Purchase Prediction Model for an Advertising Campaign on a Social Network

Carlos E Rivera

2023-11-10

Contents

Introduction

This report provides an overview of the Social Network Ads dataset, comprising user demographic data and their purchasing behavior. The primary goal is to evaluate the e ectiveness of predictive modeling techniques in forecasting user purchases based on age and estimated salary. We perform a comparative analysis using two di erent machine learning models: K-Nearest Neighbors (K-NN) and Random Forest.

Methods and Analysis

Data Preparation

The dataset was downloaded from a shared Google Drive link, featuring variables such as user ID, gender, age, estimated salary, and purchase status. We focused on 'Age', 'EstimatedSalary', and the binary 'Purchased' variable for our analysis. The dataset was split into training (75%) and test (25%) sets, ensuring reproducibility with a set random seed.

Feature Scaling

Age and Estimated Salary were scaled to standardize the data, facilitating better model performance, particularly for algorithms sensitive to variable scales like K-NN.

Model Evaluation Function

We deÅned an evaluation function using the 'caret' package to compute accuracy and Cohen's Kappaóa statistic that measures agreement between predicted and actual classiÅcations, adjusting for chance.

Data Import and Preprocessing

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:randomForest':
##
##     margin
## Loading required package: lattice
```

```
# Define the direct download URL for the dataset

download_url <- "https://drive.google.com/uc?export=download&id=183CuUb08gcK5s3Sf1OToDTu-ZYn-89pX"

# Download the dataset file to the local directory

download.file(download_url, destfile = "Social_Network_Ads.csv", mode ="wb")

# Read the dataset into R
dataset <- read.csv('Social_Network_Ads.csv')
```

Dataset Exploration

```
# Statistical summary of the dataset

summary(dataset)
```

```
##       User.ID          Gender               Age        EstimatedSalary
##  Min.   :15566689   Length:400         Min.   :18.00   Min.   : 15000
##  1st Qu.:15626764   Class :character   1st Qu.:29.75   1st Qu.: 43000
##  Median :15694342   Mode  :character   Median :37.00   Median : 70000
##  Mean   :15691540                      Mean   :37.66   Mean   : 69742
##  3rd Qu.:15750363                      3rd Qu.:46.00   3rd Qu.: 88000
##  Max.   :15815236                      Max.   :60.00   Max.   :150000
##    Purchased
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.3575
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

```
# View the structure of the dataset

str(dataset)
```

```
## 'data.frame':    400 obs. of  5 variables:
##  $ User.ID        : int  15624510 15810944 15668575 15603246 15804002 15728773 15598044 15694829 15
##  $ Gender         : chr  "Male" "Male" "Female" "Female" ...
##  $ Age            : int  19 35 26 27 19 27 27 32 25 35 ...
##  $ EstimatedSalary: int  19000 20000 43000 57000 76000 58000 84000 150000 33000 65000 ...
##  $ Purchased      : int  0 0 0 0 0 0 0 1 0 0 ...
```

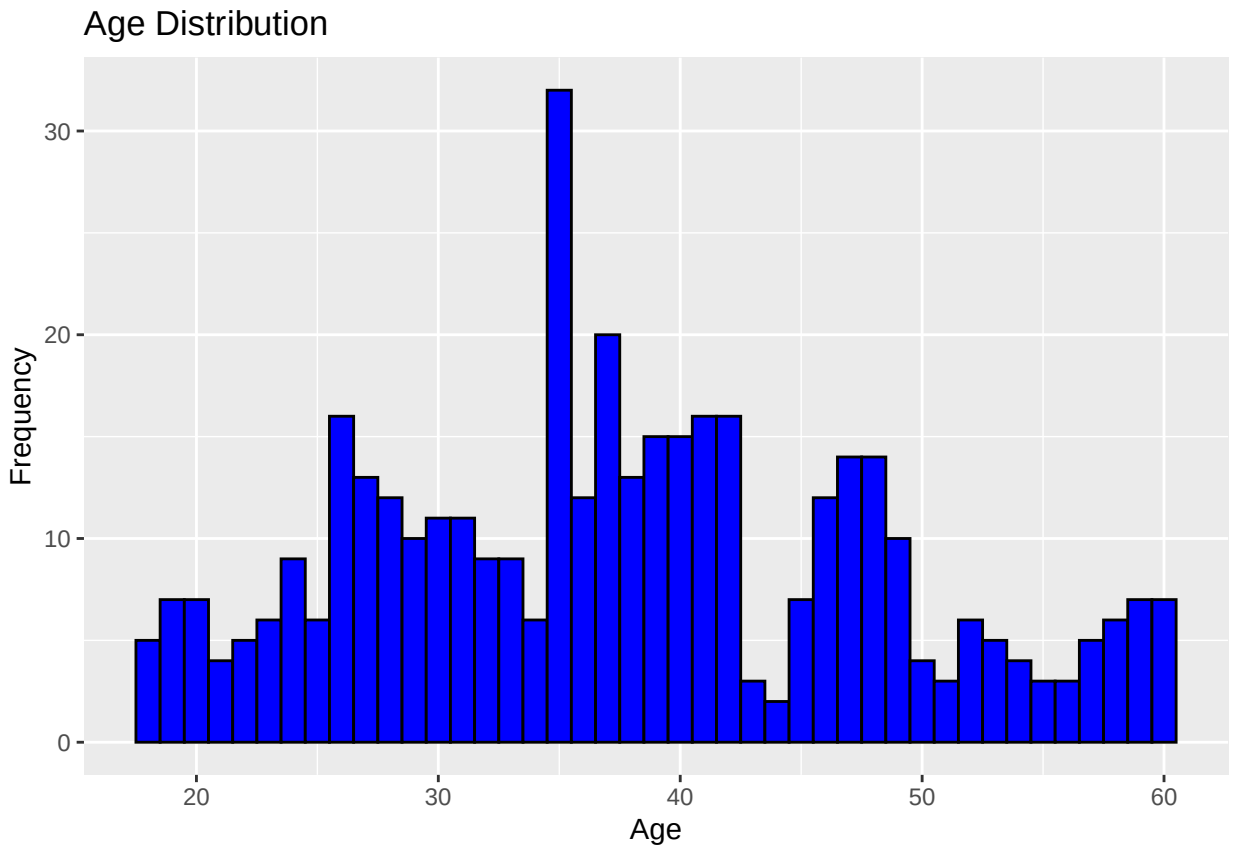
```
# Correlation analysis for numerical variables

cor(dataset[, c("Age", "EstimatedSalary", "Purchased")])
```

```
##                     Age  EstimatedSalary  Purchased
```

```
##  Age                 1.0000000          0.155238  0.6224542
##  EstimatedSalary  0.1552380             1.000000  0.3620830
##  Purchased           0.6224542          0.362083  1.0000000
```

