DSE 6311 – Data Science Capstone

Team Gamma

**Team Lead:** Canfield, Ryan

canfieldr@merrimack.edu

**Reporter:** Sabri, Abdelmalek

sabria@merrimack.edu

**Spokesperson:** Crozier, Amber

croziera@merrimack.edu

EVALUATING AI’s ROLE IN CUSTOMER SATISFACTION AND RETENTION

Preprocessing & Feature Engineering Report

Table of Contents

[**Background & Question 2**](#_gjdgxs)

[Question: 2](#_30j0zll)

[Hypothesis and Prediction 2](#_3dy6vkm)

[Hypothesis: 2](#_1t3h5sf)

[Prediction: 2](#_4d34og8)

[**Methods 2**](#_2s8eyo1)

[Preprocessing and Feature Engineering Plans: 2](#_17dp8vu)

[Methods for Preprocessing: 2](#_3rdcrjn)

[Methods for Feature Engineering: 3](#_35nkun2)

[**Results 3**](#_z337ya)

[Justification: 3](#_3j2qqm3)

[All Cleaning. 3](#_j6hz0c31emij)

[Collapsing of Variables. 3](#_ni059w6x2mgm)

[Encoding. 3](#_67qume8yrycn)

[Mathematical Transformations. 4](#_y4fs4tb0kovw)

[Preprocessing Steps: 4](#_1y810tw)

[Unsupervised Learning: 4](#_oemaymbq6kko)

[Supervised Method Plans: 5](#_rw1kt6k7fdv8)

[**Discussion and Next Steps 7**](#_6cjx1tk0ob6n)

[Summary of Key Takeaways: 7](#_b1bqdpl5ukl0)

[Modeling Plan: 8](#_eu5jkuuxy2rz)

[GitHub Link: 8](#_4i7ojhp)

[**References 9**](#_2xcytpi)

[**Appendix 10**](#_m3adjyvj41xm)

[Data Dictionary: 10](#_3rdcrjn)

# Background & Question

## Question:

Can we predict customer satisfaction with AI content to help businesses identify the type of customer base most likely to have a positive experience with AI tools they deploy?

## Hypothesis and Prediction

### Hypothesis:

While implementing AI in retail settings improves cost efficiency and reduces errors (Kasareni, 2021), excessive reliance on AI in customer service without human interaction will reduce customer satisfaction.

### Prediction:

Customer satisfaction with AI-driven customer service will be higher among specific demographic groups and for certain AI tools.

# Methods

## Preprocessing and Feature Engineering Plans:

Last week, we outlined a plan to refine our data cleaning and preprocessing process which included an emphasis on encoding the categorical variables, specifically our target variable AI\_Satisfaction. This allows us to run a wider variety of models as well as keep consistency in the way things are interpreted, avoiding things like having variables become ordered, and in some cases allowing orderedness. Also, a decent amount of time was spent on our test-train data set split. This focus allowed us to prepare a relevant split which we will then use in the next phase when building our models.

## Methods for Preprocessing:

Our preprocessing strategy focused on encoding categorical variables and creating a test-train split to optimize the data for machine learning models while preserving relevant information from the original dataset. We applied frequency encoding to the "Country" variable, which allowed us to capture geographical differences in customer preferences, specifically around AI usage and trust, as observed during our exploratory data analysis (EDA).

For the other categorical variables, we used binary encoding to reduce dimensionality without compromising the quality of the data. This decision was based on the need to keep the number of features manageable (under 25 columns) due to having a relatively small dataset of 634 observations. Standard techniques like one-hot encoding would have resulted in an excessively large number of columns, which could lead to overfitting and increased model complexity. Binary encoding allowed us to balance the complexity while also maintaining relevant information. After encoding the categorical variables, we split the dataset into training and testing sets using a 78:22 ratio, which was determined using the calcSplitRatio function (Geist, 2019) to ensure an optimal split based on the number of predictors.

## Methods for Feature Engineering:

For our project we created two new variables called AI\_Trust and AI\_Usage. To make the new AI\_Trust variable, we had to mutate and manipulate a few existing variables in our dataset. Originally we had a feature called “AI\_Privacy\_No\_Trust”. This feature showed if the respondent had any privacy concerns or lack of trust in AI. We wanted the opposite of this for our new variable. So we inverted the values making a new AI\_Privacy\_Trust feature where if the value was 0, it changed to 1, and if the value was 1, it changed to 0. Next we took a column called AI\_Endorsement and used it to make the equation for our new feature. This equation is (AI\_Trust = (AI\_Privacy\_Trust + 1 + AI\_Endorsement + 1) / 2). After we executed the calculation, it gave us three values 1, 1.5 or 2. If the AI\_Trust value was equal to one, then the respondent was considered to have a low trust with AI. If the AI\_Trust value was equal to one and a half, then the respondent was considered to have a moderate trust with AI, and finally AI\_Trust value was equal to two, then the respondent was considered to have a high trust with AI. Lastly the two columns that were used to create the new column (AI\_Privacy\_Trust and AI\_Endorsement) were dropped.

The next variable that we created by feature engineering is called AI\_Usage. To make this variable we took the three types of commonly used AI found in websites, which are chatbots, virtual assistants, and Voice/Photo AI, and added them together if they were included on the site. If all three were used a “high” usage value was given. If two were used then a “moderate” usage value was given, and if only one was used then a “low” usage value was given.

The reason why we created these two variables was due to the fact that AI\_Privacy\_No\_Trust and AI\_Endorsement were highly correlated so we removed them to reduce multicollinearity. We decided to make AI\_Usage to see if there was a difference in satisfaction depending on if they used all of them, none of them, or only some. We left the three variables that were used to make this new variable in because we still wanted to see which ones were used.

# Results

## Justification:

### All Cleaning.

Almost all of the cleaning done was to fix typos or create names that were easier to read/understand. This does not affect the models at all, it just makes it easier and more straightforward for us to understand.

### Collapsing of Variables.

We decided on dropping two variables because of their high correlation which can lead to multicollinearity which is bad for our machine learning models.

### Encoding.

For our encoding, we did three different types. For all the categorical variables which were yes or nos we performed label encoding making them turn into a 1 or a 0. For the variables where there was a ranking like Age (Gen Z, Millennials, Gen X, Baby Boomer), and Salary (Low, Medium, High) we performed Ordinal Encoding to impose an order on each of the values. Finally, we did frequency encoding on our country variable because it helps to reduce dimensionality by not recreating a ton of extra columns like what would happen with one hot encoding and it also does not impose any implicit ordinal relationship between the countries.

### Mathematical Transformations.

Due to the nature of our data being all categorical we decided to standardize our data because the majority of our data which are 0 or 1 should not be transformed, as they already represent a meaningful range. We have some of the features that have higher values and others have lower 0 or 1, standardization will scale these features to have a mean of 0 and a standard deviation of 1, ensuring they contribute equally in any distance-based algorithms like SVM or KNN.

## Preprocessing Steps:

We implemented a comprehensive preprocessing strategy to optimize our dataset for machine learning models while maintaining the integrity of the data. One key step was splitting the dataset into training and testing sets using a 78:22 ratio, determined by the calcSplitRatio function (Geist, 2019). This specific split was chosen because it maximizes the training data available for model learning while retaining a sufficient portion of the dataset for unbiased evaluation. Given our dataset’s size of 634 observations, this balance ensures robust model performance without overfitting. To further validate our decision, summary statistics confirmed that both the training and testing sets retained comparable distributions for key variables such as AI\_Satisfaction, AI\_Trust, and AI\_Usage. These statistics affirm that the split preserves the representativeness of the original dataset, enabling accurate model evaluation.

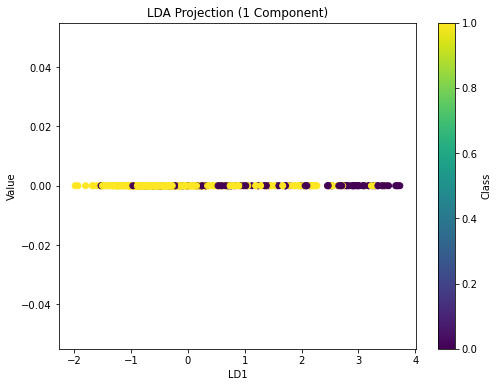
Handling categorical variables was another critical aspect of our preprocessing workflow. We addressed the challenges of dimensionality and information preservation through tailored encoding strategies. For example, we applied binary encoding to categorical variables to reduce the number of features without compromising the quality of the data. This method was particularly effective given our dataset's limited size, as it kept the feature count below 25 columns while retaining essential information. Additionally, frequency encoding was employed for the "Country" variable, leveraging insights from our exploratory data analysis (EDA) to capture geographical differences in customer preferences related to AI usage and trust. Label encoding and ordinal encoding were also selectively applied to handle binary variables and ranked categories, such as age groups and salary ranges, respectively. Together, these preprocessing decisions not only optimized the dataset for machine learning but also ensured consistency and interpretability across features. Summary statistics and visualizations confirmed that these techniques effectively preserved the dataset’s key characteristics, paving the way for meaningful analysis and model development.

## Unsupervised Learning:

For our feature engineering and dimensionality reduction, we employed two methods: Linear Discriminant Analysis (LDA) and K-Nearest Neighbors (KNN). We chose LDA because we had a categorical target variable and pre-existing labels.This meant that it was an appropriate technique for pulling out any underlying patterns or hidden features from our data. After applying LDA, we obtained a single component since we only had two classes in our target variable. The results are shown in the figure below. *Figure 1* shows all the continuous values the LDA came up with ranging from -2 to 4. To Make things simple and to leave no value undefined we applied a threshold of 0, categorizing values below zero as 0 and values above zero as 1.

Next, we used KNN to identify the top two clusters by calculating the densities of each cluster. We selected these top two clusters and added their information as two new columns to the dataset. In total, we generated three new features: one from LDA (the LDA component) and two from KNN (representing their cluster information). These new features will hopefully capture underlying patterns that were not apparent from exploratory data analysis. We believe that including these features will provide additional insights when applied to our machine learning models.

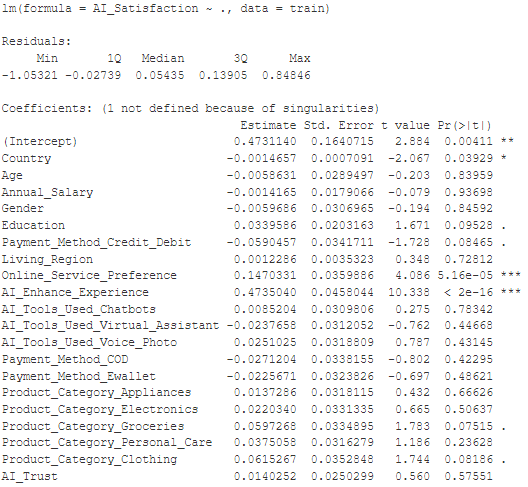
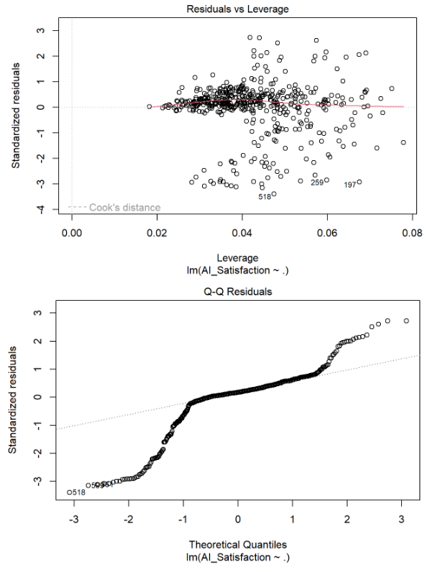
After these transformations, our dataset now contains 25 columns and over 600 rows, still within the bounds of the square root rule for feature selection (SQRT(number of rows)).



