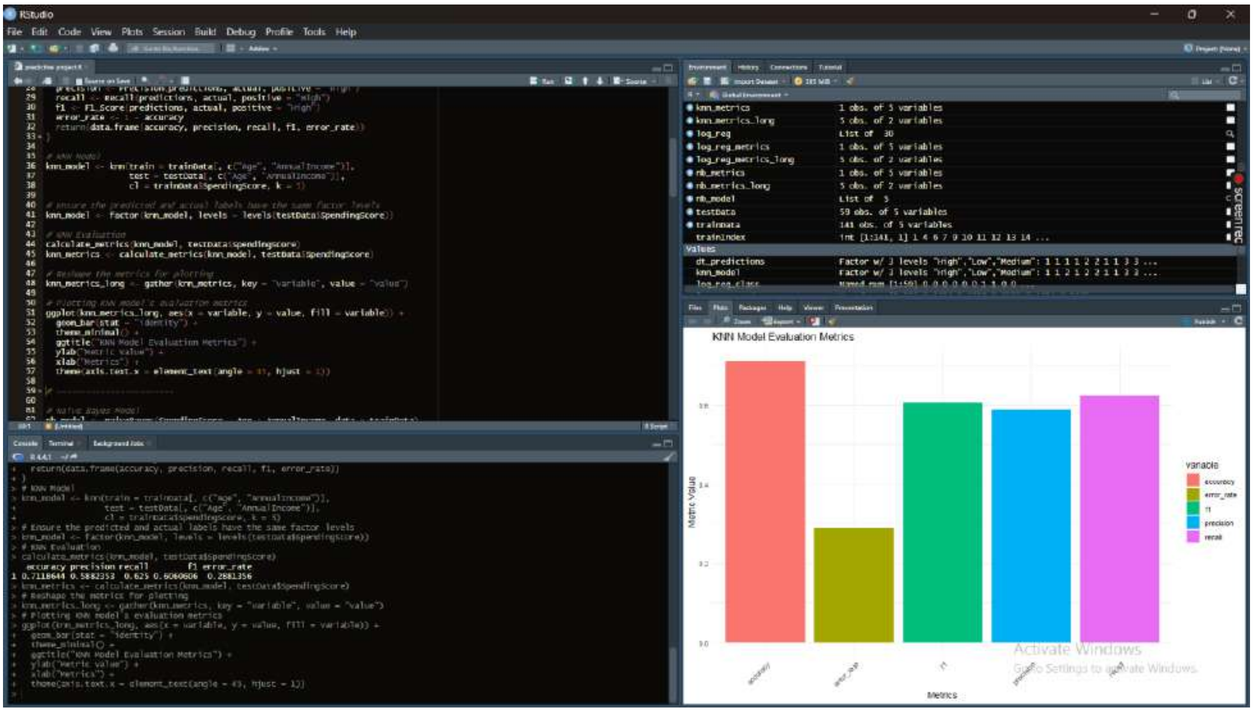
5. 📊 Performance Comparison

Each model was evaluated using metrics like accuracy, precision, recall, F1 score, and error rate to determine the best-performing algorithm for customer segmentation.



5. Algorithms Used

K-Nearest Neighbours (K-NN):



```
  precision <- Precision(predictions, actual, positive = "High")
  recall <- Recall(predictions, actual, positive = "High")
  f1 <- F1_Score(predictions, actual, positive = "High")
  error_rate <- 1 - accuracy
  return(data.frame(accuracy, precision, recall, f1, error_rate))
}

# KNN Model
knn_model <- knn(train = trainData[, c("Age", "AnnualIncome")],
                 test = testData[, c("Age", "AnnualIncome")],
                 cl = trainData$SpendingScore, k = 5)

# Ensure the predicted and actual labels have the same factor levels
knn_model <- factor(knn_model, levels = levels(testData$SpendingScore))

# KNN Evaluation
calculate_metrics(knn_model, testData$SpendingScore)
knn_metrics <- calculate_metrics(knn_model, testData$SpendingScore)

# Reshape the metrics for plotting
knn_metrics_long <- gather(knn_metrics, key = "variable", value = "value")

# Plotting KNN model's evaluation metrics
ggplot(knn_metrics_long, aes(x = variable, y = value, fill = variable)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  ggtitle("KNN Model Evaluation Metrics") +
  ylab("Metric Value") +
  xlab("Metrics") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# ----------------------------

# Naive Bayes Model
nb_model <- naiveBayes(SpendingScore ~ Age + AnnualIncome, data = trainData)
```