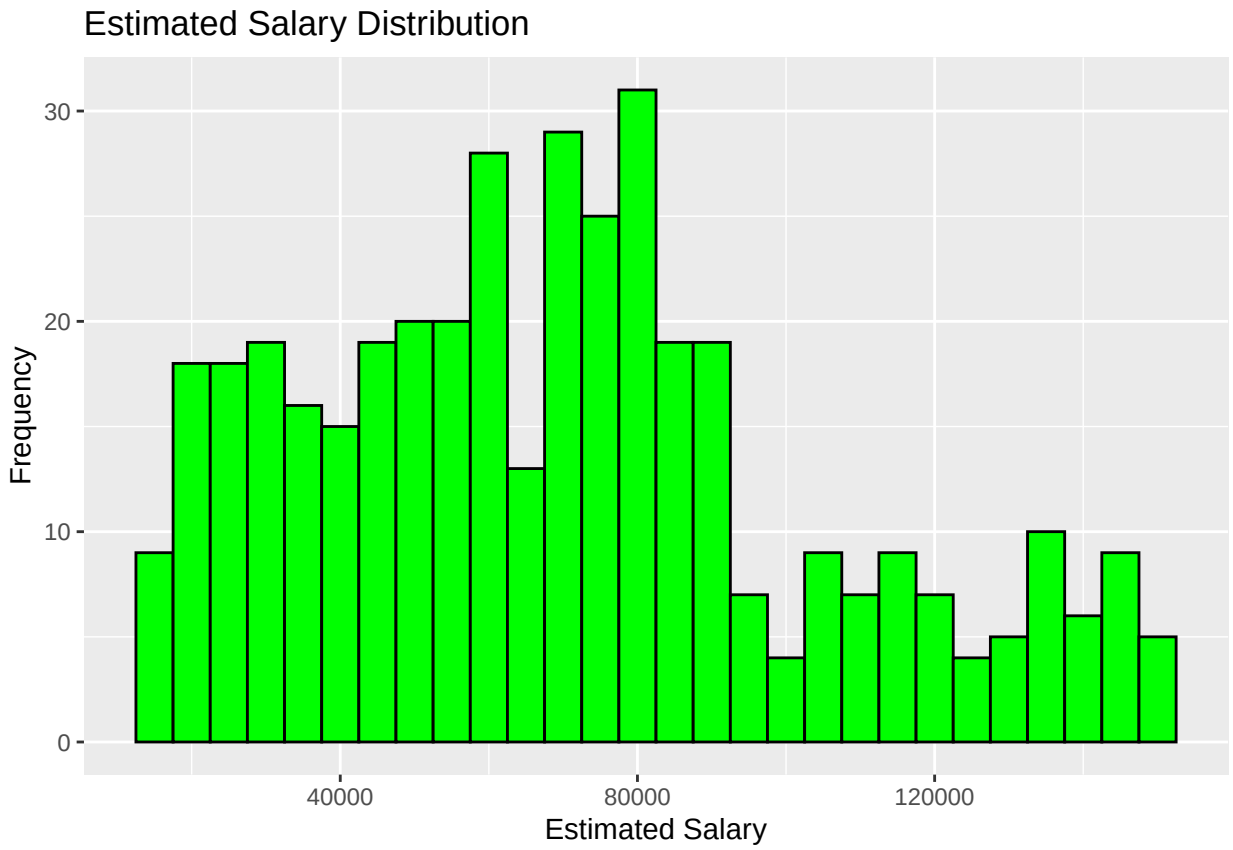
Data Visualization

```
# Visualization of the age distribution
ggplot (dataset,    aes( x = Age))  +
  geom_histogram( binwidth =  1,  fill =    "blue" ,  color =  "black" )  +
  ggtitle   ("Age Distribution"    )  +
  xlab ("Age")  +
  ylab ("Frequency" )
```



