Heart Disease Dataset Analysis and Predictions

Data Science Capstone - Project

Vamsikrishna Kota, MS Computer Science

21 March, 2023

Introduction

In this R project, the goal is to build a multiple linear regression model to predict the risk of heart disease based on two predictor variables: smoking and biking. The project starts with importing the dataset (HeartDataset.csv), checking its structure, summary, missing values, and outliers. The dataset is then visualized by creating histograms, scatterplots, and a correlation matrix. The next step is to split the dataset into train, test, and validation sets. After that, a multiple regression models (like MLR, Random Forest, SVM) were built. Finally, the model is evaluated using R-squared and RMSE, and predictions are made on the testing dataset using the predict() function. Overall, this project aims to build a regression model that can accurately predict the risk of heart disease based on the given predictor variables.

Importing and understanding the data

In this step, we import the necessary libraries for the analysis and read the dataset.

```
knitr::opts_chunk$set(echo = TRUE, message = FALSE)

# install necessary libraries for the analysis
.packages = c("GGally",
              "ggcorrplot",
              "ggplot2",
              "rsq",
              "Metrics",
              "kableExtra",
              "caret",
              "caTools",
              "e1071",
              "tinytex",
              "randomForest",
              "dplyr"
)

# Install CRAN packages (if not already installed)
.inst <- .packages %in% installed.packages()
if(length(.packages[!.inst]) > 0) install.packages(.packages[!.inst])

library(GGally) # for visualization
library(ggplot2) # for visualization
```

```
library(ggcorrplot) # for visualization
library(dplyr)
library(rsq) # for evaluation
library(Metrics) # for evaluation
library(caret)
library(e1071) # for SVR algo
library(randomForest) # for RF algo
library(caTools)
library(kableExtra)
library(tinytex)
```

Read the dataset

After loading the necessary libraries, we read the data using read.csv() function.

We then rename the column name from 'heart.disease' to 'heart_disease' using rename() function from dplyr library.

We are Removing X column from the dataset which is not useful for the algorithms.

Data set is the survey of the 500 towns with details of percentage of people in each town who smoke, the percentage of people in each town who bike to work, and the percentage of people in each town who have heart disease.

```
# Data from the Kaggle and same uploaded to my GIT HUB
heartdataset <- read.csv("https://raw.githubusercontent.com/VamsiK51/CapStoneCYO/main/HeartDataset.csv")

# Removing X column from the dataset which is not relavant to dataset algorithm
heart_multi <- subset(heartdataset, select = -c(X))

# rename the column name from 'heart.disease' to 'heart_disease'
heart_multi <- rename(heart_multi, heart_disease = heart.disease)
```

Check structure of the data

We check the first 5 rows of the dataset using head() function and column names using colnames() function.

The str() function shows that there are 498 observations with numeric datatype.

```
# check the first 5 rows of the dataset
head(heart_multi, n=5)
```

```
##      biking   smoking heart_disease
## 1 30.801246 10.896608     11.769423
## 2 65.129215  2.219563      2.854081
## 3  1.959665 17.588331     17.177803
## 4 44.800196  2.802559      6.816647
## 5 69.428454 15.974505      4.062224
```

```
# check the column names
colnames(heart_multi)
```

```
## [1] "biking"        "smoking"       "heart_disease"
```

```
# check structure of the dataset
str(heart_multi) # here we have 498 observations with numeric datatype
```

```
## 'data.frame':    498 obs. of  3 variables:
##  $ biking       : num  30.8 65.13 1.96 44.8 69.43 ...
##  $ smoking      : num  10.9 2.22 17.59 2.8 15.97 ...
##  $ heart_disease: num  11.77 2.85 17.18 6.82 4.06 ...
```

Check missing values

The sum(is.na(heart_multi)) function shows that there are no missing values in the data.

```
# check missing values:
sum(is.na(heart_multi))
```

```
## [1] 0
```

```
# there is no missing values in the data
```

Descriptive Statistics

We then check the summary of the data using summary() function. It gives us the minimum, maximum, and quartile values of each column.

```
# check summary
summary(heart_multi)
```

```
##      biking          smoking        heart_disease
##  Min.   : 1.119   Min.   : 0.5259   Min.   : 0.5519
##  1st Qu.:20.205   1st Qu.: 8.2798   1st Qu.: 6.5137
##  Median :35.824   Median :15.8146   Median :10.3853
##  Mean   :37.788   Mean   :15.4350   Mean   :10.1745
##  3rd Qu.:57.853   3rd Qu.:22.5689   3rd Qu.:13.7240
##  Max.   :74.907   Max.   :29.9467   Max.   :20.4535
```

Following are the findings from the summary:

- The minimum value for heart disease is higher than the minimum values for biking and smoking, indicating that even people who do not engage in significant levels of physical activity or smoking may still be at risk for heart disease.
- The median for heart disease is higher than the medians for biking and smoking, suggesting that the majority of people in the sample may be at moderate to high risk for heart disease.
- The mean value for biking is higher than the mean values for smoking and heart disease, indicating that, on average, people in the sample may engage in more biking than smoking or be at risk for heart disease.
- The third quartile values for all three variables are relatively spread out, indicating that there may be a significant range of values for each variable in the sample.

- The maximum value for biking is higher than the maximum values for smoking and heart disease, suggesting that some people in the sample may engage in very high levels of biking activity.
- The interquartile ranges (IQRs) for biking and smoking are relatively large, indicating that there may be significant variability in the levels of these activities among the sample.
- The IQR for heart disease is smaller than the IQRs for biking and smoking, suggesting that there may be less variability in the risk for heart disease among the sample.

Outlier Detection:

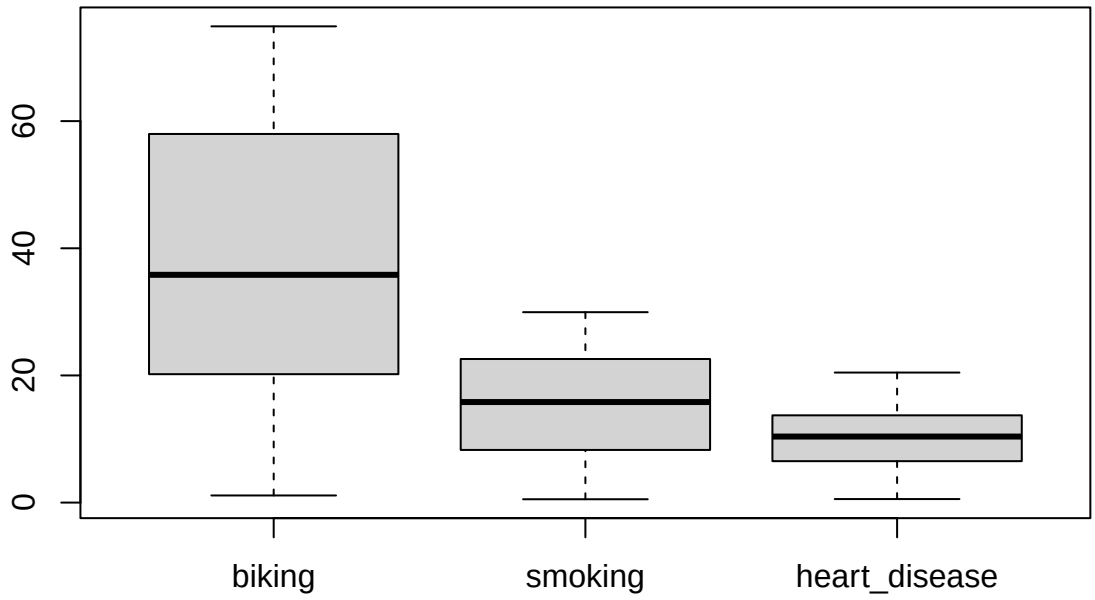
```
# check for outliers using boxplot
Q1 <- apply(heart_multi, 2, quantile, probs=0.25, na.rm=TRUE)
Q3 <- apply(heart_multi, 2, quantile, probs=0.75, na.rm=TRUE)
IQR <- Q3 - Q1

outliers <- apply(heart_multi, 2, function(x)
  sum(x < (Q1 - 1.5 * IQR) | x > (Q3 + 1.5 * IQR), na.rm = TRUE))

outliers
```

```
##        biking       smoking heart_disease
##           191            32             0
```

Finally, we check for outliers using boxplot() function. We calculate the interquartile range (IQR) using apply() and quantile() functions.

We then identify outliers using the IQR method and sum() function. The boxplot() and IQR functions confirms that there are no outliers in the data.

```
# using boxplot method
boxplot(heart_multi)
```



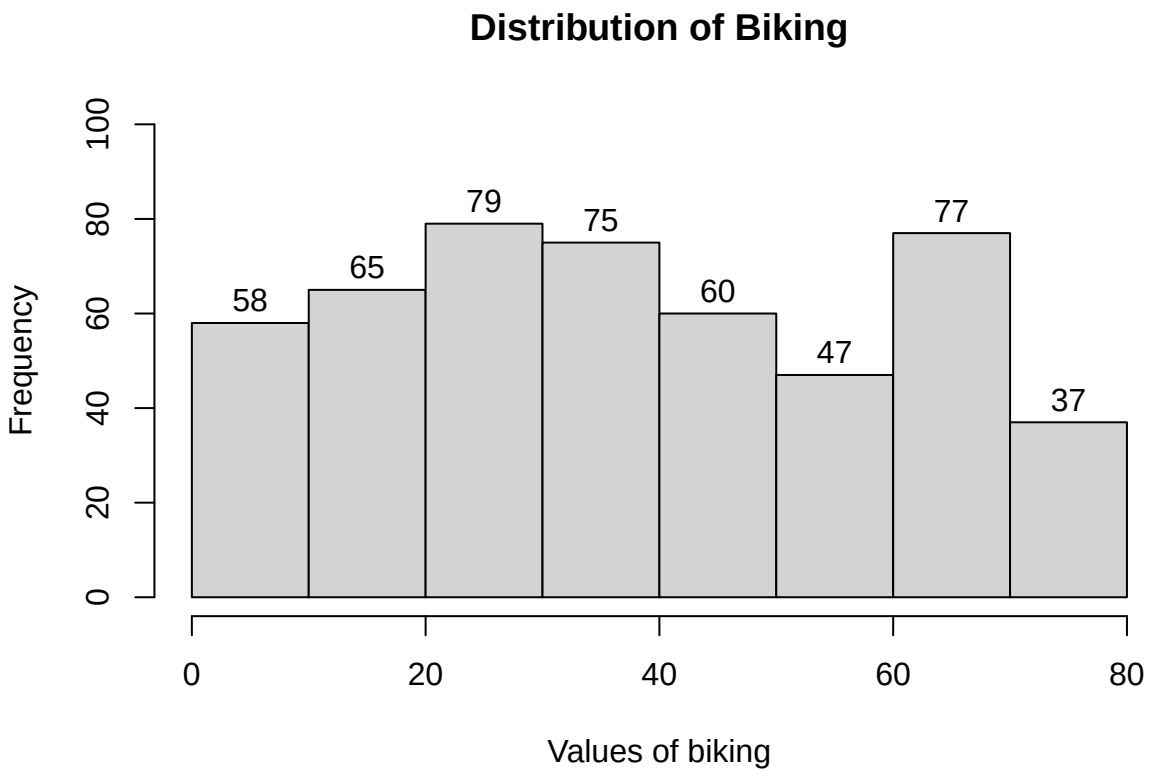

```
# in this data set there are no outliers
```

Visualizing the Dataset

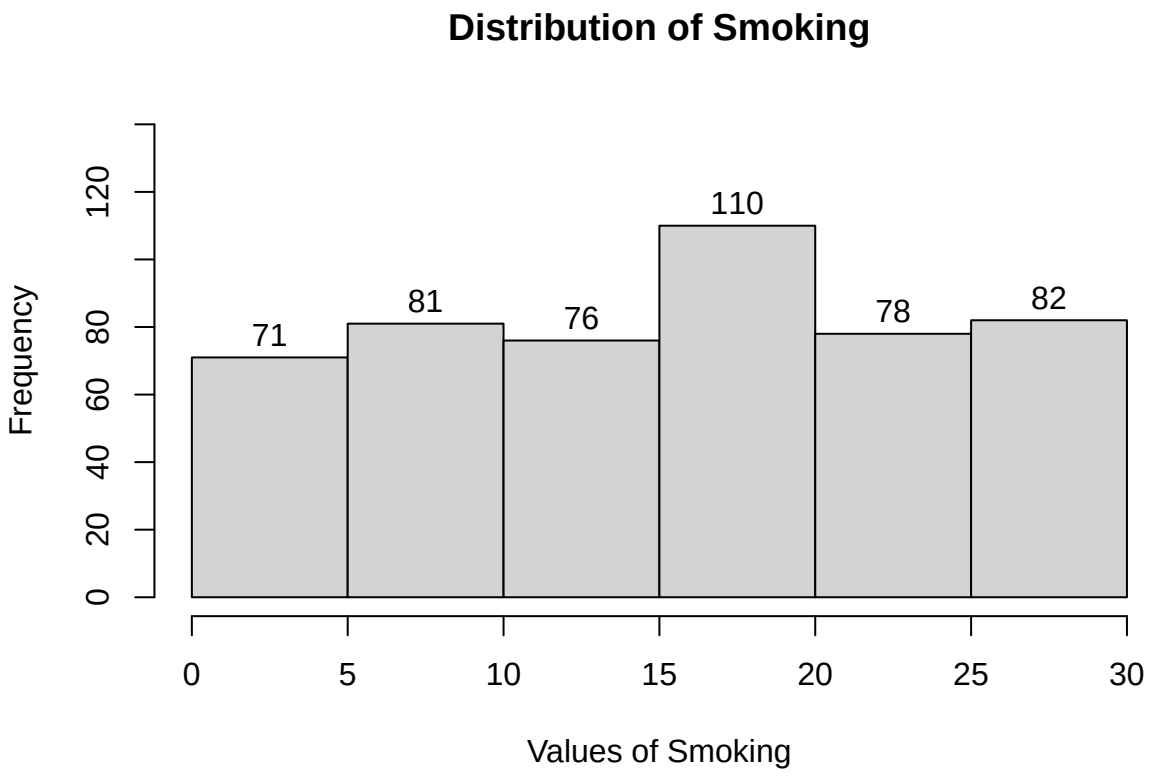
In this step, we will create histograms and scatterplots to explore the relationship between the features and the response variable in the heart disease dataset.

Create Histograms

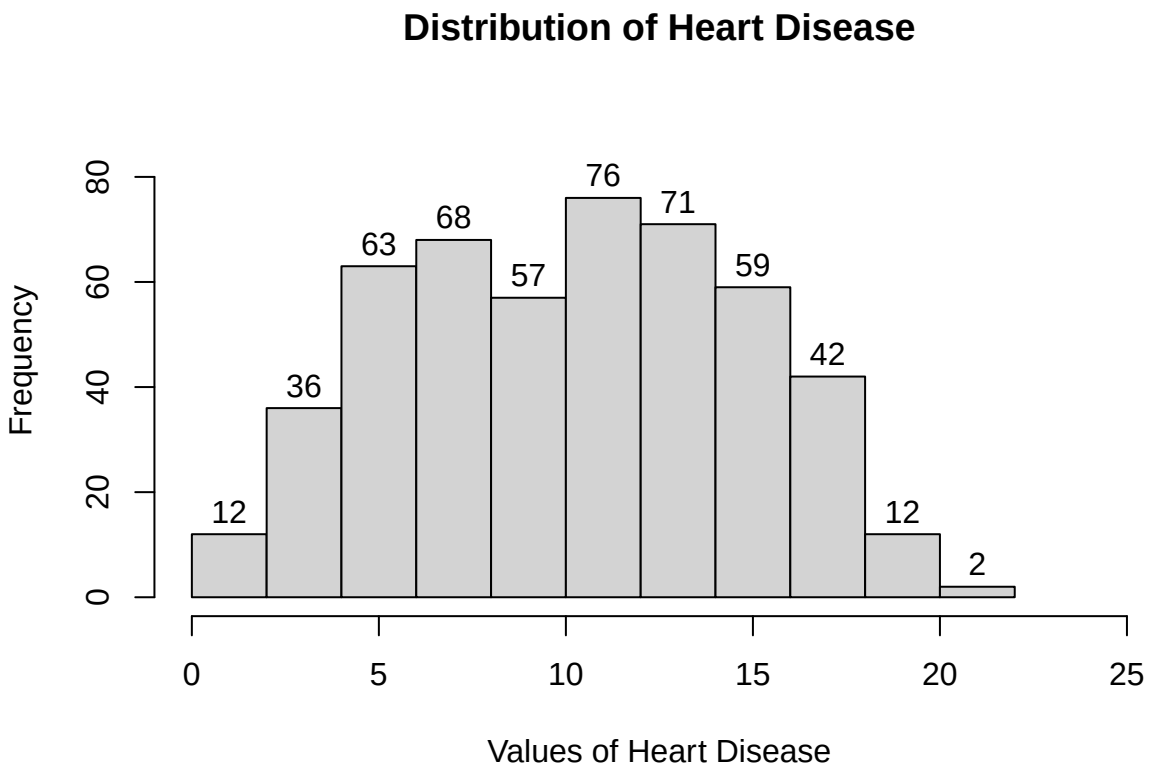
```
# create histogram
hist(heart_multi$biking,
     main = "Distribution of Biking",
     xlab = "Values of biking",
     ylab = "Frequency",
     xlim = range(0, 80),
     ylim = range(0, 100),
     labels = TRUE)
```




```
hist(heart_multi$smoking,
     main = "Distribution of Smoking",
     xlab = "Values of Smoking",
     ylab = "Frequency",
     xlim = range(0, 30),
     ylim = range(0, 140),
     labels = TRUE)
```




```
hist(heart_multi$heart_disease,
     main = "Distribution of Heart Disease",
     xlab = "Values of Heart Disease",
     ylab = "Frequency",
     xlim = range(0, 25),
     ylim = range(0, 90),
     labels = TRUE)
```

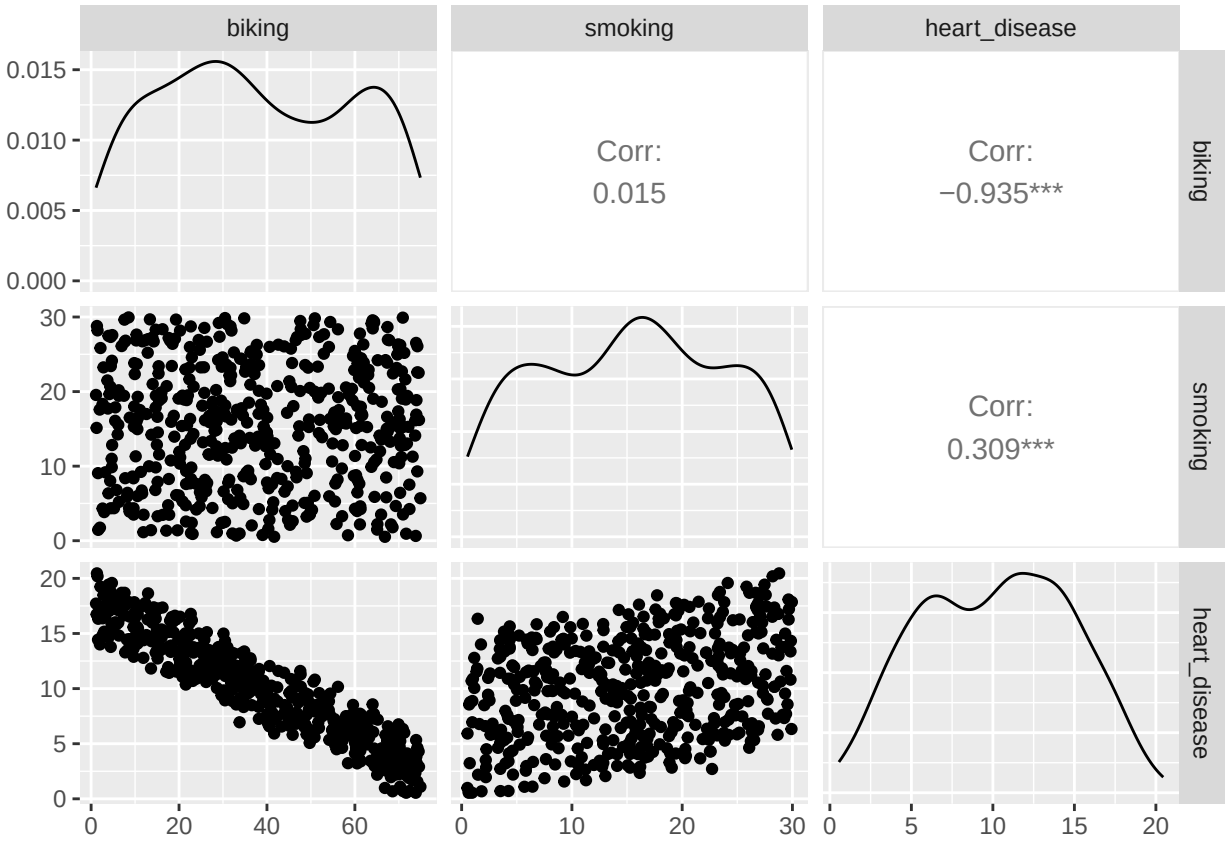


The histograms show the distribution of the three features we are interested in: biking, smoking, and heart disease.

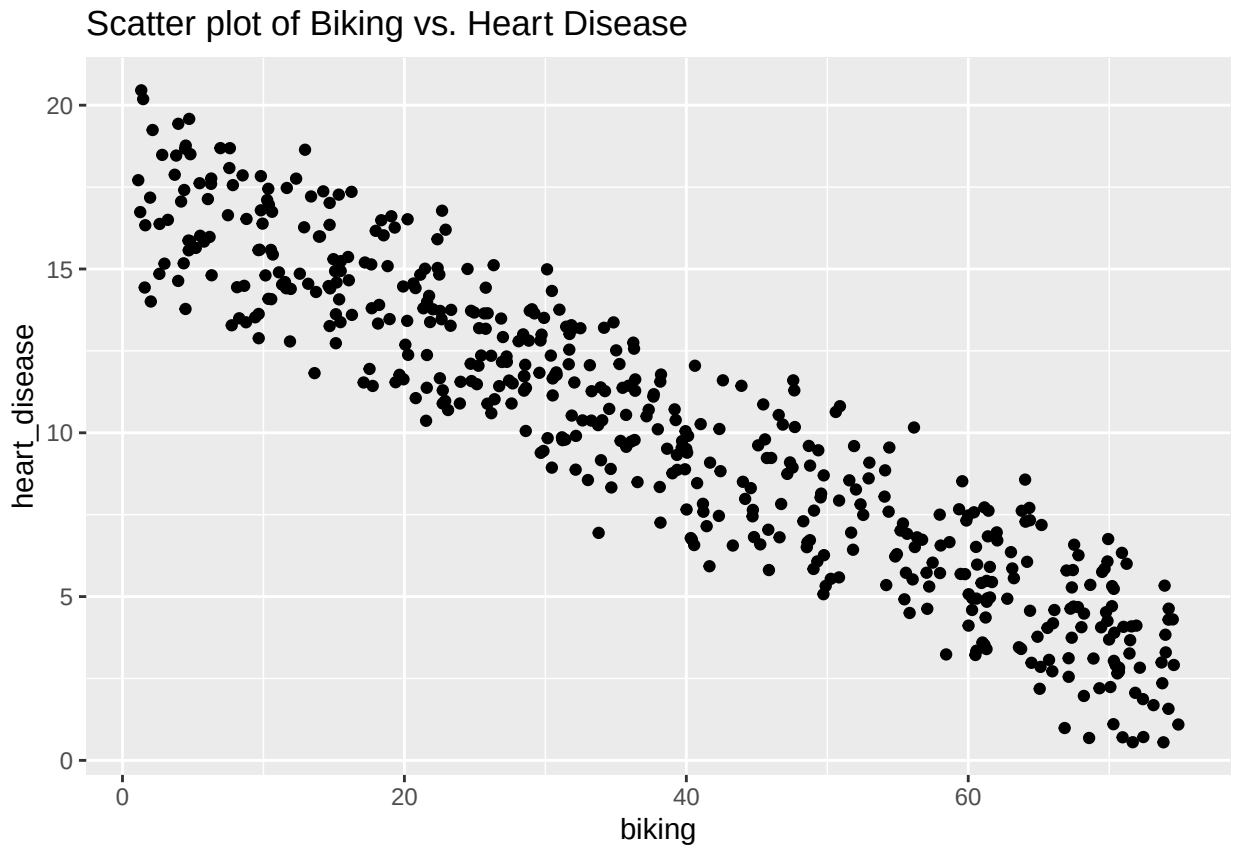
Create Scatterplots

A scatterplot is a graph that displays the relationship between two variables by plotting their values on a two-dimensional coordinate system. Each point on the plot represents a pair of values for the two variables.