*Figure 1 Linear Discriminant Analysis (LDA)*

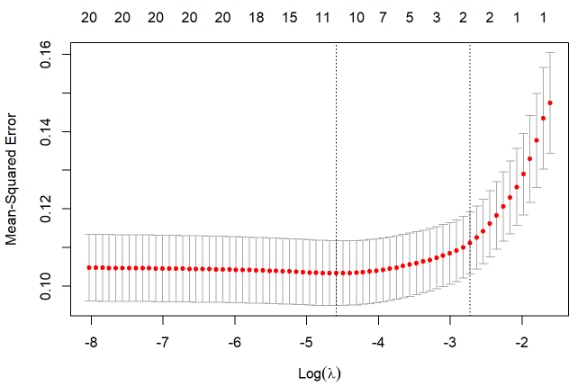
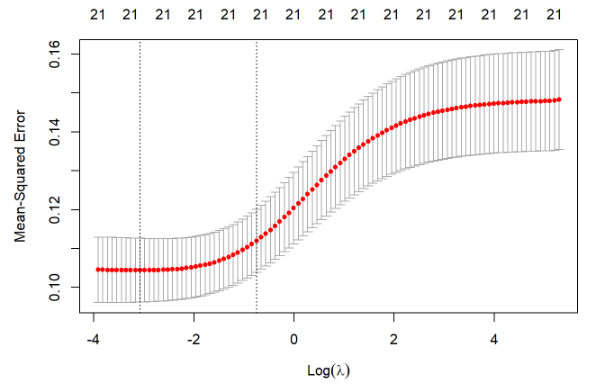
## Supervised Method Plans:

We explored feature engineering using supervised methods to identify the most relevant features while maintaining the integrity of the data and allowing for actionable insights. Once our categorical variables were encoded, we ran some supervised methods to identify features that contribute significantly to predicting AI\_Satisfaction. The first method we utilized was logistic regression to assess the importance of different variables *(Figure 2).*

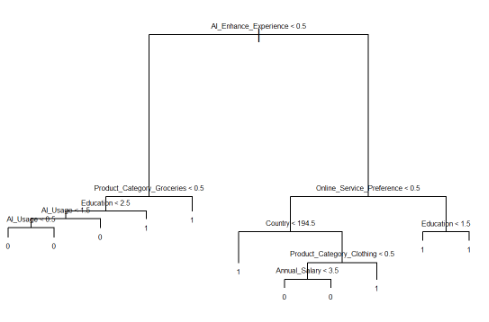
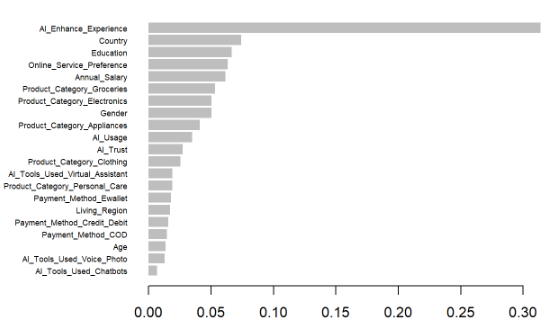
*Figure 2 Logistic Regression Analysis*

Ridge regression helps address overfitting and could be useful because we have many features that contribute weakly to the target variable (AI satisfaction). Ridge regression can allow us to keep all our features which may be more valuable to our stakeholders. The cross-validation graph from the ridge regression will help us identify the optimal penalty term. *(Figure 3).* Lasso regression, would be useful if we decide to reduce our features to focus on the most impactful predictors. The visualization of the cross-validation process (via cv.out2) shows the lambda values that provide the best trade-off between bias and variance *(Figure 4).*



*Figure 3 Ridge Regression Analysis Figure 4 Lasso Regression Analysis*

We then utilized tree-based methods, such as XGBoost and Classification Trees, to assess feature importance. A feature importance plot generated from XGBoost allowed us to visually represent which variables have the most influence on the model’s decisions *(Figure 5).* Ranking features based on their contribution to the model’s accuracy could show the value of retaining features that can provide actionable insights. Classification Trees provided a visual way to understand the relationship between the predictors and our target variable. For example, if AI\_Enhance\_Experience is low and Product\_Category\_Groceries is high, dissatisfaction is more likely *(Figure 6).*

****

*Figure 5 XGBoost Analysis of feature importance Figure 6 Classification tree analysis of feature importance*

We need to balance the complexity of the model with the relevance of the features in providing actionable business insights. Reducing features could help prevent overfitting but we want to avoid eliminating variables that could be valuable for understanding customer satisfaction with AI tools. Variables like Online Service Preference, AI Enhance Experience, and Education have significant coefficients but even though some features may have higher p-values or show weaker individual relationships with AI satisfaction (such as Age, Gender, or Living Region), these variables could have practical value when they are combined with other features in a broader context. Reducing or eliminating features might discard variables that might have more actionable insights for our stakeholders. Our decision to keep or reduce features should balance statistical performance with business goals.

# Discussion and Next Steps

## Summary of Key Takeaways:

Our analysis demonstrates that customer satisfaction with AI content can be predicted by utilizing preprocessing, feature engineering, and supervised/unsupervised machine learning techniques. Our group's focus is on whether or not we could predict customer satisfaction to identify which customer bases are most likely to have positive experiences with AI in retail environments. Our initial hypothesis was that excessive AI usage without human interaction would lead to a decreased overall customer satisfaction.

Some important insights that we have found is through the use of our engineered variables which are AI\_Trust and AI\_Usage which capture the customer attitudes and usage patterns. By addressing multicollinearity, this has helped our dataset remain interpretable while also keeping its predictability strong. Also, preprocessing methods like binary and frequency encoding streamlined dimensionality while retaining critical information. Our unsupervised learning approaches, such as LDA and KNN clustering, added three insightful features that revealed latent structures in the data. These transformations prepare the dataset for effective modeling and align closely with our original hypothesis, affirming its relevance.

## Modeling Plan:

With preprocessing and feature engineering completed, our next steps involve building and evaluating machine learning models to predict customer satisfaction. Our modeling strategy will focus on both traditional and tree-based methods to achieve an optimal balance of performance and explainability. Initially, we plan to use logistic regression and ridge regression to assess the overall fit of our model and the importance of individual features. Ridge regression, in particular, will address overfitting concerns and allow us to retain features that may hold practical value for stakeholders.

Given the clarity provided by unsupervised techniques like LDA and KNN, these features will be incorporated into supervised models for added depth. The decision on whether to reduce features (using methods like lasso regression) or retain a broader set of predictors will be guided by a tradeoff analysis that prioritizes actionable insights alongside statistical performance. By implementing this plan, we aim to deliver a predictive framework that helps businesses identify and cater to the customer bases most likely to benefit from AI tools.

## GitHub Link:

<https://github.com/Canfieldr/DSE6311-Graduate-DS-Capstone>

# References

Bergen, Mark, and Lynn Doan. “Tech Giants Are Set to Spend $200 Billion This Year Chasing AI.” *Bloomberg.com*, Bloomberg, Nov. 2024, www.bloomberg.com/news/articles/2024-11-01/tech-giants-are-set-to-spend-200-billion-this-year-chasing-ai. Accessed 3 Nov. 2024.

Chollet, F., Kalinowski, T., and Allaire, J. J. (2022). Deep Learning with R. (Second Edition). Manning.

Geist, K.S. (2019) *calcSplitRatio-3*, *GitHub*. Merrimack College. Available at: https://github.com/ksgeist (Accessed: 14 November 2024).

James G., Witten D., Hastie T., & Tibshirani R. (2015). An Introduction to Statistical Learning with Applications in R (2nd ed.). Springer.

Kannan, Rathimala; Ramakrishnan, Kannan; Ersoy, Ayse Begum; Contu, Davide (2023). Customer Satisfaction Response to Artificial Intelligence Tools Usage During Online Shopping. figshare. Dataset. https://doi.org/10.6084/m9.figshare.24633105.v1

Kasaraneni, Ramana Kumar (2021). AI-Enhanced Supply Chain Collaboration Platforms for Retail: Improving Coordination and Reducing Costs. *Journal of Bioinformatics and Artificial Intelligence*, 1(1), pp. 410–450. Available at: https://biotechjournal.org/index.php/jbai/article/view/98 (Accessed: 3 November 2024).

What is the difference between one-hot and dummy encoding? (n.d.) *Data Science Stack Exchange*. Available at: https://datascience.stackexchange.com/ questions/98172/what-is-the-difference-between-one-hot-and-dummy-encoding.

‌

# Appendix

## Data Dictionary:

|  |  |  |
| --- | --- | --- |
| Variable | Data Type | Description |
| AI\_Satisfaction | String | Overall satisfaction level |
| Country | String | Country of the respondent |
| Online\_Consumer | String | Whether the consumer shops online (YES/NO) |
| Age | String | Age group of the respondent (Gen X, Gen Z, etc.) |
| Annual\_Salary | String | Salary range of the respondent (Low, Medium, High) |
| Gender | String | Gender of the respondent (Male/Female) |
| Education | String | Highest education level achieved by the respondent (Masters' Degree, University Graduate, etc.) |
| Payment\_Method\_Credit/Debit | String | Whether the respondent uses credit or debit cards for payments (YES/NO) |
| Living\_Region | String | Type of living area (Metropolitan, Rural Areas) |
| Online\_Service\_Preference | String | Preference for using online services (YES/NO) |
| AI\_Enhance\_Experience | String | Whether the respondent feels AI enhances their shopping experience (YES/NO) |
| AI\_Trust | String | (AI\_Privacy\_No\_Trust + 1 + AI\_Endorsement+1)/2 (Will create High (2), Medium (1), Low Trust(0)) |
| AI\_Usage | String | AI\_Tools variables/3 (Will create High (3), Medium (2), Low Usage(1)) |
| AI\_Tools\_Used\_Chatbots | String | Use of chatbots for assistance (YES/NO) |
| AI\_Tools\_Used\_Virtual\_Assistant | String | Use of virtual assistants (YES/NO) |
| AI\_Tools\_Used\_Voice&Photo\_Search | String | Use of voice and photo search tools (YES/NO) |
| Paymeny\_Method\_COD | String | Use of Cash on Delivery as a payment method (YES/NO) |
| Payment\_Method\_Ewallet | String | Use of e-wallets as a payment method (YES/NO) |
| Product\_Category\_Appliances | String | Interest or purchase in appliances category (YES/NO) |
| Product\_Category\_Electronics | String | Interest or purchase in electronics category (YES/NO) |
| Product\_Category\_Groceries | String | Interest or purchase in groceries category (YES/NO) |
| Product\_Category\_Personal\_Care | String | Interest or purchase in personal care category (YES/NO) |
| Product\_Category\_Clothing | String | Interest or purchase in clothing category (YES/NO) |

