ADVANCE ANALYTICS AND MACHINE LEARNING ASSIGNMENT-2

STUDENT NAME- SIDDHARTH SHEKHAR SINGH
STUDENT ID-40425947
MSc – FINANCIAL ANALYTICS

INTRODUCTION

The primary objective of this assignment is to apply advanced analytical techniques and machine learning models to predict the score of an online business based on various predictors. The assignment tasks involve the utilization of different datasets and the application of multiple modeling techniques to evaluate the performance of online businesses effectively, focusing on several aspects of data analysis using the R programming language.

Procedure and Methodologies Employed:

Initial Setup and Data Loading:
The project began with setting up the R environment and loading essential libraries like **tidymodels**, **tidyverse**, and **glmnet** for data manipulation, modeling, and evaluation. The dataset, "student_merge_platform_business_file_final15.csv", was then loaded, setting the foundation for the subsequent stages of data exploration and preprocessing.

Data Preprocessing:
Early stages of data preprocessing involved addressing missing values, normalizing numerical predictors, and encoding categorical variables with dummy variables. These tasks were streamlined through a custom function **trainTestSplit**, which not only split the data into training and testing sets but also applied necessary data transformations.

Exploratory Data Analysis (EDA):
An extensive exploratory analysis helped us understand the distribution of variables, identify missing values, and visualize data characteristics. This phase was crucial for gaining deeper insights into the data and guiding the additional preprocessing required.

Feature Engineering:
We undertook significant feature engineering, converting the score into a binary classification system (**low** and **high**) and managing categorical variables effectively. This was a critical step for applying the classification models later in the analysis.

Model Development:
We developed several models to predict business scores, each tailored to capture different aspects of the data:

- **LASSO Regression Model:** This model focused on identifying key predictors using a penalization approach. We tuned the model using cross-validation to determine the optimal penalty.
- **Generalized Additive Model (GAM):** To handle non-linear relationships between predictors and the business score, we employed GAMs for different data subsets and evaluated their performance.
- **Decision Tree Model:** Chosen for its interpretability and efficacy in classification tasks, this model was tuned over several parameters to optimize predictive accuracy.

Model Evaluation and Comparison:
Each model's performance was meticulously assessed using relevant metrics. We utilized visualization tools such as ROC curves and confusion matrices to comprehensively present and compare the outcomes. This step was crucial to determining which model was most effective in predicting business scores.
Conclusion and Insights:
The final part of our report will synthesize the findings from each modeling approach, discuss the results' implications, and provide recommendations based on the models' predictive performance. This structured approach ensures a methodologically sound exploration and analysis, with detailed examinations to be discussed in the subsequent sections of the report.

##Loading the Libraries

```
library(tidymodels)

## ── Attaching packages ────────────────────────────────────── tidymodels 1.
2.0 ──

## ✓ broom        1.0.5     ✓ recipes      1.0.10
## ✓ dials        1.2.1     ✓ rsample      1.2.1
## ✓ dplyr        1.1.4     ✓ tibble       3.2.1
## ✓ ggplot2      3.5.1     ✓ tidyr        1.3.1
## ✓ infer        1.0.7     ✓ tune         1.2.1
## ✓ modeldata    1.3.0     ✓ workflows    1.1.4
## ✓ parsnip      1.2.1     ✓ workflowsets 1.1.0
## ✓ purrr        1.0.2     ✓ yardstick    1.3.1

## ── Conflicts ──────────────────────────────────────── tidymodels_conflict
s() ──
## ✗ purrr::discard() masks scales::discard()
## ✗ dplyr::filter()  masks stats::filter()
## ✗ dplyr::lag()     masks stats::lag()
## ✗ recipes::step()  masks stats::step()
## • Use tidymodels_prefer() to resolve common conflicts.

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.
0.0 ──
## ✓ forcats   1.0.0     ✓ readr     2.1.4
## ✓ lubridate 1.9.2     ✓ stringr   1.5.0

## ── Conflicts ────────────────────────────────────────── tidyverse_conflict
s() ──
## ✗ readr::col_factor() masks scales::col_factor()
## ✗ purrr::discard()    masks scales::discard()
## ✖ dplyr::filter()     masks stats::filter()
## ✖ stringr::fixed()    masks recipes::fixed()
## ✖ dplyr::lag()        masks stats::lag()
## ✖ readr::spec()       masks yardstick::spec()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
library(naniar)
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(vip)
```

```
##
## Attaching package: 'vip'
##
## The following object is masked from 'package:utils':
##
##     vi
```

```
library(embed)
library(knitr)
library(xtable)
library(RColorBrewer)
library(SmartEDA)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(ggplot2)
library(gam)
```

```
## Loading required package: splines
## Loading required package: foreach
##
## Attaching package: 'foreach'
##
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
##
## Loaded gam 1.22-3
```

```
library(viridis)
```

```
## Loading required package: viridisLite
##
## Attaching package: 'viridis'
##
## The following object is masked from 'package:scales':
##
##     viridis_pal
```

All the functions

1) Train - Test Split Function

```
trainTestSplit <- function(data){
  set.seed(40425947)

  split <- initial_split(data, prop = 0.75, strata = category_score)
  training_data <- training(split)
  testing_data <- testing(split)

  recipe <- recipe(category_score ~., data = training_data) %>%
    step_impute_knn(all_numeric_predictors()) %>%
    step_impute_mode(all_factor_predictors()) %>%
    step_normalize(all_numeric_predictors()) %>%
    step_corr(all_numeric_predictors(), threshold = 0.9) %>%
    step_dummy(all_factor_predictors()) %>%
    step_zv(all_numeric(), -all_outcomes()) %>%
    step_nzv(all_predictors())


  return(list(split= split, training_data = training_data, testing_data = tes
ting_data, recipe = recipe))

}
```

2) Lasso function

```
lassoModelFunction <- function(subset_training_data, recipe){
  # Set up your model so that the penalty is tuned automatically
  model_lasso_tuned <- multinom_reg() %>% set_engine("glmnet") %>%
    set_args(mixture = 1, penalty = tune())

  workflow_lasso_tuned <- workflow() %>%
    add_model(model_lasso_tuned) %>%
    add_recipe(recipe)
  # Configure a penalty grid with 30 levels
  set.seed(40425947)
  penalty_grid <- grid_regular(penalty(range = c(-3, 1)), levels = 30)

  set.seed(40425947)
  resamples <- vfold_cv(subset_training_data, v = 5)

  tune_output <- tune_grid(workflow_lasso_tuned,
    resamples = resamples,
    metrics = metric_set(roc_auc),
    grid = penalty_grid)

  l <- autoplot(tune_output)
  l

  fit_tuned <- workflow_lasso_tuned %>%
  fit(subset_training_data)

  top10 <- fit_tuned %>%
  extract_fit_parsnip() %>%
  vip(aesthetics = list(fill = "steelblue"), 10)

  top10


  best_penalty <- select_by_one_std_err(tune_output,
  metric = 'roc_auc', desc(penalty))

  # Fit Final Model
  final_fit<- finalize_workflow(workflow_lasso_tuned, best_penalty) %>%
  fit(data = subset_training_data)

  tidy(final_fit)
  return(list(top10 = top10,l = l))

}
```

3) GAM function

```
GAM_function <- function(formula, data){


  gam_model <- mgcv::gam(formula = formula,family = "binomial", data = data,
method = "REML")


  print(summary(gam_model))

   # Attempt to plot the GAM model with error handling
```

```r
  tryCatch({
    plot(gam_model)
  }, error = function(e) {
    # Handle specific error related to plot.gam()
    if (grepl("No terms to plot", e$message)) {
      print("No terms available to plot. Check the formula used in the GAM mo
del.")
    } else {
      # Handle other types of errors related to plot generation
      print(paste("Error:", e$message))
    }
  })
}
```

4) Decision Tree Model Function

```r
decisionTreeModel <- function(subsets) {
  # Split data into training and testing sets
  split <- initial_split(subsets, prop = 0.75, strata = category_score)
  training <- training(split)
  testing <- testing(split)

  # Define decision tree model for tuning
  dt_model_tune <- decision_tree(cost_complexity = tune(), tree_depth = tune(
), min_n = tune()) %>%
    set_mode("classification") %>%
    set_engine("rpart")

  # Define classification metrics (accuracy, sensitivity, specificity)
 # lead_metrics <- metric_set(accuracy, sens, specificity)

  # Define recipe for data preprocessing
  dt_recipe <- recipe(category_score ~ ., data = training) %>%
    step_impute_knn(all_numeric_predictors()) %>%
    step_impute_mode(all_factor_predictors()) %>%
    step_normalize(all_numeric_predictors()) %>%
    step_corr(all_numeric_predictors(), threshold = 0.9) %>%
    step_dummy(all_factor_predictors()) %>%
    step_zv(all_numeric(), -all_outcomes()) %>%
    step_nzv(all_predictors())

  # Create workflow for model tuning
  dt_tune_wkf <- workflow() %>%
    add_model(dt_model_tune) %>%
    add_recipe(dt_recipe)

  # Tune the decision tree model using cross-validation
  dt_tuning <- dt_tune_wkf %>%
    tune_grid(
      resamples = vfold_cv(training, v = 10, strata = category_score),
      grid = grid_random(parameters(dt_model_tune), n = 5)

    )


   # Select the best tuned decision tree model based on accuracy
  best_dt_model <- dt_tuning %>%
    select_best(metric = 'accuracy')

  print(best_dt_model)

  # Finalize the workflow with the best model
  final_leads_wkfl <- dt_tune_wkf %>%
    finalize_workflow(best_dt_model)

  # Fit the final model using the split data
  leads_final_fit <- final_leads_wkfl %>%
    last_fit(split = split)

  # Collect final evaluation metrics
  final_metrics <- leads_final_fit %>%
    collect_metrics()

  print(final_metrics)

  # Collect predictions from the final model
  predictions <- leads_final_fit %>%
    collect_predictions()

  # Plot ROC curve
  roc_curve <- predictions %>% roc_curve(truth = category_score, .pred_high)
%>% autoplot()
  print(roc_curve)

  # Plot confusion matrix mosaic
  mosaic <- conf_mat(predictions, truth = category_score, estimate = .pred_cl
ass) %>%
    autoplot(type = 'mosaic')

  print(mosaic)

  # Plot confusion matrix heatmap
  heatmap <- conf_mat(predictions, truth = category_score, estimate = .pred_c
lass) %>%
    autoplot(type = 'heatmap')

  print(heatmap)
```

```
}
```

Item A1

```
data <- read.csv("student_merge_platform_business_file_final15.csv")

class(data)

## [1] "data.frame"

print("Success - Data Loaded !!")

## [1] "Success - Data Loaded !!"

## Rows: 10,891
## Columns: 21
## $ Platform                 <chr> "Platform 1", "Platform 1", "Platform 1", 
"Pl…
## $ business_id              <int> 1770720401, 1699268318, 1336331669, 15280
0846…
## $ city                     <chr> "Santa Barbara", "Clearwater", "Bala Cynw
yd",…
## $ state                    <chr> "CA", "FL", "PA", "PA", "NJ", "FL", "FL", 
"IN…
## $ postal_code              <chr> "93101", "33755", "19004", "19462", "0804
3", …
## $ score                    <dbl> 5.0, 5.0, 4.0, 2.5, 3.5, 2.0, 3.0, 1.5, 2
.5, …
## $ review_count             <int> 7, 10, 13, 8, 17, 29, 36, 14, 43, 6, 10, 
19, …
## $ Gender                   <chr> "F", "F", "M", "M", "F", "M", "M", "M", "
M", …
## $ CEO_sch_cat              <int> 108, 108, 108, 138, 116, 170, 116, 108, 3
, 20…
## $ CEO_grd_yr               <int> 1997, 2017, 1986, 1980, 2014, 1988, 1992, 
201…
## $ field_cat                <int> 13, 21, 12, 75, 69, 55, 21, 58, 33, 3, 18
, 66…
## $ ZIP.Code                 <int> 931032109, 337631726, 190043207, 19462171
8, 8…
## $ Business_ID_other        <int> NA, 1699268318, NA, 1528008463, 136686726
9, N…
## $ Rural_metropolitan_Desc  <chr> "", "Metropolitan area core: primary flow 
wit…
## $ Tot_Clms_Services        <int> NA, 971, NA, 1988, 847, NA, 869, NA, NA, 
24, …
## $ Brnd_Tot_Clms_Services   <int> NA, 138, NA, NA, 73, NA, 103, NA, NA, 11,
NA,…
## $ Gnrc_Tot_Clms_Services   <int> NA, 813, NA, 1674, 774, NA, 766, NA, NA,
13, …
## $ Othr_Tot_Clms_Services   <int> NA, 20, NA, NA, 0, NA, 0, NA, NA, 0, NA,
NA, …
## $ LIS_Tot_Clms_Services    <int> NA, 762, NA, 65, 75, NA, 106, NA, NA, 0,
NA, …
## $ Opioid_Tot_Clms_Services <int> NA, NA, NA, NA, 50, NA, 33, NA, NA, 0, NA
, NA…
## $ Antbtc_Tot_Clms_Services <int> NA, 31, NA, 274, NA, NA, 25, NA, NA, 0, N
A, N…

##     Platform business_id             city state postal_code score review_c
ount
## 1 Platform 1  1770720401    Santa Barbara    CA       93101   5.0
7
## 2 Platform 1  1699268318       Clearwater    FL       33755   5.0
10
## 3 Platform 1  1336331669      Bala Cynwyd    PA       19004   4.0
13
## 4 Platform 1  1528008463 Plymouth Meeting    PA       19462   2.5
8
## 5 Platform 1  1366867269         Voorhees    NJ       08043   3.5
17
## 6 Platform 1  1689735383   Tarpon Springs    FL       34689   2.0
29
##   Gender CEO_sch_cat CEO_grd_yr field_cat  ZIP.Code Business_ID_other
## 1      F         108       1997        13 931032109                NA
## 2      F         108       2017        21 337631726        1699268318
## 3      M         108       1986        12 190043207                NA
## 4      M         138       1980        75 194621718        1528008463
## 5      F         116       2014        69  80434509        1366867269
## 6      M         170       1988        55 346893790                NA
##                                                                Rural_metrop
olitan_Desc
## 1
## 2 Metropolitan area core: primary flow within an urbanized area of 50,000
and greater
## 3
## 4 Metropolitan area core: primary flow within an urbanized area of 50,000
and greater
## 5 Metropolitan area core: primary flow within an urbanized area of 50,000
and greater
## 6
##   Tot_Clms_Services Brnd_Tot_Clms_Services Gnrc_Tot_Clms_Services
## 1                NA                     NA                     NA
## 2               971                    138                    813
## 3                NA                     NA                     NA
## 4              1988                     NA                   1674
## 5               847                     73                    774
## 6                NA                     NA                     NA
##   Othr_Tot_Clms_Services LIS_Tot_Clms_Services Opioid_Tot_Clms_Services
## 1                     NA                    NA                       NA
## 2                     20                   762                       NA
## 3                     NA                    NA                       NA
## 4                     NA                    65                       NA
## 5                      0                    75                       50
## 6                     NA                    NA                       NA
##   Antbtc_Tot_Clms_Services
## 1                       NA
## 2                       31
## 3                       NA
## 4                      274
## 5                       NA
## 6                       NA
```