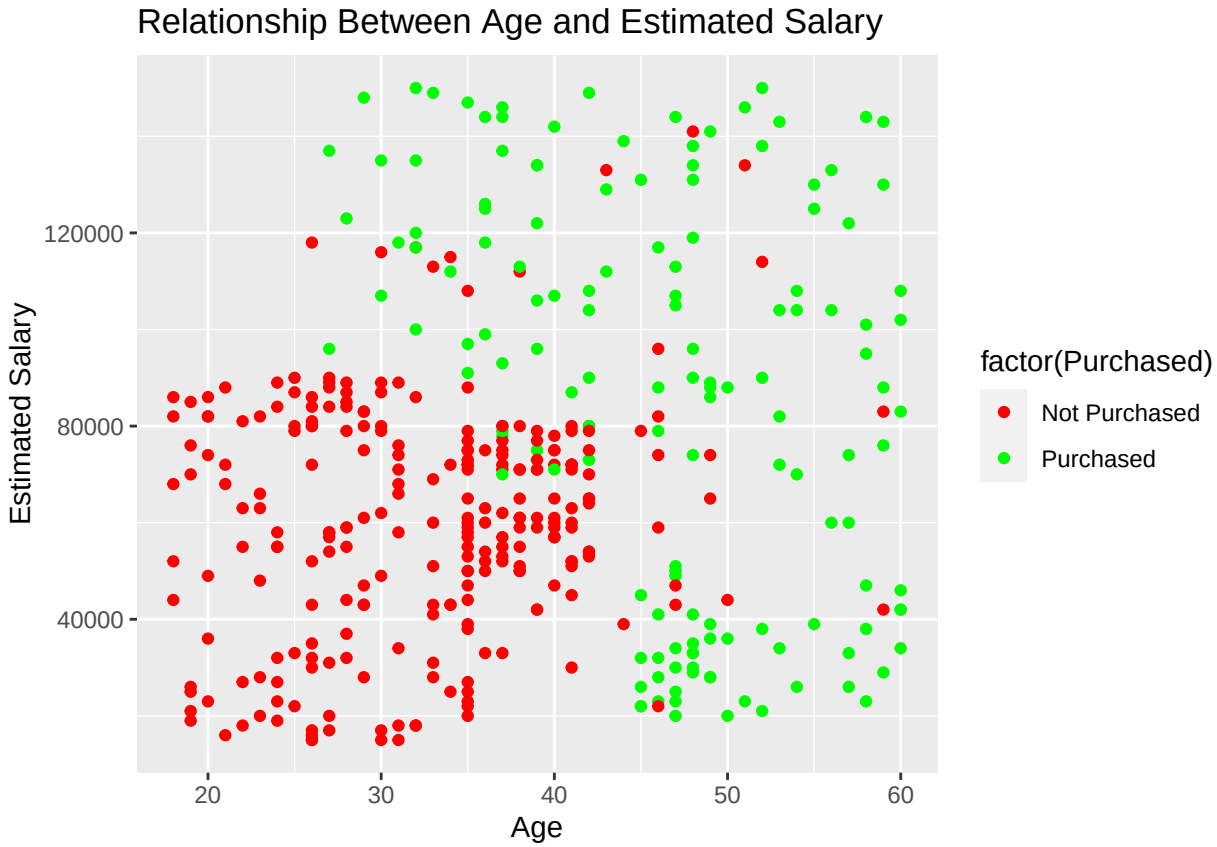
The histogram illustrates the age distribution within the dataset. It reveals that the most common age group appears to be between 30 and 40 years, as indicated by the peak in the frequency of individuals in that age range. The distribution is somewhat right-skewed, with fewer individuals in the older age groups.

```
# Visualization of the estimated salary distribution

ggplot (dataset,    aes( x = EstimatedSalary))    +
  geom_histogram( binwidth =  5000,  fill =    "green" ,  color =  "black" )  +
  ggtitle   ("Estimated Salary Distribution"      )  +
  xlab ("Estimated Salary"   )  +
  ylab ("Frequency" )
```