*## Code from Geist (2019)*

calcSplitRatio <- **function**(data, p = 21) {

*## @p = the number of parameters. by default, if none are provided, the number of columns (predictors) in the dataset are used*

*## @df = the dataframe that will be used for the analysis*

*## If the number of parameters isn't supplied, set it to the number of features minus 1 for the target*

*## Calculate the ideal number of testing set*

test\_N <- (1 / sqrt(p)) \* nrow(data)

*## Turn that into a testing proportion*

test\_prop <- round(test\_N / nrow(data), 2)

*## And find the training proportion*

train\_prop <- 1 - test\_prop

*## Output the ideal split ratio*

message("The ideal split ratio is ", train\_prop, ":", test\_prop, " (training:testing)")

*## Return training set proportion*

**return**(train\_prop)

}

*# Final split*

calcSplitRatio(data)

## The ideal split ratio is 0.78:0.22 (training:testing)

## [1] 0.78

*# AC: Create a stratified split for training and testing data (e.g., 78-22 split)*

*set.seed(123)*

*train\_index <- createDataPartition(data$AI\_Satisfaction, p = 0.78, list = FALSE)*

*# AC: Split the data*

*train <- data[train\_index, ]*

*test <- data[-train\_index, ]*

*# Least squares Model*

*set.seed(123)*

*reg <- lm(AI\_Satisfaction ~ ., data = train)*

*summary(reg)*

*##*

*## Call:*

*## lm(formula = AI\_Satisfaction ~ ., data = train)*

*##*

*## Residuals:*

*## Min 1Q Median 3Q Max*

*## -1.05321 -0.02739 0.05435 0.13905 0.84846*

*##*

*## Coefficients: (1 not defined because of singularities)*

*## Estimate Std. Error t value Pr(>|t|)*

*## (Intercept) 0.4731140 0.1640715 2.884 0.00411 \*\**

*## Country -0.0014657 0.0007091 -2.067 0.03929 \**

*## Age -0.0058631 0.0289497 -0.203 0.83959*