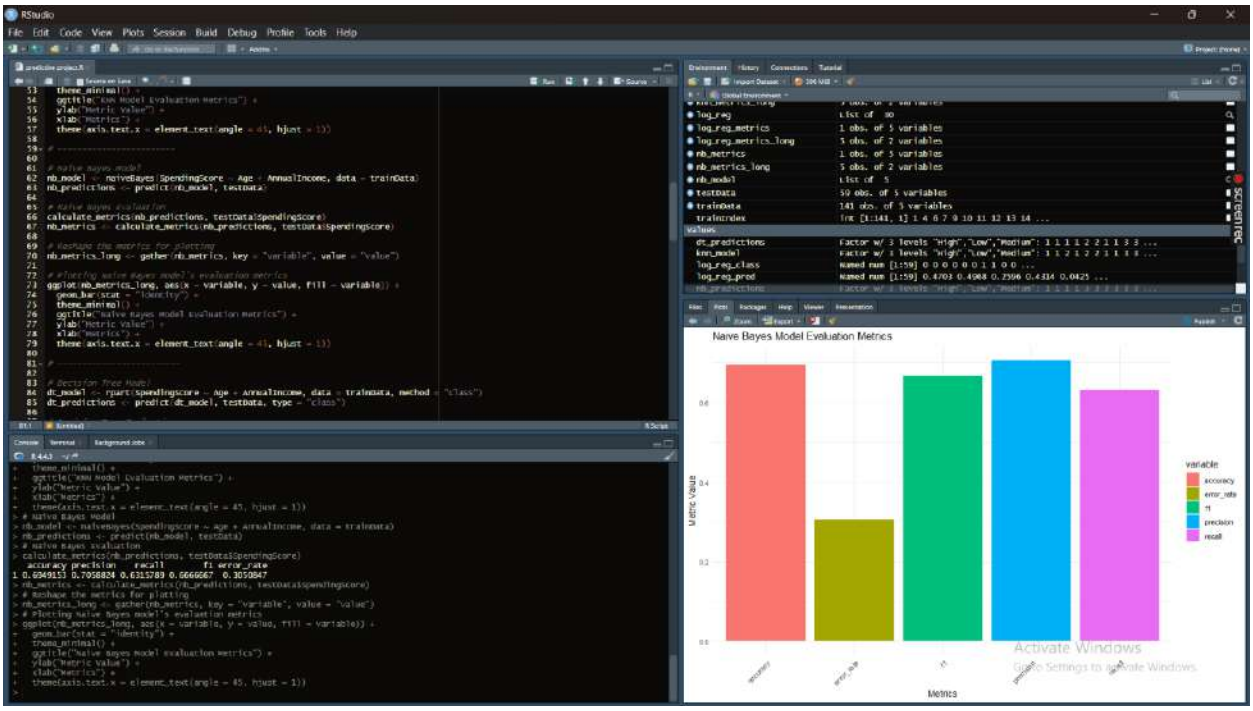
```
+   return(data.frame(accuracy, precision, recall, f1, error_rate))
+ }
> # KNN Model
> knn_model <- knn(train = trainData[, c("Age", "AnnualIncome")],
+                  test = testData[, c("Age", "AnnualIncome")],
+                  cl = trainData$SpendingScore, k = 5)
> # Ensure the predicted and actual labels have the same factor levels
> knn_model <- factor(knn_model, levels = levels(testData$SpendingScore))
> # KNN Evaluation
> calculate_metrics(knn_model, testData$SpendingScore)
   accuracy precision recall        f1 error_rate
1 0.7118644 0.5882353  0.625 0.6060606  0.2881356
> knn_metrics <- calculate_metrics(knn_model, testData$SpendingScore)
> # Reshape the metrics for plotting
> knn_metrics_long <- gather(knn_metrics, key = "variable", value = "value")
> # Plotting KNN model's evaluation metrics
> ggplot(knn_metrics_long, aes(x = variable, y = value, fill = variable)) +
+   geom_bar(stat = "identity") +
+   theme_minimal() +
+   ggtitle("KNN Model Evaluation Metrics") +
+   ylab("Metric Value") +
+   xlab("Metrics") +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1))
>
```

```
knn_metrics            1 obs. of 5 variables
knn_metrics_long       5 obs. of 2 variables
log_reg                List of 30
log_reg_metrics        1 obs. of 5 variables
log_reg_metrics_long   5 obs. of 2 variables
nb_metrics             1 obs. of 5 variables
nb_metrics_long        5 obs. of 2 variables
nb_model               List of 5
testData               59 obs. of 5 variables
trainData              141 obs. of 5 variables
trainIndex             int [1:141, 1] 1 4 6 7 9 10 11 12 13 14 ...
Values
dt_predictions         Factor w/ 3 levels "High","Low","Medium": 1 1 1 1 2 2 1 1 3 3 ...
knn_model              Factor w/ 3 levels "High","Low","Medium": 1 1 2 1 2 2 1 1 3 3 ...
log_reg_class          Named num [1:59] 0 0 0 0 0 0 1 1 0 0 ...
```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Console Terminal Background Jobs
R 4.4.1
Environment History Connections Tutorial
Import Dataset
Global Environment
Files Plots Packages Help Viewer Presentation
Zoom Export
KNN Model Evaluation Metrics
0.6
0.4
0.2
0.0
Metric Value
accuracy
error_rate
f1
precision
recall
Metrics
variable
🟥 accuracy
🟨 error_rate
🟩 f1
🟦 precision
🟪 recall
Activate Windows
Go to Settings to activate Windows.

This R code implements a K-Nearest Neighbors (KNN) model to predict customer spending behavior based on demographic features like age and annual income. It trains the model using a training dataset and makes predictions on a test dataset. The predicted labels are then adjusted to match the factor levels of the actual labels. The performance of the KNN model is evaluated using various metrics such as accuracy, precision, recall, F1 score, and error rate. These metrics are reshaped for visualization, and a bar plot is created to display the evaluation results, helping to assess the model's performance.

Naive Bayes:



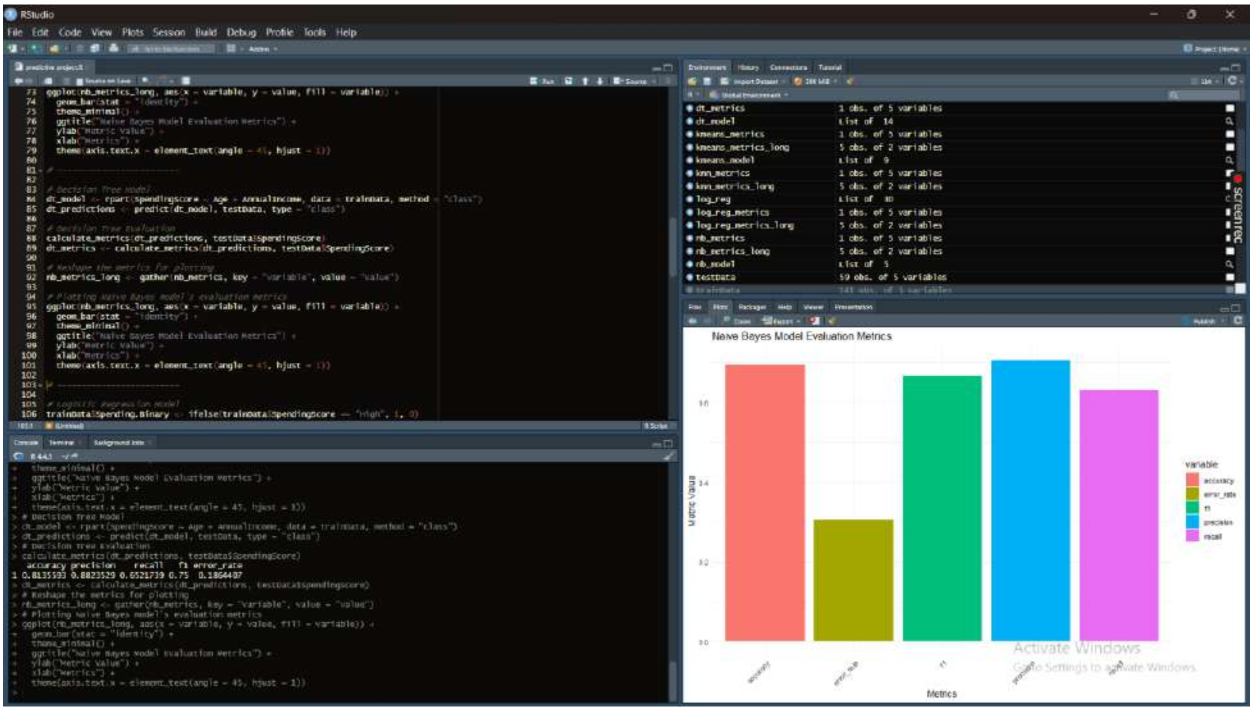
```r
  theme_minimal() +
  ggtitle("KNN Model Evaluation Metrics") +
  ylab("Metric Value") +
  xlab("Metrics") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# -----------------------------

# Naive Bayes Model
nb_model <- naiveBayes(SpendingScore ~ Age + AnnualIncome, data = trainData)
nb_predictions <- predict(nb_model, testData)

# Naive Bayes evaluation
calculate_metrics(nb_predictions, testData$SpendingScore)
nb_metrics <- calculate_metrics(nb_predictions, testData$SpendingScore)

# Reshape the metrics for plotting
nb_metrics_long <- gather(nb_metrics, key = "variable", value = "value")

# Plotting Naive Bayes model's evaluation metrics
ggplot(nb_metrics_long, aes(x = variable, y = value, fill = variable)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  ggtitle("Naive Bayes Model Evaluation Metrics") +
  ylab("Metric Value") +
  xlab("Metrics") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# -----------------------------

# Decision Tree Model
dt_model <- rpart(SpendingScore ~ Age + AnnualIncome, data = trainData, method = "class")
dt_predictions <- predict(dt_model, testData, type = "class")
```

```
+   theme_minimal() +
+   ggtitle("KNN Model Evaluation Metrics") +
+   ylab("Metric Value") +
+   xlab("Metrics") +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1))
> # Naive Bayes Model
> nb_model <- naiveBayes(SpendingScore ~ Age + AnnualIncome, data = trainData)
> nb_predictions <- predict(nb_model, testData)
> # Naive Bayes evaluation
> calculate_metrics(nb_predictions, testData$SpendingScore)
   accuracy precision    recall        f1 error_rate
1 0.6949153 0.7058824 0.6315789 0.6666667  0.3050847
> nb_metrics <- calculate_metrics(nb_predictions, testData$SpendingScore)
> # Reshape the metrics for plotting
> nb_metrics_long <- gather(nb_metrics, key = "variable", value = "value")
> # Plotting Naive Bayes model's evaluation metrics
> ggplot(nb_metrics_long, aes(x = variable, y = value, fill = variable)) +
+   geom_bar(stat = "identity") +
+   theme_minimal() +
+   ggtitle("Naive Bayes Model Evaluation Metrics") +
+   ylab("Metric Value") +
+   xlab("Metrics") +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1))
>
```

```
log_reg                  List of  30
log_reg_metrics          1 obs. of 5 variables
log_reg_metrics_long     5 obs. of 2 variables
nb_metrics               1 obs. of 5 variables
nb_metrics_long          5 obs. of 2 variables
nb_model                 List of  5
testData                 59 obs. of 5 variables
trainData                141 obs. of 5 variables
trainIndex               int [1:141, 1] 1 4 6 7 9 10 11 12 13 14 ...
Values
dt_predictions           Factor w/ 3 levels "High","Low","Medium": 1 1 1 1 2 2 1 1 3 3 ...
knn_model                Factor w/ 3 levels "High","Low","Medium": 1 1 2 1 2 2 1 1 3 3 ...
log_reg_class            Named num [1:59] 0 0 0 0 0 0 1 1 0 0 ...
log_reg_pred             Named num [1:59] 0.4703 0.4988 0.2596 0.4334 0.0425 ...
```

This R code implements a Naive Bayes model to predict customer spending behavior based on age and annual income. The model is trained using the training dataset and makes predictions on the test dataset. The predicted labels are compared to the actual labels, and the performance of the model is evaluated using metrics like accuracy, precision, recall, F1 score, and error rate. These evaluation metrics are reshaped for easier visualization and then plotted in a bar chart to assess the effectiveness of the Naive Bayes model.

Decision Tree:



```r
ggplot(nb_metrics_long, aes(x = variable, y = value, fill = variable)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  ggtitle("Naive Bayes Model Evaluation Metrics") +
  ylab("Metric Value") +
  xlab("Metrics") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# -------------------------

# Decision Tree Model
dt_model <- rpart(SpendingScore ~ Age + AnnualIncome, data = trainData, method = "class")
dt_predictions <- predict(dt_model, testData, type = "class")

# Decision Tree Evaluation
calculate_metrics(dt_predictions, testData$SpendingScore)
dt_metrics <- calculate_metrics(dt_predictions, testData$SpendingScore)

# Reshape the metrics for plotting
nb_metrics_long <- gather(nb_metrics, key = "variable", value = "value")

# Plotting Naive Bayes model's evaluation metrics
ggplot(nb_metrics_long, aes(x = variable, y = value, fill = variable)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  ggtitle("Naive Bayes Model Evaluation Metrics") +
  ylab("Metric Value") +
  xlab("Metrics") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# -------------------------

# Logistic Regression Model
trainData$Spending.Binary <- ifelse(trainData$SpendingScore == "High", 1, 0)
```