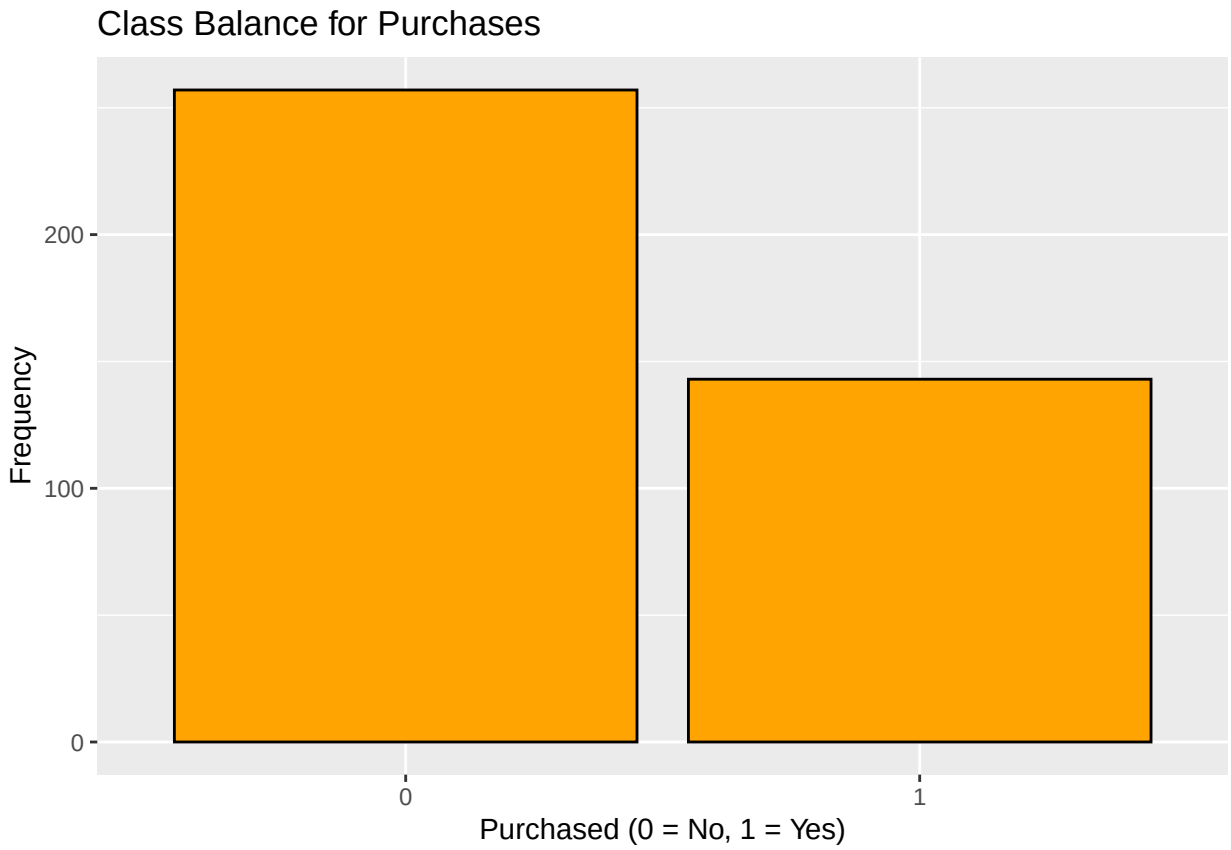
The histogram represents the distribution of estimated salaries in the dataset. It highlights that the majority of individuals have estimated salaries within the lower to mid-range values. There is a notable peak in the frequency of individuals with salaries around $40,000 to $60,000. The distribution is slightly right-skewed, with fewer individuals earning higher estimated salaries.

```
# Scatter plot to visualize the relationship between age and estimated salary
ggplot(dataset, aes(x = Age, y = EstimatedSalary, color = factor(Purchased))) +
  geom_point() +
  ggtitle("Relationship Between Age and Estimated Salary") +
  xlab("Age") +
  ylab("Estimated Salary") +
  scale_color_manual(values = c("red", "green"), labels = c("Not Purchased", "Purchased"))
```



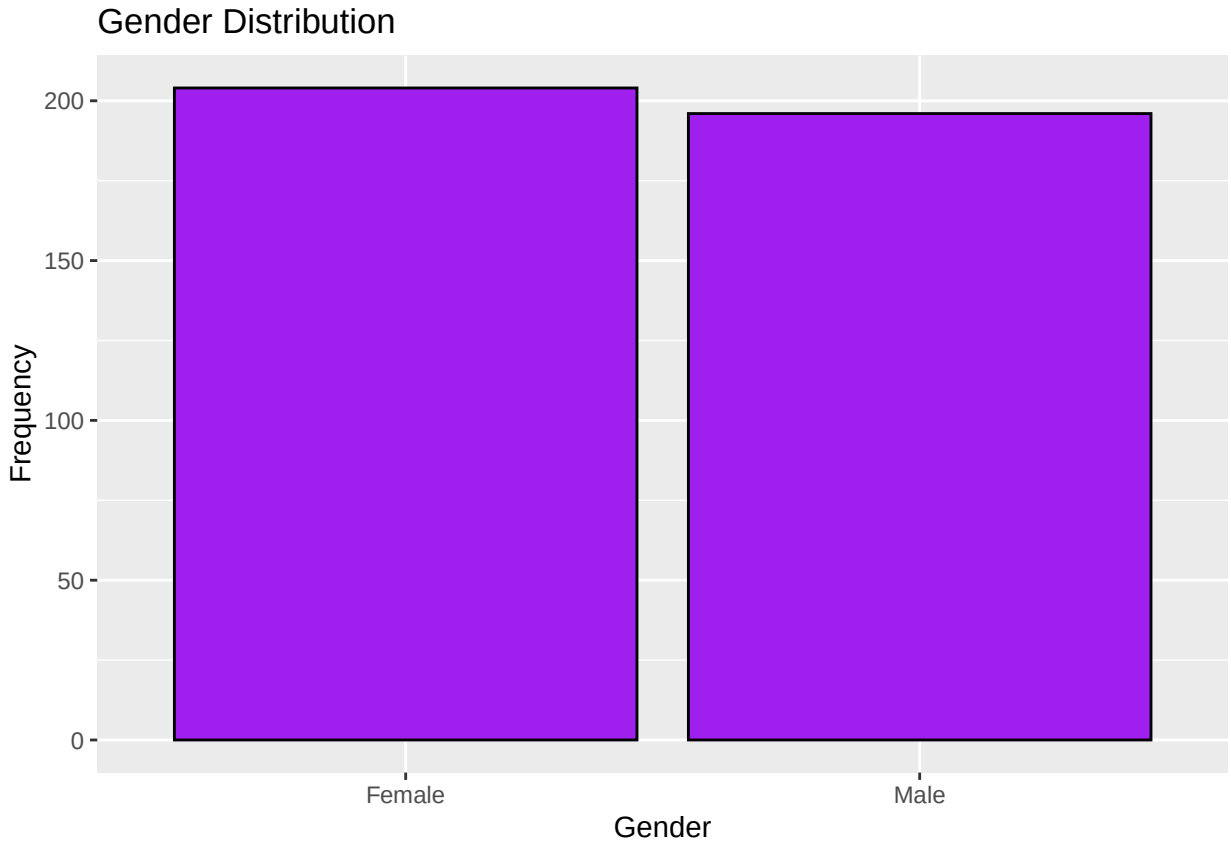
The scatter plot illustrates the relationship between age and estimated salary, with points color-coded to distinguish between individuals who made a purchase (“Purchased”) and those who did not (“Not Purchased”). It reveals that younger individuals, generally those under 40 years of age, tend to have lower estimated salaries. There is a visible divide in purchasing behavior, with younger individuals having a higher concentration in the “Purchased” group. This visualization suggests that age and estimated salary are important factors to consider in predicting purchasing decisions, indicating the potential e ectiveness of targeted marketing strategies based on age and income levels.

```
# Visualization of class balance for the target variable 'Purchased'
ggplot (dataset,    aes( x = factor  (Purchased)))    +
   geom_ba( fill =     "orange" ,  color =   "black"  )  +
   ggtitle   ("Class  Balance  for  Purchases"    )  +
   xlab ( "Purchased  (0  =  No,  1  =  Yes)" )  +
   ylab ( "Frequency" )
```



The bar chart highlights the class balance for the target variable 'Purchased,' where "0" represents individuals who did not make a purchase (No) and "1" represents those who made a purchase (Yes). It's evident from the chart that there is an imbalance in the classes, with a higher frequency of "Not Purchased" instances (0) compared to "Purchased" instances (1).

```
# Visualization of gender distribution
ggplot(dataset, aes(x = Gender)) +
  geom_bar(fill = "purple", color = "black") +
  ggtitle("Gender Distribution") +
  xlab("Gender") +
  ylab("Frequency")
```



The dataset appears to have a relatively balanced representation of both genders, with neither “Male” nor “Female” dominating the other.

Model Training and Evaluation

```
# Select relevant columns: Age, EstimatedSalary, and Purchased
dataset <- dataset[3:5]

# Encode the 'Purchased' variable as a factor
dataset$Purchased <- factor(dataset$Purchased, levels = c(0, 1))

# Split dataset into Training and Test sets with a 75% split ratio
set.seed(123)  # Ensure reproducibility
split <- sample.split(dataset$Purchased, SplitRatio = 0.75)
training_set <- subset(dataset, split == TRUE)
test_set <- subset(dataset, split == FALSE)

# Apply feature scaling to the Age and EstimatedSalary columns
training_set[-3] <- scale(training_set[-3])
test_set[-3] <- scale(test_set[-3])

# Define a function to evaluate and return model performance metrics
evaluate_model <- function(predictions, actual) {
  cm <- confusionMatrix(as.factor(predictions), as.factor(actual))
  return(list(accuracy = cm$overall['Accuracy'], kappa = cm$overall['Kappa']))
}
```

```r
# Fit a K-Nearest Neighbors (KNN) model to the training data
knn_pred <- knn(train = training_set[, -3], test = test_set[, -3], cl = training_set[, 3], k = 5)

# Fit a Random Forest classifier to the training data
rf_classifier <- randomForest(x = training_set[-3], y = training_set$Purchased, ntree = 500)

# Use the fitted Random Forest classifier to make predictions on the test set
rf_pred <- predict(rf_classifier, newdata = test_set[-3])

# Evaluate the performance of the KNN model
knn_performance <- evaluate_model(knn_pred, test_set$Purchased)

# Evaluate the performance of the Random Forest model
rf_performance <- evaluate_model(rf_pred, test_set$Purchased)
```

Results

```r
# Print out accuracy and kappa statistics for both models
cat("K-NN Accuracy:", knn_performance$accuracy, "Kappa:", knn_performance$kappa, "\n")
```

```
## K-NN Accuracy: 0.89 Kappa: 0.7598253
```

```r
cat("Random Forest Accuracy:", rf_performance$accuracy, "Kappa:", rf_performance$kappa, "\n")
```

```
## Random Forest Accuracy: 0.85 Kappa: 0.6764452
```