*## Annual\_Salary -0.0014165 0.0179066 -0.079 0.93698*

*## Gender -0.0059686 0.0306965 -0.194 0.84592*

*## Education 0.0339586 0.0203163 1.671 0.09528 .*

*## Payment\_Method\_Credit\_Debit -0.0590457 0.0341711 -1.728 0.08465 .*

*## Living\_Region 0.0012286 0.0035323 0.348 0.72812*

*## Online\_Service\_Preference 0.1470331 0.0359886 4.086 5.16e-05 \*\*\**

*## AI\_Enhance\_Experience 0.4735040 0.0458044 10.338 < 2e-16 \*\*\**

*## AI\_Tools\_Used\_Chatbots 0.0085204 0.0309806 0.275 0.78342*

*## AI\_Tools\_Used\_Virtual\_Assistant -0.0237658 0.0312052 -0.762 0.44668*

*## AI\_Tools\_Used\_Voice\_Photo 0.0251025 0.0318809 0.787 0.43145*

*## Payment\_Method\_COD -0.0271204 0.0338155 -0.802 0.42295*

*## Payment\_Method\_Ewallet -0.0225671 0.0323826 -0.697 0.48621*

*## Product\_Category\_Appliances 0.0137286 0.0318115 0.432 0.66626*

*## Product\_Category\_Electronics 0.0220340 0.0331335 0.665 0.50637*

*## Product\_Category\_Groceries 0.0597268 0.0334895 1.783 0.07515 .*

*## Product\_Category\_Personal\_Care 0.0375058 0.0316279 1.186 0.23628*

*## Product\_Category\_Clothing 0.0615267 0.0352848 1.744 0.08186 .*

*## AI\_Trust 0.0140252 0.0250299 0.560 0.57551*

*## AI\_Usage NA NA NA NA*

*## ---*

*## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1*

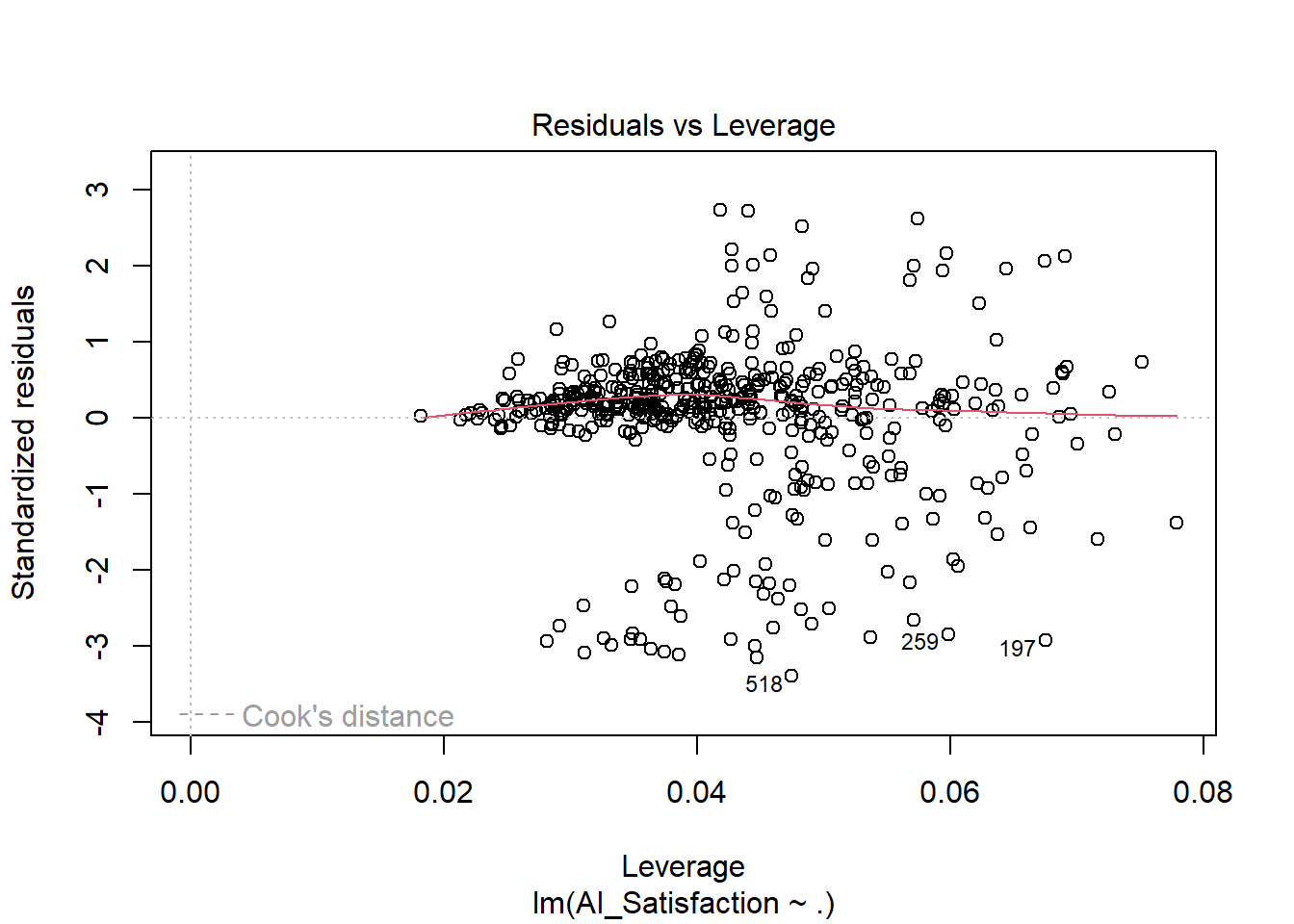
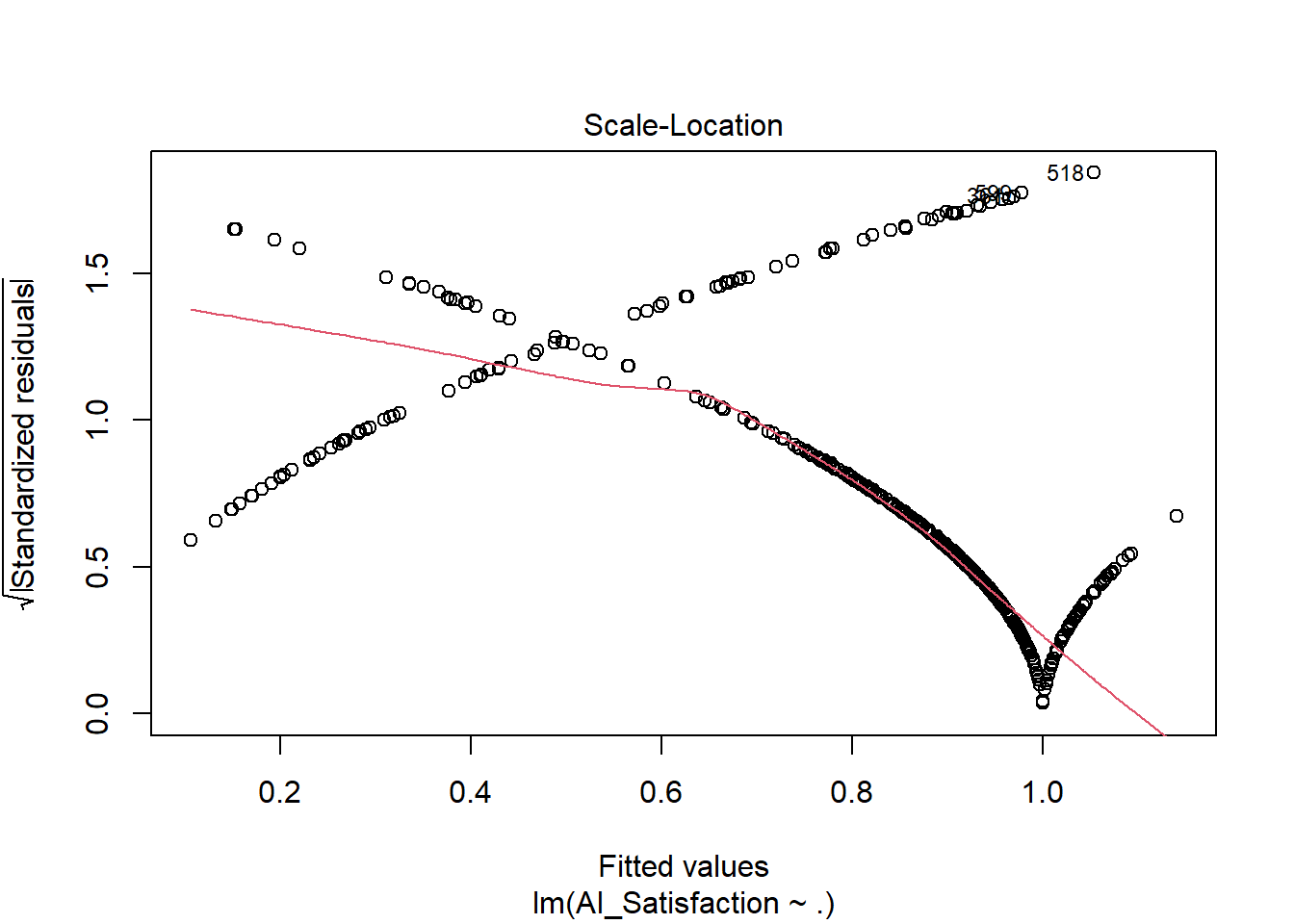
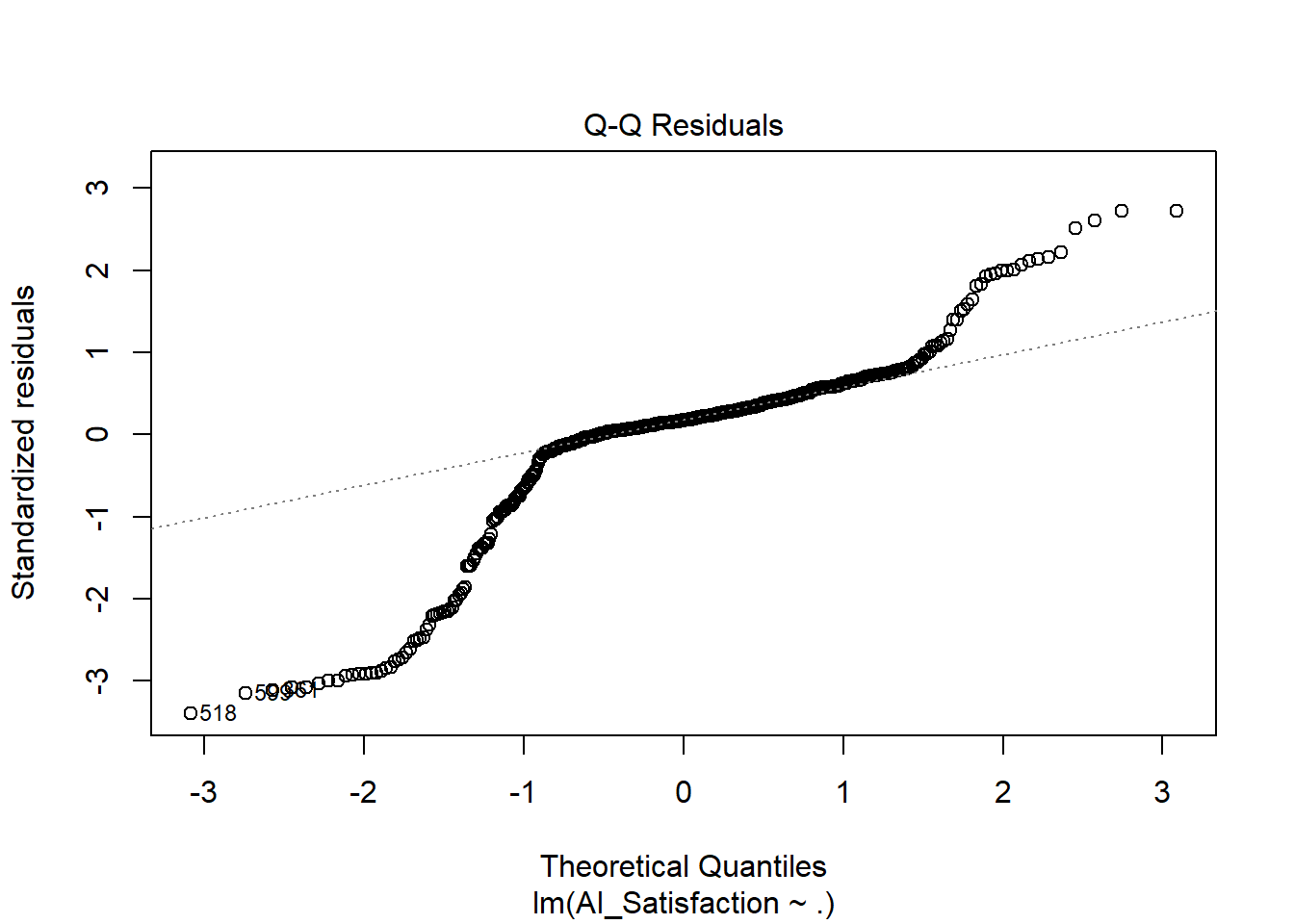
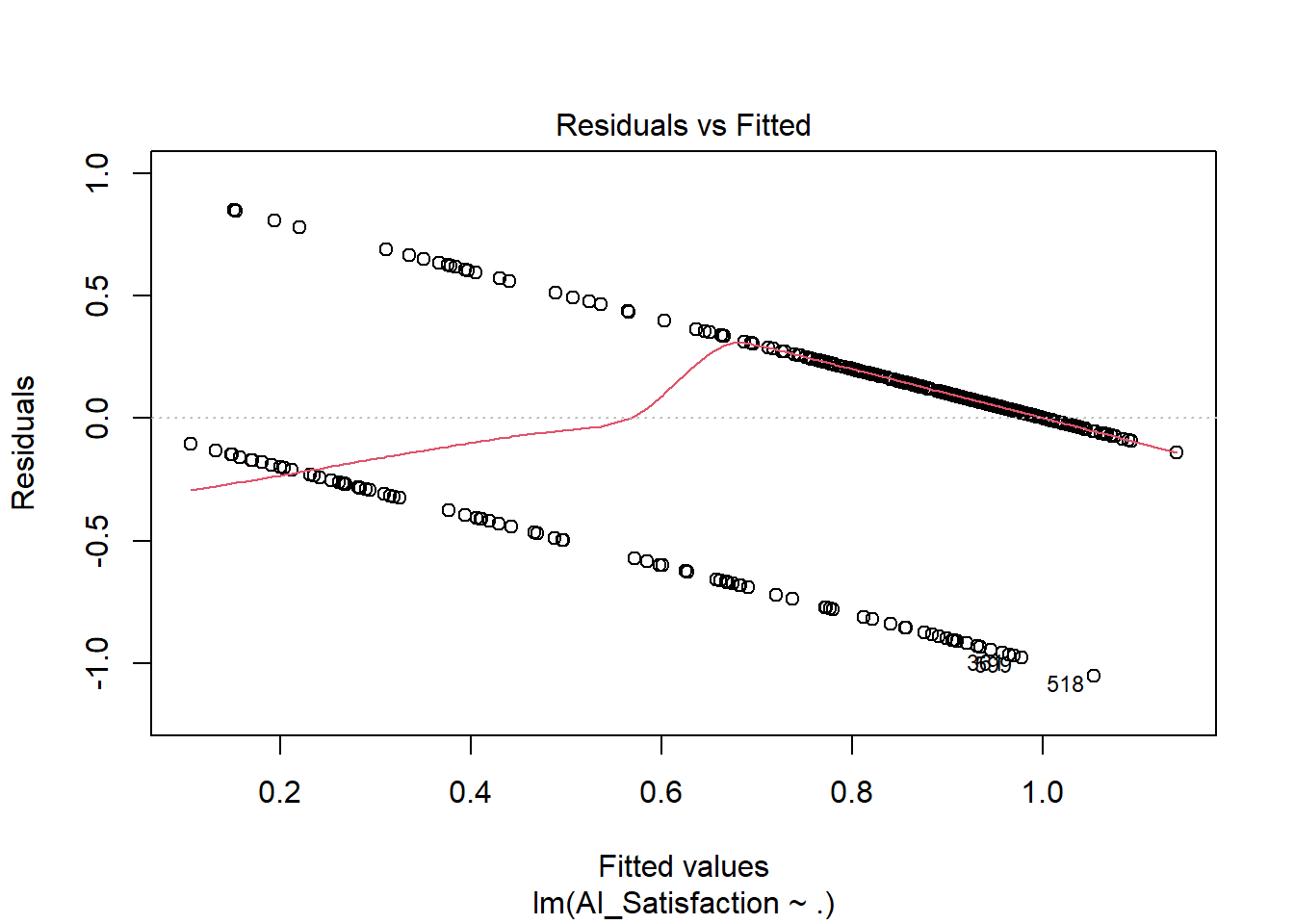
*##*

*## Residual standard error: 0.3177 on 474 degrees of freedom*

*## Multiple R-squared: 0.3446, Adjusted R-squared: 0.3169*

*## F-statistic: 12.46 on 20 and 474 DF, p-value: < 2.2e-16*

*plot(reg)*

**

*training\_residuals <- data.frame(train$AI\_Satisfaction, reg$fitted.values, reg$residuals)*