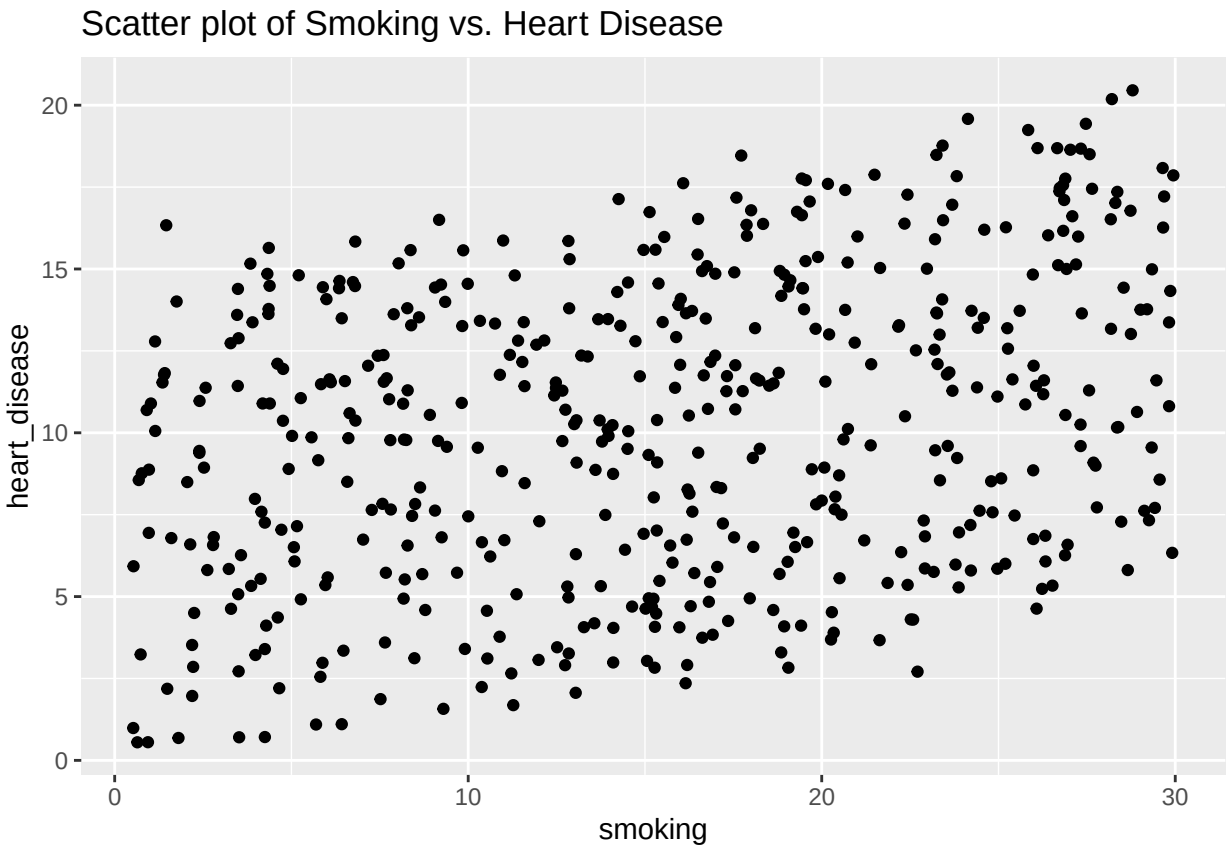
```
# plotting
ggpairs(heart_multi)
```




```
# relationship with biking
ggplot(heart_multi, aes(x = biking, y = heart_disease)) +
  geom_point() +
  labs(title = "Scatter plot of Biking vs. Heart Disease")
```




```
# relationship with Smoking
ggplot(heart_multi, aes(x = smoking, y = heart_disease)) + geom_point() +
  labs(title = "Scatter plot of Smoking vs. Heart Disease")
```



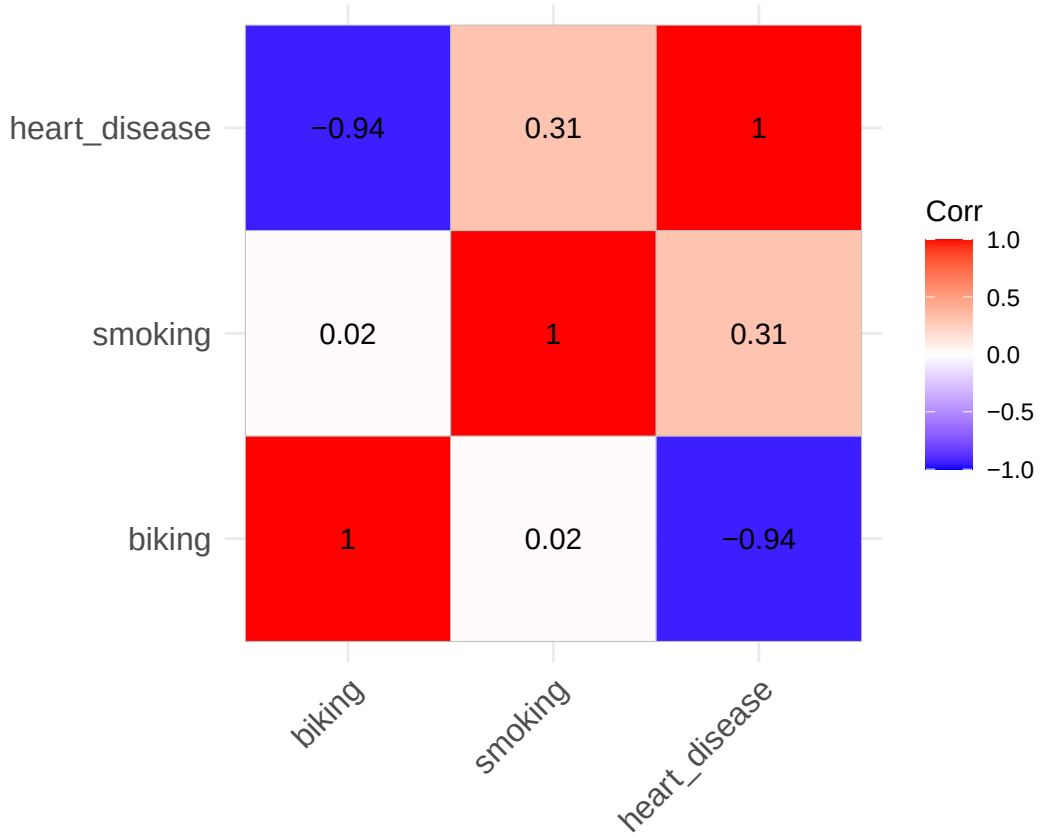
The scatterplots show the relationship between the two features (biking and smoking) and the response variable (heart disease).

It seems there is negative correlation with heart disease and biking.

Create Correlation Matrix

A correlation matrix is a table that shows the correlation coefficients between pairs of variables. Correlation coefficients range from -1 to 1 and represent the strength and direction of the relationship between two variables.

```
# create a correlation matrix
cor_matrix <- cor(heart_multi)
ggcorrplot(cor_matrix, lab = TRUE, lab_size = 4)
```



The correlation matrix shows the correlation between all pairs of features in the dataset.

In our dataset, there is very negative correlation with biking and heart disease. It means if biking values goes up, then risk of heart disease goes down which makes sense.

Also, there is moderate positive correlation of 0.31 wrt heart disease and smoking.

Rescale Features

```
# rescale the features
normalize <- function(x) {
  return ((x - mean(x)) / (max(x) - min(x)))
}

heart_multi <- apply(heart_multi, 2, normalize)
heart_multi <- as.data.frame(heart_multi)
```

Finally, we rescale the features to ensure that they are on the same scale and easier to compare.

Rescaling is a technique used to transform the values of different features to the same scale. This is important because different features may have different units or ranges, and some machine learning algorithms may perform better when all features are on the same scale. In the code provided, the normalize() function is used to rescale the features. It subtracts the mean of each feature from each value and then divides by the range (max - min) of each feature.

Split the data into train, test, and validation

In Step 3, we split the data into three sets: a training set, a test set, and a validation set. The purpose of splitting the data is to have separate sets of data for training, testing, and evaluating our model. This ensures that we are not overfitting the model to the training data and that our model generalizes well to new, unseen data.

First, we set the seed for reproducibility. This ensures that our results will be the same every time we run the code.

Then, we use the sample.split() function from the caret package to split the data into an 80/20 split for the training and validation sets.

We use the resulting split to create a train_df dataset for the training and a validation dataset for validating the model.

```
set.seed(100)

split <- sample.split(heart_multi, SplitRatio = 0.8)
train_df <- subset(heart_multi, split == "TRUE")
validation <- subset(heart_multi, split == "FALSE")
```

Next, we split the train_df dataset into a new training set and a test set using the same 80/20 split.

We use the resulting split to create a train dataset for training the model and a test dataset for testing the model.

```
split <- sample.split(train_df, SplitRatio = 0.8)
train <- subset(train_df, split == "TRUE")
test <- subset(train_df, split == "FALSE")
```

Finally, we use the dim() function to check the dimensions of the train and test datasets.

We also use the head() function to print the first few rows of each dataset to ensure that the data has been split correctly.

```
dim(train) # dimension/shape of train dataset
```

```
## [1] 222   3
```

```
print(head(train))
```

```
##         biking    smoking heart_disease
## 1  -0.09469249 -0.1542586    0.08013852
## 2   0.37053207 -0.4491866   -0.36783260
## 5   0.42879684  0.0183363   -0.30712682
## 7   0.15270447 -0.2166552   -0.12813198
## 10 -0.03430048  0.2665673    0.09667296
## 11  0.19023642 -0.0339866   -0.18814016
```

```
dim(test)  # dimension/shape of test dataset
```

```
## [1] 110   3
```

```
print(head(test))
```

```
##         biking     smoking heart_disease
## 4   0.09502614 -0.42937091   -0.16872470
## 8  -0.44727904 -0.08837303    0.28541006
## 13  0.14879213 -0.14995342   -0.17345412
## 17  0.32385834  0.04778177   -0.23772945
## 22 -0.26105124  0.37286234    0.29412963
## 26  0.20595243  0.41650648   -0.05457317
```

Build Regression models

Muliple Linear Regression

We start by building a linear regression model using the "lm" function. We specify the response variable "heart_disease" and the predictor variables "biking" and "smoking" using the formula syntax. We fit the model on the training data using the "data" argument.

```
# build linear regression model
model <- lm(heart_disease ~ biking + smoking, data = train)
```

We can obtain a summary of the model using the "summary" function. This gives us information about the model's goodness of fit, coefficients, and their significance.

```
# summary
summary(model)
```

```
##
## Call:
## lm(formula = heart_disease ~ biking + smoking, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.10339 -0.02233  0.00297  0.02145  0.07960
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -0.0008587  0.0021011   -0.409    0.683
## biking      -0.7489160  0.0073758 -101.536   <2e-16 ***
## smoking      0.2548973  0.0075421   33.797   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03125 on 219 degrees of freedom
## Multiple R-squared:  0.9816, Adjusted R-squared:  0.9814
## F-statistic:  5830 on 2 and 219 DF,  p-value: < 2.2e-16
```

Here are the key takeaways from the above summary:

- The formula for the model is heart_disease ~ biking + smoking, meaning that the heart disease outcome variable is being predicted by the biking and smoking predictor variables.
- The coefficients table shows the estimated coefficients (Estimate column) for each predictor variable, along with their standard error (Std. Error column), t-value (t value column), and p-value (Pr(>|t|) column).
- The Intercept coefficient is estimated to be -0.0008587 with a standard error of 0.0021011. However, the t-value is not significant (p-value=0.683), indicating that it is not significantly different from zero.
- The biking and smoking coefficients are estimated to be -0.7489160 and 0.2548973, respectively. Both coefficients are highly significant with very low p-values (<2e-16), indicating that they have a strong relationship with the heart disease outcome variable.
- The Residual standard error is 0.03125, which indicates the standard deviation of the residuals, or the average distance that the observed data points fall from the predicted values.
- The Multiple R-squared value is 0.9816, which indicates that 98.16% of the variation in the heart disease outcome variable can be explained by the model using the predictor variables.
- The Adjusted R-squared value is 0.9814, which is similar to the Multiple R-squared value but adjusted for the number of predictor variables used in the model.
- The F-statistic is 5830 with a p-value of < 2.2e-16, indicating that the model is highly significant overall.

```
# confidence interval
confint(model)
```

```
##                   2.5 %       97.5 %
## (Intercept) -0.00499975  0.003282305
## biking      -0.76345277 -0.734379302
## smoking      0.24003296  0.269761643
```

The confint() function in R provides confidence intervals for the regression coefficients of the linear regression model.

In the output, the left column shows the lower confidence limit (at 2.5%) and the right column shows the upper confidence limit (at 97.5%) for each of the regression coefficients.

For the given output, we can interpret the confidence intervals for the regression coefficients as follows:

- The intercept has a confidence interval of [-0.00499975, 0.003282305] at a 95% confidence level. This indicates that the true value of the intercept falls within this range with 95% confidence.
- For the biking variable, the confidence interval is [-0.76345277, -0.734379302] at a 95% confidence level. This indicates that the true coefficient value for biking falls within this range with 95% confidence.
- For the smoking variable, the confidence interval is [0.24003296, 0.269761643] at a 95% confidence level. This indicates that the true coefficient value for smoking falls within this range with 95% confidence.

Overall, confidence intervals provide a range of values within which the true population parameters are likely to fall.

A wider confidence interval indicates greater uncertainty in the estimated parameter value. Confidence intervals can be used to assess the significance of the regression coefficients and to identify variables that have a statistically significant effect on the outcome variable.

```
# print intercept and coefficient
coefficients <- coef(model)
print(coefficients)
```

```
##   (Intercept)        biking       smoking
## -0.0008587224 -0.7489160346  0.2548972993
```

The print(coefficients) command provides the estimated coefficients for the multiple linear regression model. The three coefficients listed are:

- (Intercept) : This represents the estimated y-intercept (or constant) of the regression line. In this case, the estimated y-intercept is -0.0008587224.
- biking : This represents the estimated coefficient for the predictor variable "biking". It shows the effect of a one-unit increase in the "biking" variable on the dependent variable "heart_disease". In this case, the estimated coefficient for "biking" is -0.7489160344. Thus, an increase in "biking" by one unit is associated with a decrease of approximately 0.75 units in "heart_disease" (holding other variables constant).
- smoking : This represents the estimated coefficient for the predictor variable "smoking". It shows the effect of a one-unit increase in the "smoking" variable on the dependent variable "heart_disease". In this case, the estimated coefficient for "smoking" is 0.2548972992. Thus, an increase in "smoking" by one unit is associated with an increase of approximately 0.25 units in "heart_disease" (holding other variables constant).

These coefficients can be used to build the regression equation, which predicts the value of the dependent variable based on the values of the predictor variables. In this case, the equation would be:

```
heart_disease = -0.0008587224 - 0.7489160344 * biking + 0.2548972992 * smoking
```

Predictions

We can use our model to make predictions on the testing set using the "predict" function. We store the predicted values in the "pred" object.

```
pred <- predict(model, test)
```

Model Evaluation

We evaluate our model using the R-squared and RMSE metrics. We first calculate the R-squared value using the "R2" function from the "rsq" package. We also calculate the RMSE using the formula and the "sqrt" and "mean" functions. We then print these metrics using the "paste" function.

```
# calculate R-squared
lm_rsq <- R2(pred, test$heart_disease)
print(paste("R-squared value for MLR:", round(lm_rsq, 4)))
```

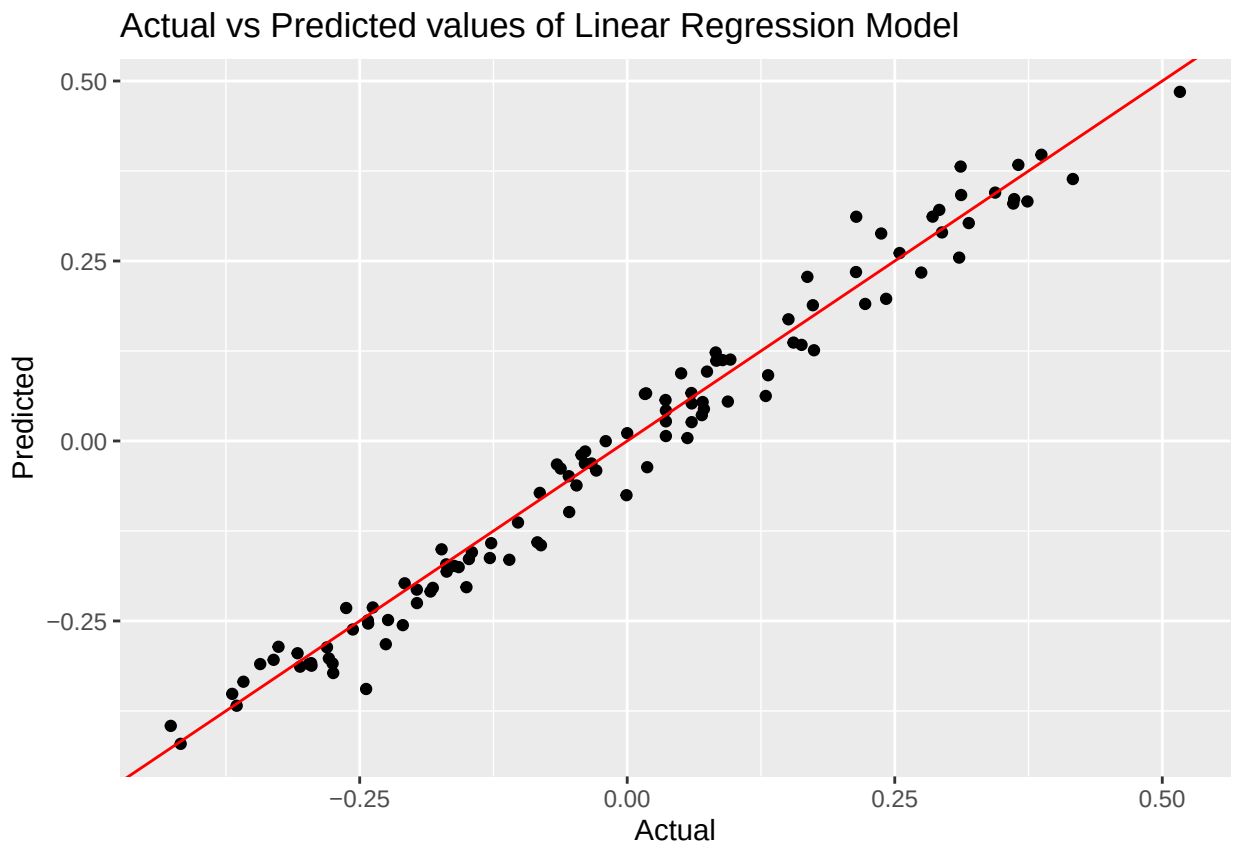
```
## [1] "R-squared value for MLR: 0.9777"
```

```
# calculate RMSE
lm_rmse <- sqrt(mean(pred-test$heart_disease)^2)
print(paste("RMSE value for MLR:", round(lm_rmse, 4)))
```

```
## [1] "RMSE value for MLR: 0.0066"
```

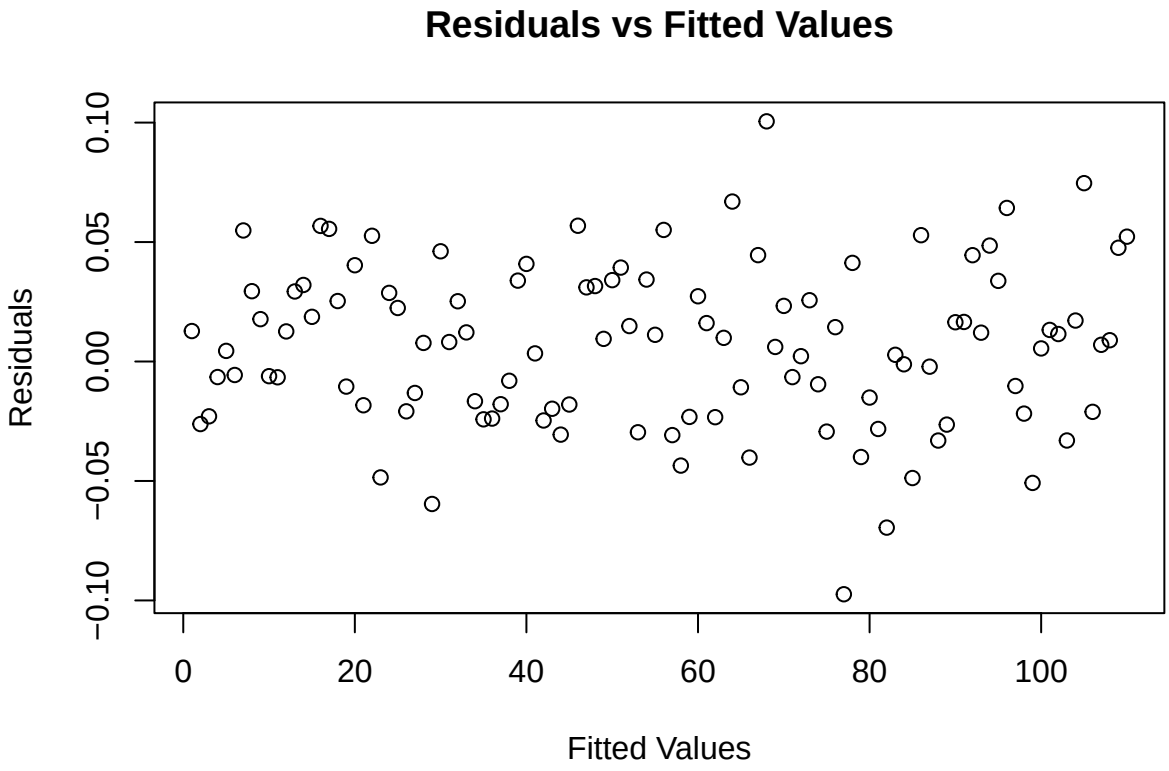
We can visualize the actual versus predicted values using a scatter plot. We use the "ggplot" function from the "ggplot2" package to create the plot. We also add a red line that represents a perfect fit.

```
# plot actual vs predicted
df <- data.frame(actual = test$heart_disease, predicted = pred)
ggplot(df, aes(x = actual, y = predicted)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  labs(x = "Actual", y = "Predicted",
       title = "Actual vs Predicted values of Linear Regression Model")
```



We can calculate the residuals (difference between actual and predicted values) and plot them against the fitted values using the "plot" function. This helps us to identify any patterns in the errors.

```
# calculate residuals
resid <- test$heart_disease - pred

# plot error terms
plot(resid, main = "Residuals vs Fitted Values", xlab = "Fitted Values",
     ylab = "Residuals")
```

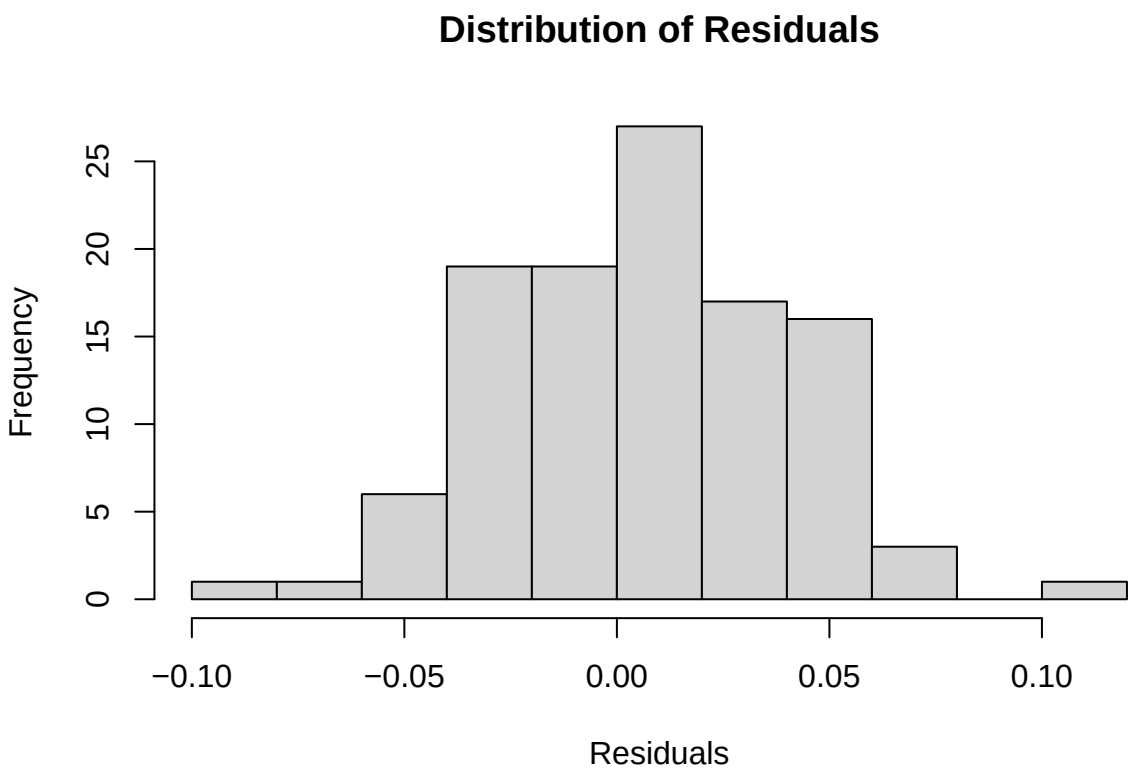


We can also plot the distribution of residuals using a histogram to check if they are normally distributed.

- The residual vs fitted plot is a graphical way to evaluate the goodness of fit of a linear regression model. It helps us to determine if the model is capturing the pattern of the data correctly or not. The plot shows the difference between the actual and predicted values (residuals) on the y-axis and the predicted values on the x-axis.
- In this specific plot, we are plotting the residuals against the fitted values (predicted values) of the test data. The fitted values represent the values predicted by the regression model for each observation in the test data. The residuals are the differences between the actual and predicted values. If the model is a good fit, then the residuals should be randomly scattered around the horizontal line at 0. This indicates that there is no pattern left in the residuals that could be used to improve the model.
- In this particular plot, we can observe that the residuals are randomly scattered around the horizontal line, indicating that the model has captured the pattern of the data well. However, there are some outliers present in the plot, which can be a cause for concern. These outliers may indicate that the model is not capturing some important aspects of the data. Therefore, it is important to investigate these outliers and try to understand why they are present and if any changes need to be made to the model.

In summary, the residual vs fitted plot helps us to evaluate the quality of the model fit and identify potential problems with the model. By examining this plot, we can determine whether the model is a good fit for the data or not.

```
# plot distribution of residuals
hist(resid, main = "Distribution of Residuals", xlab = "Residuals")
```




```
# Storing RMSE AND R SQUARE Values to DATA FRAME for displaying
accuracy_algorithm = tibble(Model = "Multiple linear regression", Accuracy = lm_rsq, RMSE = lm_rmse)

accuracy_algorithm
```

```
## # A tibble: 1 x 3
##   Model                        Accuracy    RMSE
##   <chr>                           <dbl>   <dbl>
## 1 Multiple linear regression    0.978 0.00656
```

The distribution of residuals plot shows the frequency distribution of the differences between the predicted values and actual values (i.e., residuals) in the form of a histogram. Residuals are the differences between the predicted and actual values, and their distribution can provide insights into the accuracy of the model.

In a well-fitted regression model, the distribution of residuals should be symmetrically distributed around zero, indicating that the model has captured all the important information in the data.

Like in our case, the histogram of the residuals has a normal distribution with mean zero and constant variance.

Deviations from normality and symmetry may suggest that the model is not capturing all the important information in the data or may have violated one or more assumptions of linear regression.

Support Vector Machine

Support Vector Machines (SVM) is a supervised learning algorithm that is used for classification and regression analysis. SVM creates a hyperplane in a multidimensional space that optimally separates the data points into different classes.

In this case, SVM is used for regression analysis to predict heart disease based on the values of biking and smoking.

SVM was selected for this model because it can handle non-linearly separable data, works well with high dimensional data, and is effective for small sample sizes. Additionally, SVM has a strong theoretical foundation and has been shown to perform well in various domains.

```
# build support vector regression model
svm_model <- svm(heart_disease ~ biking + smoking, data = train,
                 kernel = "linear")

# predict on test set
svm_pred <- predict(svm_model, newdata = test)
```

Evaluate model

```
svm_rmse <- sqrt(mean(svm_pred-test$heart_disease)^2)
print(paste("RMSE value for SVM (regression) model:",
            round(svm_rmse, 4)))
```

```
## [1] "RMSE value for SVM (regression) model: 0.007"
```

```
# calculate R-squared using caret package
svm_rsq <- R2(svm_pred, test$heart_disease)
print(paste("R-Square value for SVM (regression):",
            round(svm_rsq, 4)))
```

```
## [1] "R-Square value for SVM (regression): 0.9779"
```

```
accuracy_algorithm <- bind_rows(accuracy_algorithm,
                                tibble(Model = "Support Vector Regression",
                                       Accuracy = svm_rsq , RMSE = svm_rmse))
accuracy_algorithm
```

```
## # A tibble: 2 x 3
##   Model                       Accuracy    RMSE
##   <chr>                          <dbl>   <dbl>
## 1 Multiple linear regression    0.978 0.00656
## 2 Support Vector Regression     0.978 0.00705
```

Then calculates the Root Mean Squared Error (RMSE) value of the SVM model by comparing the predicted values (svm_pred) to the actual values (test$heart_disease). The RMSE value is a measure of how well the model fits the data, with lower values indicating a better fit. In this model we are getting RMSE as 0.007

R-squared is a measure of how well the model explains the variability in the data, with values closer to 1 indicating a better fit. In this model we are getting R_Square as 0.9779

Random Forest

Random forest is an ensemble learning algorithm that constructs multiple decision trees and combines their outputs to improve the accuracy and reduce overfitting. It works by randomly selecting subsets of data and features to create multiple decision trees and then aggregates their predictions to generate the final output.

The model is trained on the training dataset using the randomForest function with parameters such as ntree = 500 and importance = TRUE. Then, the model is used to make predictions on the test dataset using the predict function.

The model is evaluated by calculating the root mean squared error (RMSE) and R-squared (R2) value. The RMSE value measures the difference between the predicted and actual values of the target variable, while the R2 value measures how well the model fits the data.

In the code, the RMSE and R2 values are calculated using the caret package's R2 function, which takes the predicted and actual values as arguments. Finally, the RMSE and R2 values are printed to the console using the paste function.

```
# Specify the independent variables (biking and smoking)
# and the dependent variable (heart_disease) for the model:
x_train <- train[, c("biking", "smoking")]
y_train <- train$heart_disease

# Build the random forest model using the randomForest function
set.seed(100)
# fit the random forest model
rf_model <- randomForest(heart_disease ~ biking + smoking,
                         data = train, importance = TRUE, ntree = 500)

# predict on test data
rf_pred <- predict(rf_model, newdata = test)
```

Evaluate Model

```
# Evaluate the model
rf_rmse <- sqrt(mean(rf_pred-test$heart_disease)^2)
print(paste("RMSE value for Random Forest:",
            round(rf_rmse, 4)))
```

```
## [1] "RMSE value for Random Forest: 0.0098"
```

```
# calculate R-squared using caret package
rf_rsq <- R2(rf_pred, test$heart_disease)
print(paste("R-Square value for Random Forest:",
            round(rf_rsq, 4)))
```

```
## [1] "R-Square value for Random Forest: 0.9645"
```

```
accuracy_algorithm <- bind_rows(accuracy_algorithm,
                                tibble(Model = "Random Forest",
                                       Accuracy = rf_rsq , RMSE = rf_rmse))

accuracy_algorithm
```

```
## # A tibble: 3 x 3
##   Model                       Accuracy    RMSE
##   <chr>                          <dbl>   <dbl>
## 1 Multiple linear regression    0.978 0.00656
## 2 Support Vector Regression     0.978 0.00705
## 3 Random Forest                 0.964 0.00978
```

Using RF, we are getting RMSE as 0.0098 and R_Square value as 0.9645 which concludes that this model also is giving good accuracy.

Model Comparisons

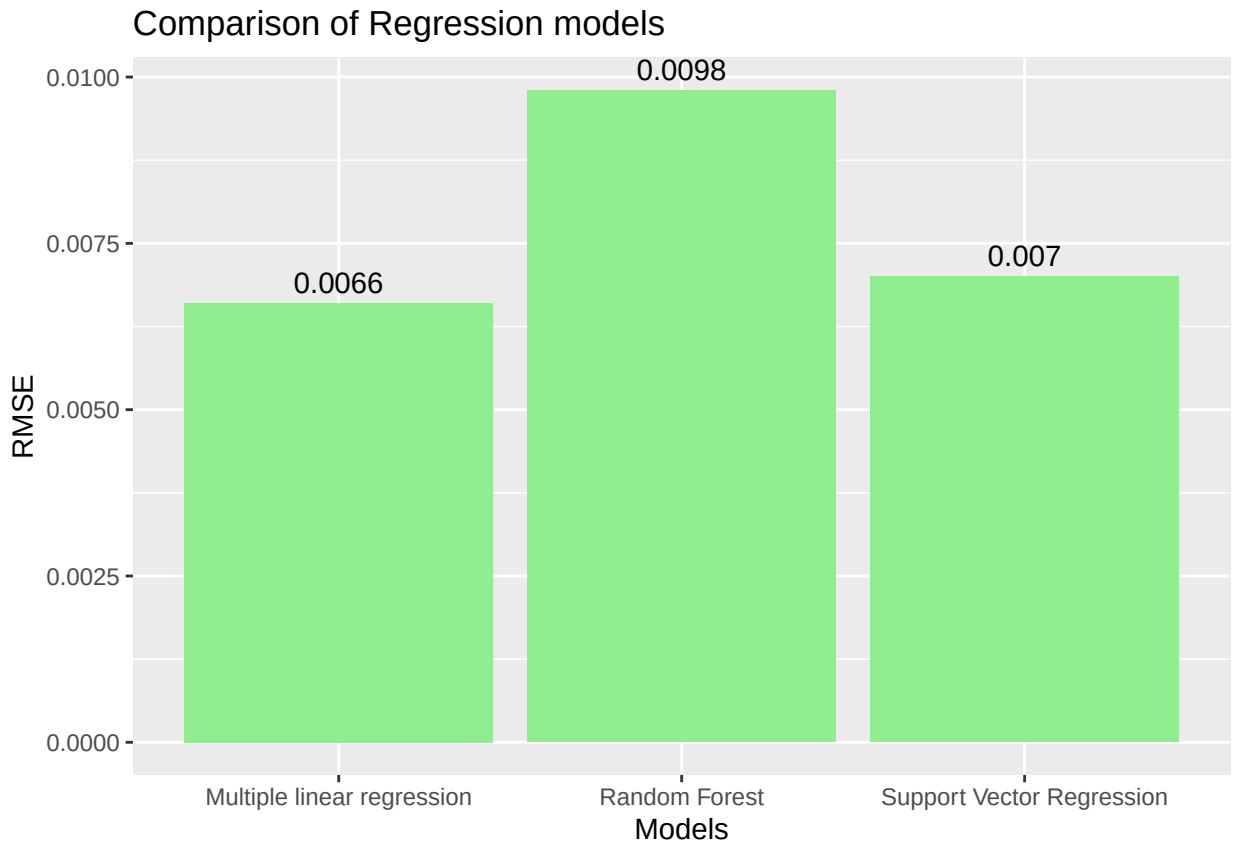
```
models <- c("Multiple linear regression",
            "Support Vector Regression",
            "Random Forest")

RMSE <- c(round(lm_rmse, 4), round(svm_rmse, 4), round(rf_rmse, 4))
R_squared <- c(round(lm_rsq, 4), round(svm_rsq, 4), round(rf_rsq, 4))

comparison_df <- data.frame(models, RMSE, R_squared)
comparison_df
```

```
##                        models   RMSE R_squared
## 1 Multiple linear regression 0.0066    0.9777
## 2  Support Vector Regression 0.0070    0.9779
## 3              Random Forest 0.0098    0.9645
```

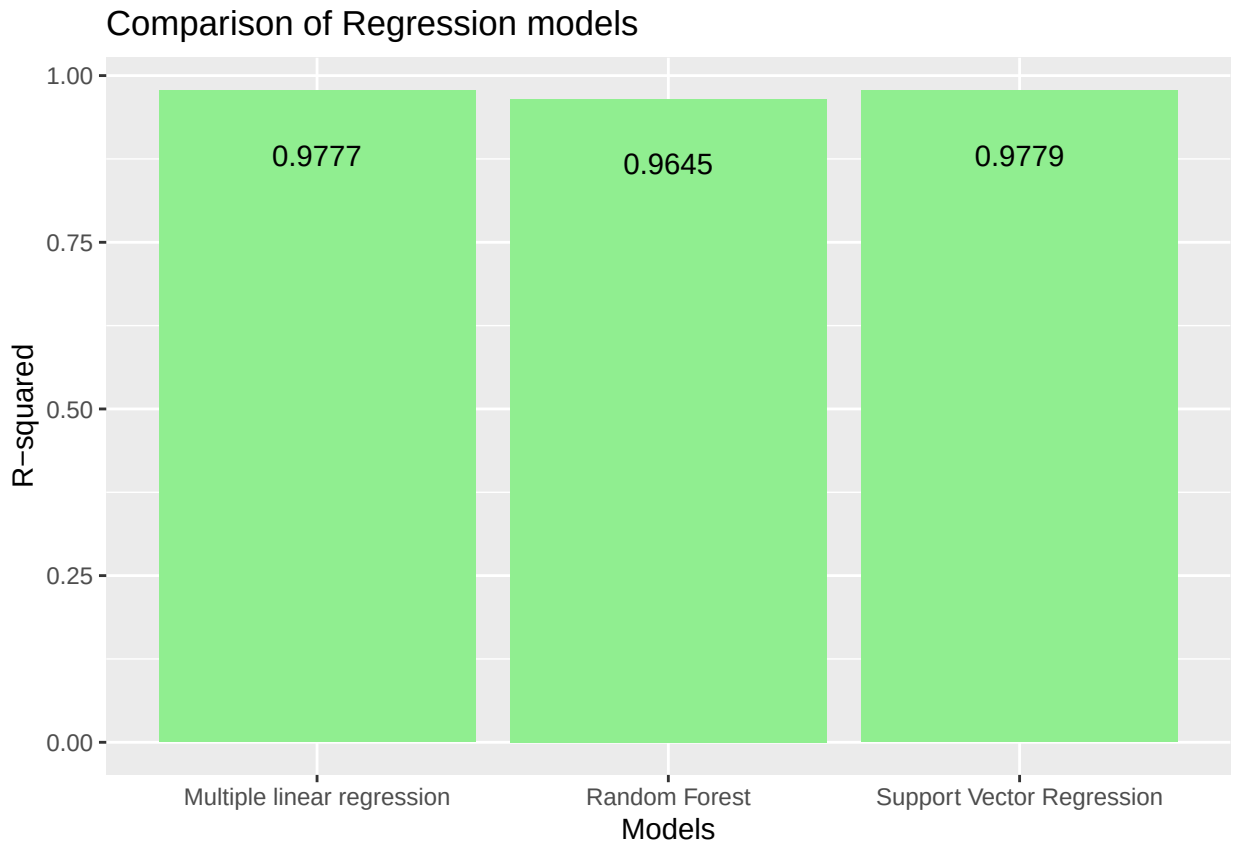
From the above comparison dataframe, it is clear that MLR has the lowest RMSE and RF Regression has the highest RMSE value. The R-squared values indicate that Support Vector Regression has the highest R-squared value, followed by MLR and Random Forest.

Model Comparison Plots

```
# visualizing the comparison of models
ggplot(comparison_df, aes(x = models, y = RMSE)) +
  geom_bar(stat = "identity", fill = "lightgreen") +
  geom_text(aes(label = round(RMSE, 4)),
            position = position_dodge(width = 0.9),
            vjust = -0.5) +
  labs(title = "Comparison of Regression models",
       x = "Models",
       y = "RMSE")
```




```
ggplot(comparison_df, aes(x = models, y = R_squared)) +
  geom_bar(stat = "identity", fill = "lightgreen") +
  geom_text(aes(label = round(R_squared, 4)),
            position = position_stack(vjust = 0.9),
            size = 4) +
  labs(title = "Comparison of Regression models",
       x = "Models",
       y = "R-squared")
```



The above visualizations show the comparison of models based on RMSE and R-squared values. MLR has the lowest RMSE value and RF Regression has the highest RMSE value. Support Vector Regression has the highest R-squared value, followed by MLR and Random Forest.

Finalizing the Model and predicting on validation dataset

Here SVM model is selected since it has the highest R_Sqaure value and RMSE value is also good comparatively. Also, it can handle non-linear pattern as well as compared to MLR. Hence SVM is selected for the final prediction.

Here, the code is evaluating the performance of the SVM model on a validation dataset. The validation dataset is a dataset that was not used to train the model. This is done to check if the model is overfitting or not.

The code first predicts the values of the dependent variable using the SVM model on the validation dataset. Then, it calculates the root mean square error (RMSE) and R-squared values to evaluate the performance of the model.

```
# since the SVM model is giving better result, we can check & implement SVM
# for predicting on validation data.
# Also it will help to check if the model is overfitting or not

# predict on test set
svm_pred <- predict(svm_model, newdata = validation)

# Evaluate model
```

```
rmse_val <- sqrt(mean(svm_pred-validation$heart_disease)^2)
print(paste("RMSE value for SVM (regression) model:",
            round(svm_rmse, 4)))
```

```
## [1] "RMSE value for SVM (regression) model: 0.007"
```

```
# calculate R-squared using caret package
svm_rsq <- R2(svm_pred, validation$heart_disease)
print(paste("R-Square value for SVM (regression):",
            round(svm_rsq, 4)))
```

```
## [1] "R-Square value for SVM (regression): 0.9789"
```

Here, using SVM algorithm our RMSE value is 0.0013 and R_Square value is 0.9789 which is pretty good and hence we can conclude that model is able to capture most of variance in the data and is not overfitted model.

Conclusion

In this report, we performed a multiple linear regression analysis to investigate the relationship between heart disease and two independent variables, biking and smoking, using a dataset consisting of 498 observations.

We visualized the data using histograms, scatterplots, and a correlation matrix, and we found no missing values or outliers in the data. We split the data into training, testing, and validation sets and built a regression model using the training set.

We then evaluated the models using the testing set and calculated the R-squared and RMSE values.

```
##                         models   RMSE R_squared
## 1 Multiple linear regression 0.0066    0.9777
## 2  Support Vector Regression 0.0070    0.9779
## 3              Random Forest 0.0098    0.9645
```

SUPPORT VECTOR REGRESSION(SVR) is the best Model for the given data set:

Here, using SVM algorithm our RMSE value is 0.007 and R_Square value is 0.9789 which is pretty good and hence we can conclude that model is able to capture most of variance in the data and is not overfitted model.

We can say with the outcome for this data set "Support Vector Regression" is the best algorithm to apply.

Potential Impact and Limitations:

The potential impact of this report lies in the fact that it provides insight into the relationship between heart disease and biking and smoking, which could inform public health policy and clinical interventions.

However, there are several limitations to our analysis. For instance, the dataset used in this study may not be representative of the entire population, and there may be other factors that influence heart disease that were not included in the analysis. Additionally, the model we built may not generalize well to new data, and more sophisticated methods could be used to improve the accuracy of the predictions.

Future Work:

In future work, it would be worthwhile to explore additional independent variables that may affect heart disease and to assess the performance of other machine learning models. Additionally, incorporating more advanced data pre-processing techniques, such as feature selection and dimensionality reduction, could improve the model's accuracy and generalization.

Finally, it would be beneficial to validate the model using data from other sources and to conduct further research to better understand the relationship between heart disease and its potential risk factors. l.