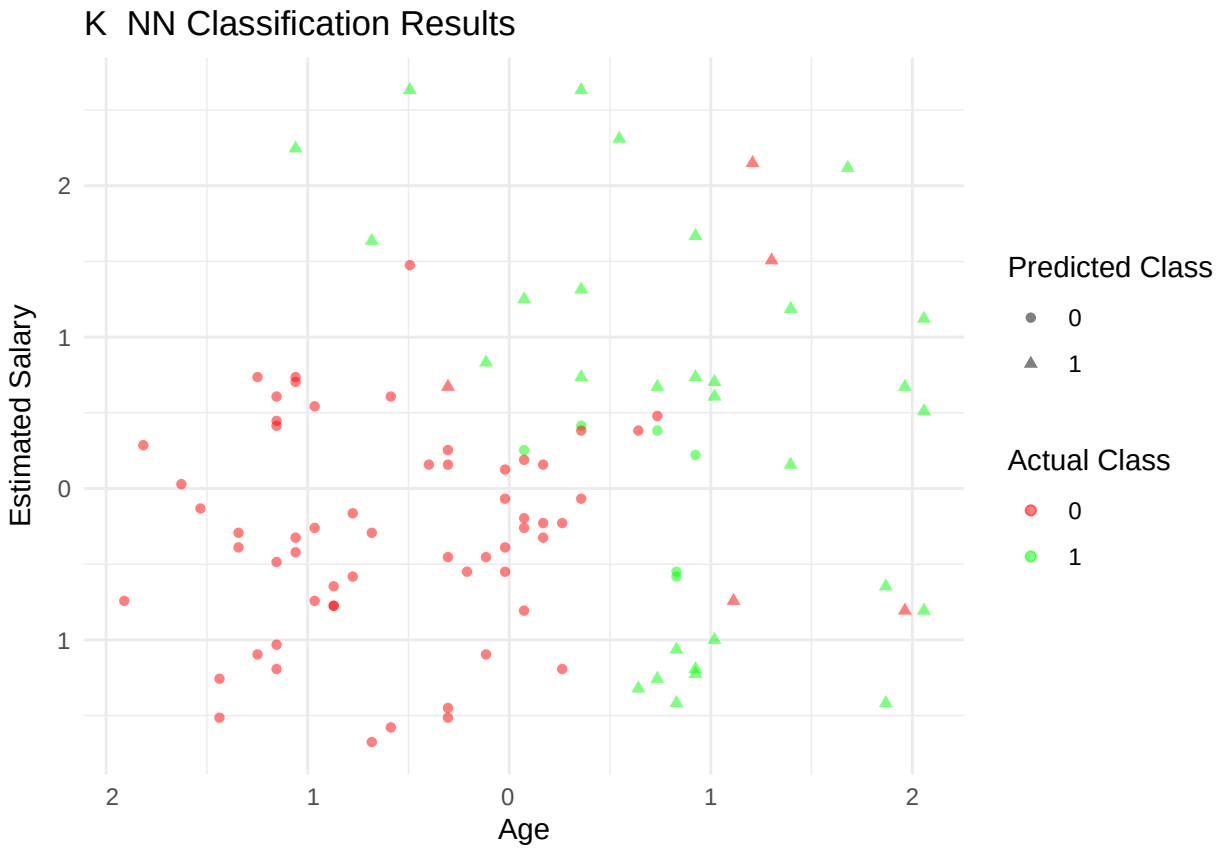
```r
# Determine which model is more efficient based on accuracy
efficient_model <- ifelse(knn_performance$accuracy > rf_performance$accuracy, "K-NN",
                          ifelse(knn_performance$accuracy < rf_performance$accuracy, "Random Forest", "E
cat("The more efficient model based on accuracy is:", efficient_model, "\n")
```

```
## The more efficient model based on accuracy is: K-NN
```

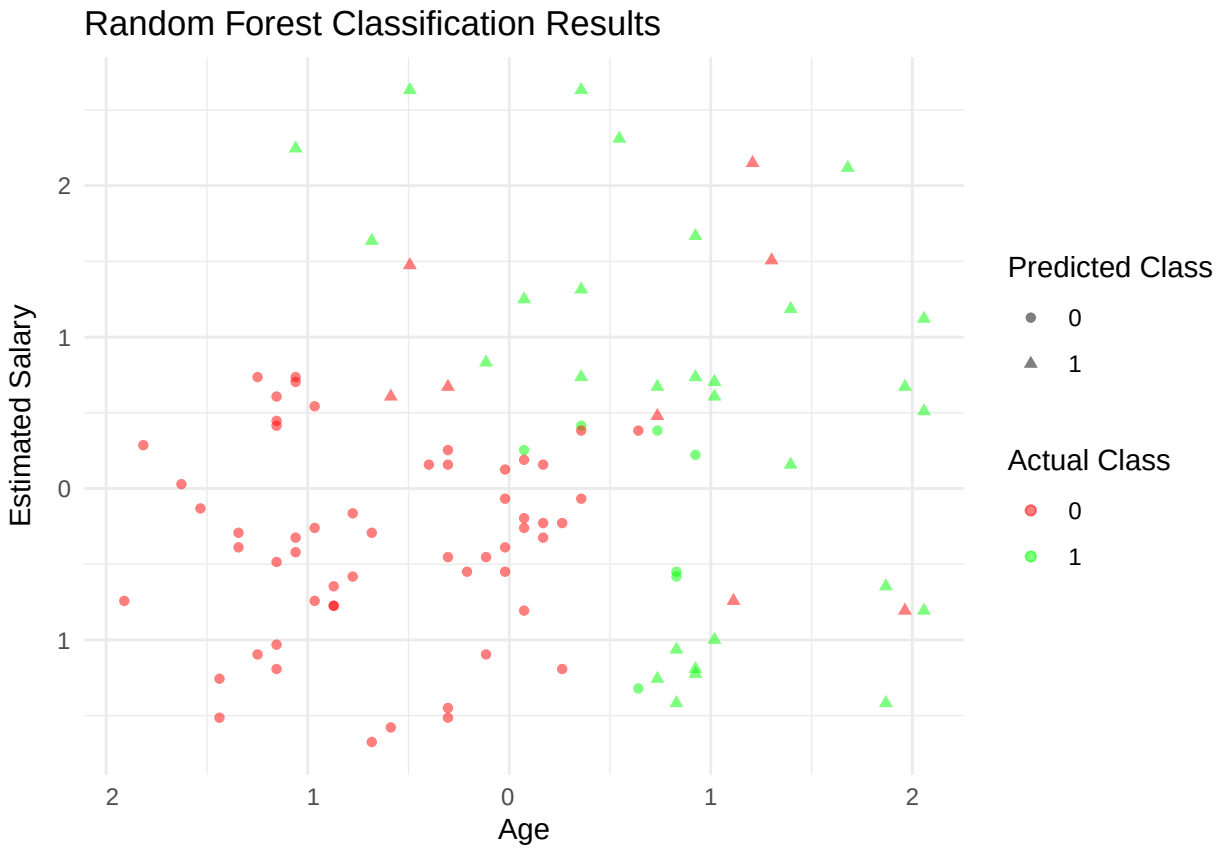
```r
# Compare models based on Kappa statistic
best_kappa_model <- ifelse(knn_performance$kappa > rf_performance$kappa, "K-NN",
                           ifelse(knn_performance$kappa < rf_performance$kappa, "Random Forest", "Both" )
cat("The model with the better Kappa score is:", best_kappa_model, "\n")
```

```
## The model with the better Kappa score is: K-NN
```

```r
# Function to plot model results using ggplot2
plot_model_results <- function(test_set, predictions, model_name) {
  test_set$Predicted <- as.factor(predictions)
  ggplot(test_set, aes(x = Age, y = EstimatedSalary, color = Purchased, shape = Predicted)) +
    geom_point(alpha = 0.5) +
    scale_color_manual(values = c('red', 'green')) +
    scale_shape_manual(values = c(16, 17)) +
    labs(title = paste(model_name, "Classification Results"), x = "Age", y = "Estimated Salary",
         color = "Actual Class", shape = "Predicted Class") +
    theme_minimal()
}

# Create and print visualizations for both models
plot_model_results(test_set, knn_pred, "K-NN")
```




```
plot_model_results (test_set, rf_pred, "Random Forest")
```



