Syracuse University
School of Information Studies

IST 687 - Introduction to Data Science

Group 1

Hitesh Patil, Jennifer Fernando, Navya Kiran V B, Shrey Sheth, Siddharth Bose, Varshin Hariharan Bhaskaran

Project Report

Healthcare Cost Analysis

Index

Project Goal

Health Management Organizations (HMOs) are responsible for managing and providing healthcare services to a large number of individuals, often through insurance plans. These organizations need to carefully analyze healthcare cost data to identify trends, patterns, and potential areas for improvement in their healthcare services. The goal of this project is to assist an HMO in analyzing their healthcare cost dataset and providing actionable insights to help them optimize their services.

Project Approach

The approach we have used to achieve this goal is:

Data Cleaning:
The first step in the analysis was to clean the data by checking for missing values, outliers, and inconsistencies. The data was transformed and standardized as needed, and duplicates and other data quality issues were identified and handled.

Exploratory Data Analysis:
Next, we performed an exploratory data analysis by creating histograms, boxplots, and other visualizations of numeric variables. We explored relationships and correlations between variables and identified potential outliers and influential observations. The results of this analysis provided us with an understanding of the nature of the data and helped us identify potential drivers of healthcare costs.

Model Selection & Interpretation:
We then selected appropriate machine learning algorithms for the prediction task and evaluated and compared different models based on performance metrics such as accuracy and precision. We interpreted model coefficients, feature importance, and other model outputs to identify the most impactful variables for predicting healthcare costs.

Recommendations:
Based on our analysis, we summarized key findings and insights and provided recommendations for how the HMO can reduce healthcare costs. We highlighted the most impactful variables and interventions, such as identifying high-risk individuals who require more proactive medical interventions and implementing cost-saving measures to reduce healthcare costs.

Key Metrics used for identifying healthcare expenses:

1. Age
2. BMI
3. Smoker
4. Exercise

Project Scope and Context

Individual Basic Information	
Variable	**Description**
X	Unique Identifier
Age	Age of the person at the end of the year
Gender	Gender of the person
education_level	Level of education
married	Marital status of the person
num_children	Number of children

Individuals Geographical Information	
Variable	**Description**
location	US States
location_type	Urban or County

Individual Health Information	
Variable	**Description**
exercise	If the person exercises or not
smoker	If the person smokes or not

hypertension	If the person has hypertension or not
bmi	Body Mass Index of the person
yearly_physical	If the person visits a medical professional during the year or not
cost	Healthcare cost of the person

Business Questions

1.Forecast who will spend a lot of money on health care in the coming year (i.e., who will have high healthcare costs).
2. Provide actionable insight to the HMO on how to reduce total health care expenses by making specific recommendations on how to reduce health care costs.

Code

Installing packages for various functions like data cleaning, visualization and prediction models

```
### Data Cleaning and Manipulation Packages
library(tidyverse)
#install.packages('tseries')
library(tseries)
#install.packages('imputeTS')
library(imputeTS)

### Data Visualization Packages
library(ggplot2)
#install.packages("ggmap", dependencies=TRUE)
library(ggmap)
#install.packages("maps",dependencies=TRUE)
library(maps)
#install.packages("mapproj", dependencies = TRUE)
library(mapproj)
#install.packages("usmap",dependencies = TRUE)
library(usmap)

### Prediction Models Packages
#install.packages("caret",dependencies=TRUE)
library(caret)
#install.packages("MASS",dependencies=TRUE)
library(MASS)
#install.packages("kernlab",dependencies=TRUE)
library(kernlab)
#install.packages("glm2",dependencies=TRUE)
library(glm2)
```

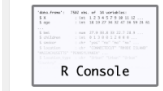
Importing Dataset

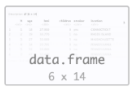
```
# Read the data from a CSV file and store it in the 'data' variable
data <- read.csv("https://intro-datascience.s3.us-east-2.amazonaws.com/HMO_data.csv", stringsAsFactors = FALSE)
```

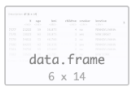
Structure of the Dataset

The dataset consists of 7582 observations of 14 variables

```
# Display the structure of the 'data' data frame
str(data)
# Show the first few rows of the 'data' data frame
head(data)
# Show the last few rows of the 'data' data frame
tail(data)
```








```
'data.frame':   7582 obs. of  14 variables:
 $ X               : int  1 2 3 4 5 7 9 10 11 12 ...
 $ age             : int  18 19 27 34 32 47 36 59 24 61 ...
 $ bmi             : num  27.9 33.8 33 22.7 28.9 ...
 $ children        : int  0 1 3 0 0 1 2 0 0 0 ...
 $ smoker          : chr  "yes" "no" "no" "no" ...
 $ location        : chr  "CONNECTICUT" "RHODE ISLAND" "MASSACHUSETTS" "PENNSYLVANIA" ...
 $ location_type   : chr  "Urban" "Urban" "Urban" "Country" ...
 $ education_level: chr  "Bachelor" "Bachelor" "Master" "Master" ...
 $ yearly_physical: chr  "No" "No" "No" "No" ...
 $ exercise        : chr  "Active" "Not-Active" "Active" "Not-Active" ...
 $ married         : chr  "Married" "Married" "Married" "Married" ...
 $ hypertension    : int  0 0 0 1 0 0 0 1 0 0 ...
 $ gender          : chr  "female" "male" "male" "male" ...
 $ cost            : int  1746 602 576 5562 836 3842 1304 9724 201 4492 ...
```

Head of the dataset

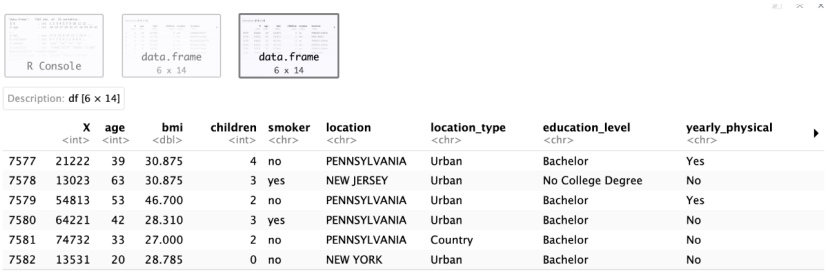
First 6 rows of the dataset

Description: df [6 × 14]

	X <int>	age <int>	bmi <dbl>	children <int>	smoker <chr>	location <chr>	location_type <chr>	education_level <chr>	yearly_physical <chr>
1	1	18	27.900	0	yes	CONNECTICUT	Urban	Bachelor	No
2	2	19	33.770	1	no	RHODE ISLAND	Urban	Bachelor	No
3	3	27	33.000	3	no	MASSACHUSETTS	Urban	Master	No
4	4	34	22.705	0	no	PENNSYLVANIA	Country	Master	No
5	5	32	28.880	0	no	PENNSYLVANIA	Country	PhD	No
6	7	47	33.440	1	no	PENNSYLVANIA	Urban	Bachelor	No

Tail of the dataset

Last 6 rows of the dataset



R Console | data.frame 6 x 14 | data.frame 6 x 14

Description: df [6 × 14]

	X <int>	age <int>	bmi <dbl>	children <int>	smoker <chr>	location <chr>	location_type <chr>	education_level <chr>	yearly_physical <chr>
7577	21222	39	30.875	4	no	PENNSYLVANIA	Urban	Bachelor	Yes
7578	13023	63	30.875	3	yes	NEW JERSEY	Urban	No College Degree	No
7579	54813	53	46.700	2	no	PENNSYLVANIA	Urban	Bachelor	Yes
7580	64221	42	28.310	3	yes	PENNSYLVANIA	Urban	Bachelor	No
7581	74732	33	27.000	2	no	PENNSYLVANIA	Country	Bachelor	No
7582	13531	20	28.785	0	no	NEW YORK	Urban	Bachelor	No

Converting the 'bmi' and 'hypertension' columns in the dataset to numeric data type.

```r
```{r}
# Converting 'bmi' and 'hypertension' columns to numeric data type in the 'data' data frame
# Convert the 'bmi' column to numeric data type
data$bmi <- as.numeric(data$bmi)
# Convert the 'hypertension' column to numeric data type by replacing '1' with 1 and other values with 0 using ifelse function
data$hypertension <- ifelse(data$hypertension == '1', 1, 0)
```
```

Removing NA values from the dataset

```r
```{r}
# Removing NAs from the 'data' data frame

# Count the number of NAs in each column of the 'data' data frame using colSums
colSums(is.na(data))

# Interpolate missing values in the 'bmi' column using na_interpolation function
data$bmi <- na_interpolation(data$bmi)

# Interpolate missing values in the 'hypertension' column using na_interpolation function
data$hypertension <- na_interpolation(data$hypertension)

# Interpolate missing values in the 'cost' column using na_interpolation function
data$cost <- na_interpolation(data$cost)

# Display the structure of the updated 'data' data frame
str(data)

```
```

Updated dataset after interpolation with the count of NA values in each variable

```
             X            age            bmi       children         smoker       location  location_type
             0              0             78              0              0              0              0
education_level yearly_physical       exercise        married   hypertension         gender           cost
             0              0              0              0             80              0              0
'data.frame':   7582 obs. of  14 variables:
 $ X              : int  1 2 3 4 5 7 9 10 11 12 ...
 $ age            : int  18 19 27 34 32 47 36 59 24 61 ...
 $ bmi            : num  27.9 33.8 33 22.7 28.9 ...
 $ children       : int  0 1 3 0 0 1 2 0 0 0 ...
 $ smoker         : chr  "yes" "no" "no" "no" ...
 $ location       : chr  "CONNECTICUT" "RHODE ISLAND" "MASSACHUSETTS" "PENNSYLVANIA" ...
 $ location_type  : chr  "Urban" "Urban" "Urban" "Country" ...
 $ education_level: chr  "Bachelor" "Bachelor" "Master" "Master" ...
 $ yearly_physical: chr  "No" "No" "No" "No" ...
 $ exercise       : chr  "Active" "Not-Active" "Active" "Not-Active" ...
 $ married        : chr  "Married" "Married" "Married" "Married" ...
 $ hypertension   : num  0 0 0 1 0 0 0 1 0 0 ...
 $ gender         : chr  "female" "male" "male" "male" ...
 $ cost           : int  1746 602 576 5562 836 3842 1304 9724 201 4492 ...
```

Calculating the mean cost of the 'cost' column in the dataset and using it as a threshold to identify expensive and non-expensive cases.

```r
# Using mean as a threshold for expensiveness in the 'data' data frame
# Calculate the mean cost of the 'cost' column
meanCost <- mean(data$cost)
# Count the number of rows with costs below the mean cost
no_below <- nrow(data[data$cost < meanCost,])
print("Values less than the mean cost")
print(no_below)
# Count the number of rows with costs above the mean cost
no_above <- nrow(data[data$cost > meanCost,])
print("Values greater than the mean cost")
print(no_above)
meanCost
```

```
[1] "Values less than the mean cost"
[1] 5222
[1] "Values greater than the mean cost"
[1] 2360
[1] 4042.961
```

By using the mean cost as a threshold, we were able to identify cases that are more expensive than the mean cost and cases that are less expensive than the mean cost.

The mean cost we calculated was 4042.961. Upon running the above snippet we found that 5222 values were less than the mean cost and 2360 were greater than the mean cost.

Calculating the median cost of the 'cost' column in the dataset and using it as a threshold to identify expensive and non-expensive cases.

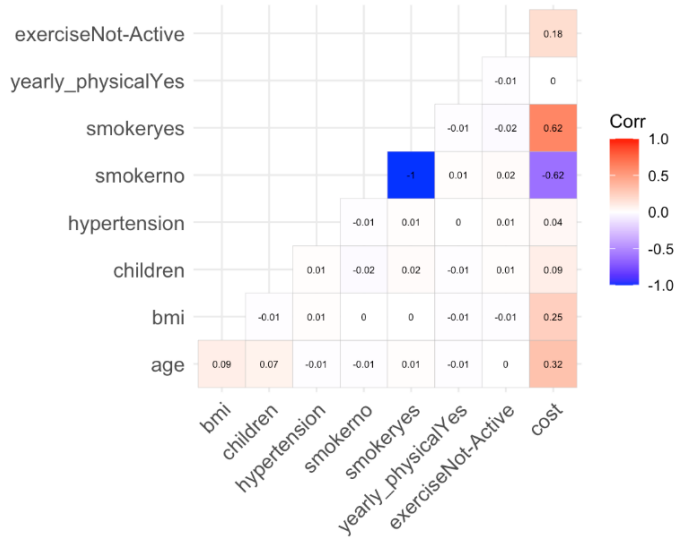
```r
# Using median as a threshold for expensiveness in the 'data' data frame
# Count the number of rows with costs less than or equal to the median cost (2500)
print("Cost values less than median cost")
n1 <- nrow(data[data$cost <= 2500,])
print(n1)
# Count the number of rows with costs greater than the median cost (2500)
print("Cost values greater than median cost")
n2 <- nrow(data[data$cost > 2500,])
print(n2)
```

```
[1] "Cost values less than median cost"
[1] 3792
[1] "Cost values greater than median cost"
[1] 3790
```

By using the median cost as a threshold, we were able to identify cases that are more expensive than the median cost and cases that are less expensive than or equal to the median cost.
The median cost we got was 2500. Upon running the above snippet we found that 3792 values were less than the mean cost and 3790 were greater than the mean cost.

Creating a correlation matrix for a subset of variables in the dataset. The variables included in the matrix are age, bmi, children, hypertension, smoker, yearly_physical, exercise, and cost.

```
# Install the ggcorrplot package.
#install.packages("ggcorrplot")

# Load the ggcorrplot package.
library(ggcorrplot)

# Create a vector of variables to include in the correlation matrix.
cols <- c("age", "bmi", "children", "hypertension","smoker","yearly_physical","exercise", "cost")

# Create a model matrix with the variables in `cols`.
model.matrix(~0+., data=data[,cols])

# Compute the correlation matrix of the model matrix.
cor(model.matrix(~0+., data=data[,cols]))

# Plot the correlation matrix using ggcorrplot.
ggcorrplot(cor(model.matrix(~0+., data=data[,cols])),
           show.diag=FALSE,
           type="lower",
           lab=TRUE,
           lab_size=2)
#The highly correlated variables with cost are smoker, age, bmi, exercise.
```



The plot highlights the highly correlated variables with cost, which are smoker, age, bmi, and exercise.

Adding a new column ‘exp’ which categorizes each row as expensive or not expensive based on median cost.

```r
# Adding a column 'exp' to the 'data' data frame for categorizing as 'Expensive' or 'Not Expensive' based on cost

# Create a new column 'exp' and assign 'Not Expensive' if cost is less than or equal to 2500, otherwise 'Expensive'
data$exp <- ifelse(data$cost <= 2500, 'Not Expensive', 'Expensive')

# Show the first few rows of the updated 'data' data frame
head(data)
```

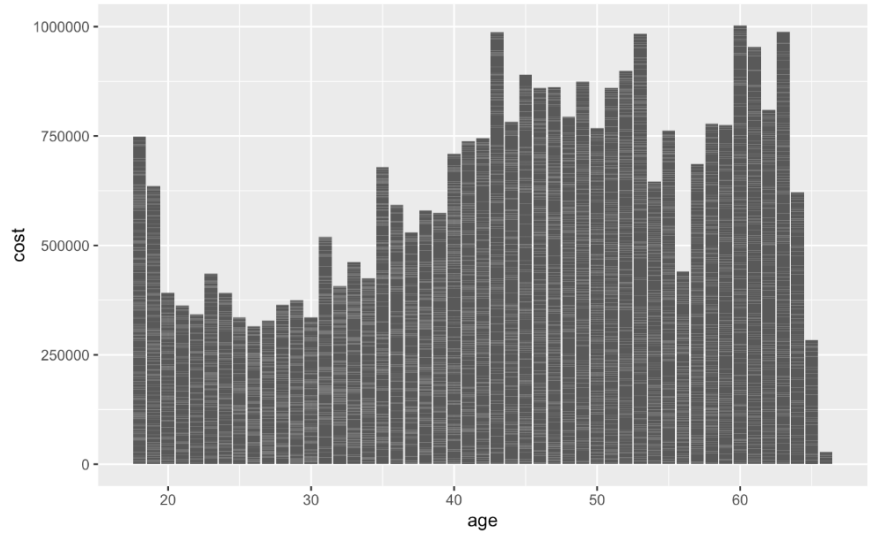
Description: df [6 × 15]

location_type <chr>	education_level <chr>	yearly_physical <chr>	exercise <chr>	married <chr>	hypertension <dbl>	gender <chr>	cost <int>	exp <chr>
Urban	Bachelor	No	Active	Married	0	female	1746	Not Expensive
Urban	Bachelor	No	Not-Active	Married	0	male	602	Not Expensive
Urban	Master	No	Active	Married	0	male	576	Not Expensive
Country	Master	No	Not-Active	Married	1	male	5562	Expensive
Country	PhD	No	Not-Active	Married	0	male	836	Not Expensive
Urban	Bachelor	No	Not-Active	Married	0	female	3842	Expensive