*head(training\_residuals)*

*## train.AI\_Satisfaction reg.fitted.values reg.residuals*

*## 4 1 0.9862163 0.01378368*

*## 5 1 1.0418611 -0.04186114*

*## 6 1 0.9709294 0.02907057*

*## 7 0 0.2416100 -0.24160995*

*## 8 0 0.4109741 -0.41097410*

*## 10 1 0.8664589 0.13354114*

*pred <- predict(reg, newdata = test)*

*lm\_metrics <- accuracy(pred, test$AI\_Satisfaction)*

*# Ridge regression model*

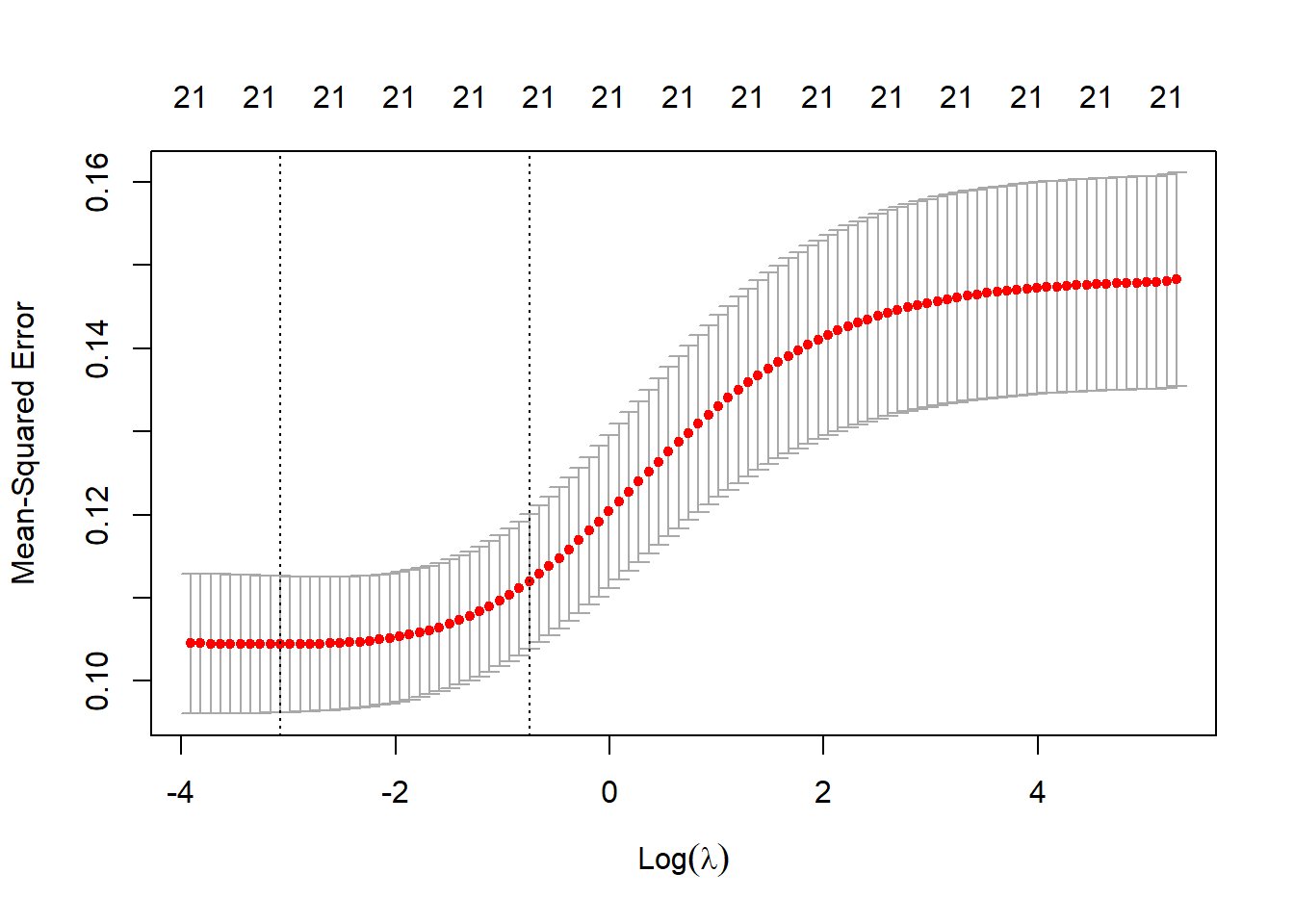
*set.seed(123)*

*x <- model.matrix(AI\_Satisfaction ~ ., data = train)[, -1]*

*y <- train$AI\_Satisfaction*

*cv.out2 <- cv.glmnet(x, y, alpha = 0)*

*plot(cv.out2)*

**

*lam\_min2 <- cv.out2$lambda.min*

*ridge.training2 <- glmnet(x, y, alpha = 0, lambda = lam\_min2, standardize = TRUE)*

*coef(ridge.training2)*

*## 22 x 1 sparse Matrix of class "dgCMatrix"*

*## s0*

*## (Intercept) 0.4647288223*

*## Country -0.0011263986*

*## Age -0.0098336017*

*## Annual\_Salary -0.0053723321*

*## Gender -0.0033980470*

*## Education 0.0345112795*

*## Payment\_Method\_Credit\_Debit -0.0449123452*

*## Living\_Region 0.0008621757*

*## Online\_Service\_Preference 0.1447355364*

*## AI\_Enhance\_Experience 0.4212089858*

*## AI\_Tools\_Used\_Chatbots 0.0080976363*

*## AI\_Tools\_Used\_Virtual\_Assistant -0.0233942676*

*## AI\_Tools\_Used\_Voice\_Photo 0.0231578988*

*## Payment\_Method\_COD -0.0226324719*

*## Payment\_Method\_Ewallet -0.0099978548*

*## Product\_Category\_Appliances 0.0112529652*

*## Product\_Category\_Electronics 0.0198255102*

*## Product\_Category\_Groceries 0.0575137487*

*## Product\_Category\_Personal\_Care 0.0330184464*

*## Product\_Category\_Clothing 0.0613439581*

*## AI\_Trust 0.0089430832*

*## AI\_Usage 0.0011293977*

*x1 <- model.matrix(AI\_Satisfaction ~ ., data = test)[, -1]*

*y1 <- test$AI\_Satisfaction*

*predict2 <- predict(ridge.training2, s = lam\_min2, newx = x1)*

*predict2 <- predict2[1:nrow(predict2),]*

*ridge\_metrics <- accuracy(predict2, y1)*

*# Lasso Model*

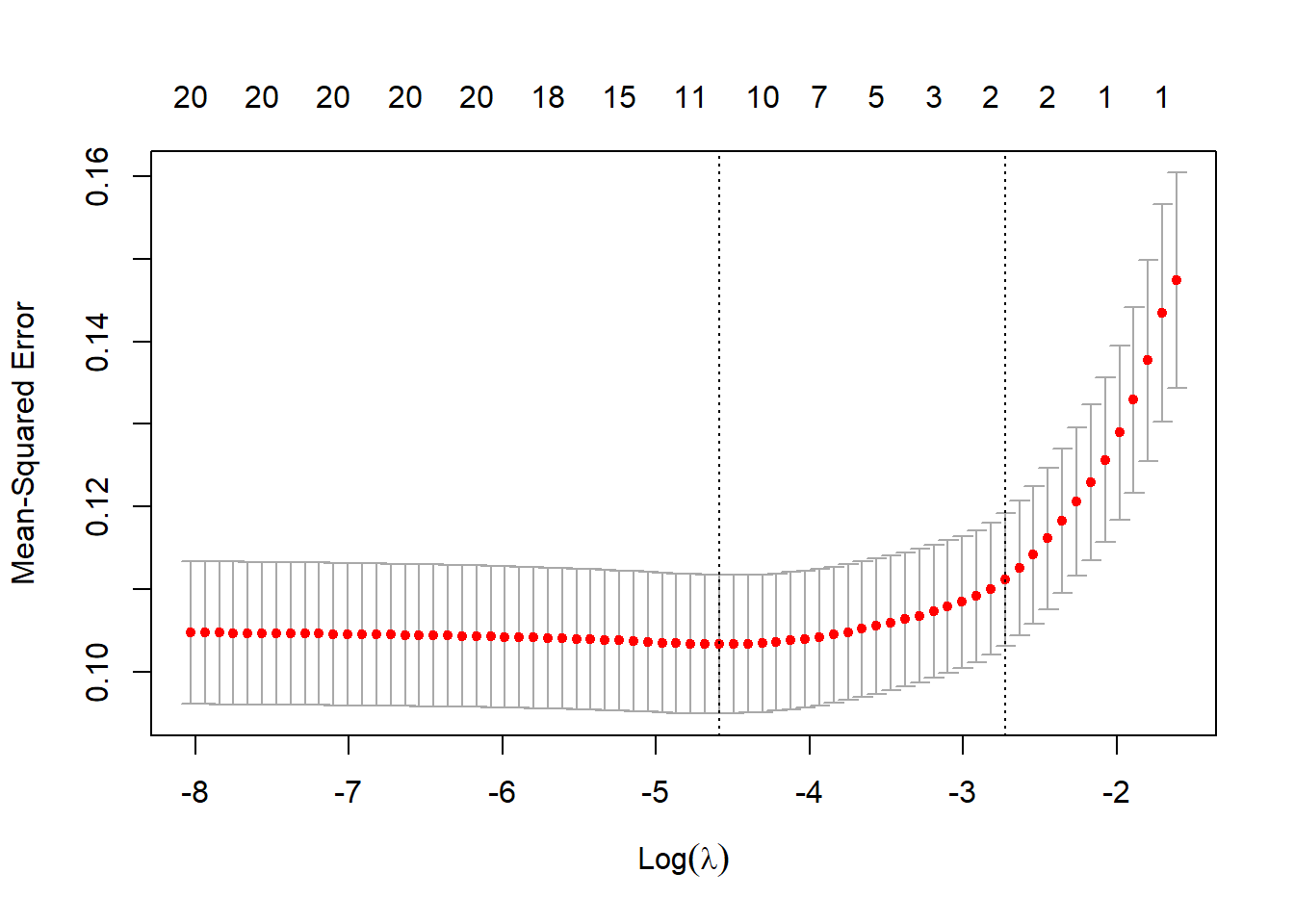
*set.seed(123)*

*x <- model.matrix(AI\_Satisfaction ~ ., data = train)[, -1]*

*y <- train$AI\_Satisfaction*

*cv.out2 <- cv.glmnet(x, y, alpha =1)*

*plot(cv.out2)*

**

*lam\_min2 <- cv.out2$lambda.min*

*lasso.training2 <- glmnet(x, y, alpha = 1, lambda = lam\_min2, standardize = TRUE)*

*coef(lasso.training2)*

*## 22 x 1 sparse Matrix of class "dgCMatrix"*

*## s0*

*## (Intercept) 0.3755449496*

*## Country -0.0007537756*

*## Age .*

*## Annual\_Salary .*

*## Gender .*

*## Education 0.0249257861*

*## Payment\_Method\_Credit\_Debit -0.0244070477*

*## Living\_Region .*

*## Online\_Service\_Preference 0.1379209686*

*## AI\_Enhance\_Experience 0.4561740120*

*## AI\_Tools\_Used\_Chatbots .*

*## AI\_Tools\_Used\_Virtual\_Assistant -0.0005314492*

*## AI\_Tools\_Used\_Voice\_Photo 0.0065612223*

*## Payment\_Method\_COD .*

*## Payment\_Method\_Ewallet .*

*## Product\_Category\_Appliances .*

*## Product\_Category\_Electronics 0.0096362970*

*## Product\_Category\_Groceries 0.0441471937*

*## Product\_Category\_Personal\_Care 0.0188184573*

*## Product\_Category\_Clothing 0.0539851406*

*## AI\_Trust .*

*## AI\_Usage .*

*x1 <- model.matrix(AI\_Satisfaction ~ ., data = test)[, -1]*

*y1 <- test$AI\_Satisfaction*

*predict2 <- predict(lasso.training2, s = lam\_min2, newx = x1)*

*predict2 <- predict2[1:nrow(predict2),]*

*lasso\_metrics <- accuracy(predict2, y1)*

*# PCR Model*

*set.seed(123)*

*pcr.fit <- pcr(AI\_Satisfaction ~ ., data = train, scale = TRUE, validation = 'CV')*

*pcr.fit2 <- pcr(AI\_Satisfaction ~ ., data = test, scale = TRUE, validation = 'CV')*