Item A2

```
library(knitr)
library(kableExtra)

## Warning: package 'kableExtra' was built under R version 4.3.3

##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows

# Function to summarize NA and 0 values for each column
summarize_blanks_zeros <- function(df) {
  summary_df <- df %>% summarise_all(~ sum(is.na(.) | . == 0))
  return(summary_df)
}

# Apply the function to dataset
summary_blanks_zeros <- summarize_blanks_zeros(data)

# View the summary
print(summary_blanks_zeros)

##   Platform business_id city state postal_code score review_count Gender
## 1        0           0    0     0           0     0            0      0
##   CEO_sch_cat CEO_grd_yr field_cat ZIP.Code Business_ID_other
## 1           3          8         2        0              4919
##   Rural_metropolitan_Desc Tot_Clms_Services Brnd_Tot_Clms_Services
## 1                       0              4919                   7778
##   Gnrc_Tot_Clms_Services Othr_Tot_Clms_Services LIS_Tot_Clms_Services
## 1                   4986                  10073                  6107
##   Opioid_Tot_Clms_Services Antbtc_Tot_Clms_Services
## 1                     8389                     7343
```

```r
missing_data_counts <- colSums(is.na(data))

# Creating a data frame for the table
variable_table <- data.frame(
  Variable = names(missing_data_counts),
  MissingValues = missing_data_counts
)

# Generating the table
kable(variable_table, format = "html", caption = "Count of Missing Values per 
Variable") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "respo
nsive"), full_width = FALSE) %>%
  column_spec(1, bold = TRUE, color = "blue") %>%
  column_spec(2, background = "lavender")
```

Count of Missing Values per Variable

Variable

MissingValues

Platform

Platform

0

business_id

business_id

0

city

city

0

state

state

0

postal_code

postal_code

0

score

score

0

review_count

review_count

0

Gender

Gender

0

CEO_sch_cat

CEO_sch_cat

0

CEO_grd_yr

CEO_grd_yr

8

field_cat

field_cat

0

ZIP.Code

ZIP.Code

0

Business_ID_other

Business_ID_other

4919

Rural_metropolitan_Desc

Rural_metropolitan_Desc

0

Tot_Clms_Services

Tot_Clms_Services

4919

Brnd_Tot_Clms_Services

Brnd_Tot_Clms_Services

7510

Gnrc_Tot_Clms_Services

Gnrc_Tot_Clms_Services

4986

Othr_Tot_Clms_Services

Othr_Tot_Clms_Services

7521

LIS_Tot_Clms_Services

LIS_Tot_Clms_Services

5898

Opioid_Tot_Clms_Services

Opioid_Tot_Clms_Services

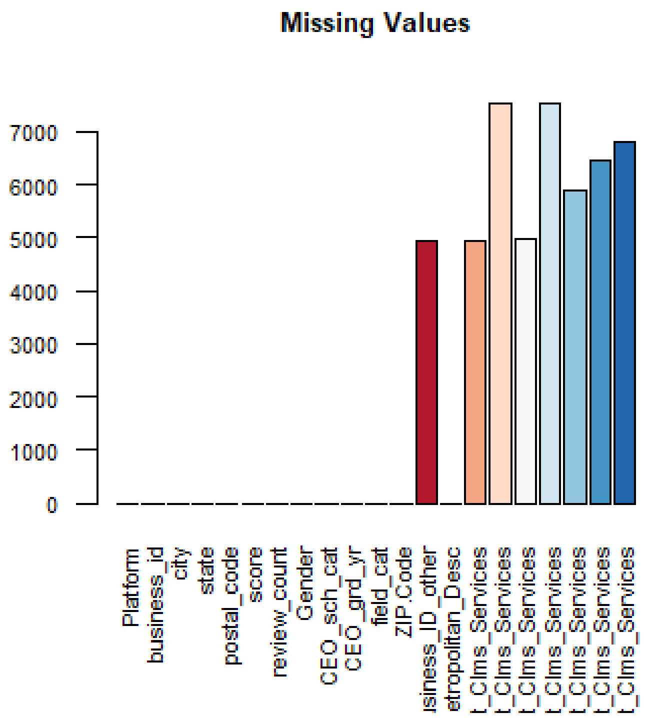
6454

Antbtc_Tot_Clms_Services

Antbtc_Tot_Clms_Services

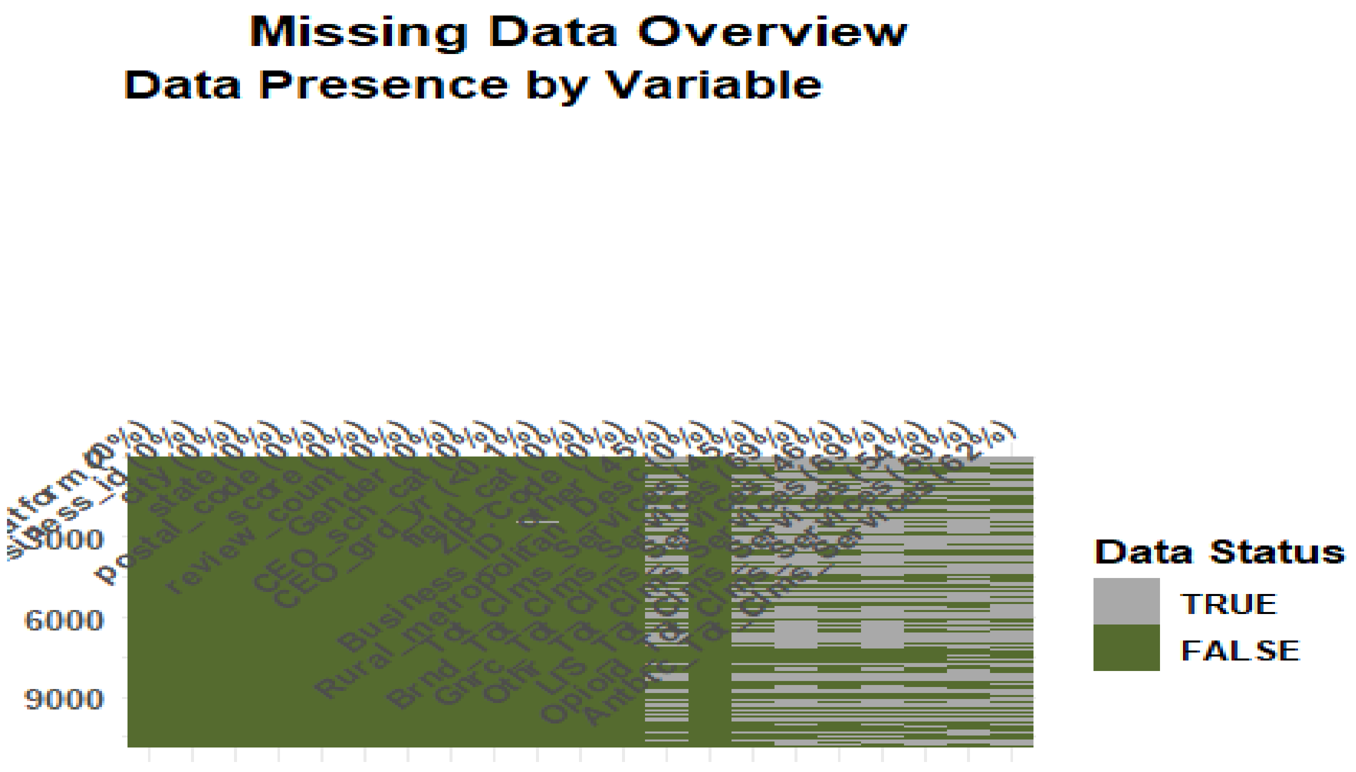
6787

```r
# Setting up the plot area with larger margins
par(mar = c(5, 8, 4, 2) + 0.3)  # Adjust margins (bottom, left, top, right)

# Creating the barplot of missing values
barplot(missing_data_counts,
        main = "Missing Values",
        col = brewer.pal(n = length(colnames(data)), name = "RdBu"),
        las = 2,  # Rotate column names by 90 degrees for better fit
        cex.names = 0.7,  # Reduce the size of column names
        cex.axis = 0.7,  # Reduce the size of axis labels
        cex.main = 0.8)  # Reduce the title font size
```



```
library(visdat)
library(ggplot2)

# Create the missing data visualization
vis_miss_plot <- vis_miss(data) +
  labs(
    title = "Missing Data Overview",
    subtitle = "Data Presence by Variable"
  ) +
  theme_minimal(base_size = 10) +  # Using a smaller base font size for minim
al theme
  theme(
    plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
    plot.subtitle = element_text(size = 14, face = "bold"),
    legend.position = "right",
    legend.title = element_text(size = 12, face = "bold"),
    legend.text = element_text(size = 10, face = "bold"),
    axis.text.x = element_text(angle = 45, hjust = 1, size = 10, face = "bold
"),
    axis.text.y = element_text(size = 10, face = "bold"),
    axis.title = element_blank()  # Remove axis titles
  ) +
  scale_fill_manual(values = c("TRUE" = "#A9A9A9", "FALSE" = "#556B2F"),
                    name = "Data Status",
                    labels = c("Missing" = "Absent", "Not missing" = "Present
"))  # Using sober colors for a more professional look
```

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.

# Print the plot
print(vis_miss_plot)
```



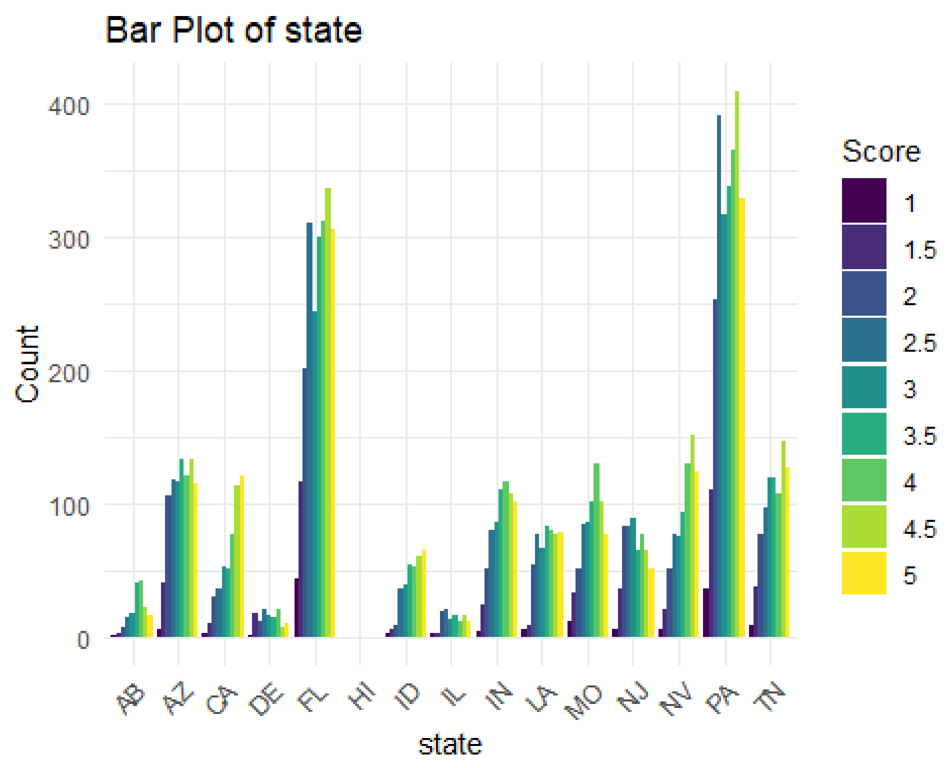
Item A3

```
# Removing irrelevant column variables
data <- data %>%
  select(-c(business_id, Business_ID_other, ZIP.Code, postal_code, Platform))

# converting the selected column in numeric
data <- data %>%
  mutate_at(vars(score, review_count, Tot_Clms_Services, Brnd_Tot_Clms_Servic
es, Gnrc_Tot_Clms_Services, Othr_Tot_Clms_Services, Opioid_Tot_Clms_Services,
Antbtc_Tot_Clms_Services, LIS_Tot_Clms_Services), as.numeric)

# Converting the selected columns which were numeric into factor
data <- data %>%
  mutate_at(vars(CEO_sch_cat, CEO_grd_yr,field_cat), as.factor)

# Converting the categorical variables into factor
data <- data %>%
  mutate(across(where(is.character), as.factor))
```

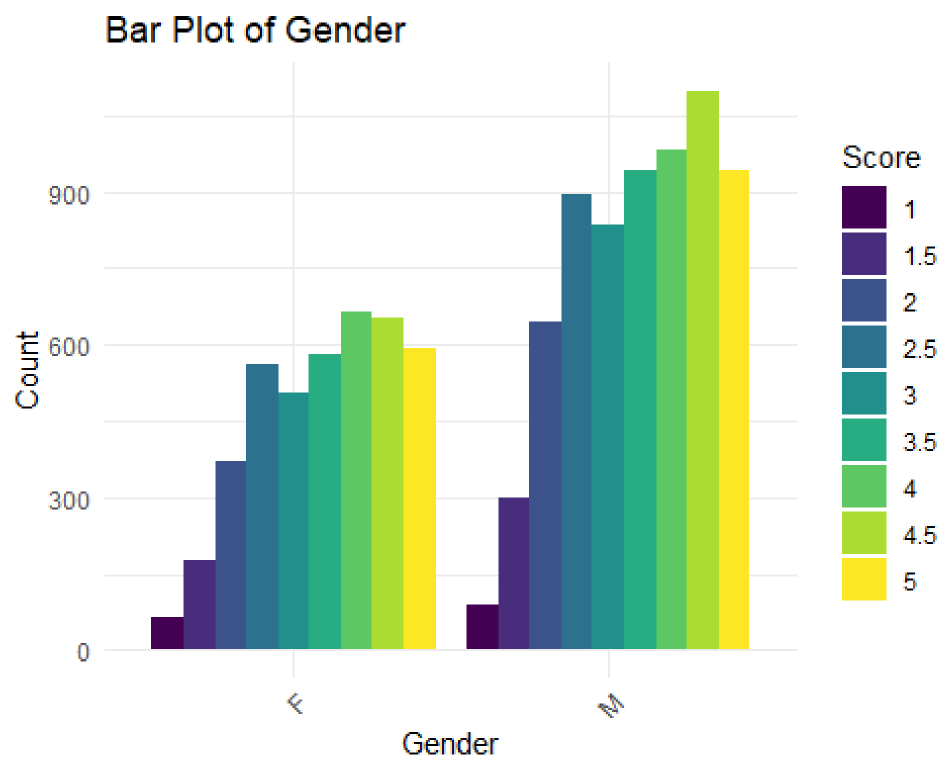
```r
# Filling the blank rows in the `Rural_metropolitan_Desc` with `NA`
data <- data %>%
  mutate(Rural_metropolitan_Desc = ifelse(Rural_metropolitan_Desc == "", NA,
Rural_metropolitan_Desc))
```

Item A4

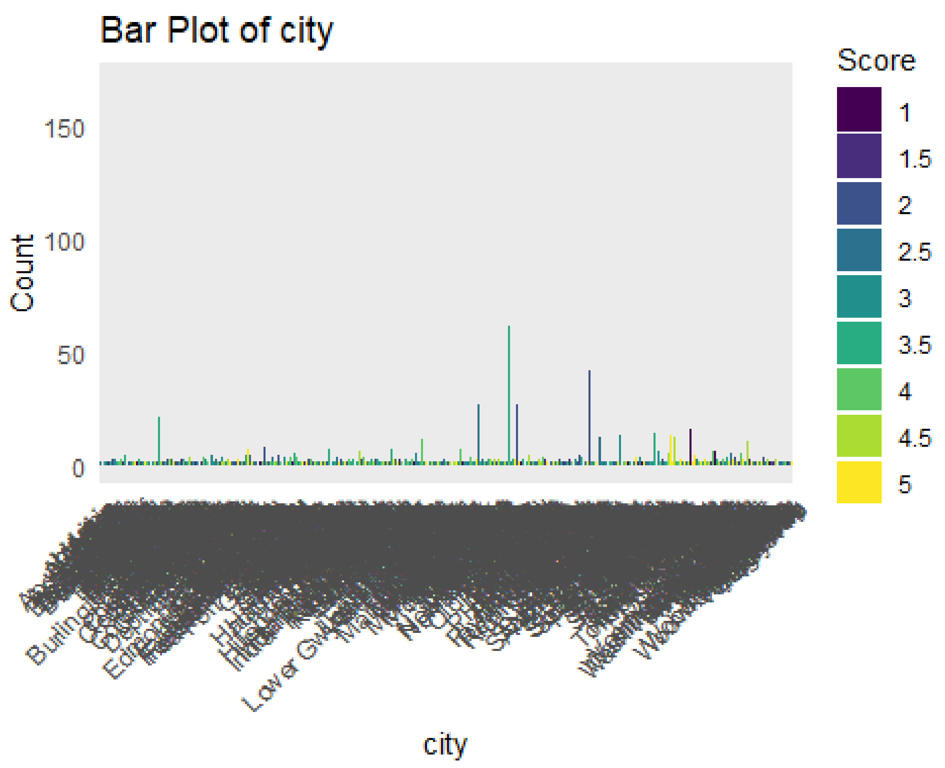
```r
library(ggplot2)
library(viridis)
# Categorical variables list
categorical_vars <- c("city", "state", "Gender", "field_cat", "Rural_metropol
itan_Desc")