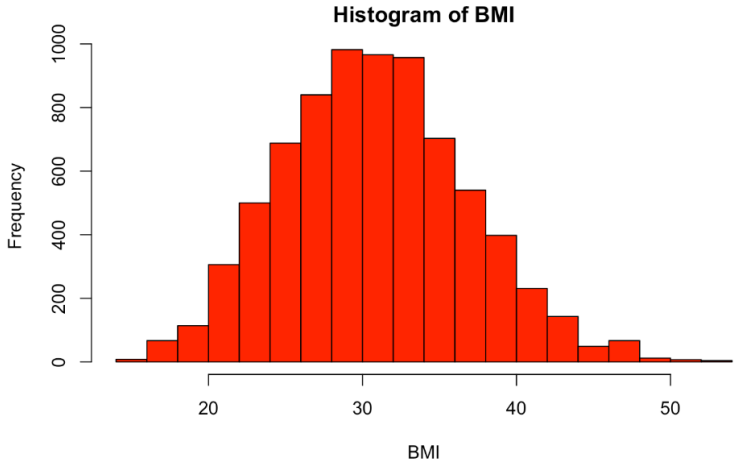
Plotting graphs with the data till now to find some relation between variables

```r
#age to cost relation. Not conclusive
ggplot(data,aes(x=age, y=cost)) +geom_bar(stat="identity")

#
hist(data$bmi, col = "red", main = "Histogram of BMI", xlab = "BMI", ylab = "Frequency")

#
hist(data$cost, breaks = 25, col = "red", main = "Cost Histogram", xlab = "Cost", ylab = "Frequency")


#age to bmi scatter plot
ggplot(data)+geom_point(aes(x=age ,y=cost ,color=bmi))+
ylab('cost')+xlab('age')+ggtitle("")


#
ggplot(data)+geom_point(aes(x=age ,y=cost ,color=smoker))+
ylab('cost')+xlab('age')+ggtitle("")


#
ggplot(data)+geom_point(aes(x=age ,y=cost ,color=exercise))+
ylab('cost')+xlab('age')+ggtitle("")


#
box_plot1 <- ggplot(data, aes(x = smoker, y = cost)) + geom_boxplot()
box_plot1
```

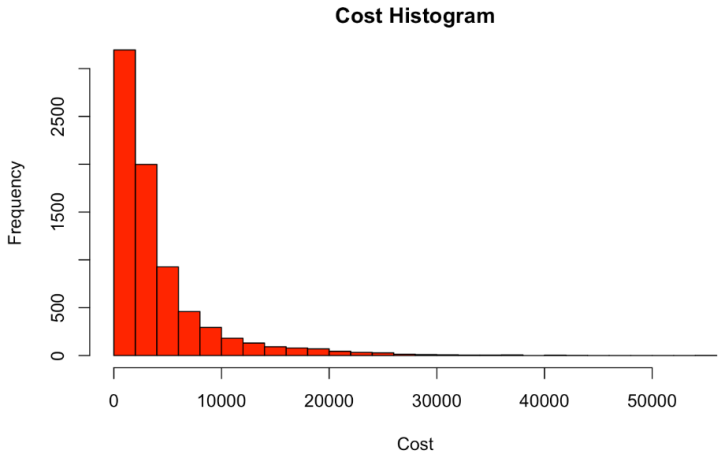
Age to cost relation (inconclusive)



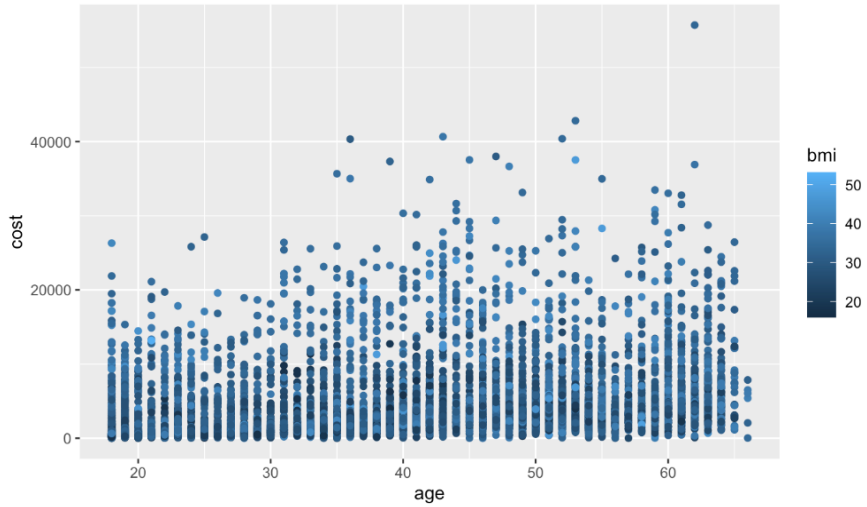
Histogram of BMI



Histogram of Cost

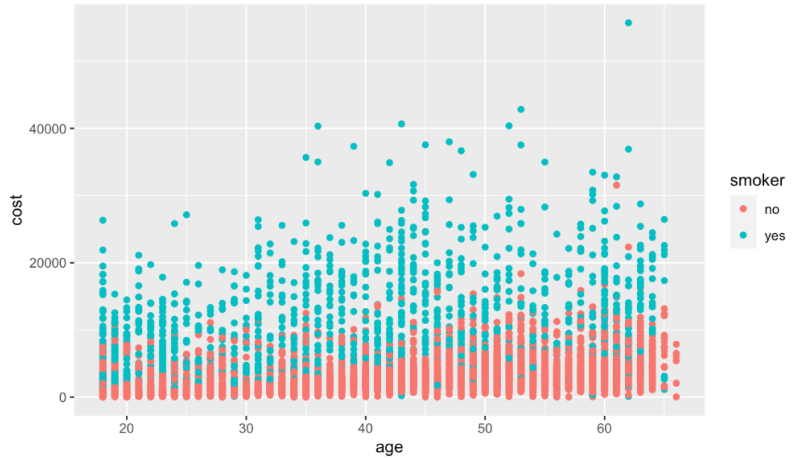


Scatterplot of Age to BMI



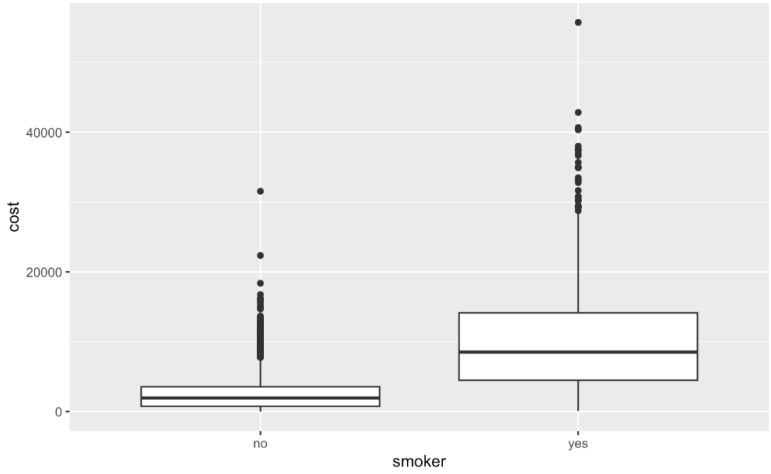
An active person has far lower healthcare costs than inactive people

Scatterplot of Age to Smoker



Scatterplot of Age to Exercise

Box plot for smokers and non-smokers



Identifying Significant Predictors for Healthcare Costs by Location

```r
```{r}
library(readr)
library(dplyr)
library(caret)
library(car)

# Load the dataset
data <- read.csv("https://intro-datascience.s3.us-east-2.amazonaws.com/HMO_data.csv")

# Preprocessing
categorical_vars <- c("location", "exercise", "smoker", "yearly_physical", "gender", "education_level", "married")
encoded_data <- model.matrix(~., data = data[, categorical_vars])

encoded_data <- encoded_data[, -1] # Remove the intercept column
numeric_cols <- c("age", "bmi", "hypertension")

ss <- preProcess(as.data.frame(data[numeric_cols]), method=c("range"))
data[numeric_cols] <- predict(ss, as.data.frame(data[numeric_cols]))

processed_data <- cbind(data[, c("location", numeric_cols, "cost")], encoded_data)
colnames(processed_data) <- gsub(" ", "_", colnames(processed_data))

# Separate the dataset based on the 'location' column
data_by_location <- split(processed_data, processed_data$location)

# Function to fit a linear regression model for a specific location and extract significant predictors
significant_predictors_by_location <- function(location_data) {
  lm_model <- lm(cost ~ ., data = location_data[, -1])
  summary_model <- summary(lm_model)
  p_values <- summary_model$coefficients[, "Pr(>|t|)"]
  significant_predictors <- names(p_values)[p_values < 0.05]
  return(significant_predictors)
}

# Calculate significant predictors for each location
result <- lapply(data_by_location, significant_predictors_by_location)

# Display the result
result
```
```

The above code loads the dataset and performs some preprocessing steps. Then, it splits the dataset based on the 'location' column and defines a function to fit a linear regression model and extract significant predictors for each location.

```
$CONNECTICUT
[1] "(Intercept)"          "age"                  "bmi"                  "hypertension"         "`exerciseNot-Active`"
[6] "smokeryes"

$MARYLAND
[1] "(Intercept)"          "age"                  "bmi"                  "`exerciseNot-Active`" "smokeryes"
[6] "education_levelPhD"

$MASSACHUSETTS
[1] "(Intercept)"          "age"                  "bmi"                  "`exerciseNot-Active`" "smokeryes"
[6] "gendermale"           "education_levelPhD"

$`NEW JERSEY`
[1] "(Intercept)"          "age"                  "bmi"                  "`exerciseNot-Active`" "smokeryes"

$`NEW YORK`
[1] "(Intercept)"          "age"                  "bmi"                  "`exerciseNot-Active`" "smokeryes"

$PENNSYLVANIA
[1] "(Intercept)"          "age"                  "bmi"                  "`exerciseNot-Active`" "smokeryes"

$`RHODE ISLAND`
[1] "(Intercept)"          "age"                  "bmi"                  "hypertension"         "`exerciseNot-Active`"
[6] "smokeryes"
```

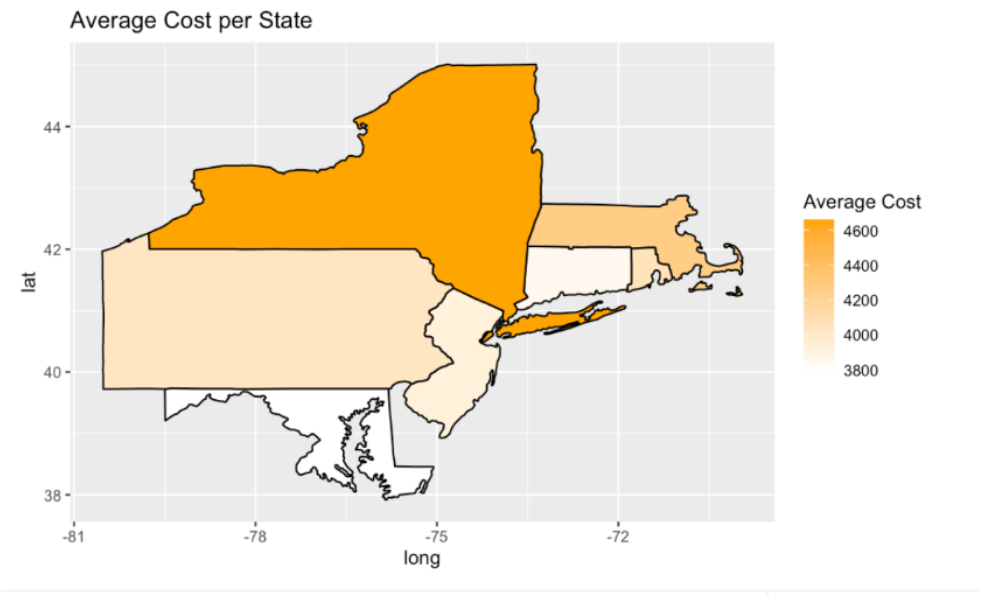
Finally, it applies the function to each location's data and returns a list of significant predictors for each location.

Plotting a map showing the average cost per state in the US

```r
# Convert the state names to lowercase
data$location <- tolower(data$location)

# Aggregate the average cost per state
state_avg_cost <- data %>%
  group_by(location) %>%
  summarise(avg_cost = mean(cost))
states_of_interest <- c("connecticut", "maryland", "massachusetts", "new jersey", "new york", "pennsylvania", "rhode island")
# Get the map data for the US
map_data <- map_data("state",regions=states_of_interest )

# Merge the map data with the average cost data
map_data_avg_cost <- merge(map_data, state_avg_cost, by.x = "region", by.y = "location")

# Create the plot
ggplot(map_data_avg_cost, aes(x = long, y = lat, group = group, fill = avg_cost)) +
  geom_polygon(color = "black") +
  scale_fill_gradient(low = "White", high = "orange") +
  labs(title = "Average Cost per State", fill = "Average Cost")
  #coord_fixed()+
 # theme_void()
View(state_avg_cost)
```

The code is aggregating the average cost per state from a given dataset and plotting the results on a US map using the ggplot2 package. The state names are converted to lowercase and merged with map data for the desired states.

The resulting plot shows a heatmap of the average cost per state.



The resulting plot shows each state filled with a color representing the average cost of HMO coverage in that state. States with higher average costs are shaded in darker orange, while states with lower average costs are shaded in lighter orange.

The average cost per state

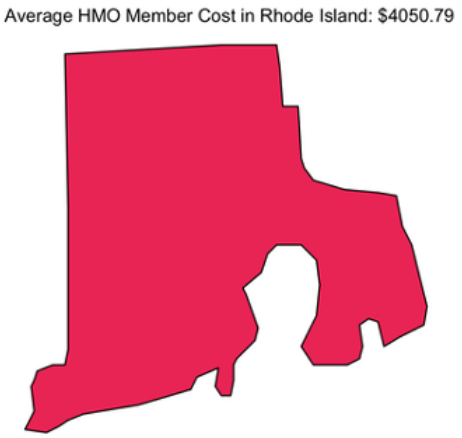
	location	avg_cost
1	connecticut	3847.519
2	maryland	3784.174
3	massachusetts	4267.540
4	new jersey	3930.564
5	new york	4661.506
6	pennsylvania	4023.115
7	rhode island	4050.791

Mapping Mean cost based on Location

1. **Rhode Island**

```r
#Rhode Island

# Load required packages
library(dplyr)
library(maps)
library(ggplot2)

# Load the data
data <- read.csv("https://intro-datascience.s3.us-east-2.amazonaws.com/HMO_data.csv")

# Convert the state names to lowercase
data$location <- tolower(data$location)

# Filter the data for Rhode Island
ri_data <- data %>%
  filter(location == "rhode island")

# Calculate the average HMO member cost for Rhode Island
ri_avg_cost <- mean(ri_data$cost)

# Get the map data for Rhode Island using the 'maps' package
map_data <- map_data("state", region = "rhode island")

# Create the plot using ggplot2 package
ggplot(map_data, aes(x = long, y = lat, group = group)) + # Define aesthetics for the plot
  geom_polygon(fill = ri_avg_cost, color = "black") + # Draw polygons for Rhode Island and set the border color to black
  ggtitle(paste0("Average HMO Member Cost in Rhode Island: $", round(ri_avg_cost, 2))) + # Add title for the plot
  theme_void() + # Remove axes and gridlines for a clean map appearance
  coord_fixed() # Fix the aspect ratio of the plot to keep the correct shape of Rhode Island map
```

The above code focuses on Rhode Island by filtering the data and calculating the average cost per member for Rhode Island. It then uses the ggplot2 package to draw a polygon map of Rhode Island and fill it with the calculated average cost per member. The theme_void() function is used to remove axes and gridlines, and coord_fixed() fixes the aspect ratio of the plot.

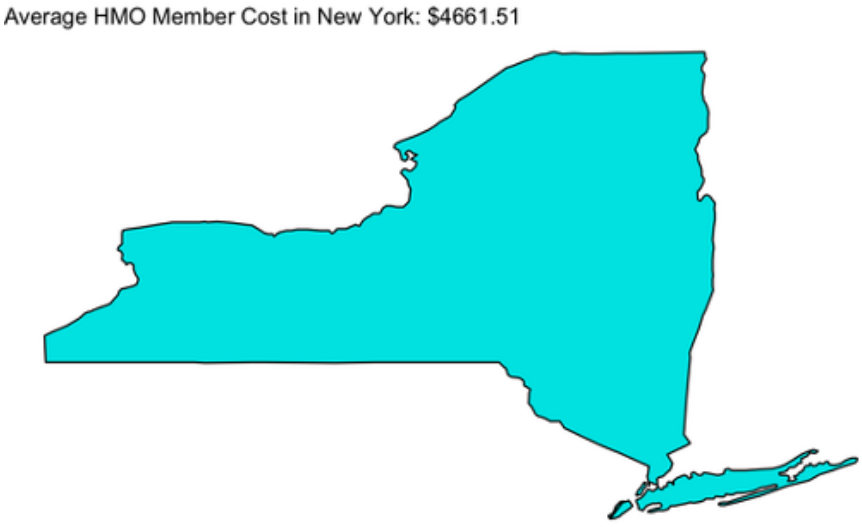


Plot for the calculated average cost per member in Rhode Island.

2. New York

```r
#NY

# Filter the data for new york
ny_data <- data %>%
  filter(location == "new york")

# Calculate the average HMO member cost for New York
ny_avg_cost <- mean(ny_data$cost)

# Get the map data for New York using the 'maps' package
map_data <- map_data("state", region = "new york")

# Create the plot using ggplot2 package
ggplot(map_data, aes(x = long, y = lat, group = group)) + # Define aesthetics for the plot
  geom_polygon(fill = ny_avg_cost, color = "black") + # Draw polygons for New York and set the border color to black
  ggtitle(paste0("Average HMO Member Cost in New York: $", round(ny_avg_cost, 2))) + # Add title for the plot
  theme_void() + # Remove axes and gridlines for a clean map appearance
  coord_fixed() # Fix the aspect ratio of the plot to keep the correct shape of New York map
```

The above code focuses on New York

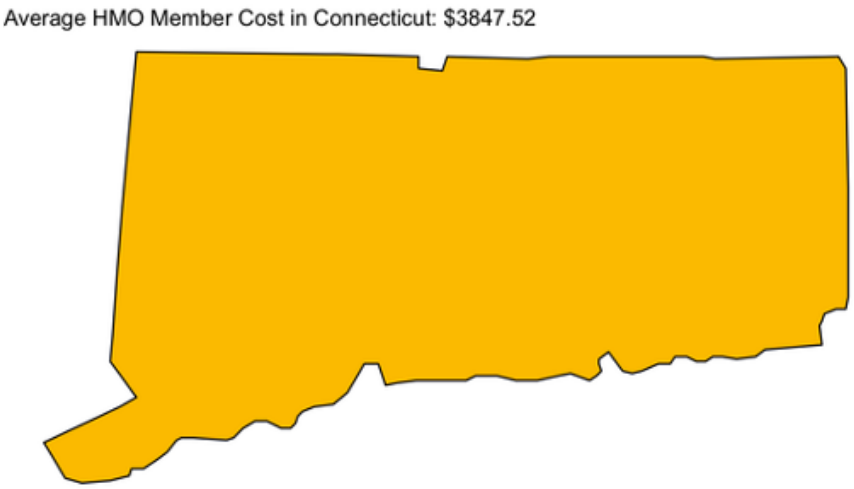


Plot for the calculated average cost per member in New York

3. Connecticut

```r
# connecticut

# Filter the data for connecticut
cc_data <- data %>%
  filter(location == "connecticut")

# Calculate the average HMO member cost for Connecticut
cc_avg_cost <- mean(cc_data$cost)

# Get the map data for connecticut using the 'maps' package
map_data <- map_data("state", region = "connecticut")

# Create the plot using ggplot2 package
ggplot(map_data, aes(x = long, y = lat, group = group)) + # Define aesthetics for the plot
  geom_polygon(fill = cc_avg_cost, color = "black") + # Draw polygons for connecticut and set the border color to black
  ggtitle(paste0("Average HMO Member Cost in Connecticut: $", round(cc_avg_cost, 2))) + # Add title for the plot
  theme_void() + # Remove axes and gridlines for a clean map appearance
  coord_fixed() # Fix the aspect ratio of the plot to keep the correct shape of connecticut map

# Maryland
```

The above code focuses on Connecticut

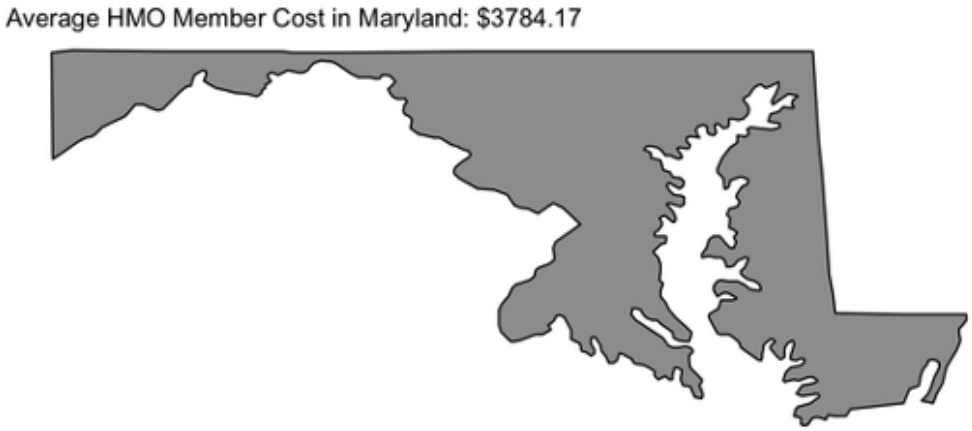


Plot for the calculated average cost per member in Connecticut

4. Maryland

```r
# Maryland

m_data <- data %>%
  filter(location == "maryland")

# Calculate the average HMO member cost for maryland
m_avg_cost <- mean(m_data$cost)

# Get the map data for Maryland using the 'maps' package
map_data <- map_data("state", region = "maryland")

# Create the plot using ggplot2 package
ggplot(map_data, aes(x = long, y = lat, group = group)) + # Define aesthetics for the plot
  geom_polygon(fill = m_avg_cost, color = "black") + # Draw polygons for maryland and set the border color to black
  ggtitle(paste0("Average HMO Member Cost in Maryland: $", round(m_avg_cost, 2))) + # Add title for the plot
  theme_void() + # Remove axes and gridlines for a clean map appearance
  coord_fixed() # Fix the aspect ratio of the plot to keep the correct shape of maryland map
```

The above code focuses on Maryland

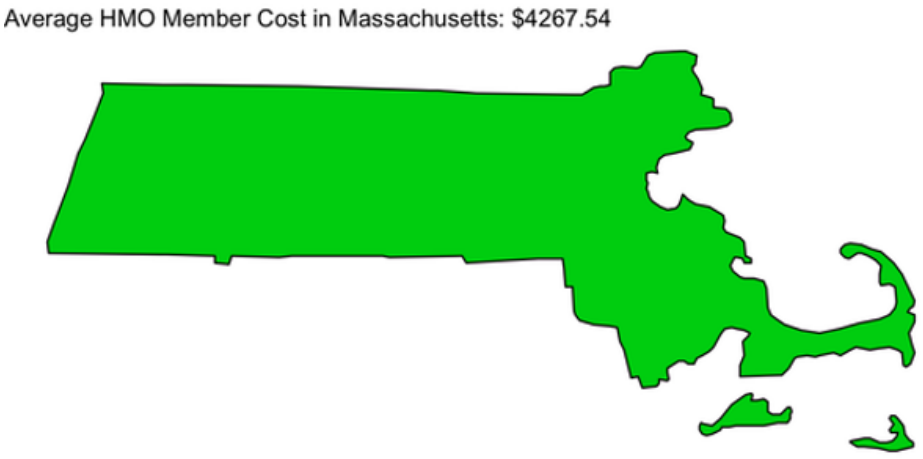


Plot for the calculated average cost per member in Maryland

5. Massachusetts

```r
#Massachusetts

ms_data <- data %>%
  filter(location == "massachusetts")

# Calculate the average HMO member cost for massachusetts
ms_avg_cost <- mean(ms_data$cost)

# Get the map data for Rhode Island using the 'maps' package
map_data <- map_data("state", region = "massachusetts")

# Create the plot using ggplot2 package
ggplot(map_data, aes(x = long, y = lat, group = group)) + # Define aesthetics for the plot
  geom_polygon(fill = ms_avg_cost, color = "black") + # Draw polygons for maryland and set the border color to black
  ggtitle(paste0("Average HMO Member Cost in Massachusetts: $", round(ms_avg_cost, 2))) + # Add title for the plot
  theme_void() + # Remove axes and gridlines for a clean map appearance
  coord_fixed() # Fix the aspect ratio of the plot to keep the correct shape of massachusetts map
```

The above code focuses on Massachusetts

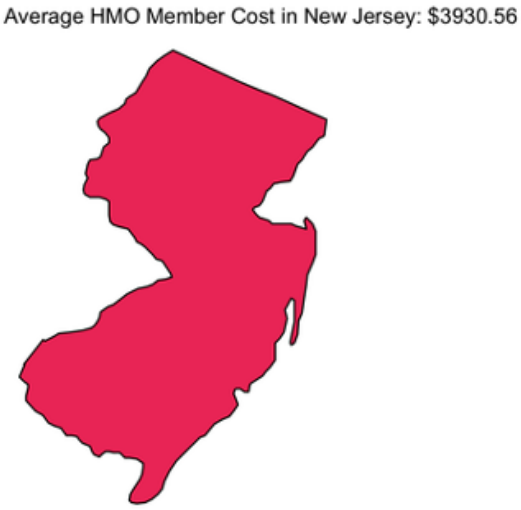


Plot for the calculated average cost per member in Massachusetts

6. New Jersey

```r
#New Jersey

nj_data <- data %>%
  filter(location == "new jersey")

# Calculate the average HMO member cost for Connecticut
nj_avg_cost <- mean(nj_data$cost)

# Get the map data for Rhode Island using the 'maps' package
map_data <- map_data("state", region = "new jersey")

# Create the plot using ggplot2 package
ggplot(map_data, aes(x = long, y = lat, group = group)) + # Define aesthetics for the plot
  geom_polygon(fill = nj_avg_cost, color = "black") + # Draw polygons for New Jersey and set the border color to black
  ggtitle(paste0("Average HMO Member Cost in New Jersey: $", round(nj_avg_cost, 2))) + # Add title for the plot
  theme_void() + # Remove axes and gridlines for a clean map appearance
  coord_fixed() # Fix the aspect ratio of the plot to keep the correct shape of massachusetts map
```

The above code focuses on New Jersey



Plot for the calculated average cost per member in New Jersey

7. Pennsylvania

```r
# Pennsylvania

pn_data <- data %>%
  filter(location == "pennsylvania")

# Calculate the average HMO member cost for Pennsylvania
pn_avg_cost <- mean(pn_data$cost)

# Get the map data for Rhode Island using the 'maps' package
map_data <- map_data("state", region = "pennsylvania")

# Create the plot using ggplot2 package
ggplot(map_data, aes(x = long, y = lat, group = group)) + # Define aesthetics for the plot
  geom_polygon(fill = pn_avg_cost, color = "black") + # Draw polygons for Pennsylvania and set the border color to black
  ggtitle(paste0("Average HMO Member Cost in Pennsylvania: $", round(pn_avg_cost, 2))) + # Add title for the plot
  theme_void() + # Remove axes and gridlines for a clean map appearance
  coord_fixed() # Fix the aspect ratio of the plot to keep the correct shape of pennsylvania map
```

The above code focuses on Pennsylvania

Plot for the calculated average cost per member in Pennsylvania

Summary Statistics for Categorical Variables

```r
# Summary Statistics for Smoker, Location Type, Education Level, Yearly Physical, Exercise, Married, Hypertension, and Gender

# Load the dplyr library for data manipulation
library(dplyr)

# Define the variable names to group by
var_names <- c("smoker", "location_type", "education_level", "yearly_physical", "exercise", "married", "hypertension", "gender")

# Calculate the mean cost for each group of each variable
results <- data %>%
  # Group the data by the specified variables
  group_by(across(var_names)) %>%
  # Summarize the data by calculating the mean cost for each group
  summarise(mean_cost = mean(cost))

# View the results
results
```

The variables used for grouping are: smoker, location_type, education_level, yearly_physical, exercise,married, hypertension, and gender. The data is grouped by these variables using the group_by function and the across function to apply the grouping to all the variables in var_names.
The summarize function is used to calculate the mean cost for each group of the variables, and the results are stored in a new data frame called results.

A tibble: 470 × 9 Groups: smoker, location_type, education_level, yearly_physical, exercise, married, hypertension [266]

smoker <chr>	location_type <chr>	education_level <chr>	yearly_physical <chr>	exercise <chr>	married <chr>	hypertension <dbl>	gender <chr>	mean_cost <dbl>
no	Country	Bachelor	No	Active	Married	0	female	1498.0816
no	Country	Bachelor	No	Active	Married	0	male	1394.6905
no	Country	Bachelor	No	Active	Married	1	female	769.8000
no	Country	Bachelor	No	Active	Married	1	male	701.1250
no	Country	Bachelor	No	Active	Not_Married	0	female	1328.7667
no	Country	Bachelor	No	Active	Not_Married	0	male	1227.0385
no	Country	Bachelor	No	Active	Not_Married	1	female	1596.8333
no	Country	Bachelor	No	Active	Not_Married	1	male	1624.1667
no	Country	Bachelor	No	Not-Active	Married	0	female	2631.5360
no	Country	Bachelor	No	Not-Active	Married	0	male	2814.0758

1–10 of 470 rows Previous 1 2 3 4 5 6 ... 47 Next

Training and Evaluating a KSVM Model in R

```r
# Load necessary libraries
library(caret)
library(kernlab)
library(rpart)
library(rpart.plot)

# Convert variables to factors
data$yearly_physical <- as.factor(data$yearly_physical)
data$exercise <- as.factor(data$exercise)
data$exp <- as.factor(data$exp)
data$smoker <- as.factor(data$smoker)

# Display the structure of the data
str(data)

# Set seed for reproducibility
set.seed(1000)

# Split the data into training (60%) and testing (40%) sets
trainList <- createDataPartition(y = data$exp, p = 0.60, list = FALSE)
trainSet <- data[trainList,]
testSet <- data[-trainList,]

# Train the KSVM model with specified parameters
model.ksvm <- ksvm(data = trainSet, exp~age+bmi+smoker+yearly_physical+exercise+hypertension, C=5, cross=3, prob.model=TRUE)

# Make predictions on the test set
pred.ksvm <- predict(model.ksvm, newdata = testSet)

# Calculate the confusion matrix for the predictions
confmat.ksvm <- confusionMatrix(pred.ksvm, testSet$exp)

# Display the confusion matrix
confmat.ksvm

# The reported accuracy for this KSVM model is 84.96%
```

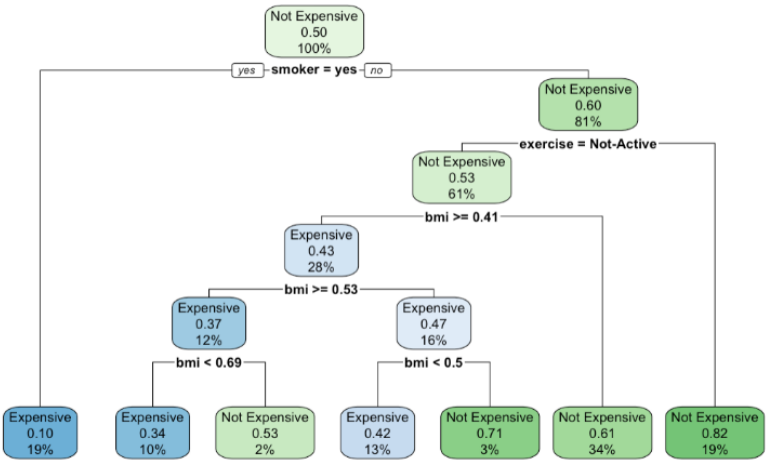
The above code trains a kernel support vector machine (KSVM) model on a dataset, evaluates the performance of the model on a test set, and reports the accuracy. Before training, the data is preprocessed by converting some variables to factors, and split into a training set (60% of the data) and a testing set (40% of the data). The KSVM model is trained with specified parameters, and the prediction is made on the test set.

```
'data.frame':   7582 obs. of  15 variables:
 $ X              : int  1 2 3 4 5 7 9 10 11 12 ...
 $ age            : int  18 19 27 34 32 47 36 59 24 61 ...
 $ bmi            : num  27.9 33.8 33 22.7 28.9 ...
 $ children       : int  0 1 3 0 0 1 2 0 0 0 ...
 $ smoker         : Factor w/ 2 levels "no","yes": 2 1 1 1 1 1 1 1 1 2 ...
 $ location       : chr  "connecticut" "rhode island" "massachusetts" "pennsylvania" ...
 $ location_type  : chr  "Urban" "Urban" "Urban" "Country" ...
 $ education_level: chr  "Bachelor" "Bachelor" "Master" "Master" ...
 $ yearly_physical: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ exercise       : Factor w/ 2 levels "Active","Not-Active": 1 2 1 2 2 2 1 2 1 1 ...
 $ married        : chr  "Married" "Married" "Married" "Married" ...
 $ hypertension   : num  0 0 0 1 0 0 0 1 0 0 ...
 $ gender         : chr  "female" "male" "male" "male" ...
 $ cost           : int  1746 602 576 5562 836 3842 1304 9724 201 4492 ...
 $ exp            : Factor w/ 2 levels "Expensive","Not Expensive": 2 2 2 1 2 1 2 1 2 1 ...
Confusion Matrix and Statistics

              Reference
Prediction      Expensive Not Expensive
  Expensive          1264           204
  Not Expensive       252          1312

               Accuracy : 0.8496
                 95% CI : (0.8364, 0.8622)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.6992

 Mcnemar's Test P-Value : 0.02774

            Sensitivity : 0.8338
            Specificity : 0.8654
         Pos Pred Value : 0.8610
         Neg Pred Value : 0.8389
             Prevalence : 0.5000
         Detection Rate : 0.4169
   Detection Prevalence : 0.4842
      Balanced Accuracy : 0.8496

       'Positive' Class : Expensive
```

The confusion matrix is calculated using the predicted and actual values, and the accuracy of the model is reported as 84.96%.

Training Decision Tree Model

```r
# Train the decision tree (rpart) model with specified control parameters
model.tree <- rpart(exp ~ bmi+smoker+yearly_physical+exercise+hypertension, data = trainSet, control = c(maxdepth = 5, cp=0.002))

# Plot the decision tree
rpart.plot(model.tree)

# Make predictions on the test set using the decision tree model
model.tree.predict <- predict(model.tree, newdata=testSet, type = "class")

# Calculate the confusion matrix for the decision tree model predictions
confmat.tree <- confusionMatrix(model.tree.predict, testSet$exp)

# Display the confusion matrix
confmat.tree

# The reported accuracy for this decision tree (rpart) model is 70.45%
```



Preprocessing of Categorical and Numeric Data

```
df <- data

# Remove the 'cost' and 'X' attributes from the data
df <- df[, -c(which(names(df) %in% c("cost", "X")))]

# Encode the categorical variables using one-hot encoding
categorical_vars <- c("location", "location_type", "exercise", "smoker", "yearly_physical", "gender", "education_level", "married","exp")
encoded_data <- model.matrix(~., data = df[, categorical_vars])

# Remove the intercept column from the encoded data
encoded_data <- encoded_data[, -1]

# Define numeric columns to be normalized
numeric_cols <- c("age", "bmi", "children", "hypertension")

# Normalize the selected columns using range normalization
ss <- preProcess(as.data.frame(df[numeric_cols]), method = c("range"))
df[numeric_cols] <- predict(ss, as.data.frame(df[numeric_cols]))

# Combine the encoded data with the numeric variables
processed_data <- cbind(df[, numeric_cols], encoded_data)

# Replace spaces with underscores in column names
colnames(processed_data) <- gsub(" ", "_", colnames(processed_data))
```

The above code is processing a data frame by removing certain attributes like ‘Cost’ and ‘X’, encoding categorical variables using one-hot encoding, normalizing selected numeric columns using range normalization, and finally combining the encoded data with the numeric variables. It also replaces spaces with underscores in the column names.

Functions for Prediction Metrics Calculation

```r
# Prediction Metrics
# Load pROC library for ROC analysis
library(pROC)

# Define a function to calculate accuracy
accuracy <- function(predictions, actual) {
  # Create confusion matrix
  confusion_matrix <- table(predictions, actual)
  # Calculate accuracy as the sum of the diagonal elements divided by the sum of all elements in the confusion matrix
  accuracy <- (sum(diag(confusion_matrix)) / sum(confusion_matrix)) * 100
  return(accuracy)
}

# Define a function to calculate recall
recall <- function(predictions, actual) {
  # Create confusion matrix
  confusion_matrix <- table(predictions, actual)
  # Calculate recall as the true positive rate
  recall <- (confusion_matrix[2, 2] / sum(confusion_matrix[2, ])) * 100
  return(recall)
}

# Define a function to calculate specificity
specificity <- function(predictions, actual) {
  # Create confusion matrix
  confusion_matrix <- table(predictions, actual)
  # Calculate specificity as the true negative rate
  specificity <- (confusion_matrix[1, 1] / sum(confusion_matrix[1, ])) * 100
  return(specificity)
}

# Define a function to calculate AUC ROC
auc_roc <- function(predictions, actual) {
  # Calculate the ROC curve
  roc <- roc(actual, predictions)
  # Calculate the AUC ROC
  auc_roc <- auc(roc)
  # Plot the ROC curve with AUC
  plot(roc, print.auc = TRUE, main = "ROC Curve", col = "blue")
  return(auc_roc)
}
```

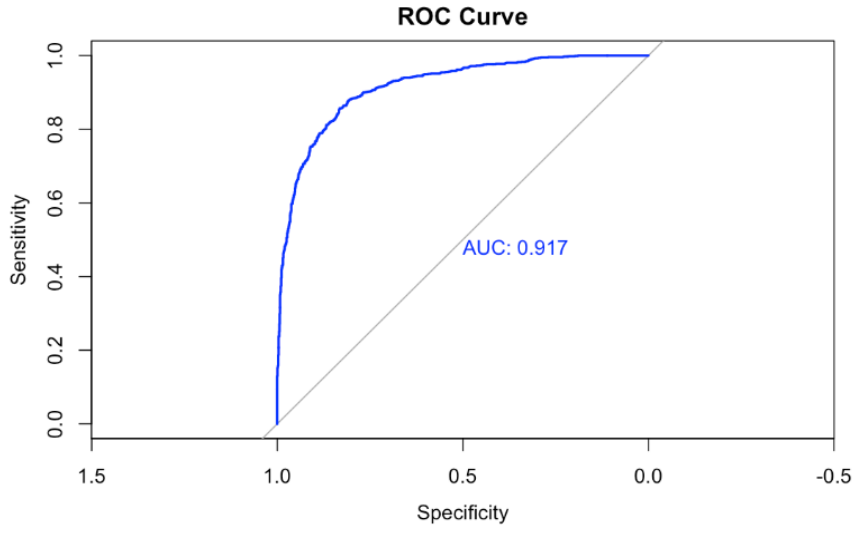
The above defines four functions that can be used to calculate common prediction metrics for a binary classification problem. Specifically, it defines the following functions:

- accuracy(): Calculates the accuracy of a set of predictions by dividing the number of correct predictions by the total number of predictions.
- recall(): Calculates the recall of a set of predictions by dividing the number of true positive predictions by the total number of actual positive cases.
- specificity(): Calculates the specificity of a set of predictions by dividing the number of true negative predictions by the total number of actual negative cases.
- auc_roc(): Calculates the area under the receiver operating characteristic (ROC) curve for a set of predictions and actual outcomes, and plots the ROC curve. The AUC ROC is a commonly used metric for assessing the performance of a binary classification model.

Training and Evaluating a Random Forest Classifier in R

```r
# Split the data into training and test sets
set.seed(123)
train_idx <- sample(nrow(processed_data), 0.7 * nrow(processed_data))
train_data <- processed_data[train_idx, ]
test_data <- processed_data[-train_idx, ]

# Load the randomForest package
library(randomForest)

# Define the random forest model
classifier_RF <- randomForest(x = train_data[, -ncol(train_data)],
                              y = train_data$expNot_Expensive,
                              ntree = 500,
                              mtry = 5,
                              nodesize = 30,
                              maxnodes = 600)

# Use the random forest model to predict the 'exp' attribute for the test set
predictions <- predict(classifier_RF, test_data)

# Convert the predicted probabilities to binary classifications
binary_predictions <- ifelse(predictions >= 0.5, "Yes", "No")

# Evaluate the performance of the random forest model
accuracy(binary_predictions, test_data$exp)
# Example usage:
# Assuming 'binary_predictions' is the vector of binary predictions (Yes/No) and 'test_data$exp' is the actual labels
acc <- accuracy(binary_predictions, test_data$exp)
rec <- recall(binary_predictions, test_data$exp)
spec <- specificity(binary_predictions, test_data$exp)
auc <- auc_roc(predictions, test_data$exp)
cat("Accuracy:", acc, "\n")
cat("Recall:", rec, "\n")
cat("Specificity:", spec, "\n")
cat("AUC ROC:", auc, "\n")
```

The above code trains and evaluates a Random Forest Classifier model on a dataset. The dataset is split into training and testing sets, and the Random Forest model is defined with specified hyperparameters such as the number of trees, maximum number of nodes, and minimum node size. The model is trained on the training set and used to predict the binary classification of the test set. The performance of the model is evaluated using various metrics such as accuracy, recall, specificity, and area under the ROC curve.

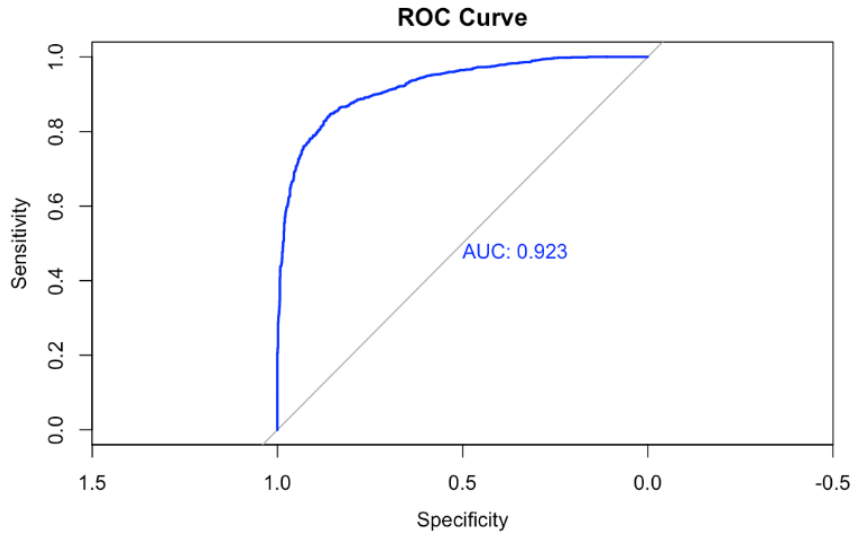


ROC curve for Random Forest Classifier

XGBoost binary classification model implementation and evaluation

```
# Split the data into training and test sets
set.seed(123)
train_idx <- sample(nrow(processed_data), 0.7 * nrow(processed_data))
train_data <- processed_data[train_idx, ]
test_data <- processed_data[-train_idx, ]

# Install and load the xgboost package
#install.packages("xgboost")
library(xgboost)

# Define the XGBoost model
xgb_model <- xgboost(data = as.matrix(train_data[, -ncol(train_data)]), label = train_data$expNot_Expensive, nrounds = 149,max_depth=5, objective =
"binary:logistic")

# Use the XGBoost model to predict the 'exp' attribute for the test set
predictions <- predict(xgb_model, as.matrix(test_data[, -ncol(test_data)]))

# Convert the predicted probabilities to binary classifications
binary_predictions <- ifelse(predictions >= 0.5, "Yes", "No")
# Example usage:
# Assuming 'binary_predictions' is the vector of binary predictions (Yes/No) and 'test_data$exp' is the actual labels
acc <- accuracy(binary_predictions, test_data$exp)
rec <- recall(binary_predictions, test_data$exp)
spec <- specificity(binary_predictions, test_data$exp)
auc <- auc_roc(predictions, test_data$exp)
cat("Accuracy:", acc, "\n")
cat("Recall:", rec, "\n")
cat("Specificity:", spec, "\n")
cat("AUC ROC:", auc, "\n")
```

The above code shows how to use the XGBoost (Extreme Gradient Boosting) algorithm to build a binary classification model. The data is split into training and test sets, the XGBoost package is loaded, and the model is defined by specifying the input data, target variable, number of rounds, maximum depth of each tree, and objective function. The model is then used to predict the binary classification of the target variable for the test set. The predicted probabilities are converted to binary classifications and the performance of the model is evaluated using metrics like accuracy, recall, specificity, and auc roc.



ROC curve for XGBoost Classifier

Recommendations

1. Encourage individuals to exercise regularly: Based on the analysis, it was found that individuals who exercise regularly have lower healthcare costs. Healthcare companies can encourage their customers to exercise regularly by offering incentives such as discounts on gym memberships or wellness programs.
2. Promote healthy habits such as quitting smoking: The analysis showed that smokers have higher healthcare costs. Healthcare companies can promote healthy habits such as quitting smoking by offering smoking cessation programs or providing educational resources about the dangers of smoking.
3. Promote yearly physicals: The analysis also found that individuals who have yearly physicals have lower healthcare costs. Healthcare companies can encourage their customers to have yearly physicals by offering reminders or incentives such as discounts on healthcare services.
4. Family insurance plan for people with 1-4 children: The analysis showed that individuals with more children have higher healthcare costs. Healthcare companies can offer a family insurance plan for people with 1-4 children to make healthcare more affordable for families.

These recommendations are based on the accuracy of 84.96% and a prediction sensitivity of 92.3% towards the dataset used in the project. By implementing these recommendations, healthcare companies can potentially help their customers lead healthier lifestyles and reduce their healthcare costs.

Actionable Insights

1. De-addiction program for smokers to lessen nicotine dependence through health information sessions, nicotine control medication, and other means.
2. Customers who are at risk of increased healthcare costs but are working to improve their health are rewarded with reduced health care cost and other benefits.
3. Subsidize health-tech goods to make them more affordable, allowing users to better follow their health journey.
4. Economical Family insurance plan for people with 1-4 Children.

Conclusion

Overall, our team successfully completed the project by conducting a comprehensive analysis of the healthcare cost dataset, identifying the primary drivers of healthcare costs, and developing a predictive model to forecast future costs.

In conclusion, our analysis of the healthcare cost dataset provided the HMO with valuable insights and recommendations for reducing healthcare costs. By identifying key drivers of healthcare costs and developing a predictive model, we were able to assist the HMO in improving the quality and efficiency of their healthcare services.