*# Extract the cross-validated errors*

*cv\_errors <- RMSEP(pcr.fit)$val[1,,]*

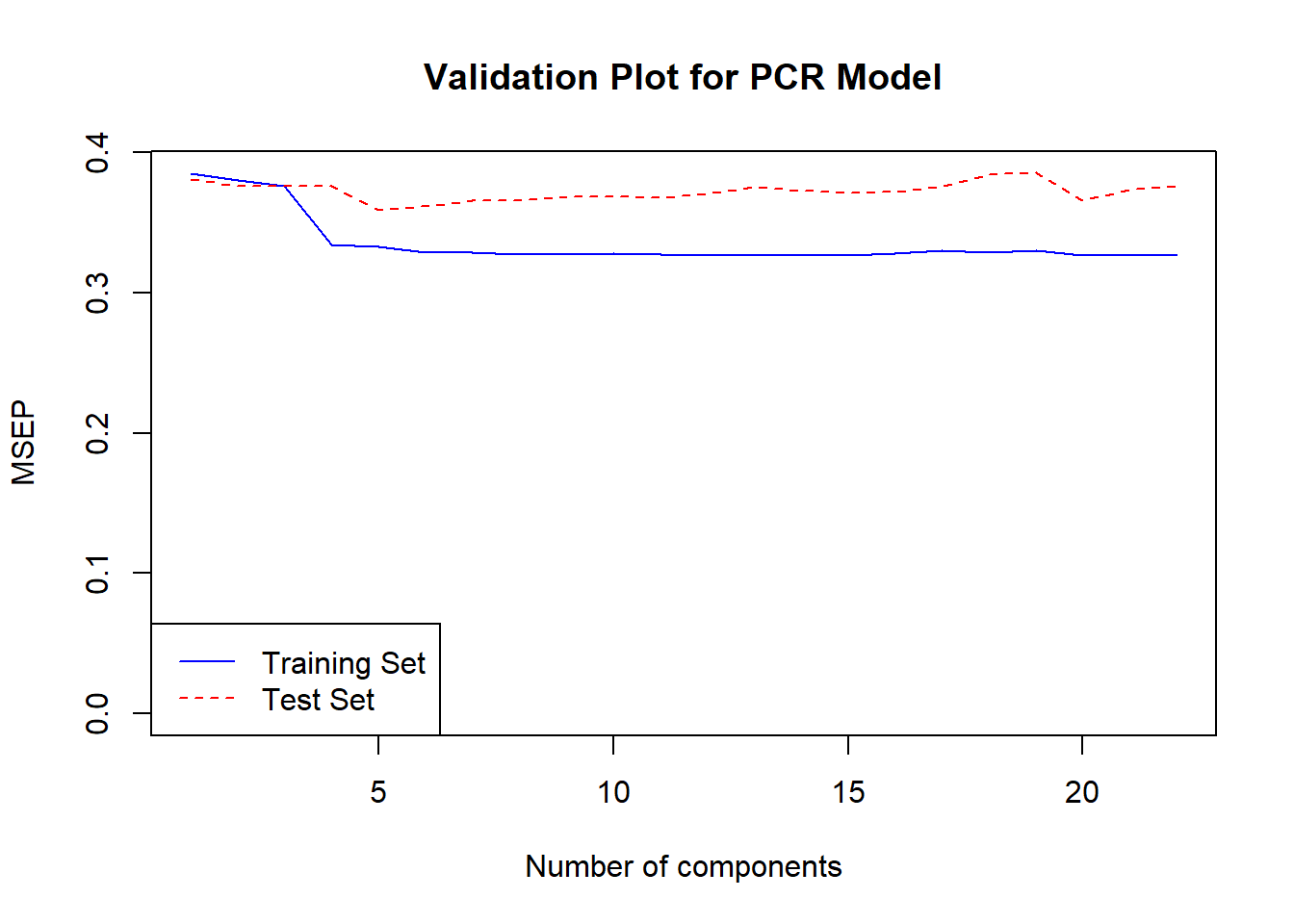
*cv\_errors2 <- RMSEP(pcr.fit2)$val[1,,]*

*# Plot the validation plot for both models*

*matplot(1:length(cv\_errors), cv\_errors, type = "l", col = "blue", xlab = "Number of components", ylab = "MSEP", ylim = c(0, max(cv\_errors, cv\_errors2)), main = "Validation Plot for PCR Model")*

*matlines(1:length(cv\_errors2), cv\_errors2, col = "red", lty = 2)*

*legend("bottomleft", legend = c("Training Set", "Test Set"), col = c("blue", "red"), lty = c(1, 2))*

**

*x <- model.matrix(AI\_Satisfaction ~ ., data = train)[, -1]*

*cv2 <- RMSEP(pcr.fit2)$val[1,,]*

*cv.min <- which.min(cv2) -1*

*cv.min*

*## 4 comps*

*## 4*

*x <- model.matrix(AI\_Satisfaction ~ ., data = train)[, -1]*

*pcr.pred2 <- predict(pcr.fit2, x, ncomp = cv.min)*

*pcr.pred2 <- pcr.pred2[1:nrow(pcr.pred2),,]*

*pcr\_metrics <- accuracy(pcr.pred2, test$AI\_Satisfaction)*

*# PLS Model*

*set.seed(123)*

*pls.fit <- plsr(AI\_Satisfaction ~ ., data = train, scale = TRUE, validation = 'CV')*

*pls.fit2 <- plsr(AI\_Satisfaction ~ ., data = test, scale = TRUE, validation = 'CV')*

*# Extract the cross-validated errors*

*cv\_errors <- RMSEP(pls.fit)$val[1,,]*

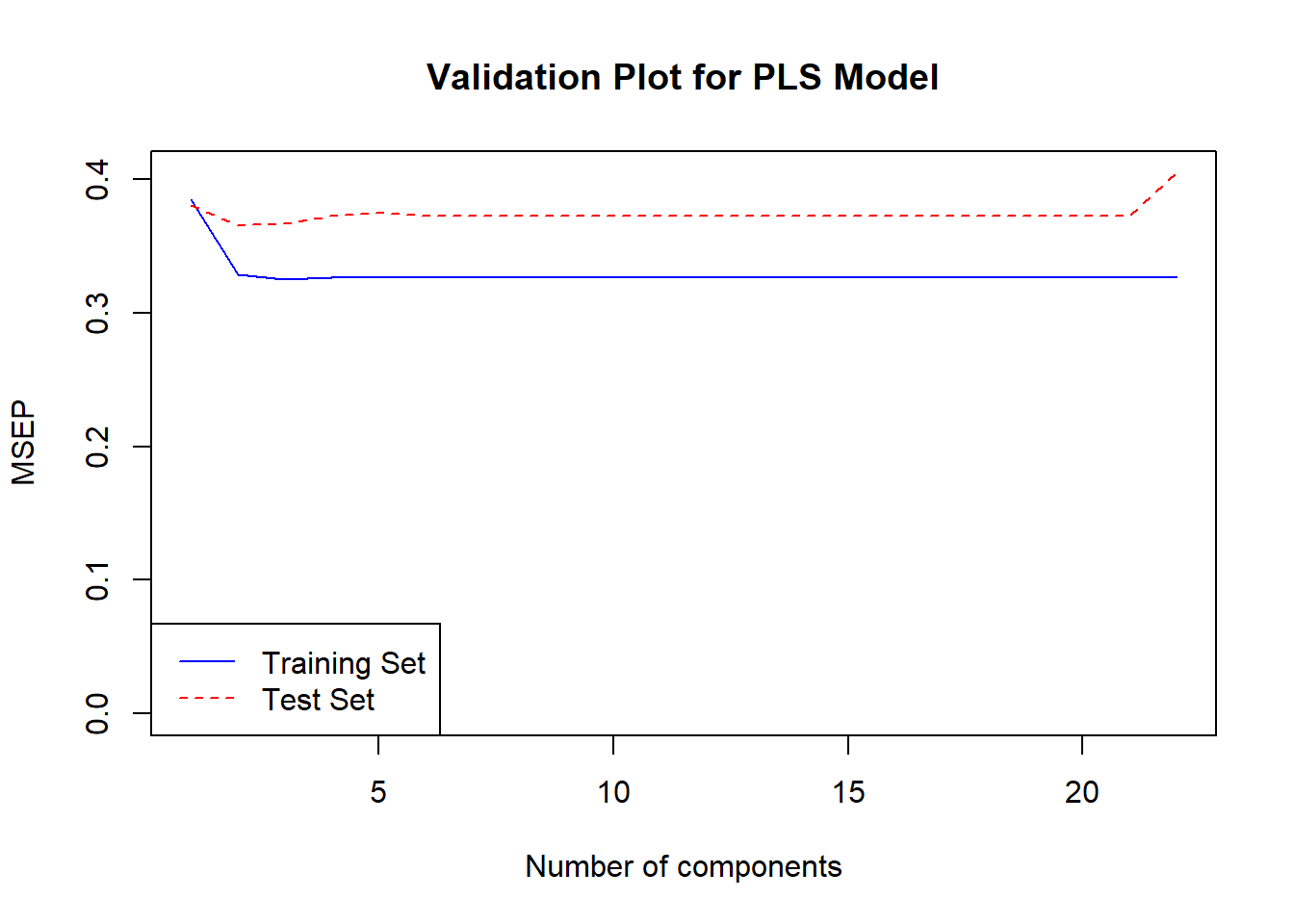
*cv\_errors2 <- RMSEP(pls.fit2)$val[1,,]*

*# Plot the validation plot for both models*

*matplot(1:length(cv\_errors), cv\_errors, type = "l", col = "blue", xlab = "Number of components", ylab = "MSEP", ylim = c(0, max(cv\_errors, cv\_errors2)), main = "Validation Plot for PLS Model")*

*matlines(1:length(cv\_errors2), cv\_errors2, col = "red", lty = 2)*

*legend("bottomleft", legend = c("Training Set", "Test Set"), col = c("blue", "red"), lty = c(1, 2))*