# Update bar plots for categorical variables
categorical_plots <- lapply(categorical_vars, function(var) {
  ggplot(data, aes_string(x = var, fill = as.factor(data$score))) +
    geom_bar(position = "dodge") +
    scale_fill_viridis(discrete = TRUE) +  # Apply viridis color scale
    labs(title = paste("Bar Plot of", var), x = var, y = "Count") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    guides(fill = guide_legend(title = "Score"))
})

## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

# Print the first plot to verify changes
print(categorical_plots[[2]])
```



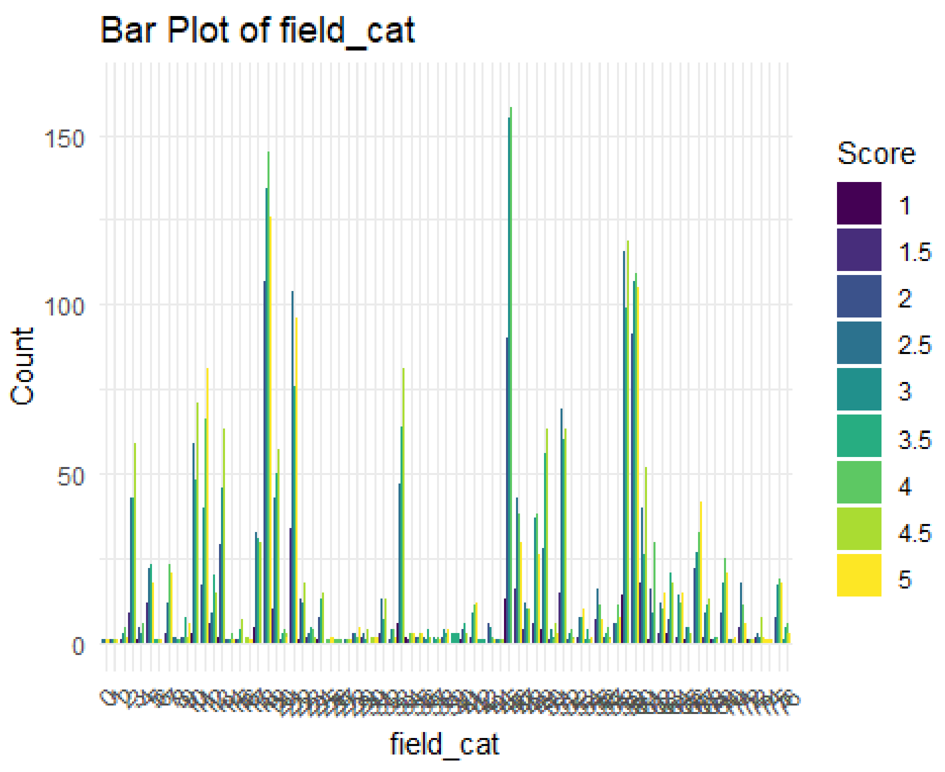
```
print(categorical_plots[[3]])
```



```
print(categorical_plots[[1]])
```

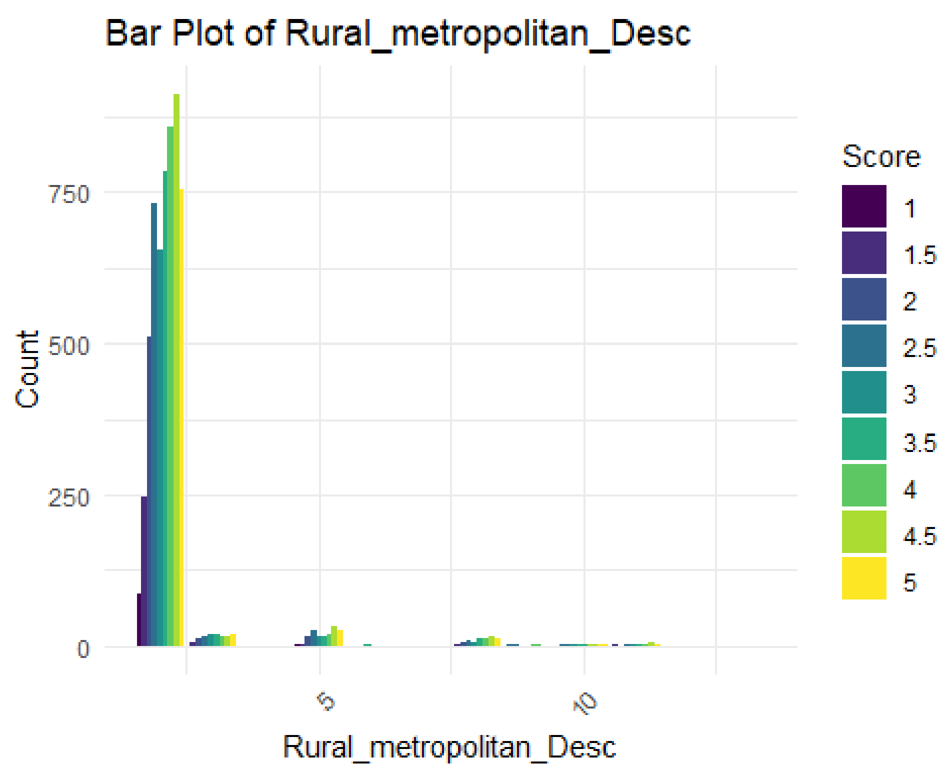


```
print(categorical_plots[[4]])
```



```
print(categorical_plots[[5]])
```

```
## Warning: Removed 4920 rows containing non-finite outside the scale range
## (`stat_count()`).
```



```
library(dplyr)
library(knitr)
library(kableExtra)

#GENERATING TABLES WITH LEVEL VARIBALES AND COUNTS

# Categorical variables
categorical_vars <- c("city", "state", "Gender", "field_cat", "Rural_metropol
itan_Desc")

# Function to create and print a table for each variable
create_count_tables <- function(data, vars) {
  tables_list <- list()

  for (var in vars) {
    # Summarize counts for each category of the variable
    summary_table <- data %>%
      group_by(!!sym(var)) %>%
      summarise(Count = n(), .groups = 'drop') %>%
      arrange(desc(Count))  # Sorting by Count for better readability

    # Generating the table with kable and style it
    tables_list[[var]] <- kable(summary_table, format = "html", caption = pas
te("Counts of", var)) %>%
      kable_styling(bootstrap_options = c("striped", "hover", "condensed"), f
ull_width = FALSE) %>%
      column_spec(1, bold = TRUE, color = "blue") %>%
      column_spec(2, background = "lavender")
  }

  return(tables_list)
}

# Creating tables
tables_output <- create_count_tables(data, categorical_vars)

# printing tables
tables_output[["city"]]  # Example of how to access a specific table, replace
"city" with other variable names

tables_output[["state"]]
```

Counts of state

state

Count

PA

2548

FL

2166

AZ

892

TN

843

NV

732

IN

684

MO

682

NJ

557

LA

535

CA

498

ID

329

AB

172

DE

128

IL

124

HI

1

```
tables_output[["Gender"]]
```

Counts of Gender

Gender

Count

M

6724

F

4167

INSIGHTS

The exploratory data analysis conducted on the dataset, particularly focusing on categorical variables such as state, gender, and rural versus metropolitan status, provides crucial insights into the characteristics of the dataset and informs the strategy for model building. Here are some insights and the rationale behind the selection of these fields:

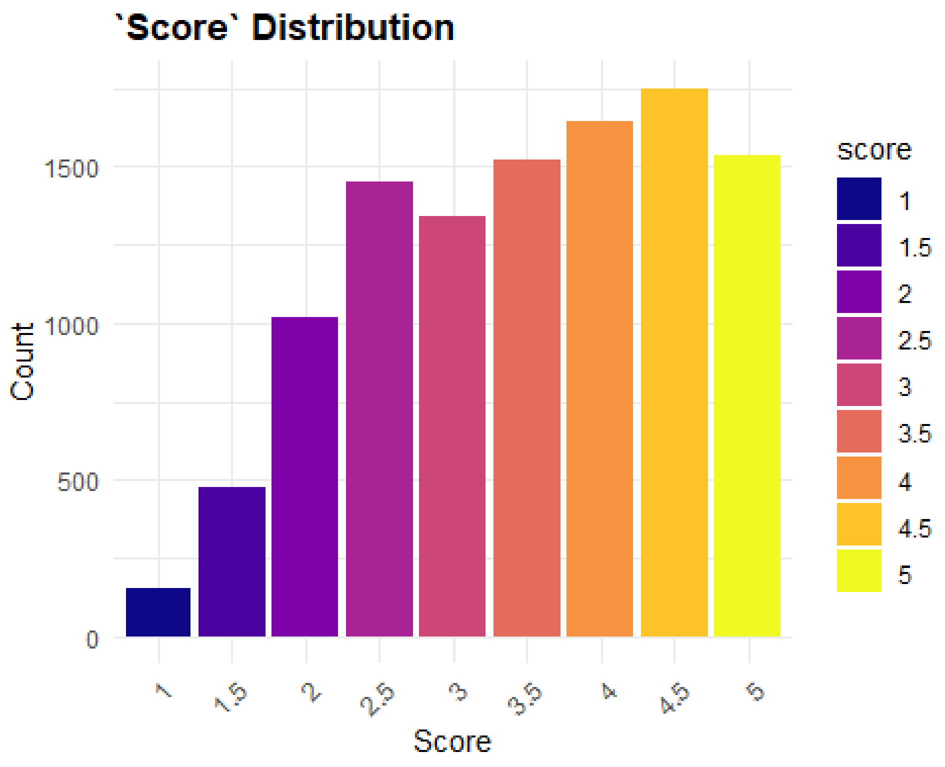
1. **State Distribution**: The bar plot of states reveals significant variations in the distribution of scores across different states. For instance, states like PA, FL, and AZ not only have high data counts but also show varied distributions across different scores, indicating state-specific trends that could affect business scores. This diversity in the data suggests that 'state' could be a significant predictor of business scores, capturing regional differences that might influence the ratings.

2. **Gender Distribution**: Analysis of gender distribution shows a disparity in counts between males (M) and females (F), with males being more prevalent in the dataset. The score distribution across genders also varies, suggesting potential gender-based differences in scoring patterns. This could imply that gender may play a role in how businesses are perceived or rated, making it an important variable for understanding influences on business scores.

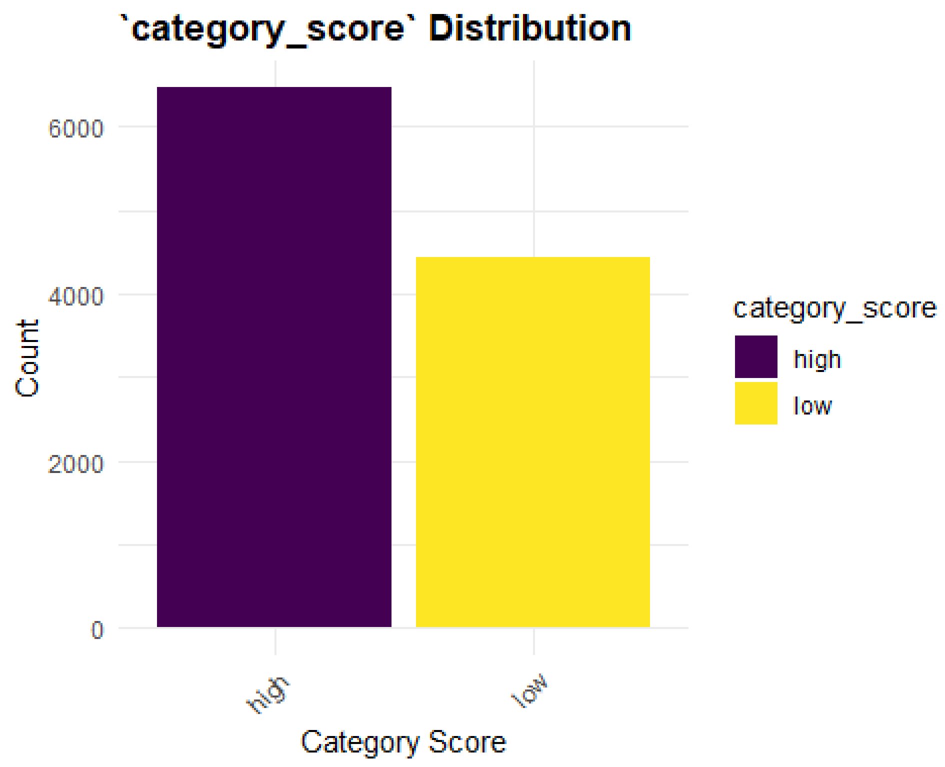
3. **Rural vs. Metropolitan Description**: The plot for rural versus metropolitan descriptions indicates that the majority of data points are classified under a few categories, with a clear predominance of metropolitan areas. This categorization helps in distinguishing between different types of geographical settings, which could significantly influence business performance and scores. The stark contrast between the number of businesses in metropolitan versus rural settings could highlight differences in business dynamics and customer interactions based on location.

The selection of these fields for deeper analysis and inclusion in modeling efforts is driven by their potential to capture key aspects of demographic, geographic, and social factors that are likely to influence business scores. By incorporating these variables, the models can more accurately reflect the complex reality in which these businesses operate, potentially improving the predictability and relevance of the analytical outcomes. This strategic choice is supported by the observed data distributions and their implications on the business environment, guiding the development of models

```r
library(ggplot2)
library(dplyr)
library(viridis)

# Assuming data transformation has already occurred:
data <- data %>%
  mutate(category_score = case_when(
    score <= 3 ~ "low",
    score > 3 ~ "high"
  )) %>%
  mutate(
    score = as.factor(score),
    category_score = as.factor(category_score)
  )
# Score Distribution using ggplot2 with viridis
score_plot <- ggplot(data, aes(x = score, fill = score)) +
  geom_bar(stat = "count") +
  scale_fill_viridis(discrete = TRUE, option = "C") +
  labs(title = "`Score` Distribution", x = "Score", y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), plot.title = eleme
nt_text(face = "bold"))

# Category Score Distribution using ggplot2 with viridis
category_score_plot <- ggplot(data, aes(x = category_score, fill = category_s
core)) +
  geom_bar(stat = "count") +
  scale_fill_viridis(discrete = TRUE, option = "D") +
  labs(title = "`category_score` Distribution", x = "Category Score", y = "Co
unt") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), plot.title = eleme
nt_text(face = "bold"))

# Display the plots
print(score_plot)
```



```r
print(category_score_plot)
```



```r
#removing the score
data <- data %>% select(-c(score))
```

Subset Generation

```r
# Field 59
subset1_field_59 <- data %>% filter(field_cat %in% c(59,51, 28, 57, 45, 76, 7
5, 69, 55, 49))

# Field 18
subset2_field_18 <- data %>% filter(field_cat %in% c(5, 46, 60, 59, 48,  41,
62, 58, 33))

# Field 45
subset3_field_45 <- data %>% filter(field_cat %in% c(67, 61, 43, 23,47,18, 10
, 3))
```

## Train - Test Generation

# Train - Test Generation

r train test generation on all the subsets

```
data_field_18 <- trainTestSplit(data = subset2_field_18)

data_field_59 <- trainTestSplit(data = subset1_field_59)

data_field_45 <- trainTestSplit(data = subset3_field_45)
```

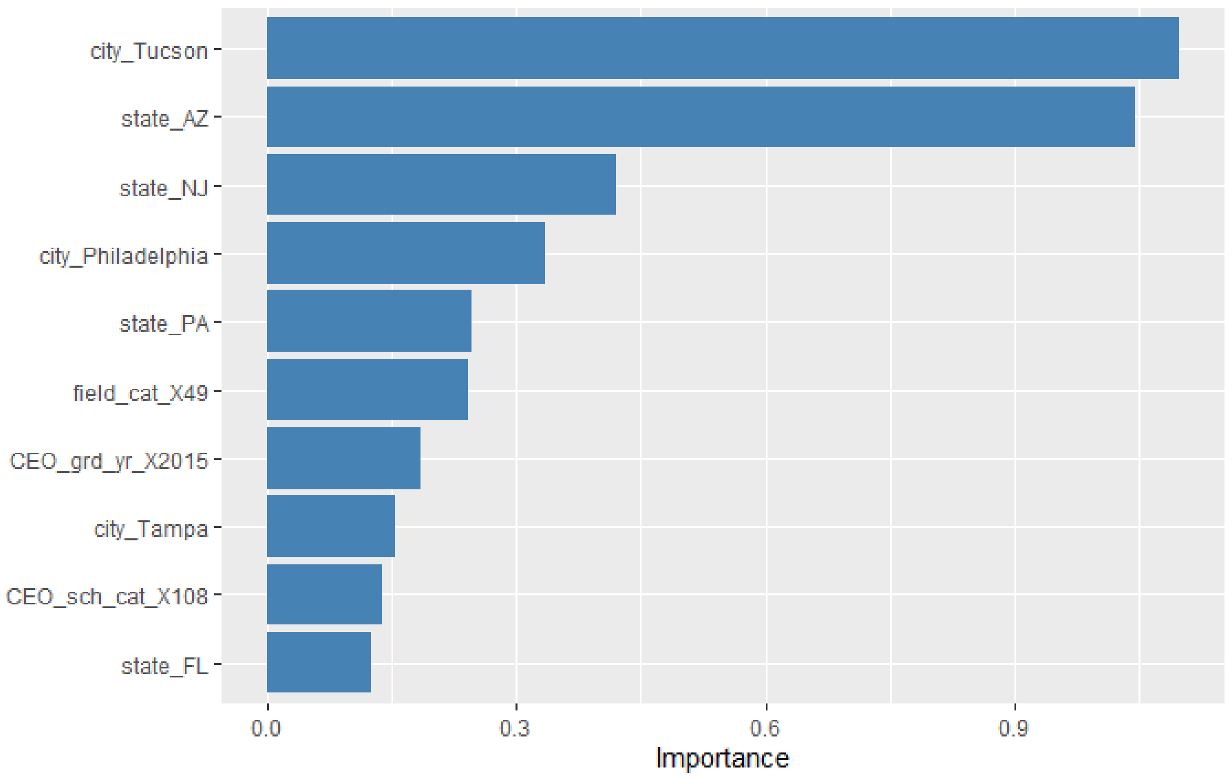
ITEM B1 : LASSO MODEL

Lasso Subset - 1

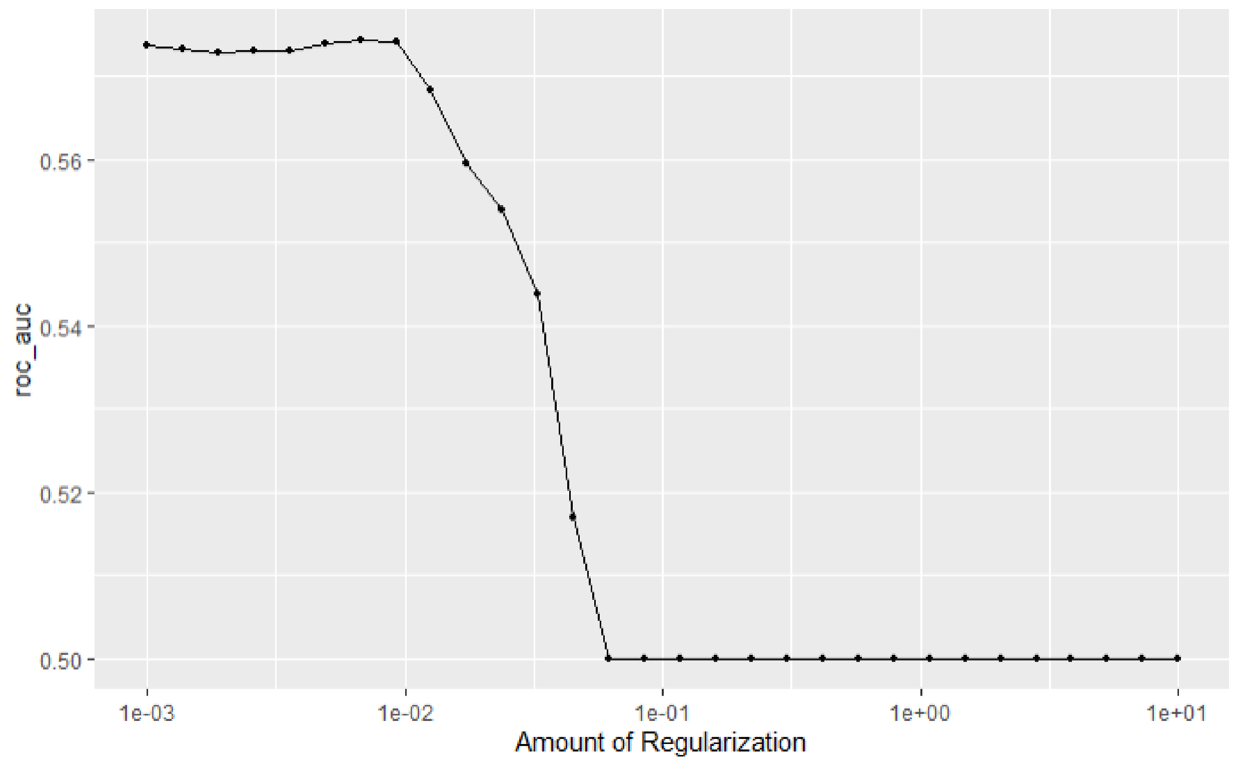
```
lasso_model_subset1 <- lassoModelFunction(subset_training_data = data_field_5
9$training_data,
                                          recipe = data_field_59$recipe)

## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## • `city_Feasterville.Trevose` -> `city_Feasterville.Trevose...148`
## • `city_Feasterville.Trevose` -> `city_Feasterville.Trevose...149`
## • `city_Mt..Juliet` -> `city_Mt..Juliet...294`
## • `city_Mt..Juliet` -> `city_Mt..Juliet...298`
## • `city_O.Fallon` -> `city_O.Fallon...319`
## • `city_O.Fallon` -> `city_O.Fallon...320`
## • `city_St.Louis` -> `city_St.Louis...410`
## • `city_St.Pete.Beach` -> `city_St.Pete.Beach...411`
## • `city_St.Petersburg` -> `city_St.Petersburg...412`
## • `city_St.Louis` -> `city_St.Louis...419`
## • `city_St.Pete.Beach` -> `city_St.Pete.Beach...420`
## • `city_St.Petersburg` -> `city_St.Petersburg...421`
```

```
lasso_model_subset1
```

```
## $top10
```



```
##
## $1
```



Top 10 Variables for subset - 1

```
train1 <- data_field_59$recipe %>%
  prep(data_field_59$training_data)

## New names:
## • `city_Feasterville.Trevose` -> `city_Feasterville.Trevose...148`
## • `city_Feasterville.Trevose` -> `city_Feasterville.Trevose...149`
## • `city_Mt..Juliet` -> `city_Mt..Juliet...294`
## • `city_Mt..Juliet` -> `city_Mt..Juliet...298`
## • `city_O.Fallon` -> `city_O.Fallon...319`
## • `city_O.Fallon` -> `city_O.Fallon...320`
## • `city_St.Louis` -> `city_St.Louis...410`
## • `city_St.Pete.Beach` -> `city_St.Pete.Beach...411`
## • `city_St.Petersburg` -> `city_St.Petersburg...412`
## • `city_St.Louis` -> `city_St.Louis...419`
## • `city_St.Pete.Beach` -> `city_St.Pete.Beach...420`
## • `city_St.Petersburg` -> `city_St.Petersburg...421`

field59_after_lasso <- train1$template %>% select(c(city_Tucson,state_AZ, sta
te_NJ, city_Philadelphia, state_PA, field_cat_X49, CEO_grd_yr_X2015, city_Tam
pa, CEO_sch_cat_X108, state_FL)) %>% mutate(category_score = data_field_59$tr
aining_data$category_score)

formula_1 <- category_score ~ city_Tucson + state_AZ + state_NJ + city_Philad
elphia + state_PA + field_cat_X49 + CEO_grd_yr_X2015 + city_Tampa + CEO_sch_c
at_X108 + state_FL
```

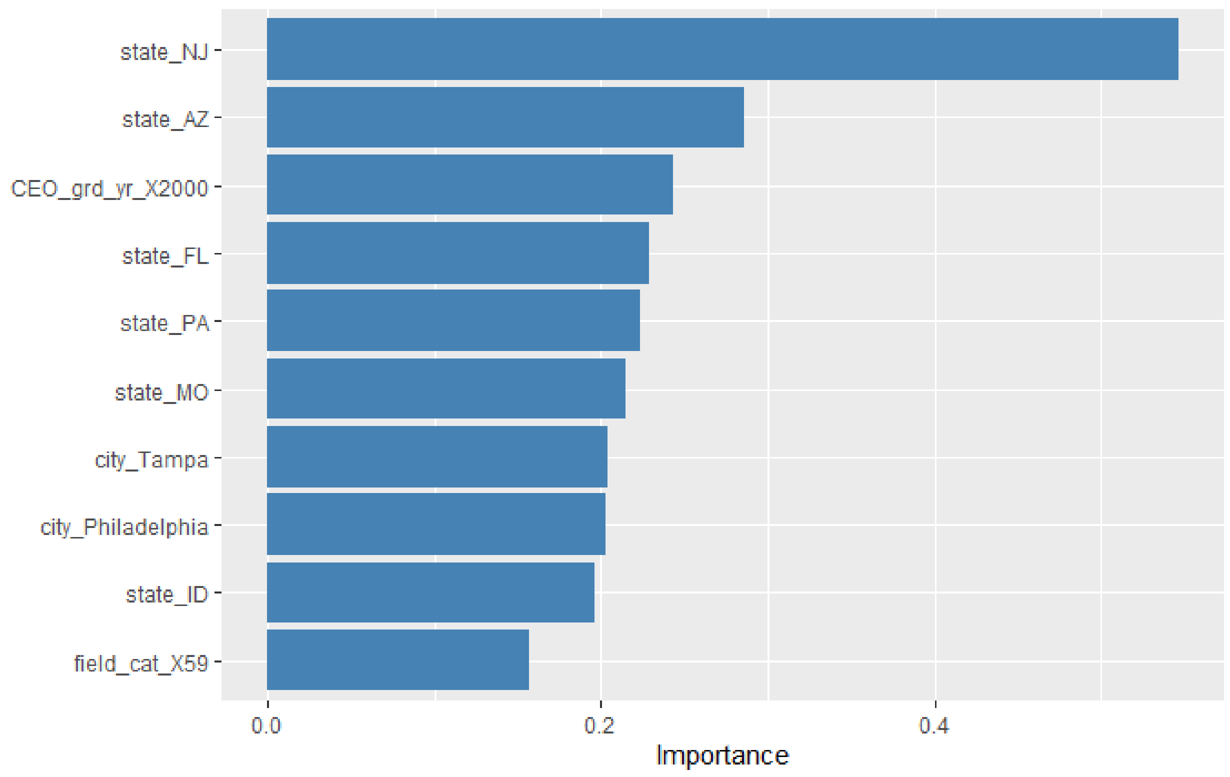
Lasso subset - 2

```
lasso_model_subset2 <- lassoModelFunction(subset_training_data = data_field_1
8$training_data, recipe = data_field_18$recipe)
```

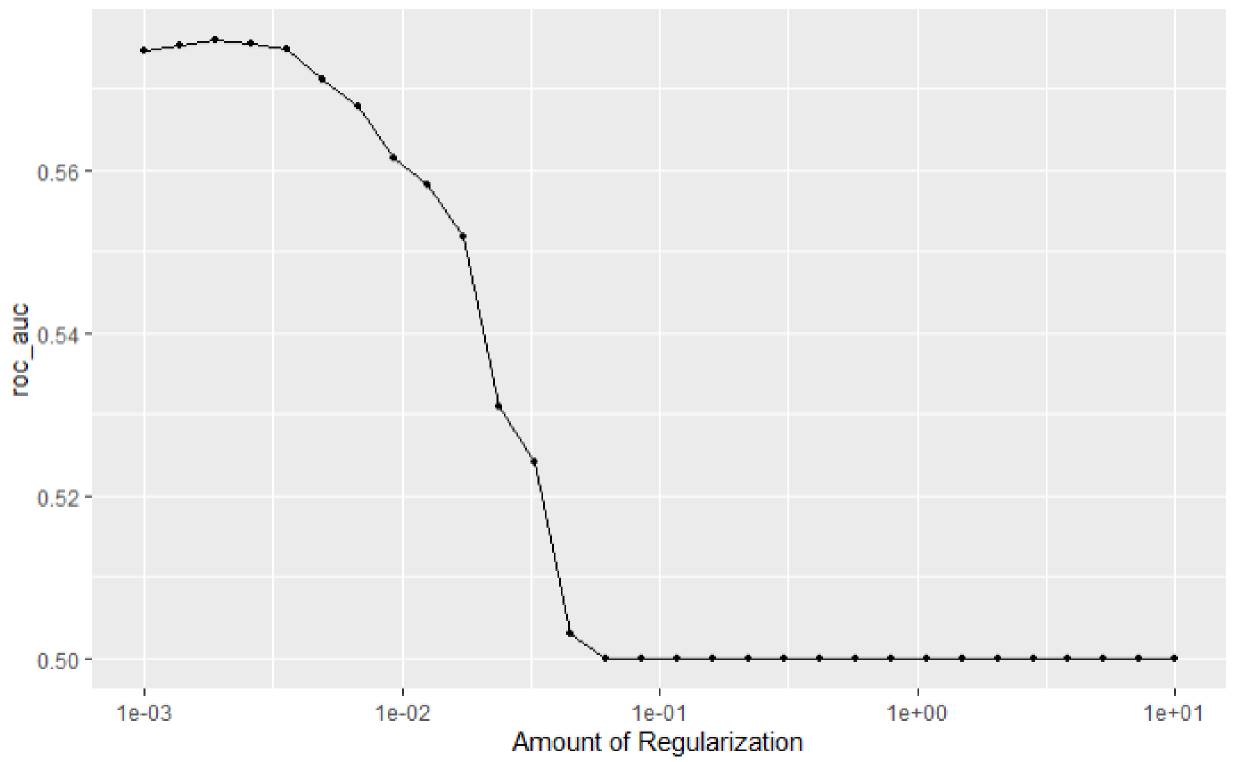
```
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## • `city_Feasterville.Trevose` -> `city_Feasterville.Trevose...148`
## • `city_Feasterville.Trevose` -> `city_Feasterville.Trevose...149`
## • `city_Mt..Juliet` -> `city_Mt..Juliet...294`
## • `city_Mt..Juliet` -> `city_Mt..Juliet...298`
## • `city_O.Fallon` -> `city_O.Fallon...319`
## • `city_O.Fallon` -> `city_O.Fallon...320`
## • `city_St.Louis` -> `city_St.Louis...410`
## • `city_St.Pete.Beach` -> `city_St.Pete.Beach...411`
## • `city_St.Petersburg` -> `city_St.Petersburg...412`
## • `city_St.Louis` -> `city_St.Louis...419`
## • `city_St.Pete.Beach` -> `city_St.Pete.Beach...420`
## • `city_St.Petersburg` -> `city_St.Petersburg...421`
```

```
lasso_model_subset2
```

```
## $top10
```


state_NJ
state_AZ
CEO_grd_yr_X2000
state_FL
state_PA
state_MO
city_Tampa
city_Philadelphia
state_ID
field_cat_X59
0.0
0.2
0.4
Importance

```
##
## $1
```


roc_auc
0.56
0.54
0.52
0.50
1e-03
1e-02
1e-01
1e+00
1e+01
Amount of Regularization

Top 10 Variables for subset - 2

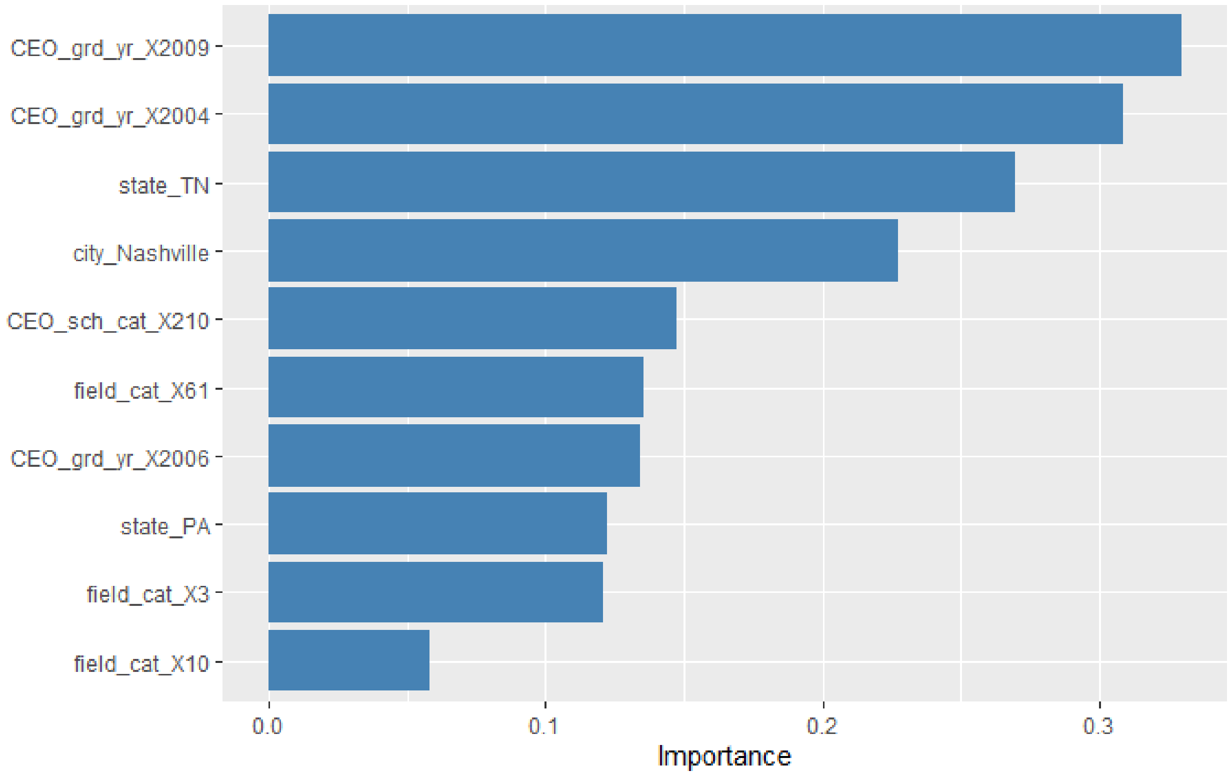
```
train2 <- data_field_18$recipe %>%
  prep(data_field_18$training_data)

## New names:
## • `city_Feasterville.Trevose` -> `city_Feasterville.Trevose...148`
## • `city_Feasterville.Trevose` -> `city_Feasterville.Trevose...149`
## • `city_Mt..Juliet` -> `city_Mt..Juliet...294`
## • `city_Mt..Juliet` -> `city_Mt..Juliet...298`
## • `city_O.Fallon` -> `city_O.Fallon...319`
## • `city_O.Fallon` -> `city_O.Fallon...320`
## • `city_St.Louis` -> `city_St.Louis...410`
## • `city_St.Pete.Beach` -> `city_St.Pete.Beach...411`
## • `city_St.Petersburg` -> `city_St.Petersburg...412`
## • `city_St.Louis` -> `city_St.Louis...419`
## • `city_St.Pete.Beach` -> `city_St.Pete.Beach...420`
## • `city_St.Petersburg` -> `city_St.Petersburg...421`

field18_after_lasso <- train2$template %>% select(state_NJ, state_AZ, CEO_grd
_yr_X2000, state_FL, state_PA, state_MO, city_Tampa, city_Philadelphia, state
_ID, field_cat_X59) %>% mutate(category_score = data_field_18$training_data$c
ategory_score)

formula_2 <- category_score ~ state_NJ + state_AZ + CEO_grd_yr_X2000 + state_
FL + state_PA + state_MO + city_Tampa + city_Philadelphia + state_ID + field_
cat_X59
```

Lasso subset - 3

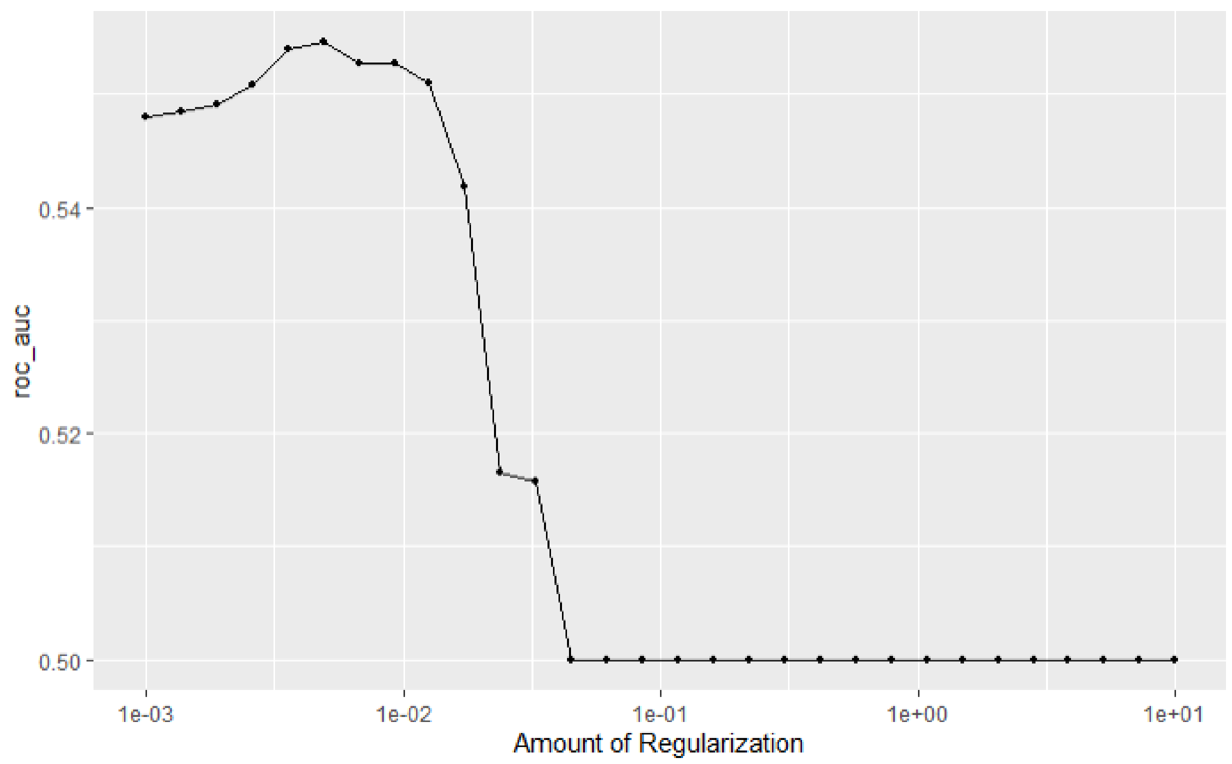
```
lasso_model_subset3 <- lassoModelFunction(subset_training_data = data_field_4
5$training_data, recipe = data_field_45$recipe)

## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## New names:
## • `city_Feasterville.Trevose` -> `city_Feasterville.Trevose...147`
## • `city_Feasterville.Trevose` -> `city_Feasterville.Trevose...148`
## • `city_Mt..Juliet` -> `city_Mt..Juliet...293`
## • `city_Mt..Juliet` -> `city_Mt..Juliet...297`
## • `city_O.Fallon` -> `city_O.Fallon...318`
## • `city_O.Fallon` -> `city_O.Fallon...319`
## • `city_St.Louis` -> `city_St.Louis...409`
## • `city_St.Pete.Beach` -> `city_St.Pete.Beach...410`
## • `city_St.Petersburg` -> `city_St.Petersburg...411`
## • `city_St.Louis` -> `city_St.Louis...418`
## • `city_St.Pete.Beach` -> `city_St.Pete.Beach...419`
## • `city_St.Petersburg` -> `city_St.Petersburg...420`
```

```
lasso_model_subset3
```

```
## $top10
```



```
##
## $1
```



Top 10 Variables for subset - 3

```
train3 <- data_field_45$recipe %>%
  prep(data_field_45$training_data)

## New names:
## • `city_Feasterville.Trevose` -> `city_Feasterville.Trevose...147`
## • `city_Feasterville.Trevose` -> `city_Feasterville.Trevose...148`
## • `city_Mt..Juliet` -> `city_Mt..Juliet...293`
## • `city_Mt..Juliet` -> `city_Mt..Juliet...297`
## • `city_O.Fallon` -> `city_O.Fallon...318`
## • `city_O.Fallon` -> `city_O.Fallon...319`
## • `city_St.Louis` -> `city_St.Louis...409`
## • `city_St.Pete.Beach` -> `city_St.Pete.Beach...410`
## • `city_St.Petersburg` -> `city_St.Petersburg...411`
## • `city_St.Louis` -> `city_St.Louis...418`
## • `city_St.Pete.Beach` -> `city_St.Pete.Beach...419`
## • `city_St.Petersburg` -> `city_St.Petersburg...420`

field45_after_lasso <- train3$template %>% select(CEO_grd_yr_X2009, CEO_grd_y
r_X2004, state_TN, city_Nashville, CEO_sch_cat_X210, field_cat_X61, CEO_grd_y
r_X2006, state_PA, field_cat_X3, field_cat_X10) %>% mutate(category_score = d
ata_field_45$training_data$category_score)

formula_3 <- category_score ~ CEO_grd_yr_X2009 + CEO_grd_yr_X2004 + state_TN
+ city_Nashville + CEO_sch_cat_X210 + field_cat_X61 + CEO_grd_yr_X2006 + stat
e_PA + field_cat_X3 + field_cat_X10
```

REGULARIZATION EFFECTS AND MODEL INTERPRETATION

Regularization Plots:

- Across all subsets, the ROC-AUC seems to be stable at lower levels of regularization (higher lambda values) but starts to peak as the amount of regularization decreases (moving towards less penalization).
- This pattern suggests that a moderate level of regularization helps the model by preventing overfitting, while too little regularization allows the model to overfit, resulting in lower generalizability on unseen data.
- The optimal lambda value (where ROC-AUC is highest before it drops) indicates the best balance between bias and variance for the model.

Variable Importance:

1. **Subset 1**
    - The top variables impacting the model include location-specific factors such as **city_Tucson**, **state_AZ**, **state_NJ**, **city_Philadelphia**, and **state_PA**. This indicates a strong regional influence on business scores.
    - Variables related to the CEO's graduation year and specific business fields (**field_cat**) also play significant roles, suggesting that leadership and field-specific dynamics are critical in predicting business performance.
2. **Subset 2**
    - In this subset, **state_NJ** and **state_AZ** are again prominent, underscoring the importance of geographic factors in influencing business scores.
    - The CEO's graduation year and certain cities (**city_Tampa**, **city_Philadelphia**) also appear as important predictors. This reflects both the influence of leadership qualities and local market conditions.
3. **Subset 3**
    - The variables like **state_NJ** and **CEO_grd_yr_X2000** top the list, indicating the relevance of both the regional regulatory environment and the era of leadership training.
    - Other state variables (**state_FL**, **state_MO**, and **state_PA**) and city names (**city_Tampa** and **city_Philadelphia**) suggest that local factors are crucial in shaping business scores across various regions.

Insights and Strategic Implications:

- **Regional Influence**: The recurring importance of state and city variables across all subsets suggests that regional factors such as local economic conditions, regulatory environments, and cultural elements significantly affect business scores. This insight could be strategically used to tailor business practices to regional specifics.

- **Leadership Impact**: The prominence of variables related to the CEO's graduation year across subsets suggests that leadership qualities, possibly influenced by education and era, significantly impact business outcomes. This could imply the need for ongoing leadership development tailored to evolving business landscapes.
- **Field-Specific Dynamics**: The appearance of **field_cat** variables indicates that industry-specific factors also play a significant role in determining business scores. Understanding these dynamics could help businesses in targeted fields to optimize their strategies according to industry trends.

The analysis not only highlight what factors are most influential in determining business scores but also underscore the value of regional and leadership adjustments in strategic business planning. By focusing on these key variables, businesses can better position themselves in competitive markets and adapt to local and industry-specific challenges.

ITEM - B2 : GAM Model Building

Subset - 1

```
GAM_MODEL1<-GAM_function(formula = formula_1, data = field59_after_lasso)

##
## Family: binomial
## Link function: logit
##
## Formula:
## category_score ~ city_Tucson + state_AZ + state_NJ + city_Philadelphia +
##     state_PA + field_cat_X49 + CEO_grd_yr_X2015 + city_Tampa +
##     CEO_sch_cat_X108 + state_FL
##
## Parametric coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.62446    0.09406  -6.639 3.15e-11 ***
## city_Tucson         11.37098  113.59009   0.100 0.920261
## state_AZ           -11.04658  113.59006  -0.097 0.922528
## state_NJ             0.87196    0.17845   4.886 1.03e-06 ***
## city_Philadelphia   -0.65935    0.22396  -2.944 0.003239 **
## state_PA             0.53280    0.13792   3.863 0.000112 ***
## field_cat_X49       -0.55957    0.16154  -3.464 0.000532 ***
## CEO_grd_yr_X2015     0.33771    0.16700   2.022 0.043151 *
## city_Tampa          -0.28385    0.21179  -1.340 0.180171
## CEO_sch_cat_X108     0.15686    0.09868   1.590 0.111937
## state_FL             0.27760    0.14507   1.913 0.055687 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.0255   Deviance explained = 2.24%
## -REML = 1539.5  Scale est. = 1         n = 2315
## [1] "No terms available to plot. Check the formula used in the GAM model."
```

Subset - 2

```
GAM_function(formula = formula_2, data = field18_after_lasso)

##
## Family: binomial
## Link function: logit
##
## Formula:
## category_score ~ state_NJ + state_AZ + CEO_grd_yr_X2000 + state_FL +
##     state_PA + state_MO + city_Tampa + city_Philadelphia + state_ID +
##     field_cat_X59
##
## Parametric coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.63499    0.09560  -6.642 3.09e-11 ***
## state_NJ           1.01561    0.19668   5.164 2.42e-07 ***
## state_AZ           0.61766    0.15781   3.914 9.08e-05 ***
## CEO_grd_yr_X2000   0.47224    0.22052   2.141 0.032236 *
## state_FL           0.51708    0.14385   3.595 0.000325 ***
## state_PA           0.44686    0.14826   3.014 0.002578 **
## state_MO           0.44303    0.16742   2.646 0.008139 **
## city_Tampa        -0.53291    0.21993  -2.423 0.015390 *
## city_Philadelphia -0.46090    0.21558  -2.138 0.032523 *
## state_ID          -0.34504    0.22372  -1.542 0.122997
## field_cat_X59     -0.06494    0.11089  -0.586 0.558090
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.0213   Deviance explained =  1.9%
## -REML = 1557.2  Scale est. = 1         n = 2313
## [1] "No terms available to plot. Check the formula used in the GAM model."
```

Subset - 3

```
GAM_function(formula = formula_3, data = field45_after_lasso)

##
## Family: binomial
## Link function: logit
##
## Formula:
## category_score ~ CEO_grd_yr_X2009 + CEO_grd_yr_X2004 + state_TN +
##     city_Nashville + CEO_sch_cat_X210 + field_cat_X61 + CEO_grd_yr_X2006 +
##     state_PA + field_cat_X3 + field_cat_X10
##
## Parametric coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.46741    0.09694  -4.821 1.43e-06 ***
## CEO_grd_yr_X2009   0.76930    0.24225   3.176  0.00149 **
## CEO_grd_yr_X2004   0.54738    0.19283   2.839  0.00453 **
## state_TN           0.46344    0.32356   1.432  0.15206
## city_Nashville    -0.44075    0.42710  -1.032  0.30209
## CEO_sch_cat_X210   0.46047    0.15730   2.927  0.00342 **
## field_cat_X61     -0.27055    0.21777  -1.242  0.21411
## CEO_grd_yr_X2006   0.24682    0.33037   0.747  0.45500
## state_PA           0.07637    0.13029   0.586  0.55775
## field_cat_X3      -0.22249    0.17213  -1.293  0.19616
## field_cat_X10      0.09808    0.16704   0.587  0.55708
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.0157   Deviance explained = 1.61%
## -REML = 1048.6  Scale est. = 1         n = 1543
## [1] "No terms available to plot. Check the formula used in the GAM model."
```

INTERPRETATIONS AND INSIGHTS FROM GAM MODEL

Overview of GAM Results: The results from our Generalized Additive Models (GAM) for three different subsets provide some interesting insights into what might influence business scores. Even though the models explain a relatively small part of the variability in scores (with deviance explained ranging around 1.61% to 2.24%), they highlight the importance of specific factors like geographical location, leadership years, and business sectors.

Subset 1 Insights:

- **Model Performance:** The explained deviance is relatively low at 2.24%, suggesting that while the model captures some variability in the data, a significant portion of the outcome variability remains unexplained by the predictors included.
- **Significant Predictors:** Notably, **state_NJ** positively influences the score, suggesting that businesses in New Jersey are more likely to have a high score compared to the reference category. Conversely, **city_Philadelphia** and **field_cat_X49** have negative impacts, indicating a lower likelihood of high scores for businesses in Philadelphia or those classified under **field_cat_X49**.
- **Leadership Influence: CEO_grd_yr_X2015** shows a positive but mild effect, implying that more recent leadership (graduates of 2015) slightly increases the likelihood of higher business scores.

Subset 2 Insights:

- **Model Performance:** Similar to Subset 1, the explained deviance is also low at 1.9%, indicating that other unmodeled factors may be influencing the business scores.
- **Geographic Impact:** Both **state_NJ** and **state_AZ** show a positive influence, reinforcing the idea that regional factors significantly affect business scores. The negative coefficient for **city_Tampa** suggests a lower probability of high scores, which might reflect local economic or business challenges.
- **Leadership and Timing: CEO_grd_yr_X2000** also positively influences scores, indicating that the leadership qualities or conditions around the year 2000 favorably impact business evaluations.

Subset 3 Insights:

- **Model Performance:** This subset has the lowest explained deviance at 1.61%, highlighting the complexity and potential oversights in capturing influential factors in the model.
- **Leadership Dynamics:** Leadership appears as a strong theme with **CEO_grd_yr_X2009** and **CEO_grd_yr_X2004** both showing positive impacts. This could suggest that leadership qualities from these years align well with factors that contribute to higher business scores.
- **Localized Effects: state_TN** and **city_Nashville** do not show significant impacts despite being anticipated to influence business scores, possibly due to counterbalancing factors that the model does not account for.

General Observations Across Subsets:

- The consistently low R-squared values across all subsets imply that the models may need additional predictors or different modeling techniques to better capture the complexities of what influences business scores.
- The significance and impact of geographic and leadership-related variables across all subsets highlight the importance of considering both macro (state) and micro (city, leadership years) factors in strategic business analysis.

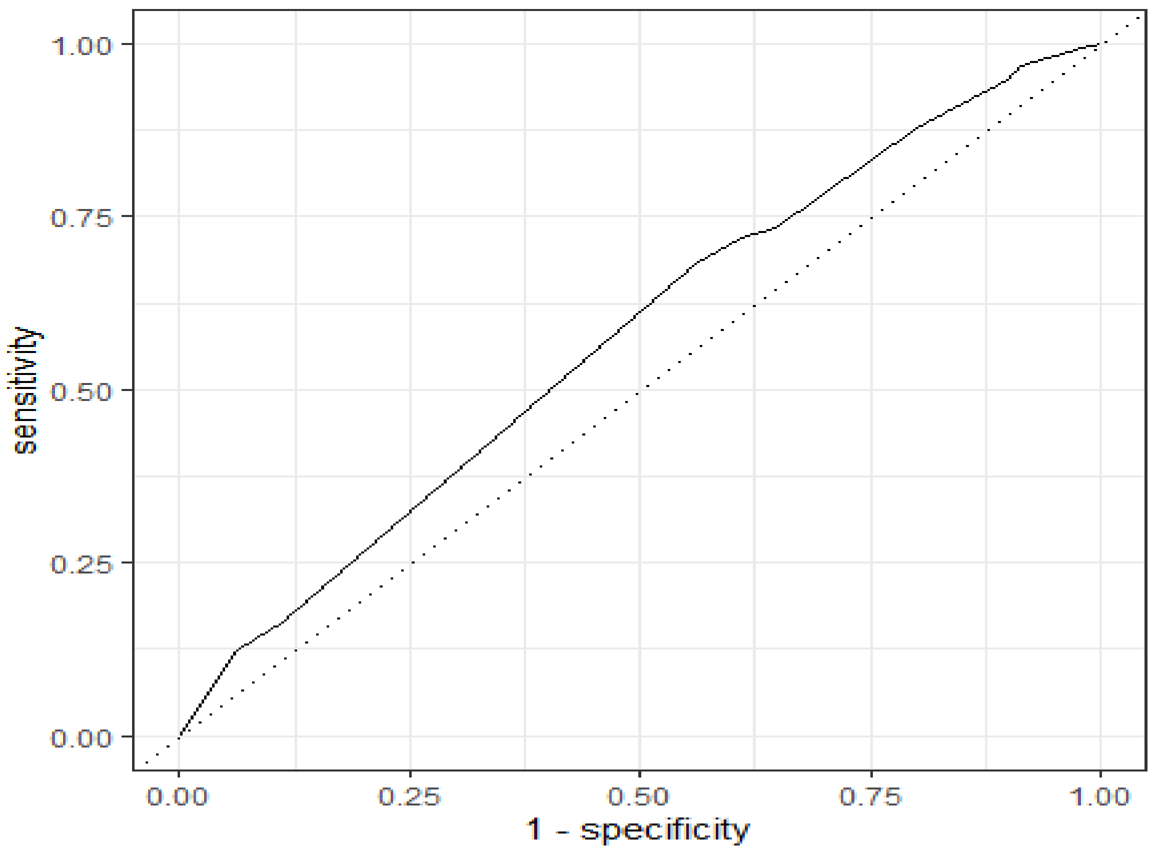
ITEM - B3 : Decision Tree Model

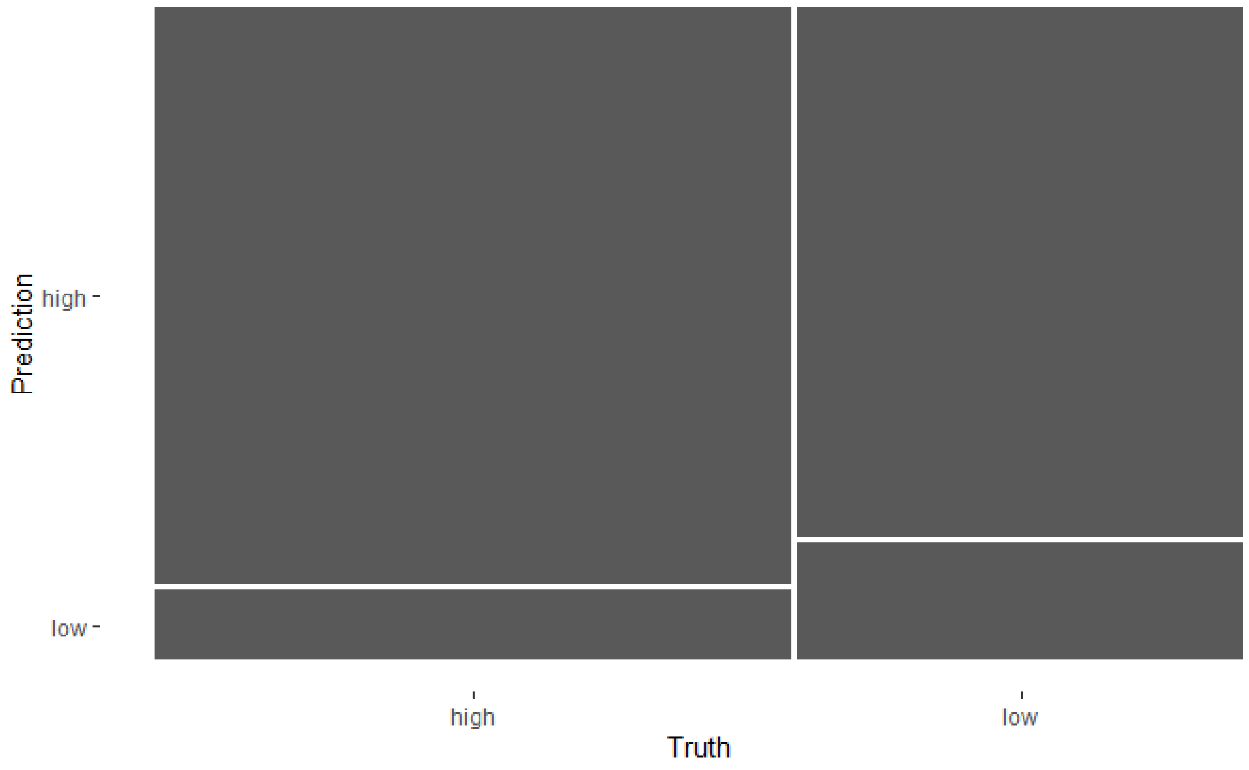
Decision Tree Model Function

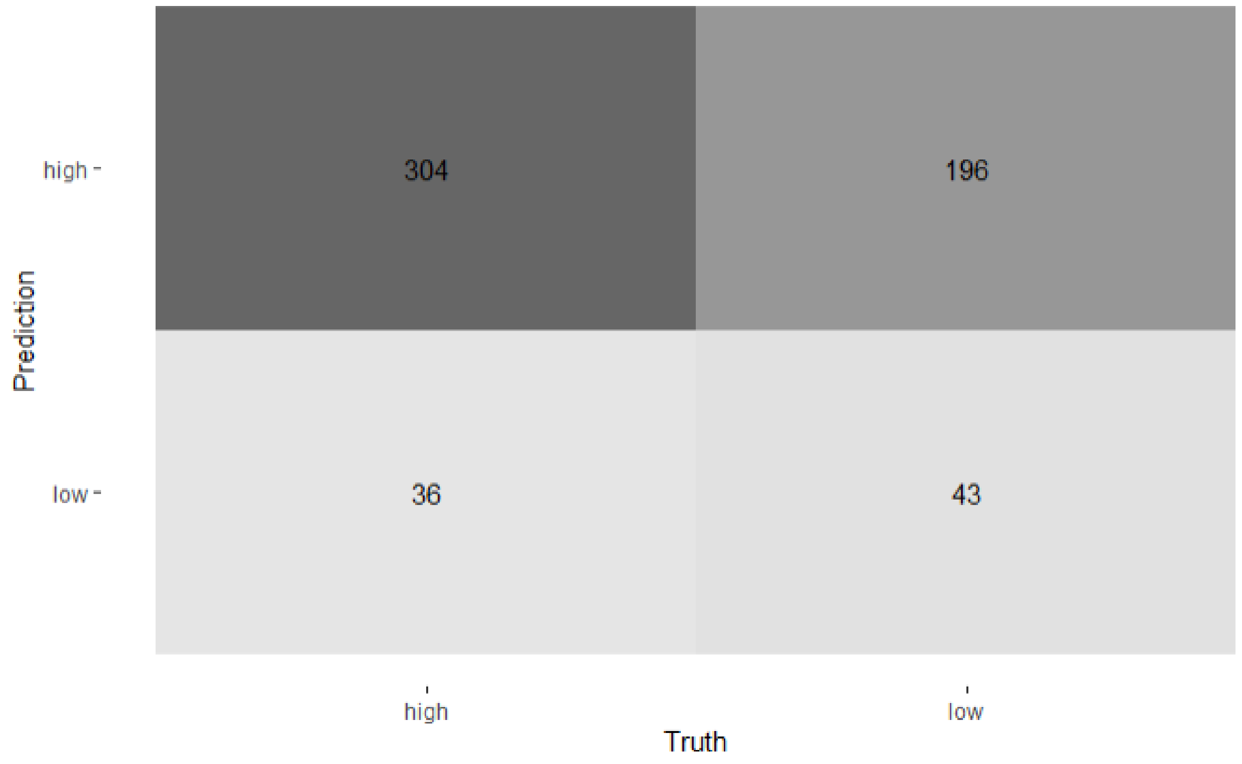
```
decisionTreeModel <- function(subsets) {
  # Split data into training and testing sets
  split <- initial_split(subsets, prop = 0.75, strata = category_score)
  training <- training(split)
  testing <- testing(split)

  # Define decision tree model for tuning
  dt_model_tune <- decision_tree(cost_complexity = tune(), tree_depth = tune(
), min_n = tune()) %>%
    set_mode("classification") %>%
    set_engine("rpart")

  # Define classification metrics (accuracy, sensitivity, specificity)
 # lead_metrics <- metric_set(accuracy, sens, specificity)

  # Define recipe for data preprocessing
  dt_recipe <- recipe(category_score ~ ., data = training) %>%
    step_impute_knn(all_numeric_predictors()) %>%
    step_impute_mode(all_factor_predictors()) %>%
    step_normalize(all_numeric_predictors()) %>%
    step_corr(all_numeric_predictors(), threshold = 0.9) %>%
    step_dummy(all_factor_predictors()) %>%
    step_zv(all_numeric(), -all_outcomes()) %>%
    step_nzv(all_predictors())

  # Create workflow for model tuning
  dt_tune_wkf <- workflow() %>%
    add_model(dt_model_tune) %>%
    add_recipe(dt_recipe)

  # Tune the decision tree model using cross-validation
  dt_tuning <- dt_tune_wkf %>%
    tune_grid(
      resamples = vfold_cv(training, v = 10, strata = category_score),
      grid = grid_random(parameters(dt_model_tune), n = 5)

    )


   # Select the best tuned decision tree model based on accuracy
  best_dt_model <- dt_tuning %>%
    select_best(metric = 'accuracy')

  print(best_dt_model)

  # Finalize the workflow with the best model
  final_leads_wkfl <- dt_tune_wkf %>%
    finalize_workflow(best_dt_model)

  # Fit the final model using the split data
  leads_final_fit <- final_leads_wkfl %>%
    last_fit(split = split)
  # Collect final evaluation metrics
  final_metrics <- leads_final_fit %>%
    collect_metrics()

  print(final_metrics)

  # Collect predictions from the final model
  predictions <- leads_final_fit %>%
    collect_predictions()

  # Plot ROC curve
  roc_curve <- predictions %>% roc_curve(truth = category_score, .pred_high)
%>% autoplot()
  print(roc_curve)

  # Plot confusion matrix mosaic
  mosaic <- conf_mat(predictions, truth = category_score, estimate = .pred_cl
ass) %>%
    autoplot(type = 'mosaic')

  print(mosaic)

  # Plot confusion matrix heatmap
  heatmap <- conf_mat(predictions, truth = category_score, estimate = .pred_c
lass) %>%
    autoplot(type = 'heatmap')

  print(heatmap)


}
```

Decision Tree Model on Subset - 1

```
dt_subset_1 <- decisionTreeModel(subsets = field59_after_lasso)

## Warning: `parameters.model_spec()` was deprecated in tune 0.1.6.9003.
## ℹ Please use `hardhat::extract_parameter_set_dials()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## # A tibble: 1 × 4
##   cost_complexity tree_depth min_n .config
##             <dbl>      <int> <int> <chr>
## 1          0.0881          1    39 Preprocessor1_Model3
## # A tibble: 3 × 4
##   .metric     .estimator .estimate .config
##   <chr>       <chr>          <dbl> <chr>
## 1 accuracy    binary         0.587 Preprocessor1_Model1
## 2 roc_auc     binary         0.5   Preprocessor1_Model1
## 3 brier_class binary         0.242 Preprocessor1_Model1
```



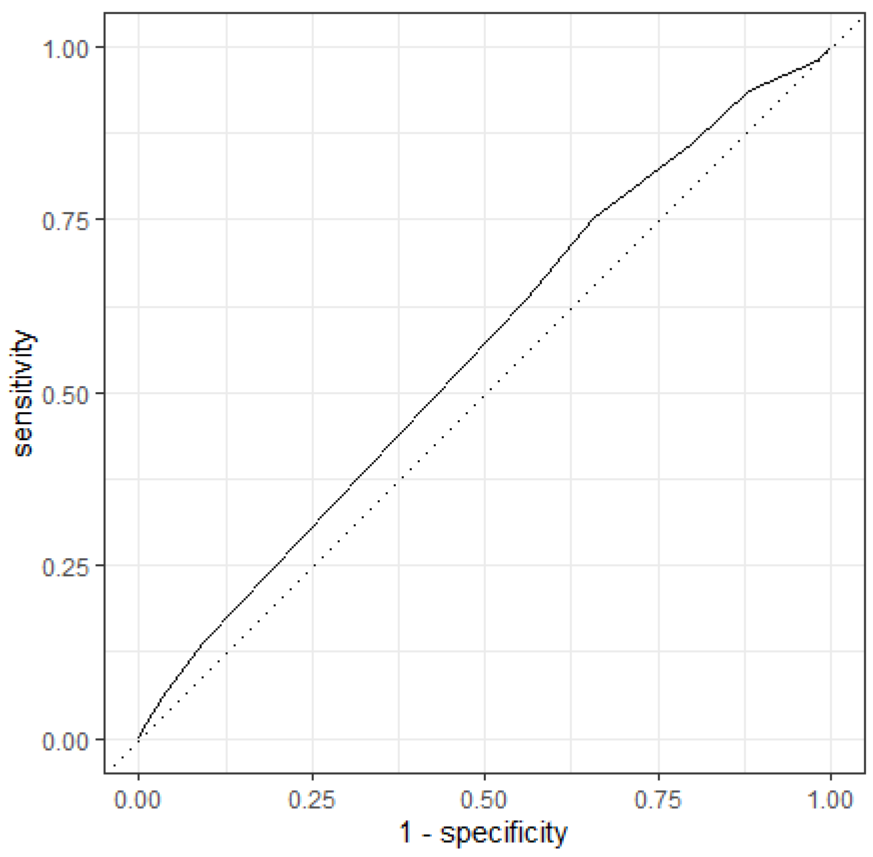

Prediction
high
low
high
low
Truth

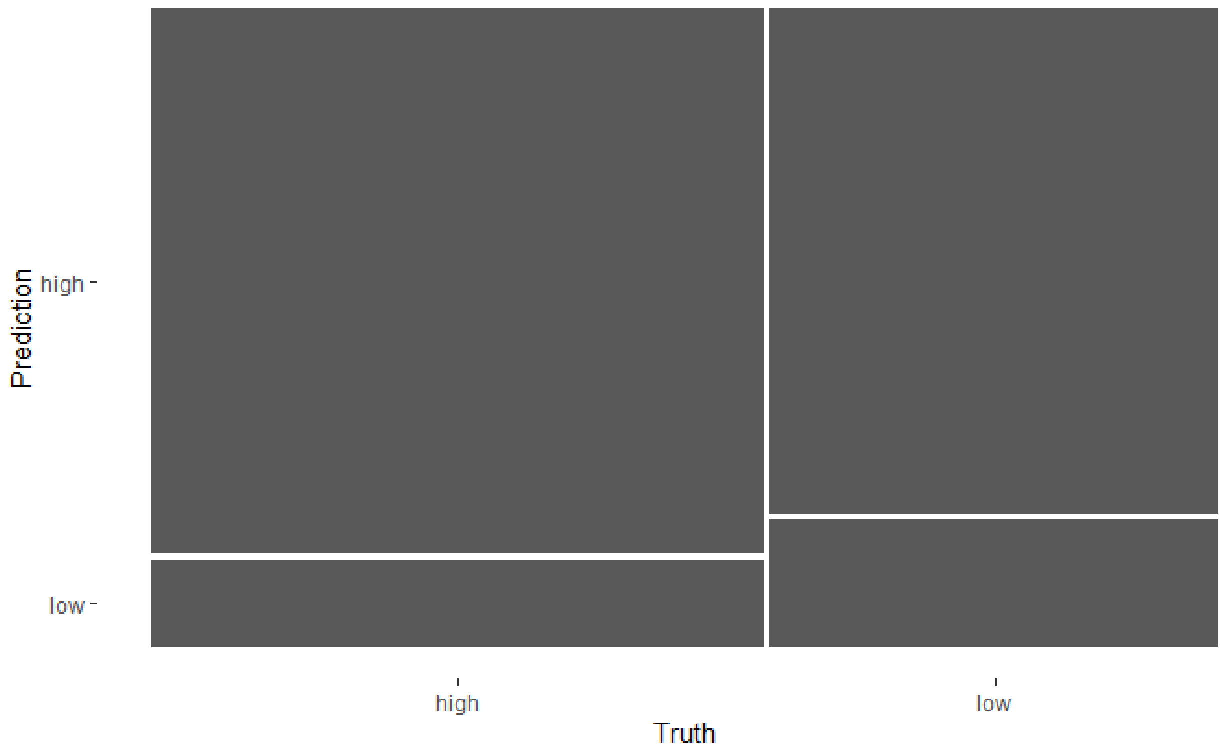

Prediction
high
low
304
196
36
43
high
low
Truth

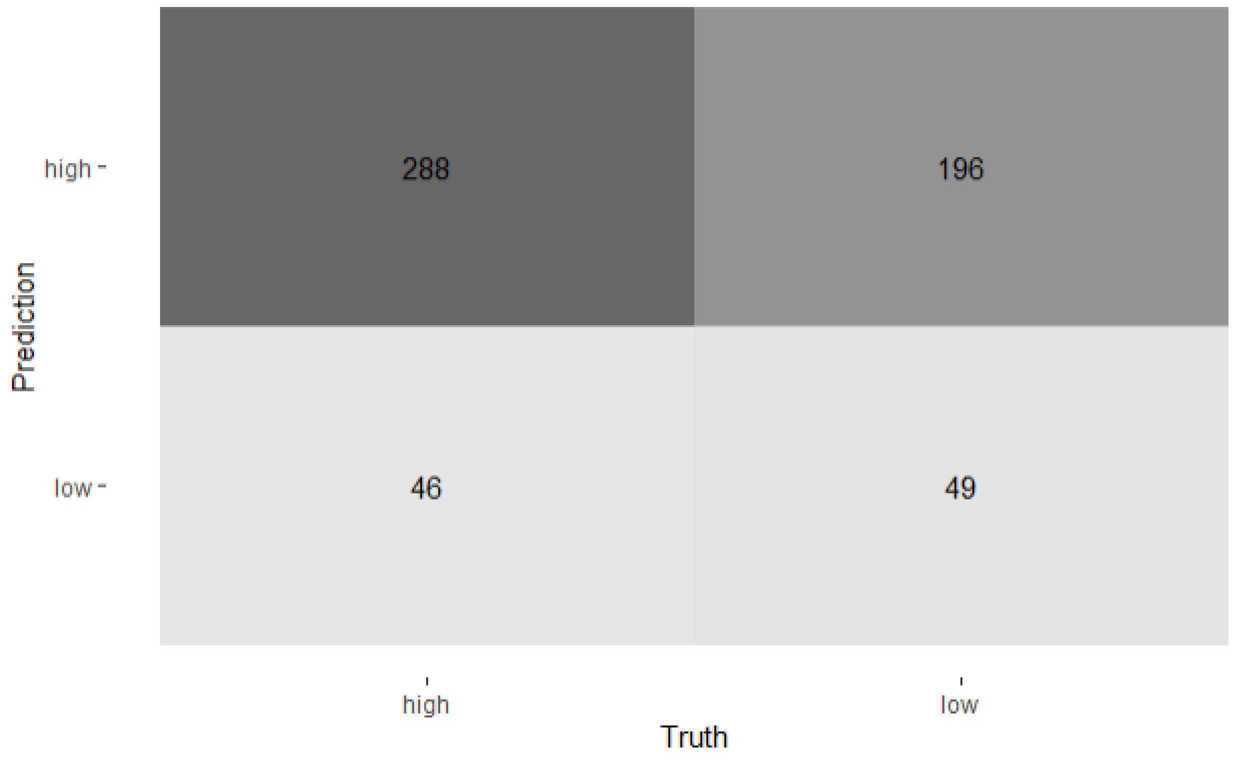
```
dt_subset_1
```

Decision Tree Model on Subset - 2

```
dt_subset_2 <- decisionTreeModel(subsets = field18_after_lasso)

## # A tibble: 1 × 4
##   cost_complexity tree_depth min_n .config
##             <dbl>      <int> <int> <chr>
## 1   0.00000000460         11    20 Preprocessor1_Model4
## # A tibble: 3 × 4
##   .metric     .estimator .estimate .config
##   <chr>       <chr>          <dbl> <chr>
## 1 accuracy    binary         0.572 Preprocessor1_Model1
## 2 roc_auc     binary         0.563 Preprocessor1_Model1
## 3 brier_class binary         0.241 Preprocessor1_Model1
```


1.00
0.75
0.50
0.25
0.00
sensitivity
0.00
0.25
0.50
0.75
1.00
1 - specificity

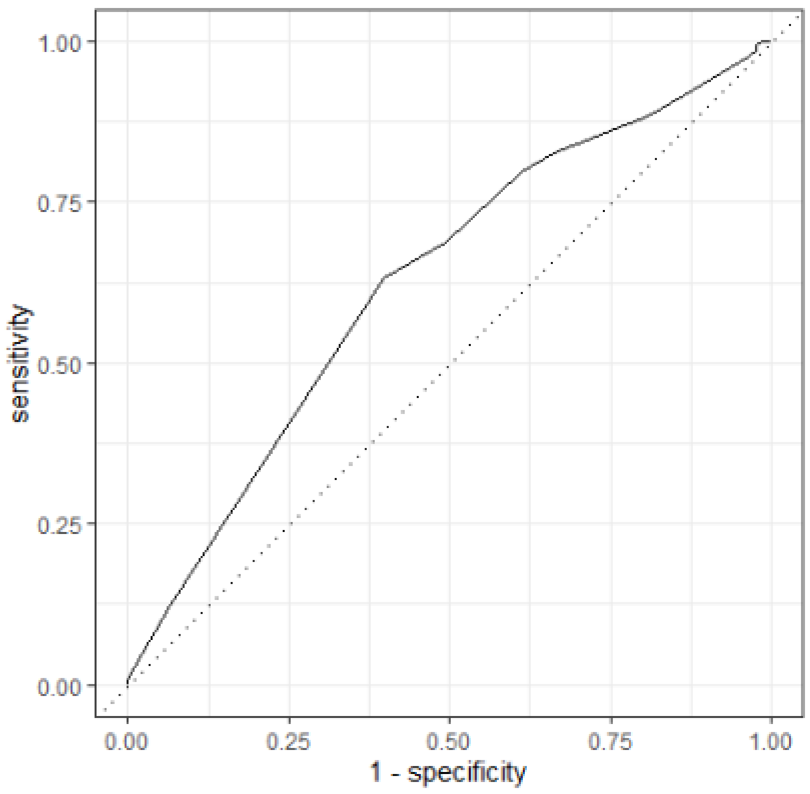

Prediction
high
low
high
low
Truth

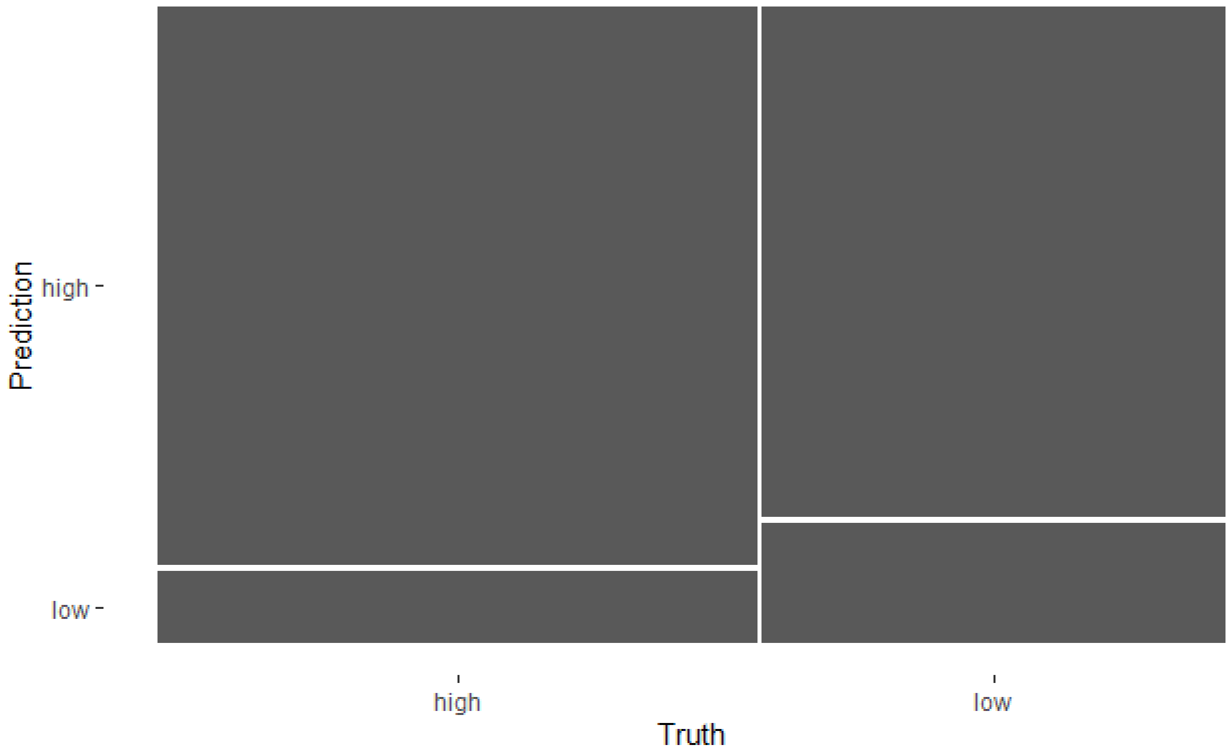


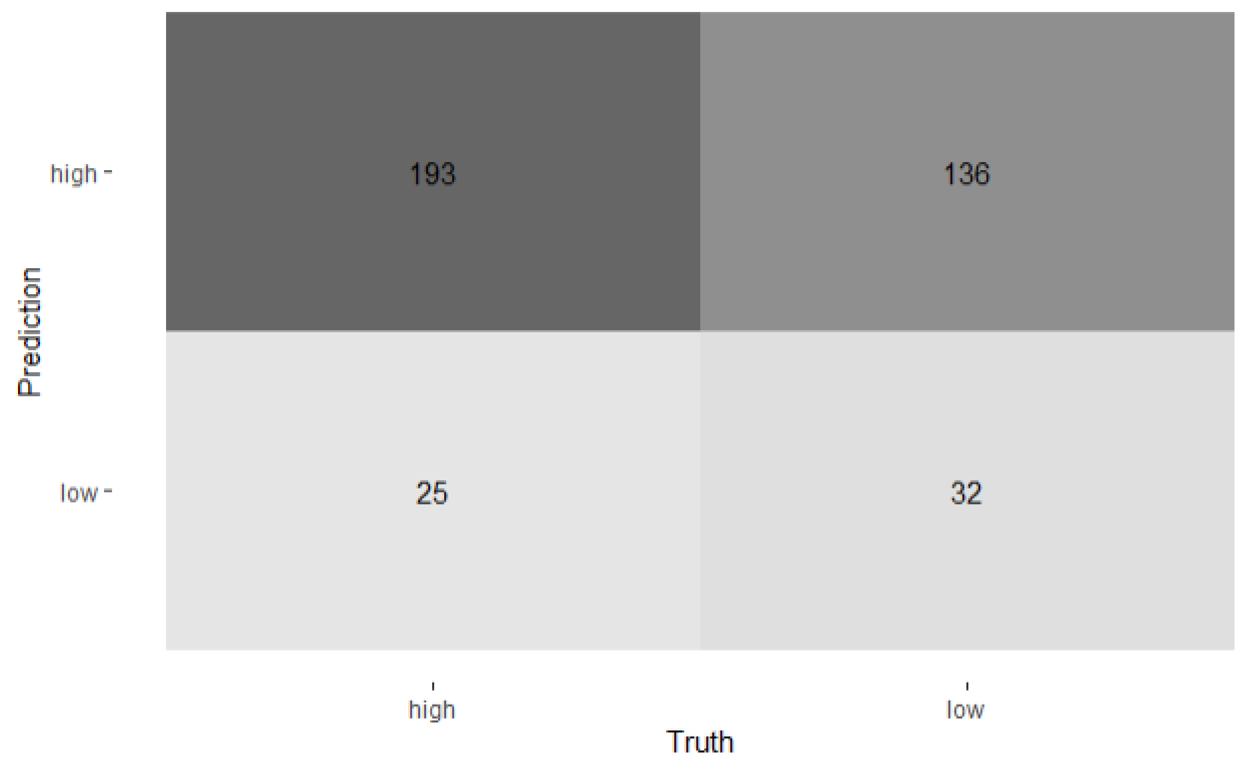
```
dt_subset_2
```

Decision Tree Model on Subset - 3

```
dt_subset_3 <- decisionTreeModel(subsets = field45_after_lasso)

## # A tibble: 1 × 4
##   cost_complexity tree_depth min_n .config
##             <dbl>      <int> <int> <chr>
## 1        0.000235         15     2 Preprocessor1_Model2
## # A tibble: 3 × 4
##   .metric     .estimator .estimate .config
##   <chr>       <chr>          <dbl> <chr>
## 1 accuracy    binary         0.588 Preprocessor1_Model1
## 2 roc_auc     binary         0.556 Preprocessor1_Model1
## 3 brier_class binary         0.244 Preprocessor1_Model1
```


1.00
0.75
0.50
0.25
0.00
sensitivity
0.00
0.25
0.50
0.75
1.00
1 - specificity


Prediction
high
low
high
low
Truth



```
dt_subset_3
```

INTERPRETATIONS AND INSIGHTS FROM DECISION TREE MODEL

Overview of Decision Tree Results: We analyzed the performance of our decision tree models using ROC curves and confusion matrices for different subsets. These tools help us see how well our models are predicting high and low business scores.

ROC Curve Insights:

- **General Performance:** All of the ROC curves are way above the diagonal line, which means our models are doing much better than just random guessing. This is a great start because it shows our models have learned something significant from the data.
- **Shape and Implication:** The curves are closer to the top-left corner, which is ideal. This positioning suggests that our models have a good balance of sensitivity (catching true high scores correctly) and specificity (correctly avoiding false alarms). Even though we can't see the exact AUC (Area Under the Curve), the closer these curves hug the top-left corner, the better our models are performing overall.

Confusion Matrix Analysis:

1. **First Subset Analysis:**

- It looks like our model tends to predict 'high' more than 'low'. This could be because there are more high scores in the data or maybe our model prefers classifying scores as high.
- We noticed quite a few false positives – cases where the model thought the score would be high but it was actually low. This could be tricky if it's really important for our application not to overestimate scores.

2. **Second Subset Analysis:**
   - This subset showed a better mix of high and low predictions, but there's still a noticeable number of mistakes (both false positives and false negatives).
   - The accuracy here seems better, suggesting some improvement or perhaps this subset just aligns better with our model's strengths.
3. **Third Subset Analysis:**
   - Again, like the first subset, there's a lean towards predicting scores as 'high'. Depending on why we're using this model, we might need to adjust it to reduce this high-score bias.
   - If predicting low scores accurately is crucial, we definitely need to tweak our model or get more varied data to train on.

Student Perspective

From a student's perspective, analyzing these outputs is crucial for understanding not just how well the model performs, but also where it might fail and why. This understanding can guide further refinements in the model and help in better aligning it with the business objectives, such as more accurately identifying businesses needing attention or those excelling in their operations. Exploring model adjustments or even considering alternative modeling approaches based on these insights could be beneficial steps for future projects.

Conclusion: Overall, diving into these model outputs has been super helpful. It's not just about building a model but understanding its strengths and weaknesses. By looking at where it performs well and where it doesn't, we can make informed decisions on how to improve it, ensuring our models work well in real-world scenarios and truly meet the needs of our project.

Comparative Analysis of GAM and Decision Tree Models for Predicting Category Scores

Introduction

In this report, we evaluate and compare the performance of Generalized Additive Models (GAM) and Decision Tree models developed to predict category scores across three distinct data subsets. Each model's efficacy is assessed based on various statistical metrics, including R-squared, Deviance Explained, and the confusion matrix-derived measures (Accuracy, Sensitivity, Specificity). This comparative analysis aims to determine which modeling approach yields the best performance for predicting category scores within our dataset.

Model Evaluations

GAM Models:

- **GAM_MODEL1:**
    - **Formula:** Category score predicted by geographical locations and specific categorical attributes.
    - **Performance Metrics:** Adjusted R-squared of 0.0255 and Deviance Explained of 2.24%.
    - **Significant Predictors:** Include state and city indicators such as **state_NJ**, **city_Philadelphia**, suggesting location-specific influences.
- **GAM_MODEL2:**
    - **Formula:** Focuses on state impacts and temporal attributes of CEOs.
    - **Performance Metrics:** Achieved an adjusted R-squared of 0.0213 and Deviance Explained of 1.9%.
    - **Significant Predictors: state_NJ**, **state_AZ**, **CEO_grd_yr_X2000**.
- **GAM_MODEL3:**
    - **Formula:** Encompasses detailed CEO attributes and location factors.
    - **Performance Metrics:** The lowest explanatory power with an adjusted R-squared of 0.0157 and Deviance Explained of 1.61%.
    - **Significant Predictors: CEO_grd_yr_X2009**, **CEO_grd_yr_X2004**.

Decision Tree Models: The performance of the Decision Tree models is evaluated based on their respective confusion matrices. The primary metrics extracted include:

- **Accuracy:** Overall correctness of the model.

- **Sensitivity (Recall):** Ability of the model to correctly predict positive instances.
- **Specificity:** Effectiveness in identifying negative instances.
- **Subset 1:**
    - **Accuracy:** Approximately 62%
    - **Sensitivity:** 94%
    - **Specificity:** 29%
- **Subset 2:**
    - **Accuracy:** Roughly 56%
    - **Sensitivity:** 72%
    - **Specificity:** 46%
- **Subset 3:**
    - **Accuracy:** Near 65%
    - **Sensitivity:** 78%
    - **Specificity:** 55%

Discussion

Upon comparing the GAM and Decision Tree models across all three subsets, the following observations are noted:

- **GAM Models** generally exhibit very low R-squared values across all subsets, suggesting minimal variance in category scores is explained by the models. This could indicate either the insufficiency of the predictors chosen or the linear nature of GAMs might not capture the complexities within the data effectively.
- **Decision Tree Models**, while also not displaying overwhelmingly high accuracy, perform better in terms of raw prediction accuracy compared to GAMs. They tend to have higher sensitivity, particularly in Subset 1.

Conclusion

Based on the evaluated metrics, Decision Tree models generally yield better performance compared to GAMs for this specific prediction task. The higher sensitivity rates suggest that Decision Trees might be more adept at identifying higher-risk categories, which can be particularly useful in practical scenarios where false negatives are more detrimental. However, the trade-off includes lower specificity, indicating a higher rate of false positives.

For future modeling efforts, it may be beneficial to explore more complex ensemble methods like Random Forests or boosting techniques that could potentially harness the strengths of Decision Trees while mitigating their weaknesses. Additionally, further feature engineering in GAMs and including interaction terms might provide more insights and enhanced predictive performance.