The visualizations provide a comparative view of the classiÅcation results for two di erent machine learning models, K-Nearest Neighbors (K-NN) and Random Forest.

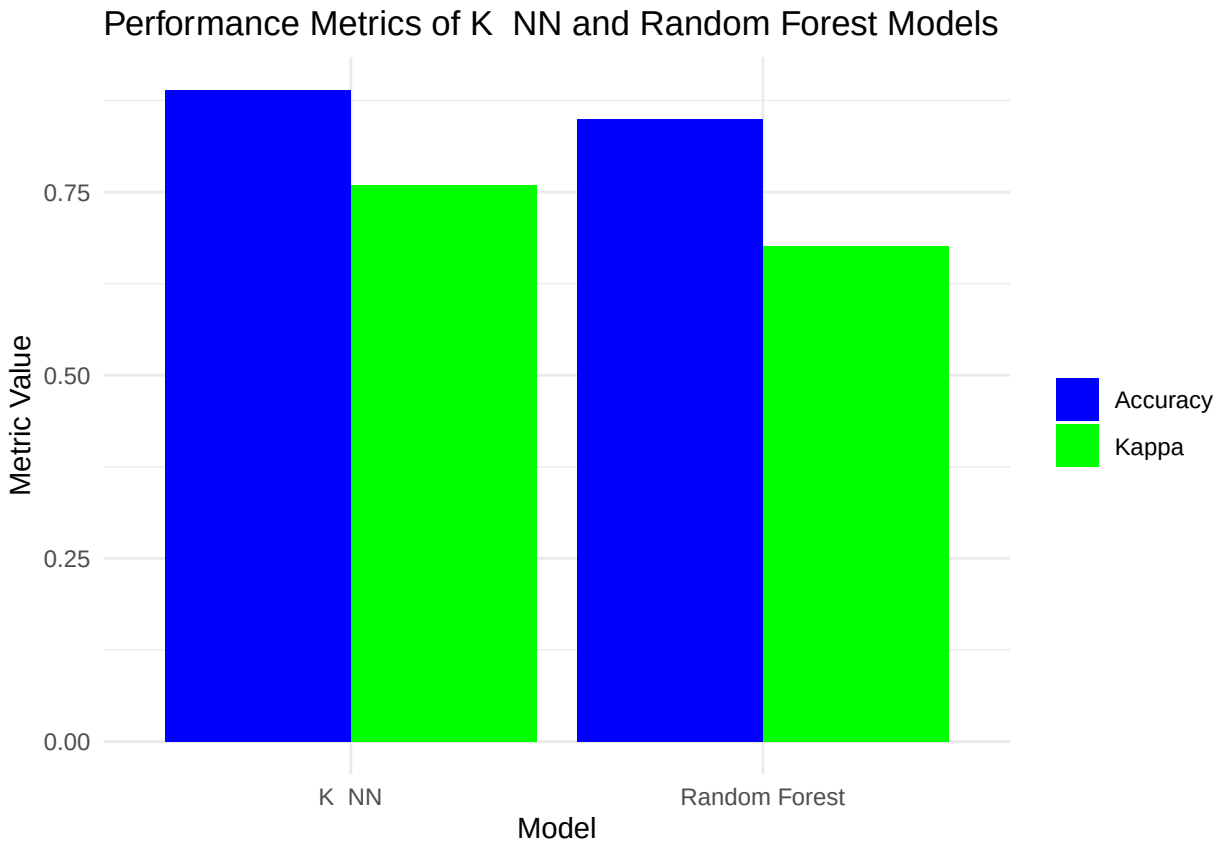
1. K-NN ClassiÅcation Results:
   - Red points represent actual non-purchases, while green points represent actual purchases.
   - The plot illustrates the K-NN model's ability to distinguish between these two classes based on age and estimated salary.
   - The chart conveys the model's strengths and weaknesses in classifying users' purchasing behavior.
2. Random Forest ClassiÅcation Results:
   - Similar to the K-NN plot, this chart shows actual non-purchases (red) and actual purchases (green) based on the Random Forest model's predictions.
   - It provides insights into the performance of the Random Forest model in predicting user purchases.
   - A comparison of both charts helps evaluate and contrast the two models' classiÅcation capabilities.

These visualizations aid in understanding how well each model performs in classifying users' purchasing behavior, facilitating model selection and further analysis.

Performance Metrics Comparison

```
# Combine the performance metrics into a data frame for plotting
performance_data <- data.frame(
  Model = c("K-NN", "Random Forest"),
  Accuracy = c(knn_performance$accuracy, rf_performance$accuracy),
  Kappa = c(knn_performance$kappa, rf_performance$kappa)
)
```

```r
# Melt the data into a long format for ggplot2
performance_melted <- melt(performance_data, id.vars = "Model", variable.name = "Metric", value.name = "

# Plot the performance metrics in a bar chart
performance_plot <- ggplot(performance_melted, aes(x = Model, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  scale_fill_manual(values = c("Accuracy" = "blue", "Kappa" = "green")) +
  labs(title = "Performance Metrics of K-NN and Random Forest Models", x = "Model", y = "Metric Value") +
  theme_minimal() +
  theme(legend.title = element_blank())

# Print the performance plot
performance_plot
```



The bar chart comparing the performance metrics of the K-NN and Random Forest models provides a clear visual representation of their accuracy and Kappa values. It is evident that the K-NN model outperforms the Random Forest model in both accuracy and Kappa score. This visualization underscores the superior performance of the K-NN model in this speciÅc analysis, o ering valuable insights for model selection.

Analysis of Results

Model Training and Prediction Two models were trained: K-NN with k = 5 and a Random Forest with 500 trees. Predictions were made on the test set.

Performance Metrics The K-NN model achieved an accuracy of 89% and a Kappa score of 0.76, while the Random Forest model showed an accuracy of 85% and a Kappa score of 0.68. Thus, K-NN performed better on both metrics.

Visualizations The results were visualized using `ggplot2`, illustrating the classification accuracy of both models. Furthermore, a bar chart compared the performance metrics.

Conclusion

The analysis indicates that the K-NN model outperforms the Random Forest in this scenario, which could be due to the nature of the data or the specific configuration of the models. The models' predictive power could assist in tailoring marketing strategies to potential buyers, thereby increasing conversion rates.

Limitations

The study's limitations include potential overfitting, the simplicity of the models, and the exclusion of other predictive factors such as user engagement.

Future Work

Future analyses could incorporate additional variables, apply hyperparameter tuning, and test ensemble methods to enhance predictive performance.

Appendix

The Kappa score

The Kappa score, also known as Cohen's Kappa, is a statistical measure of inter-rater reliability (or agreement) for categorical items. It is generally thought to be a more robust measure than simple percent agreement calculation since Kappa takes into account the agreement occurring by chance.

Here's what the Kappa score indicates:

- Value of 1: This represents perfect agreement between the two sets of ratings.
- Value of 0: This indicates that the agreement is no better than what would be expected by chance.
- Value less than 0: This suggests less agreement than would be expected by random chance.

In the context of predictive modeling, the Kappa score is used to compare the observed accuracy with the expected accuracy (random chance). It is a useful metric when dealing with imbalanced classes in the dataset, which can make metrics like accuracy less informative. For example, in a dataset where 90% of the instances belong to a single class, a model that always predicts the majority class will have high accuracy but won't be particularly useful.

The formula to calculate Cohen's Kappa is:

$$\ddot{Y} = \frac{p_o \quad p_e}{1 \quad p_e}$$

where p_o is the relative observed agreement among raters (i.e., the accuracy of the classifier), and p_e is the hypothetical probability of chance agreement.

Using the confusion matrix to calculate Cohen's Kappa, p_e can be calculated by adding the products of the sums of each row and column, then dividing by the square of the total number of observations. This reflects the probability that raters agree by just guessing.

Kappa is an important metric in machine learning for classification problems, especially where the classes are imbalanced or the cost of different types of errors varies significantly.

The confusion matrix

A confusion matrix is a table often used to describe the performance of a classiÅcation model (or “classiÅer”) on a set of data for which the true values are known. It is particularly useful for summarizing the performance of a classiÅcation algorithm.

Here’s how a confusion matrix looks for a binary classiÅcation problem:

		Predicted Positive	Predicted Negative
Actual	Positive	TP	FN
	Negative	FP	TN

- True Positives (TP) : These are cases in which we predicted positive, and the actual output was also positive.
- True Negatives (TN): We predicted negative, and the actual output was also negative.
- False Positives (FP) , also known as Type I error: We predicted positive, but the actual output was negative.
- False Negatives (FN) , also known as Type II error: We predicted negative, but the actual output was positive.

The confusion matrix allows us to calculate various performance metrics, such as:

- Accuracy: Overall, how often the classiÅer is correct.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

- Precision (or Positive Predictive Value): When it predicts positive, how often is it correct?

$$\text{Precision} = \frac{TP}{(TP + FP)}$$

- Recall (or Sensitivity or True Positive Rate): How many of the actual positives did it capture through labeling it positive?

$$\text{Recall} = \frac{TP}{(TP + FN)}$$

- F1 Score: A weighted average of precision and recall.

$$\text{F1 Score} = 2 \times \frac{(\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})}$$

- SpeciÅcity (or True Negative Rate): How many of the actual negatives did it capture through labeling it negative?

$$\text{Specificity} = \frac{TN}{(TN + FP)}$$

In multi-class classiÅcation problems, the confusion matrix can be extended to include all classes. For each class, you can calculate these metrics, and then you can Ånd an average metric across classes, often using either the macro or weighted average approach.

The confusion matrix is a powerful tool because it not only gives you insights into the errors being made by your classiÅer but also types of errors that are occurring. This can guide you toward speciÅc aspects of the problem that need more attention and can inform you about the potential trade-o s between di erent types of errors for a particular application.