**

*# XGBoost model*

xgb\_model <- xgboost(

data = as.matrix(data[, categorical\_vars]),

label = as.numeric(data$AI\_Satisfaction) - 1,

nrounds = 10,

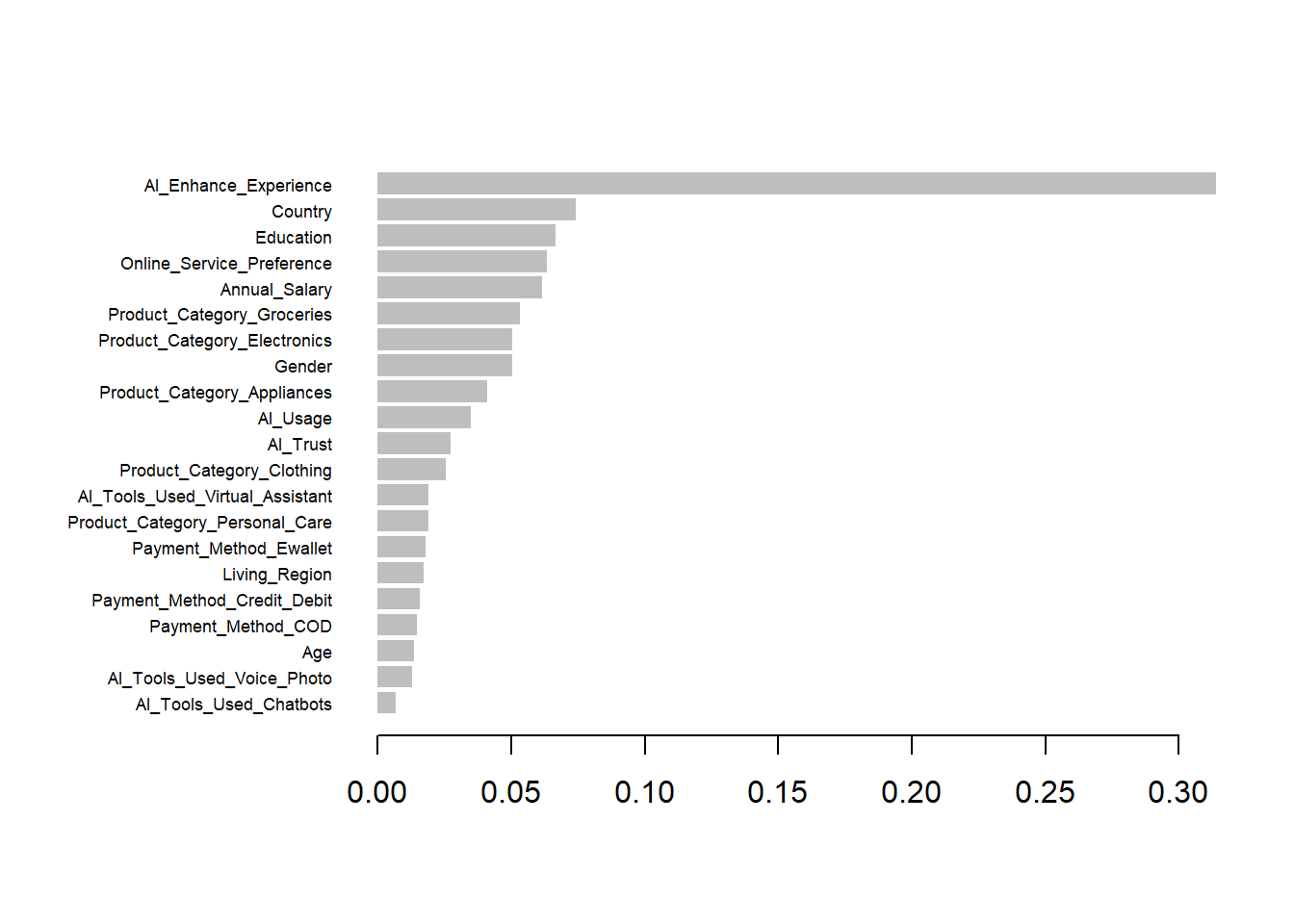
verbose = 0

)

*# Feature importance plot for XGBoost*

importance\_matrix <- xgb.importance(feature\_names = categorical\_vars, model = xgb\_model)

xgb.plot.importance(importance\_matrix)



importance\_matrix

## Feature Gain Cover Frequency

## <char> <num> <num> <num>

## 1: AI\_Enhance\_Experience 0.313929788 0.18307826 0.02178649

## 2: Country 0.074058467 0.11928963 0.08061002

## 3: Education 0.066705850 0.07069015 0.07843137

## 4: Online\_Service\_Preference 0.063422567 0.09174126 0.05882353

## 5: Annual\_Salary 0.061633923 0.04522091 0.10021786

## 6: Product\_Category\_Groceries 0.053165961 0.02763500 0.04139434

## 7: Product\_Category\_Electronics 0.050508932 0.04894600 0.06100218

## 8: Gender 0.050420860 0.04305515 0.07407407

## 9: Product\_Category\_Appliances 0.041004915 0.02982963 0.05446623

## 10: AI\_Usage 0.035066338 0.04545192 0.06100218

## 11: AI\_Trust 0.027225492 0.04311291 0.05664488

## 12: Product\_Category\_Clothing 0.025684034 0.01033786 0.02614379

## 13: AI\_Tools\_Used\_Virtual\_Assistant 0.019207933 0.02223506 0.03485839

## 14: Product\_Category\_Personal\_Care 0.019077416 0.01276350 0.02614379

## 15: Payment\_Method\_Ewallet 0.018120544 0.01914525 0.03703704

## 16: Living\_Region 0.017252941 0.04842622 0.03703704

## 17: Payment\_Method\_Credit\_Debit 0.015695585 0.06604100 0.04793028

## 18: Payment\_Method\_COD 0.014666228 0.01706613 0.03267974

## 19: Age 0.013457792 0.03447878 0.03267974

## 20: AI\_Tools\_Used\_Voice\_Photo 0.012919050 0.01140630 0.02178649

## 21: AI\_Tools\_Used\_Chatbots 0.006775385 0.01004909 0.01525054

## Feature Gain Cover Frequency

## Importance

## <num>

## 1: 0.313929788

## 2: 0.074058467

## 3: 0.066705850

## 4: 0.063422567

## 5: 0.061633923

## 6: 0.053165961

## 7: 0.050508932

## 8: 0.050420860

## 9: 0.041004915

## 10: 0.035066338

## 11: 0.027225492

## 12: 0.025684034

## 13: 0.019207933

## 14: 0.019077416

## 15: 0.018120544

## 16: 0.017252941

## 17: 0.015695585

## 18: 0.014666228

## 19: 0.013457792

## 20: 0.012919050

## 21: 0.006775385

## Importance

tree\_satisfaction <- tree(AI\_Satisfaction ~ . , train)

summary(tree\_satisfaction)

##

## Classification tree:

## tree(formula = AI\_Satisfaction ~ ., data = train)

## Variables actually used in tree construction:

## [1] "AI\_Enhance\_Experience" "Product\_Category\_Groceries"

## [3] "Education" "AI\_Usage"

## [5] "Online\_Service\_Preference" "Country"

## [7] "Product\_Category\_Clothing" "Annual\_Salary"

## Number of terminal nodes: 11

## Residual mean deviance: 0.5646 = 273.3 / 484

## Misclassification error rate: 0.1111 = 55 / 495

tree\_satisfaction

## node), split, n, deviance, yval, (yprob)

## \* denotes terminal node

##

## 1) root 495 466.40 1 ( 0.17980 0.82020 )

## 2) AI\_Enhance\_Experience < 0.5 70 88.64 0 ( 0.67143 0.32857 )

## 4) Product\_Category\_Groceries < 0.5 50 52.69 0 ( 0.78000 0.22000 )

## 8) Education < 2.5 40 33.82 0 ( 0.85000 0.15000 )

## 16) AI\_Usage < 1.5 26 28.09 0 ( 0.76923 0.23077 )

## 32) AI\_Usage < 0.5 9 0.00 0 ( 1.00000 0.00000 ) \*

## 33) AI\_Usage > 0.5 17 22.07 0 ( 0.64706 0.35294 ) \*

## 17) AI\_Usage > 1.5 14 0.00 0 ( 1.00000 0.00000 ) \*

## 9) Education > 2.5 10 13.86 1 ( 0.50000 0.50000 ) \*

## 5) Product\_Category\_Groceries > 0.5 20 26.92 1 ( 0.40000 0.60000 ) \*

## 3) AI\_Enhance\_Experience > 0.5 425 274.10 1 ( 0.09882 0.90118 )

## 6) Online\_Service\_Preference < 0.5 83 93.89 1 ( 0.25301 0.74699 )

## 12) Country < 194.5 27 0.00 1 ( 0.00000 1.00000 ) \*

## 13) Country > 194.5 56 74.10 1 ( 0.37500 0.62500 )

## 26) Product\_Category\_Clothing < 0.5 22 28.84 0 ( 0.63636 0.36364 )

## 52) Annual\_Salary < 3.5 17 23.51 0 ( 0.52941 0.47059 ) \*

## 53) Annual\_Salary > 3.5 5 0.00 0 ( 1.00000 0.00000 ) \*

## 27) Product\_Category\_Clothing > 0.5 34 34.57 1 ( 0.20588 0.79412 ) \*

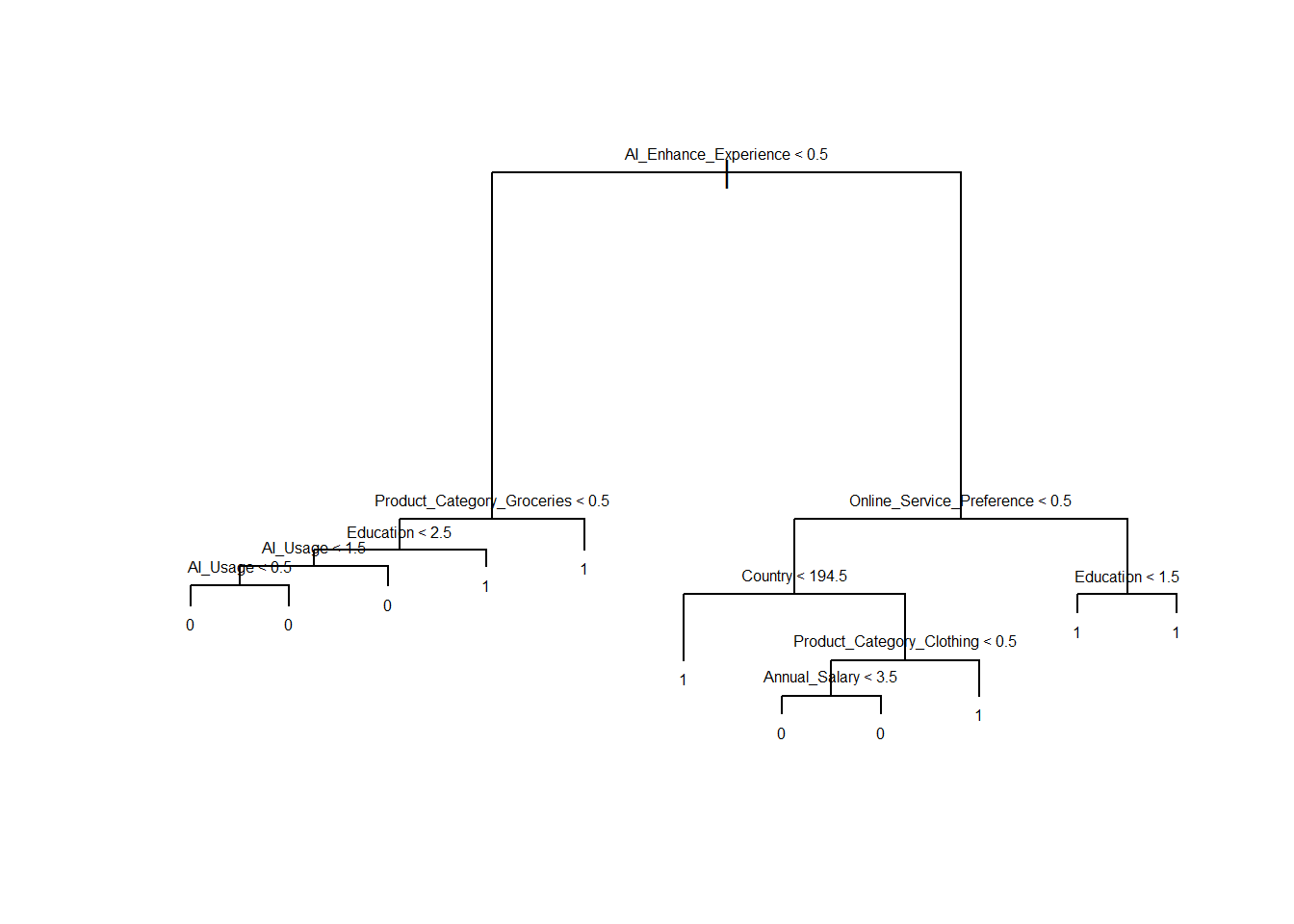
## 7) Online\_Service\_Preference > 0.5 342 157.90 1 ( 0.06140 0.93860 )