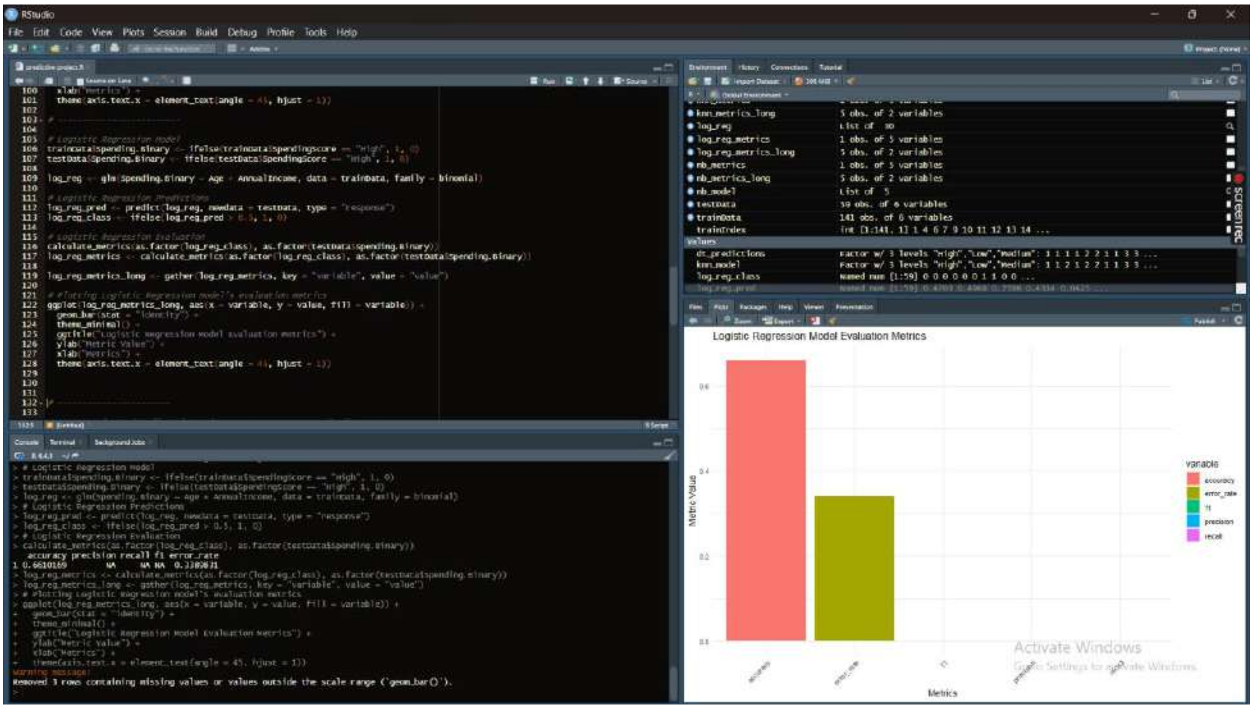
```
+   theme_minimal() +
+   ggtitle("Naive Bayes Model Evaluation Metrics") +
+   ylab("Metric Value") +
+   xlab("Metrics") +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1))
> # Decision Tree Model
> dt_model <- rpart(SpendingScore ~ Age + AnnualIncome, data = trainData, method = "class")
> dt_predictions <- predict(dt_model, testData, type = "class")
> # Decision Tree Evaluation
> calculate_metrics(dt_predictions, testData$SpendingScore)
   accuracy precision    recall   f1 error_rate
1 0.8135593 0.8823529 0.6521739 0.75  0.1864407
> dt_metrics <- calculate_metrics(dt_predictions, testData$SpendingScore)
> # Reshape the metrics for plotting
> nb_metrics_long <- gather(nb_metrics, key = "variable", value = "value")
> # Plotting Naive Bayes model's evaluation metrics
> ggplot(nb_metrics_long, aes(x = variable, y = value, fill = variable)) +
+   geom_bar(stat = "identity") +
+   theme_minimal() +
+   ggtitle("Naive Bayes Model Evaluation Metrics") +
+   ylab("Metric Value") +
+   xlab("Metrics") +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1))
>
```

```
dt_metrics             1 obs. of 5 variables
dt_model               List of 14
kmeans_metrics         1 obs. of 5 variables
kmeans_metrics_long    5 obs. of 2 variables
kmeans_model           List of 9
knn_metrics            1 obs. of 5 variables
knn_metrics_long       5 obs. of 2 variables
log_reg                List of 30
log_reg_metrics        1 obs. of 5 variables
log_reg_metrics_long   5 obs. of 2 variables
nb_metrics             1 obs. of 5 variables
nb_metrics_long        5 obs. of 2 variables
nb_model               List of 5
testData               59 obs. of 5 variables
trainData              141 obs. of 5 variables
```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Environment History Connections Tutorial
Import Dataset
Global Environment
Files Plots Packages Help Viewer Presentation
Zoom Export
Publish
Naive Bayes Model Evaluation Metrics
0.6
0.4
0.2
0.0
Metric Value
accuracy
error_rate
f1
precision
recall
Metrics
variable
accuracy
error_rate
f1
precision
recall
Activate Windows
Go to Settings to activate Windows.
screenrec

This code implements a Decision Tree model for classifying customer spending behavior based on the features "Age" and "AnnualIncome". It first trains the model using the rpart function with a classification method, then makes predictions on the test data. The calculate_metrics function is used to evaluate the model's performance by calculating various metrics like accuracy, precision, recall, F1 score, and error rate. The results are then reshaped into a long format using the gather function, and the evaluation metrics are visualized in a bar plot using ggplot2 to assess the model's effectiveness in predicting the spending behavior categories.

Logistic Regression:



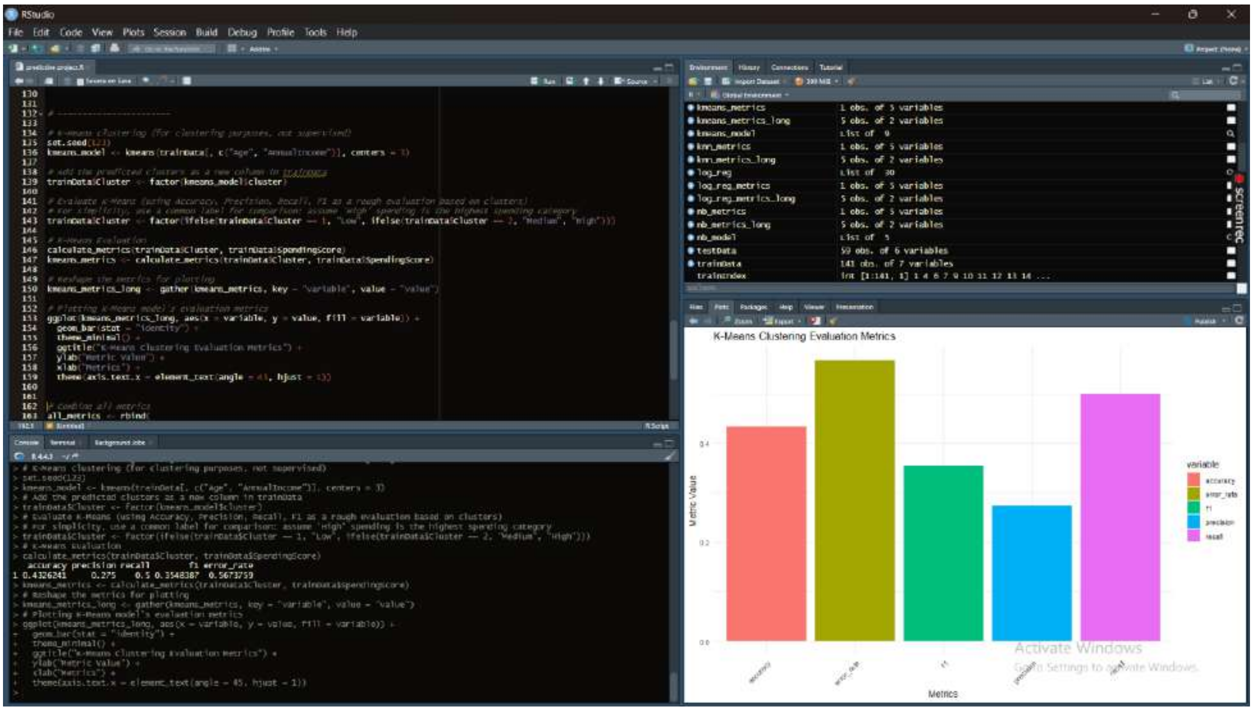
```
  xlab("Metrics") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# ---------------------------

# Logistic Regression Model
trainData$Spending.Binary <- ifelse(trainData$SpendingScore == "High", 1, 0)
testData$Spending.Binary <- ifelse(testData$SpendingScore == "High", 1, 0)

log_reg <- glm(Spending.Binary ~ Age + AnnualIncome, data = trainData, family = binomial)

# Logistic Regression Predictions
log_reg_pred <- predict(log_reg, newdata = testData, type = "response")
log_reg_class <- ifelse(log_reg_pred > 0.5, 1, 0)

# Logistic Regression Evaluation
calculate_metrics(as.factor(log_reg_class), as.factor(testData$Spending.Binary))
log_reg_metrics <- calculate_metrics(as.factor(log_reg_class), as.factor(testData$Spending.Binary))

log_reg_metrics_long <- gather(log_reg_metrics, key = "variable", value = "value")

# Plotting Logistic Regression model's evaluation metrics
ggplot(log_reg_metrics_long, aes(x = variable, y = value, fill = variable)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  ggtitle("Logistic Regression Model Evaluation Metrics") +
  ylab("Metric Value") +
  xlab("Metrics") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))



# ---------------------------
```

```
> # Logistic Regression Model
> trainData$Spending.Binary <- ifelse(trainData$SpendingScore == "High", 1, 0)
> testData$Spending.Binary <- ifelse(testData$SpendingScore == "High", 1, 0)
> log_reg <- glm(Spending.Binary ~ Age + AnnualIncome, data = trainData, family = binomial)
> # Logistic Regression Predictions
> log_reg_pred <- predict(log_reg, newdata = testData, type = "response")
> log_reg_class <- ifelse(log_reg_pred > 0.5, 1, 0)
> # Logistic Regression Evaluation
> calculate_metrics(as.factor(log_reg_class), as.factor(testData$Spending.Binary))
   accuracy precision recall f1 error_rate
1 0.6610169        NA     NA NA  0.3389831
> log_reg_metrics <- calculate_metrics(as.factor(log_reg_class), as.factor(testData$Spending.Binary))
> log_reg_metrics_long <- gather(log_reg_metrics, key = "variable", value = "value")
> # Plotting Logistic Regression model's evaluation metrics
> ggplot(log_reg_metrics_long, aes(x = variable, y = value, fill = variable)) +
+   geom_bar(stat = "identity") +
+   theme_minimal() +
+   ggtitle("Logistic Regression Model Evaluation Metrics") +
+   ylab("Metric Value") +
+   xlab("Metrics") +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1))
Warning message:
Removed 3 rows containing missing values or values outside the scale range (`geom_bar()`).
>
```

```
knn_metrics_long        5 obs. of 2 variables
log_reg                 List of 30
log_reg_metrics         1 obs. of 5 variables
log_reg_metrics_long    5 obs. of 2 variables
nb_metrics              1 obs. of 5 variables
nb_metrics_long         5 obs. of 2 variables
nb_model                List of 5
testData                59 obs. of 6 variables
trainData               141 obs. of 6 variables
trainIndex              int [1:141, 1] 1 4 6 7 9 10 11 12 13 14 ...
Values
dt_predictions          Factor w/ 3 levels "High","Low","Medium": 1 1 1 1 2 2 1 1 3 3 ...
knn_model               Factor w/ 3 levels "High","Low","Medium": 1 1 2 1 2 2 1 1 3 3 ...
log_reg_class           Named num [1:59] 0 0 0 0 0 0 1 1 0 0 ...
log_reg_pred            Named num [1:59] 0.4703 0.4369 0.7596 0.4334 0.0425 ...
```

This code implements a Decision Tree model for classifying customer spending behavior based on the features "Age" and "AnnualIncome". It first trains the model using the rpart function with a classification method, then makes predictions on the test data. The calculate_metrics function is used to evaluate the model's performance by calculating various metrics like accuracy, precision, recall, F1 score, and error rate. The results are then reshaped into a long format using the gather function, and the evaluation metrics are visualized in a bar plot using ggplot2 to assess the model's effectiveness in predicting the spending behavior categories.

K-Means Clustering:



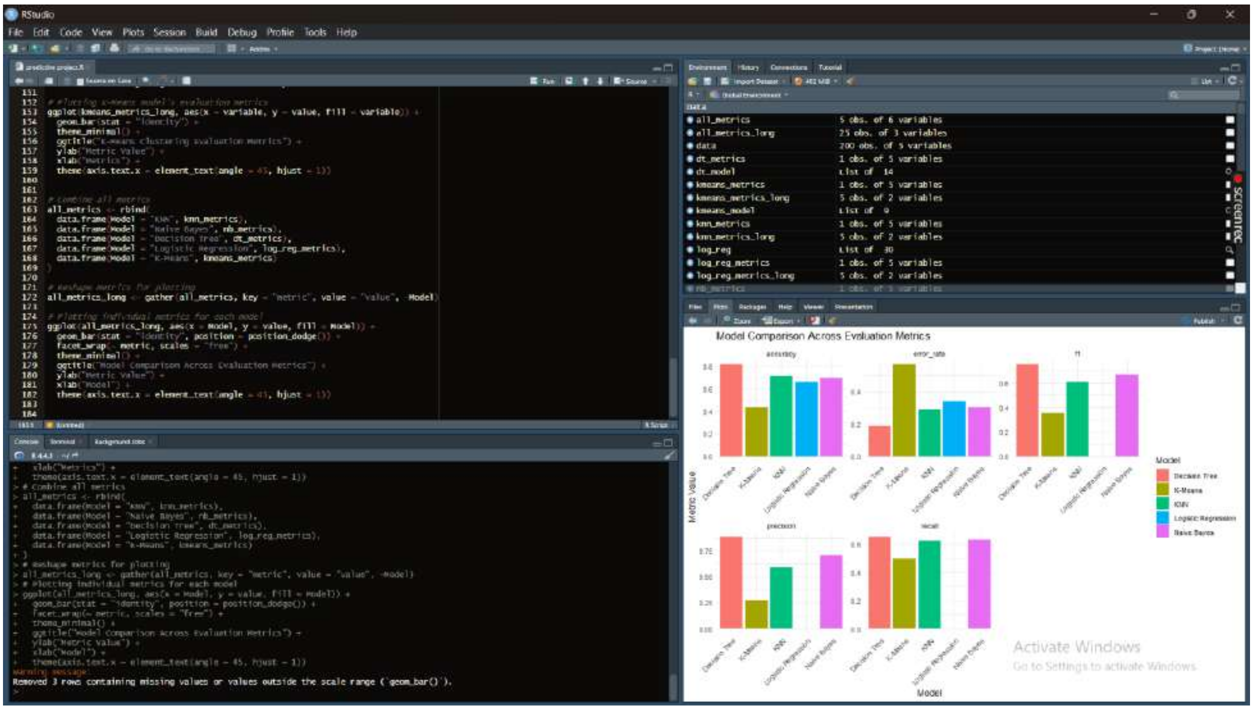
```
# ---------------------------

# K-Means clustering (for clustering purposes, not supervised)
set.seed(123)
kmeans_model <- kmeans(trainData[, c("Age", "AnnualIncome")], centers = 3)

# Add the predicted clusters as a new column in trainData
trainData$Cluster <- factor(kmeans_model$cluster)

# Evaluate K-Means (using Accuracy, Precision, Recall, F1 as a rough evaluation based on clusters)
# For simplicity, use a common label for comparison: assume 'High' spending is the highest spending category
trainData$Cluster <- factor(ifelse(trainData$Cluster == 1, "Low", ifelse(trainData$Cluster == 2, "Medium", "High")))

# K-Means Evaluation
calculate_metrics(trainData$Cluster, trainData$SpendingScore)
kmeans_metrics <- calculate_metrics(trainData$Cluster, trainData$SpendingScore)

# Reshape the metrics for plotting
kmeans_metrics_long <- gather(kmeans_metrics, key = "variable", value = "value")

# Plotting K-Means model's evaluation metrics
ggplot(kmeans_metrics_long, aes(x = variable, y = value, fill = variable)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  ggtitle("K-Means Clustering Evaluation Metrics") +
  ylab("Metric Value") +
  xlab("Metrics") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


# Combine all metrics
all_metrics <- rbind(
```

```
> # K-Means Clustering (for clustering purposes, not supervised)
> set.seed(123)
> kmeans_model <- kmeans(trainData[, c("Age", "AnnualIncome")], centers = 3)
> # Add the predicted clusters as a new column in trainData
> trainData$Cluster <- factor(kmeans_model$cluster)
> # Evaluate K-Means (using Accuracy, Precision, Recall, F1 as a rough evaluation based on clusters)
> # For simplicity, use a common label for comparison: assume 'High' spending is the highest spending category
> trainData$Cluster <- factor(ifelse(trainData$Cluster == 1, "Low", ifelse(trainData$Cluster == 2, "Medium", "High")))
> # K-Means Evaluation
> calculate_metrics(trainData$Cluster, trainData$SpendingScore)
   accuracy precision recall        f1 error_rate
1 0.4326241     0.275    0.5 0.3548387  0.5673759
> kmeans_metrics <- calculate_metrics(trainData$Cluster, trainData$SpendingScore)
> # Reshape the metrics for plotting
> kmeans_metrics_long <- gather(kmeans_metrics, key = "variable", value = "value")
> # Plotting K-Means model's evaluation metrics
> ggplot(kmeans_metrics_long, aes(x = variable, y = value, fill = variable)) +
+   geom_bar(stat = "identity") +
+   theme_minimal() +
+   ggtitle("K-Means Clustering Evaluation Metrics") +
+   ylab("Metric Value") +
+   xlab("Metrics") +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1))
>
```

```
kmeans_metrics          1 obs. of 5 variables
kmeans_metrics_long     5 obs. of 2 variables
kmeans_model            List of 9
knn_metrics             1 obs. of 5 variables
knn_metrics_long        5 obs. of 2 variables
log_reg                 List of 30
log_reg_metrics         1 obs. of 5 variables
log_reg_metrics_long    5 obs. of 2 variables
nb_metrics              1 obs. of 5 variables
nb_metrics_long         5 obs. of 2 variables
nb_model                List of 5
testData                59 obs. of 6 variables
trainData               141 obs. of 7 variables
trainIndex              int [1:141, 1] 1 4 6 7 9 10 11 12 13 14 ...
```

This code performs K-Means clustering on customer data using the features "Age" and "AnnualIncome" to group customers into three clusters. The number of clusters is set to 3, with each customer assigned to one of the clusters. The cluster assignments are mapped to spending categories ("Low", "Medium", "High") for evaluation purposes, and the calculate_metrics function is used to assess the clustering results by comparing the clusters with the actual spending scores. The performance is evaluated using metrics such as accuracy, precision, recall, and F1 score, and the results are visualized in a bar plot to present the effectiveness of the clustering model.

Result:



```
# Plotting K-Means model's evaluation metrics
ggplot(kmeans_metrics_long, aes(x = variable, y = value, fill = variable)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  ggtitle("K-Means Clustering Evaluation Metrics") +
  ylab("Metric Value") +
  xlab("Metrics") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


# Combine all metrics
all_metrics <- rbind(
  data.frame(Model = "KNN", knn_metrics),
  data.frame(Model = "Naive Bayes", nb_metrics),
  data.frame(Model = "Decision Tree", dt_metrics),
  data.frame(Model = "Logistic Regression", log_reg_metrics),
  data.frame(Model = "K-Means", kmeans_metrics)
)

# Reshape metrics for plotting
all_metrics_long <- gather(all_metrics, key = "metric", value = "value", -Model)

# Plotting individual metrics for each model
ggplot(all_metrics_long, aes(x = Model, y = value, fill = Model)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  facet_wrap(~ metric, scales = "free") +
  theme_minimal() +
  ggtitle("Model Comparison Across Evaluation Metrics") +
  ylab("Metric Value") +
  xlab("Model") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
+   xlab("Metrics") +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1))
> # Combine all metrics
> all_metrics <- rbind(
+   data.frame(Model = "KNN", knn_metrics),
+   data.frame(Model = "Naive Bayes", nb_metrics),
+   data.frame(Model = "Decision Tree", dt_metrics),
+   data.frame(Model = "Logistic Regression", log_reg_metrics),
+   data.frame(Model = "K-Means", kmeans_metrics)
+ )
> # Reshape metrics for plotting
> all_metrics_long <- gather(all_metrics, key = "metric", value = "value", -Model)
> # Plotting individual metrics for each model
> ggplot(all_metrics_long, aes(x = Model, y = value, fill = Model)) +
+   geom_bar(stat = "identity", position = position_dodge()) +
+   facet_wrap(~ metric, scales = "free") +
+   theme_minimal() +
+   ggtitle("Model Comparison Across Evaluation Metrics") +
+   ylab("Metric Value") +
+   xlab("Model") +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1))
Warning message:
Removed 3 rows containing missing values or values outside the scale range (`geom_bar()`).
>
```

```
Data
all_metrics            5 obs. of 6 variables
all_metrics_long       25 obs. of 3 variables
data                   200 obs. of 5 variables
dt_metrics             1 obs. of 5 variables
dt_model               List of  14
kmeans_metrics         1 obs. of 5 variables
kmeans_metrics_long    5 obs. of 2 variables
kmeans_model           List of  9
knn_metrics            1 obs. of 5 variables
knn_metrics_long       5 obs. of 2 variables
log_reg                List of  30
log_reg_metrics        1 obs. of 5 variables
log_reg_metrics_long   5 obs. of 2 variables
nb_metrics             1 obs. of 5 variables
```

This code combines the evaluation metrics from multiple models (KNN, Naive Bayes, Decision Tree, Logistic Regression, and K-Means) into a single data frame, then reshapes the data for plotting. The ggplot function is used to create a bar plot that compares the performance of each model across various evaluation metrics (such as accuracy, precision, recall, F1 score, etc.). The plot is faceted by the type of metric, allowing for a clear visual comparison of how each model performs in different aspects of evaluation. The result is a comprehensive visualization that helps to assess and compare the strengths of each model in the analysis.

7. 🔍 Conclusion

Each algorithm demonstrated unique strengths:

- **Decision Tree and Naive Bayes** models excelled in interpretability and provided good predictive accuracy.
- **KNN and Logistic Regression** offered alternative insights for binary and multi-class spending predictions.
- **K-Means Clustering** revealed natural groupings, adding value through unsupervised segmentation.

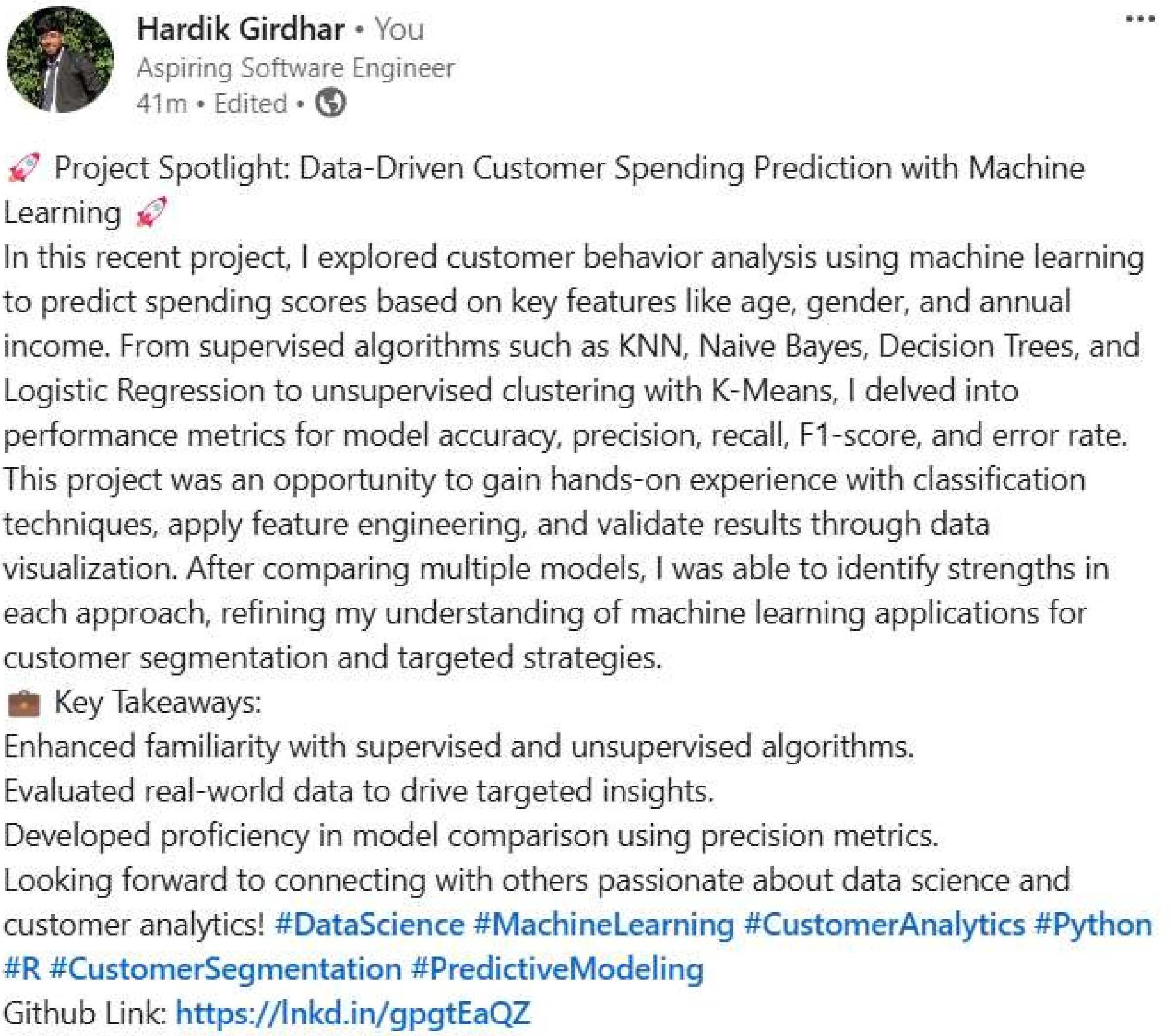
For practical applications in customer segmentation, the **Decision Tree model** may be most advantageous due to its high accuracy and straightforward interpretability.

8. GitHub Link

Hardik-Girdhar/Machine-Learning-R-Algorithms

9. LINKEDIN Post

hardik-girdhar_datascience-machinelearning-customeranalytics-activity



Hardik Girdhar • You

Aspiring Software Engineer

41m • Edited •

🚀 Project Spotlight: Data-Driven Customer Spending Prediction with Machine Learning 🚀

In this recent project, I explored customer behavior analysis using machine learning to predict spending scores based on key features like age, gender, and annual income. From supervised algorithms such as KNN, Naive Bayes, Decision Trees, and Logistic Regression to unsupervised clustering with K-Means, I delved into performance metrics for model accuracy, precision, recall, F1-score, and error rate. This project was an opportunity to gain hands-on experience with classification techniques, apply feature engineering, and validate results through data visualization. After comparing multiple models, I was able to identify strengths in each approach, refining my understanding of machine learning applications for customer segmentation and targeted strategies.

💼 Key Takeaways:

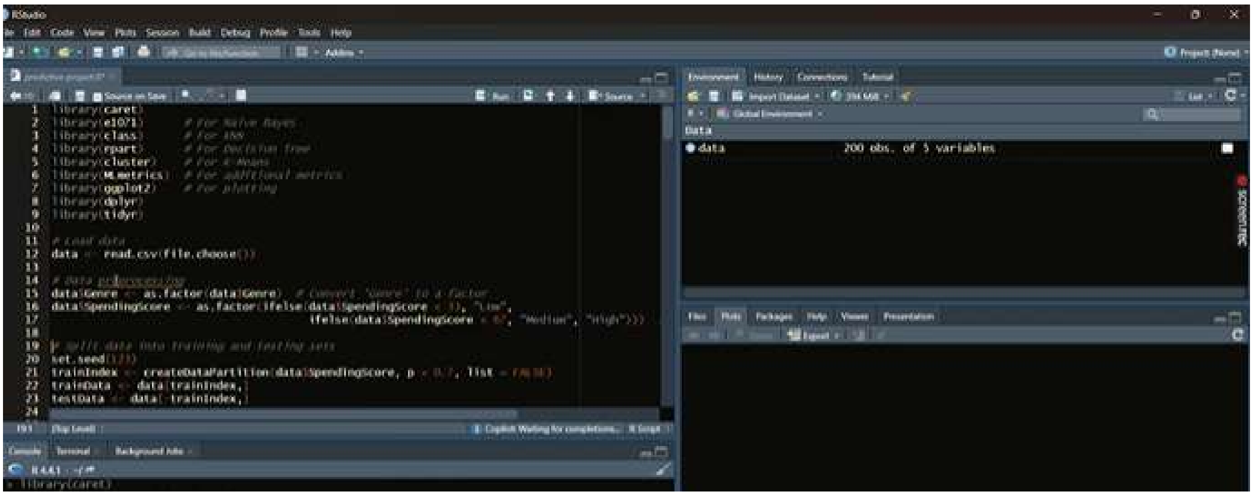
Enhanced familiarity with supervised and unsupervised algorithms.

Evaluated real-world data to drive targeted insights.

Developed proficiency in model comparison using precision metrics.

Looking forward to connecting with others passionate about data science and customer analytics! #DataScience #MachineLearning #CustomerAnalytics #Python #R #CustomerSegmentation #PredictiveModeling

Github Link: https://lnkd.in/gpgtEaQZ



RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Addins
Source on Save
Run
Source
Environment History Connections Tutorial
Import Dataset
Global Environment
Files Plots Packages Help Viewer Presentation
Export
Console Terminal Background Jobs
R 4.4.1
(Top Level)
R Script
Project: (None)

```
library(caret)
library(e1071)      # For Naive Bayes
library(class)      # For KNN
library(rpart)      # For Decision Tree
library(cluster)    # For K-Means
library(MLmetrics)  # For additional metrics
library(ggplot2)    # For plotting
library(dplyr)
library(tidyr)

# Load data
data <- read.csv(file.choose())

# Data preprocessing
data$Genre <- as.factor(data$Genre)  # Convert 'Genre' to a factor
data$SpendingScore <- as.factor(ifelse(data$SpendingScore < 33, "Low",
                                ifelse(data$SpendingScore < 67, "Medium", "High")))

# Split data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(data$SpendingScore, p = 0.7, list = FALSE)
trainData <- data[trainIndex,]
testData <- data[-trainIndex,]
```

```
Data
data          200 obs. of 5 variables
```

```
> library(caret)
```