## 14) Education < 1.5 59 46.83 1 ( 0.13559 0.86441 ) \*

## 15) Education > 1.5 283 105.50 1 ( 0.04594 0.95406 ) \*

plot(tree\_satisfaction)

text(tree\_satisfaction, cex = 0.5)



test\_predictions <- predict(tree\_satisfaction, newdata = test)

test\_predictions <- ifelse(test\_predictions > 0.5, 1, 0)

cv\_model <- cv.tree(tree\_satisfaction)

cv\_model

## $size

## [1] 11 10 9 6 5 4 3 2 1

##

## $dev

## [1] 453.8559 434.6320 434.4782 434.4782 326.7373 328.3529 366.6696 373.1525

## [9] 467.6472

##

## $k

## [1] -Inf 5.333051 5.556859 5.584469 9.031463 10.679302 19.797852

## [8] 22.348408 103.616676

##

## $method

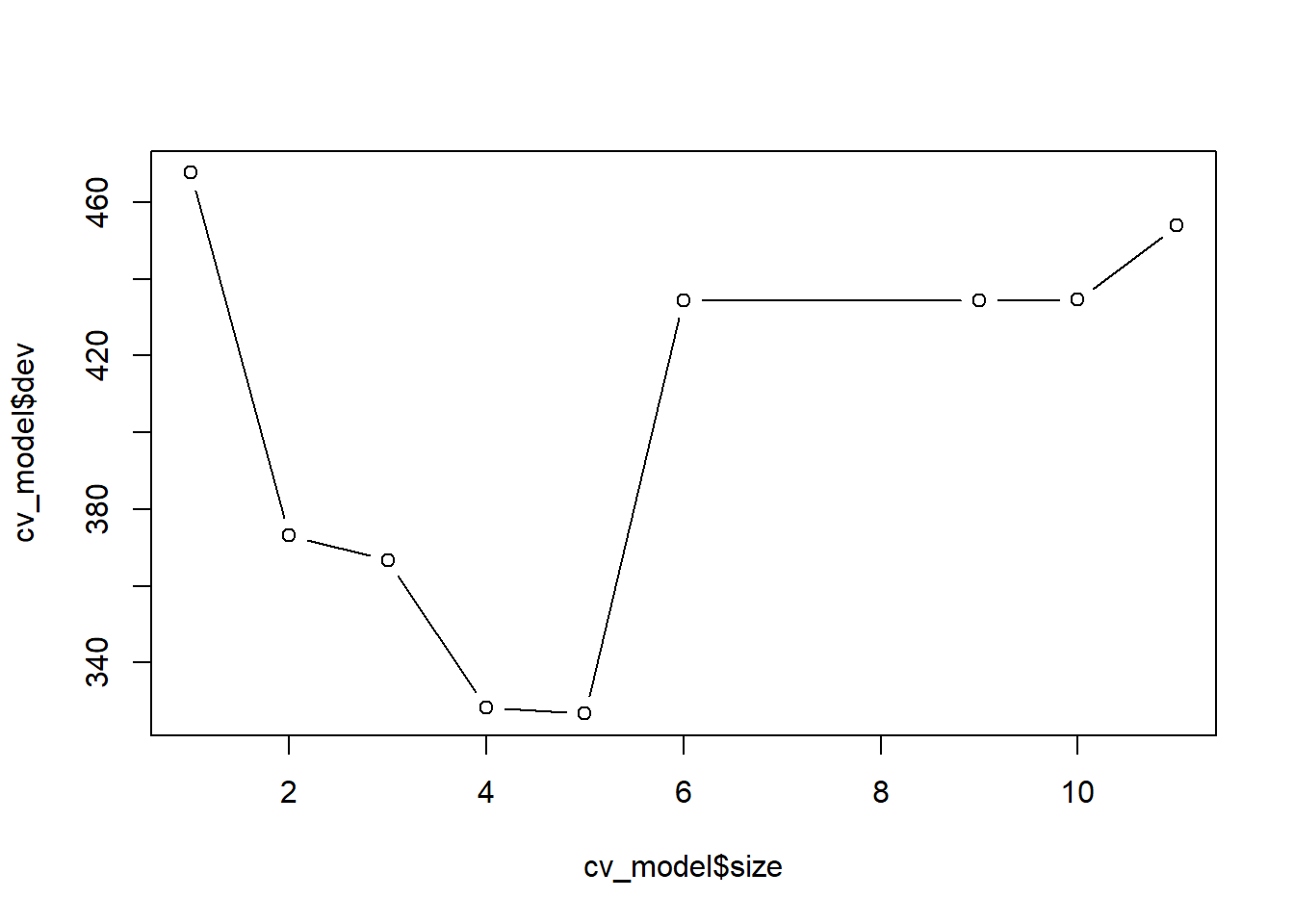
## [1] "deviance"

##

## attr(,"class")

## [1] "prune" "tree.sequence"

plot(cv\_model$size, cv\_model$dev, type = "b")



optimal\_tree\_size <- cv\_model$size[which.min(cv\_model$dev)]

optimal\_tree\_size

## [1] 5

satisfaction\_pred <- predict(tree\_satisfaction, newdata = test)

test\_mse <- mean((satisfaction\_pred - test$AI\_Satisfaction)^2)

test\_mse

## [1] 0.4200736

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**from** sklearn.decomposition **import** PCA

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.cluster **import** KMeans

**from** sklearn.discriminant\_analysis **import** LinearDiscriminantAnalysis

*# Load in the full dataset*

df **=** pd**.**read\_csv("encoded\_data.csv")

*# Make a copy of the full dataset; this is the one we will manipulate while still retaining the full original data*

df\_subset **=** df

df\_subset**.**head()

*# This chunk adds the target variable to the end of the dataset.*

AI\_Satisfaction **=** 'AI\_Satisfaction'

just\_AI\_Satisfaction **=** df\_subset**.**pop(AI\_Satisfaction)

df\_subset[AI\_Satisfaction] **=** just\_AI\_Satisfaction

df\_subset**.**head()

*# Splitting the data to get the features in one and the target variable in another*

X **=** df\_subset**.**iloc[:, :**-**1]**.**values *# The features are all the columns except the last one*

y **=** df\_subset**.**iloc[:, **-**1]**.**values *# Target column which was changed to the last column*

*# This chunk standardizes the data*

*# This is important because for both PCA and LDA, they are sensitive to the scale of the data.*

*# And since we have a column with larger numbers (Country and Living Region) it needs to be scaled and adjusted.*

scaler **=** StandardScaler()

X\_Scaled **=** scaler**.**fit\_transform(X)

*# This chunk applies the LDA from the sklearn library we use two components to make it easier for graphing*

*# df\_Subset\_Scaled.shape[0] represents a place holder for the target to make it unsupervised.*

*# Get the number of classes and features*

n\_classes **=** len(np**.**unique(y))

n\_features **=** X\_Scaled**.**shape[1]

*# Apply LDA with the adjusted number of components*

lda **=** LinearDiscriminantAnalysis(n\_components **=** 1)

df\_Subset\_LDA **=** lda**.**fit\_transform(X\_Scaled, y)

# Apply decision threshold at 0

# Assign class 1 if LDA value is greater than 0, else assign class 0

predictions = np.where(df\_Subset\_LDA[:, 0] > 0, 1, 0)

# Add the predicted class as a new column to the dataframe

df\_subset['LDA\_Prediction'] = predictions

print(f"LDA components shape: {df\_Subset\_LDA**.**shape}")

print(f"Explained Variance Ratio: {lda**.**explained\_variance\_ratio\_}")

plt**.**figure(figsize**=**(8, 6))

plt**.**scatter(df\_Subset\_LDA[:, 0], np**.**zeros\_like(df\_Subset\_LDA[:, 0]), c**=**y, cmap**=**'viridis')

plt**.**title('LDA Projection (1 Component)')

plt**.**xlabel('LD1')

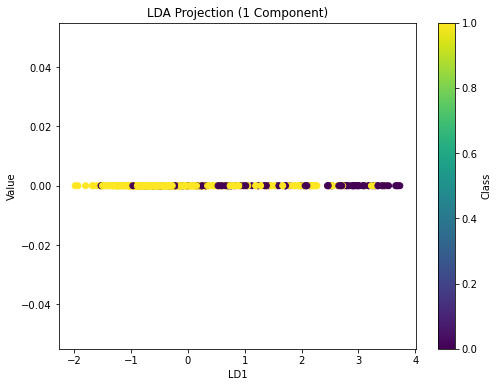
plt**.**ylabel('Value')

plt**.**colorbar(label**=**'Class')

plt**.**show()

LDA components shape: (634, 1)

Explained Variance Ratio: [1.]



**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.neighbors **import** NearestNeighbors

**from** sklearn.metrics **import** silhouette\_score

In [2]:

*# Load in the full dataset*

df **=** pd**.**read\_csv("encoded\_data.csv")

*# Make a copy of the full dataset; this is the one we will manipulate while still retaining the full original data*

df\_subset **=** df

Just\_AI\_Satisfaction **=** df\_subset['AI\_Satisfaction']

df\_subset **=** df\_subset**.**drop('AI\_Satisfaction', axis**=**1) *# Taking out target variable*

In [3]:

*# Scale the data like within LDA*

scaler **=** StandardScaler()

scaled\_df **=** scaler**.**fit\_transform(df\_subset)

In [5]:

*# Using KNN to calculate the nearest neighbors*

knn **=** NearestNeighbors(n\_neighbors **=** 5)

knn**.**fit(scaled\_df)

distances, indices **=** knn**.**kneighbors(scaled\_df)

In [6]:

*# Calculating how many points are closest to each of the 5 clusters*

*# This works by counting how many times each point appears as a neighbor to others*

neighbor\_counts **=** np**.**zeros(scaled\_df**.**shape[0])

**for** i **in** range(scaled\_df**.**shape[0]):

neighbor\_counts[indices[i]] **+=** 1

In [7]:

*# Finding the two most frequent or biggest clusters based on neighbor counts*

*# We are selecting the two largest sets of neighbors or dense regions to add back to our data set*

top\_clusters **=** np**.**argsort(neighbor\_counts)[**-**2:]

In [8]:

*# Putting the top clusters back into the dataframe*

df\_subset['KNNCluster1'] **=** np**.**where(np**.**isin(np**.**arange(scaled\_df**.**shape[0]), indices[top\_clusters[0]]), 1, 0)

df\_subset['KNNCluster2'] **=** np**.**where(np**.**isin(np**.**arange(scaled\_df**.**shape[0]), indices[top\_clusters[1]]), 1, 0)

*# Adding the target variable back into the dataset*

df\_subset['AI\_Satisfaction'] **=** Just\_AI\_Satisfaction

*# Printing the updated dataset*

print(df\_subset**.**head())

**from** os **import** read

*# Export DataFrame to CSV*

df\_subset**.**to\_csv('FE\_final\_data.csv', index**=False**)

In [ ]: