Practicum2

Anish Raju Ramakrishna Amara & Aditya Elayavalli

2023-11-08

Note: Before we start of with going through this assignment. We want to let those reading that each code chunk was created and worked on by both Aditya and Anish, the two authors of this code. Each code was worked on with the other present.

Question 1

Data Preperation

The data set we are going to obtain is from https://archive.ics.uci.edu/dataset/2/adult. The Datasets were downloaded as a kaggle file and the file paths were used. Please note to change the file path when running the code on your computer.

```
library(readr)

column_names <- c("age", "workclass", "fnlwgt", "education", "education_num",
                  "marital_status", "occupation", "relationship", "race",
                  "sex", "capital_gain", "capital_loss", "hours_per_week",
                  "native_country", "income")
# Read the adult data set with show_col_types set to FALSE
adult_data <- read.csv("/Users/anishamara/Desktop/DA5030/Practicum2/adult/adult.data",
                       stringsAsFactors = TRUE,header = F)

# Read the adult test data set with show_col_types set to FALSE
adult_test <- read.csv("/Users/anishamara/Desktop/DA5030/Practicum2/adult/adult.test",
                       #col_names = column_names,
                       skip = 1,
                       stringsAsFactor = TRUE,
                       header = F)

colnames(adult_data) <- column_names
colnames(adult_test) <- column_names
# Combine the two data sets into a single data frame
combined_df <- rbind(adult_data, adult_test)
combined_df$income <- as.factor(gsub("\\.", "", combined_df$income))
# Display specific rows and the first four columns
specified_rows <- combined_df[c(11, 112, 199, 203), 1:4]
print(specified_rows)
```

```
##     age workclass fnlwgt     education
## 11   37   Private 280464  Some-college
## 112  38   Private  65324   Prof-school
## 199  35   Private 138992       Masters
## 203  51   Private 259323     Bachelors
```

```r
# Display specific rows and the first four columns
head(combined_df[c(11, 112, 199, 203), 1:4])
```

```
##     age workclass fnlwgt     education
## 11   37   Private 280464  Some-college
## 112  38   Private  65324   Prof-school
## 199  35   Private 138992       Masters
## 203  51   Private 259323     Bachelors
```

We have loaded the dataset into R. As can be seen from the above data set, the rows 11, 112, 199 and 203 are showcased along with the first four columns. Now the below command will provide the first 10 rows of the data set. The R chunk below will take a random seed sample and afterwards we will distribute the sample with a 75:25 split, 75 is for training and 25 is for testing(validating) the models.

```r
# Set the seed for reproducibility
set.seed(33452)

# Calculate the size of the training set (75% of the combined dataset)
training_size <- round(nrow(combined_df) * 0.75)

# Generate a random sample of row indices for the training set
training_indices <- sample(nrow(combined_df), training_size,replace = F)

# Create the training dataset
training_df <- combined_df[training_indices, ]

# Create the validation dataset
validation_df <- combined_df[-training_indices, ]
```

Naive bayes Modeling The below code will run the Naive bayes model. In order to do so we have selected certain columns(aka "features") to help build a binary classifier that predicts whether an individual earns more than or less than US$50k. In order to do so we need to transform continuous variables into categorical variables by binning and eliminate any rows that contain missing values in any of the selected columns.

```r
# Load necessary libraries
library(klaR)
```

```
## Loading required package: MASS
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# Selecting the relevant columns
```

```r
selected_columns <- c(1, 2, 5, 9, 10, 13, 14, 15)

# Subsetting the dataframes to include only selected columns and removing NAs
training_df_selected <- training_df[selected_columns] %>% na.omit()
validation_df_selected <- validation_df[selected_columns] %>% na.omit()


clean_data <- function(df) {
  df[df == " ?"] <- NA
  df[df == " "] <- NA
  df[df == "?"] <- NA
  df[df == "? "] <- NA
  df <- na.omit(df)
  return(df)
}
training_df_selected <- clean_data(training_df_selected)
validation_df_selected <- clean_data(validation_df_selected)


# Binning continuous variables (age, education_num, hours_per_week)

for (col in names(training_df_selected[, sapply(training_df_selected, is.numeric)])) {
  training_df_selected[, col] <- cut(training_df_selected[[col]], breaks = 3)
}

#transform continuous variables in validation data into categorical variables by binning
for (cl in names(validation_df_selected[, sapply(validation_df_selected, is.numeric)])) {
  validation_df_selected[, cl] <- cut(validation_df_selected[[cl]], breaks = 3)
}


columns_to_convert <- c("workclass", "race", "sex", "native_country")

# Convert the specified columns to factors in the training data
for (col in columns_to_convert) {
  training_df_selected[[col]] <- as.factor(training_df_selected[[col]])
}
# Convert the specified columns to factors in the test data
for (col in columns_to_convert) {
  validation_df_selected[[col]] <- as.factor(validation_df_selected[[col]])
}

training_df_selected$income <- as.factor(training_df_selected$income)
# Building the Naive Bayes model
nb_model <- NaiveBayes(income~age+workclass+education_num+race+sex+hours_per_week+native_country, data=

# Making predictions on the validation set without showing warnings
predictions <- suppressWarnings(predict(nb_model, newdata = validation_df_selected))
```

The below code chunk produces the confusion matrix as well as the matrix in the form of a crosstable

```r
library(gmodels)
# Creating the confusion matrix
the_matrix_is_real <- table(predictions$class, validation_df_selected$income)
```

```
the_matrix_is_real
```

```
##
##           <=50K  >50K
##    <=50K   7903  1764
##    >50K     633  1011
```

```
d <- CrossTable(the_matrix_is_real,
    prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
    dnn = c('predicted', 'actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  11311
##
##
##              | actual
##    predicted |     <=50K |      >50K | Row Total |
## -------------|-----------|-----------|-----------|
##        <=50K |      7903 |      1764 |      9667 |
##              |     0.699 |     0.156 |           |
## -------------|-----------|-----------|-----------|
##         >50K |       633 |      1011 |      1644 |
##              |     0.056 |     0.089 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      8536 |      2775 |     11311 |
## -------------|-----------|-----------|-----------|
##
##
```

```
d
```

```
## $t
##
##           <=50K  >50K
##    <=50K   7903  1764
##    >50K     633  1011
##
## $prop.row
##
##              <=50K      >50K
##    <=50K 0.8175235 0.1824765
##    >50K  0.3850365 0.6149635
##
## $prop.col
##
##               <=50K       >50K
##    <=50K 0.92584349 0.63567568
```

```
##    >50K  0.07415651 0.36432432
##
## $prop.tbl
##
##               <=50K       >50K
##    <=50K 0.69870038 0.15595438
##    >50K  0.05596322 0.08938202
```

-*prop.row* : *Givestheproportionofeachpredictionwithineachpredictedclass.*−prop.col: Gives the proportion of actual classes within each prediction -$prop.tbl: Gives the overall proportion of each prediction in the total dataset.

```
true_negative <- the_matrix_is_real[1, 1]  # Correctly predicted No
true_positive <- the_matrix_is_real[2, 2]  # Correctly predicted Yes
false_positive <- the_matrix_is_real[1, 2] # Incorrectly predicted Yes
false_negative <- the_matrix_is_real[2, 1] # Incorrectly predicted No
```

The provided above output represents a confusion matrix for a classification model, along with some additional statistics. Here's a breakdown of what each part means:

Confusion Matrix: -True Negatives (TN): The model correctly predicted " <=50K" 7903. -False Positives (FP): The model incorrectly predicted " >50K" 1764 times when the actual class was " <=50K". -False Negatives (FN): The model incorrectly predicted " <=50K" 633 times when the actual class was " >50K". -True Positives (TP): The model correctly predicted " >50K" 1011 times.

```
## F1 score for Logistic Regression model: 75.37 %
```

The accuracy of the model was 0.788 and the precision was 0.818. This indicates that the model accurately predicts 78.808% correctly with a precision of 81.8%.

Logistic regression model Now we will create a Logistic regression model.

```
## [1] "age"            "workclass"      "education_num"  "race"
## [5] "sex"            "hours_per_week" "native_country" "income"
```

```
## [1] "age"            "workclass"      "education_num"  "race"
## [5] "sex"            "hours_per_week" "native_country" "income"
```

The below code chunk runs the model

```
logistic_model <- glm(income ~ ., data = training_df_selected, family = binomial)
predictions_prob <- predict(logistic_model, newdata = validation_df_selected, type = "response")
```

```
# Creating the confusion matrix
# Convert probabilities to binary predictions using a threshold of 0.5
binary_predictions <- ifelse(predictions_prob > 0.5, 1, 0)


# Now create the confusion matrix
the_matrixis_real <- table(Predicted = binary_predictions, Actual = validation_df_selected$income)

# Print the confusion matrix
the_matrixis_real
```

```
##          Actual
## Predicted  <=50K  >50K
##         0   8131  1911
##         1    405   864
```

```
d2 <- CrossTable(the_matrixis_real,
    prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
    dnn = c('predicted', 'actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  11311
##
##
##              | actual
##    predicted |     <=50K |      >50K | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |      8131 |      1911 |     10042 |
##              |     0.719 |     0.169 |           |
## -------------|-----------|-----------|-----------|
##            1 |       405 |       864 |      1269 |
##              |     0.036 |     0.076 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      8536 |      2775 |     11311 |
## -------------|-----------|-----------|-----------|
##
##
```

```
d2
```

```
## $t
##          Actual
## Predicted  <=50K  >50K
##         0   8131  1911
##         1    405   864
##
## $prop.row
##          Actual
## Predicted     <=50K      >50K
##         0 0.8096993 0.1903007
##         1 0.3191489 0.6808511
##
## $prop.col
##          Actual
## Predicted      <=50K       >50K
##         0 0.95255389 0.68864865
##         1 0.04744611 0.31135135
##
## $prop.tbl
##          Actual
## Predicted      <=50K       >50K
##         0 0.71885775 0.16895058
##         1 0.03580585 0.07638582
```

```
true_negative2 <- the_matrixis_real[1, 1]  # Correctly predicted No
true_positive2 <- the_matrixis_real[2, 2]  # Correctly predicted Yes
false_positive2 <- the_matrixis_real[1, 2] # Incorrectly predicted Yes
false_negative2 <- the_matrixis_real[2, 1] # Incorrectly predicted No

accuracy2 <- (true_positive + true_negative) / (true_negative+true_positive+false_positive+false_negati

precision_NB2<-round(d2$t[1]/ sum(d2$t[3],d2$t[1]),3)
recall2 <- true_negative2 / (true_negative2+true_positive2+false_positive2+false_negative2)
f1_precision_NB2 <- 2 * (precision_NB2 * recall2) / (precision_NB2 + recall2)
cat("F1 score for Logistic Regression model:", round(f1_precision_NB2 * 100, 2), "%\n")
```

```
## F1 score for Logistic Regression model: 76.17 %
```

The provided above output represents a confusion matrix for a classification model, along with some additional statistics. Here's a breakdown of what each part means:

Confusion Matrix: -True Negatives (TN): The model correctly predicted " <=50K" 8131. -False Positives (FP): The model incorrectly predicted " >50K" 1911 times when the actual class was " <=50K". -False Negatives (FN): The model incorrectly predicted " <=50K" 405 times when the actual class was " >50K". -True Positives (TP): The model correctly predicted " >50K" 864 times.

According to the above matrix result, the model was had an accuracy of 0.788 or 78.808% and a precision of 0.81 or 0.81%

Decision Tree The below code will aid in building a Decision Tree model that predicts whether an individual earns more than or less than US$50k

```
library(rpart)

decision_tree_model <- rpart(income ~ ., data = training_df_selected, method = "class")

# Make predictions on the validation set
tree_predictions <- predict(decision_tree_model, newdata = validation_df_selected, type = "class")

# Create a confusion matrix to evaluate the model
confusion_matrix_tree <- table(Predicted = tree_predictions, Actual = validation_df_selected$income)

# Print the confusion matrix
confusion_matrix_tree
```

```
##          Actual
## Predicted  <=50K  >50K
##     <=50K   8194  2037
##     >50K     342   738
```

```
d3 <- CrossTable(confusion_matrix_tree,
    prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
    dnn = c('predicted', 'actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
```

```
##
##
## Total Observations in Table:  11311
##
##
##              | actual
##    predicted |     <=50K |      >50K | Row Total |
## -------------|-----------|-----------|-----------|
##        <=50K |      8194 |      2037 |     10231 |
##              |     0.724 |     0.180 |           |
## -------------|-----------|-----------|-----------|
##         >50K |       342 |       738 |      1080 |
##              |     0.030 |     0.065 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      8536 |      2775 |     11311 |
## -------------|-----------|-----------|-----------|
##
##
```

```
d3
```

```
## $t
##          Actual
## Predicted  <=50K  >50K
##     <=50K   8194  2037
##     >50K     342   738
##
## $prop.row
##          Actual
## Predicted     <=50K      >50K
##     <=50K 0.8008992 0.1991008
##     >50K  0.3166667 0.6833333
##
## $prop.col
##          Actual
## Predicted     <=50K      >50K
##     <=50K 0.9599344 0.7340541
##     >50K  0.0400656 0.2659459
##
## $prop.tbl
##          Actual
## Predicted      <=50K       >50K
##     <=50K 0.72442755 0.18009018
##     >50K  0.03023605 0.06524622
```

```
true_negative3 <- confusion_matrix_tree[1, 1]  # Correctly predicted No
true_positive3 <- confusion_matrix_tree[2, 2]  # Correctly predicted Yes
false_positive3 <- confusion_matrix_tree[1, 2] # Incorrectly predicted Yes
false_negative3 <- confusion_matrix_tree[2, 1] # Incorrectly predicted No

accuracy3 <- (true_positive3 + true_negative3) / (true_negative3+true_positive3+false_positive3+false_ne

precision_NB3<-round(d3$t[1]/ sum(d3$t[3],d3$t[1]),3)

recall3 <- true_negative3 / (true_negative3+true_positive3+false_positive3+false_negative3)
```

```r
f1_dec <- 2 * (precision_NB3 * recall3) / (precision_NB3 + recall3)
cat("F1 score for Decision tree model:", round(f1_dec * 100, 2), "%\n")
```

```
## F1 score for Decision tree model: 76.08 %
```

The provided above output represents a confusion matrix for a classification model, along with some additional statistics. Here's a breakdown of what each part means:

Confusion Matrix: -True Negatives (TN): The model correctly predicted " <=50K" 8194. -False Positives (FP): The model incorrectly predicted " >50K" 2037 times when the actual class was " <=50K". -False Negatives (FN): The model incorrectly predicted " <=50K" 342 times when the actual class was " >50K". -True Positives (TP): The model correctly predicted " >50K" 738 times.

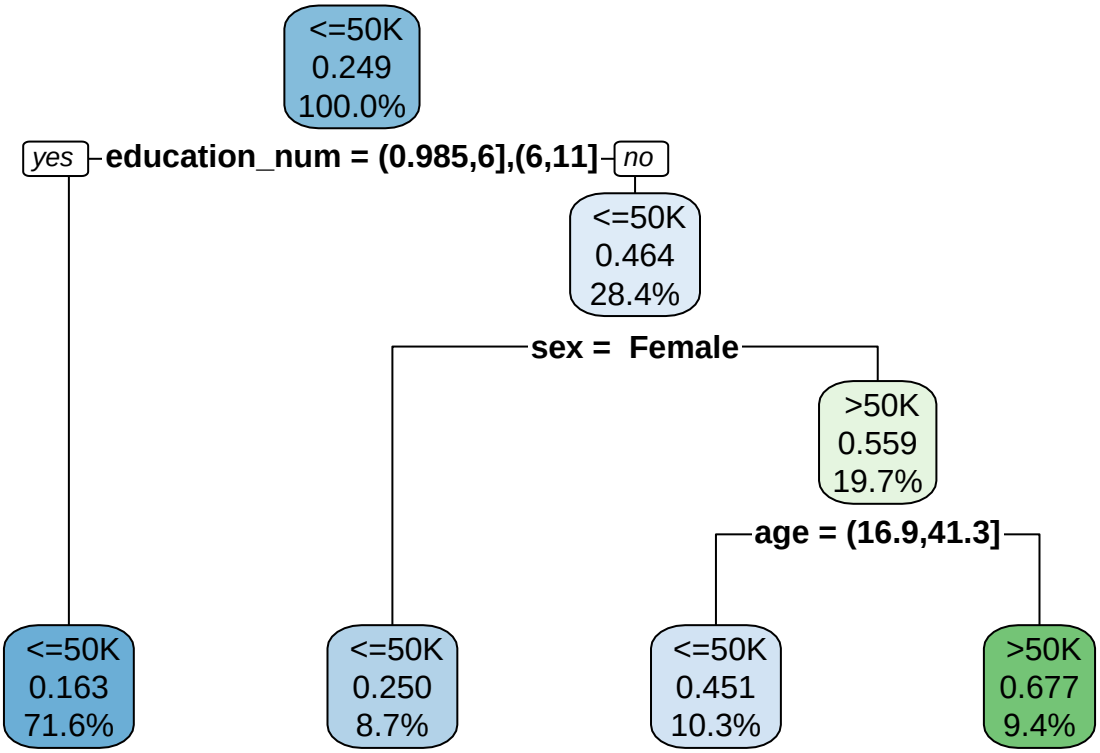
According to the above matrix result, the model was had an accuracy of 0.79 or 78.967% and a precision of 0.801 or 0.801%

```r
decision_tree_model
```

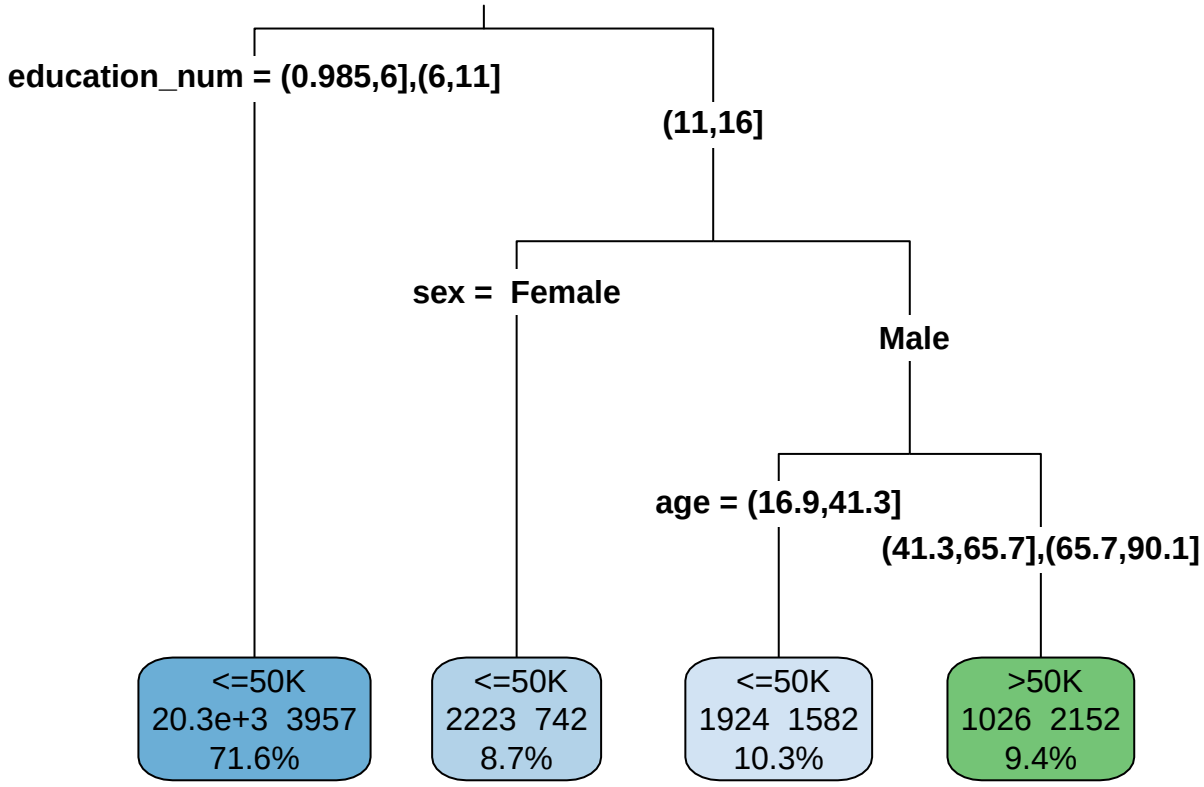
```
## n= 33922
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 33922 8433  <=50K (0.7514003 0.2485997)
##    2) education_num=(0.985,6],(6,11] 24273 3957  <=50K (0.8369794 0.1630206) *
##    3) education_num=(11,16] 9649 4476  <=50K (0.5361177 0.4638823)
##      6) sex= Female 2965  742  <=50K (0.7497470 0.2502530) *
##      7) sex= Male 6684 2950  >50K (0.4413525 0.5586475)
##       14) age=(16.9,41.3] 3506 1582  <=50K (0.5487735 0.4512265) *
##       15) age=(41.3,65.7],(65.7,90.1] 3178 1026  >50K (0.3228446 0.6771554) *
```

The above code produces the nodes and terminal nodes used for the decision tree. The code below will produce a plot for the decision tree

```r
library(rpart.plot)
rpart.plot(decision_tree_model, digits = 3)
```




```
rpart.plot(decision_tree_model, digits = 3, fallen.leaves = TRUE,
               type = 3, extra = 101)
```



Predicting

The below code junk will help is predict an individuals income via a function we built predictEarningsClass().

```
predictEarningsClass <- function(data) {
    # Assuming logistic_model, decision_tree_model, nb_model are already trained and available

    # Predict using logistic regression
    predictions_prob <- predict(logistic_model, newdata = data, type = "response")

    # Predict using decision tree
    tree_predictions <- predict(decision_tree_model, newdata = data, type = "class")

    # Predict using Naive Bayes
    predictions_nb <- suppressWarnings(predict(nb_model, newdata = data, type = "class")$class)

    # Combine individual predictions
    combined_predictions <- cbind(as.numeric(predictions_prob > 0.5), as.numeric(tree_predictions), as.n

    # Ensemble prediction using majority voting
    ensembl_pred <- apply(combined_predictions, 1, function(row) {
        ifelse(sum(row) >= 2, 1, 0)
    })

    return(ensembl_pred)
}
```

```r
# Prepare the individual data for prediction
individual_data <- data.frame(
  age = 38,
  workclass = " Private",
  education_num = 13,
  race = " Black",
  sex = " Female",
  hours_per_week = 40,
  native_country = " Peru",
  income = NA
)



#individual_data$age <- as.integer(factor(training_df_selected2$age))
#individual_data$workclass <- as.integer(factor(training_df_selected2$workclass))

#individual_data
#head(training_df_selected,10)

# Binning age, education_num, and hours_per_week

individual_data$age <- cut(individual_data$age, breaks = c(16.9, 41.3, 65.7, 90.1), labels = c("(16.9,4


individual_data$education_num <- cut(individual_data$education_num, breaks = c(0.985, 6, 11, 16), labels

individual_data$hours_per_week <- cut(individual_data$hours_per_week, breaks = c(0.902, 33.7, 66.3, 99.


# List of column names to convert to factors for new data
columns_to_convert <- c("age", "education_num", "hours_per_week", "workclass", "race", "sex","native_cou


# Convert the specified columns to factors in the test data
for (col in columns_to_convert) {
  individual_data[[col]] <- as.factor(individual_data[[col]])
}

head(training_df_selected,10)
```

```
##               age         workclass education_num   race     sex hours_per_week
## 42045 (16.9,41.3]         State-gov       (11,16]  White    Male    (33.7,66.3]
## 32854 (16.9,41.3]           Private        (6,11]  White    Male    (33.7,66.3]
## 38009 (41.3,65.7]  Self-emp-not-inc        (6,11]  White    Male    (33.7,66.3]
## 45945 (65.7,90.1]         Local-gov     (0.985,6]  White  Female   (0.902,33.7]
## 40966 (16.9,41.3]         Local-gov       (11,16]  White    Male   (0.902,33.7]
## 15281 (16.9,41.3]           Private       (11,16]  White    Male    (33.7,66.3]
## 18939 (41.3,65.7]       Federal-gov        (6,11]  White    Male    (33.7,66.3]
## 10897 (16.9,41.3]           Private       (11,16]  White    Male    (33.7,66.3]
## 25752 (16.9,41.3]         Local-gov       (11,16]  White  Female    (33.7,66.3]
## 42363 (16.9,41.3]           Private        (6,11]  Black  Female    (33.7,66.3]
```

```
##       native_country income
## 42045  United-States   >50K
## 32854  United-States  <=50K
## 38009  United-States   >50K
## 45945  United-States  <=50K
## 40966        Germany  <=50K
## 15281  United-States  <=50K
## 18939  United-States   >50K
## 10897  United-States   >50K
## 25752  United-States   >50K
## 42363  United-States  <=50K
```

```
#individual_data
# Predict income class using the ensemble model
individual_prediction <- predictEarningsClass(individual_data)
#cat("Predicted income class:", ifelse(individual_prediction == 1, ">50K", "<=50K"), "\n")
```

According to the above code a 38-year-old black female adult who is privately employed, has 13 years of education, and who immigrated from Peru earns >50K

Model ensemble accuracy

The below code chunk will evaluate the F1-Score for the ensemble

```
# Predict income class using the ensemble model
final_predictions <- predictEarningsClass(individual_data)
cat("Predicted income class:", ifelse(final_predictions == 1, ">50K", "<=50K"), "\n")
```

```
## Predicted income class: >50K
```

```
validation_df_prediction <- predictEarningsClass(validation_df_selected)
# Create a confusion matrix
confusion_matrix2 <- table(Predicted = validation_df_prediction, Actual = validation_df_selected$income)
print(confusion_matrix2)
```

```
##          Actual
## Predicted  <=50K  >50K
##         1   8536  2775
```

```
# Calculate accuracy
accuracy4 <- sum(diag(confusion_matrix2)) / sum(confusion_matrix2)

# Calculate precision
precise <- confusion_matrix2[1,1] / sum(confusion_matrix2[1,])

# Calculating recall for ensemble model
ensemblerecall <- confusion_matrix2[1,1] / sum(confusion_matrix2[, 1 ])

# Calculating F1 score for ensemble
ensemble_f1_score <- 2 *(precise * ensemblerecall) / (precise + ensemblerecall)
```

F1 score that is produced from the Ensemble model is 86.02

Best Model

The following chunk of code helps us predict the best model.

```r
# Calculate the F1-Score for the ensemble from (11) using the validation data. How does its performance

f1 <- c(
  ensemble = ensemble_f1_score,
  NaiveBayes = f1_precision_NB,
  DecisionTree = f1_dec,
  LogisticRegression = f1_precision_NB2
)
best_model <-which.max(f1)

the_end_is_near <- names(f1)[best_model]
the_end_is_near
```

```
## [1] "ensemble"
```

```r
score <- f1[best_model]
# Print the F1-Scores and compare the models
cat("F1-Scores:\n")
```

```
## F1-Scores:
```

```r
cat(paste(names(f1), ": ", round(f1, 4), "\n"))
```

```
## ensemble :  0.8602
##  NaiveBayes :  0.7537
##  DecisionTree :  0.7608
##  LogisticRegression :  0.7617
```

```r
cat("Best Model: ", the_end_is_near, " (F1-Score: ", round(f1[the_end_is_near], 4),")")
```

```
## Best Model:  ensemble  (F1-Score:  0.8602 )
```

The best scoring model is ensemble with an F1-Score of 0.8602. Higher the F1 score, the greater the accuracy and thus we should use this model to predict.

Problem 2

Reading the data and identification of Outliers

Identify the outliers and the load the data to understand the dataset.

```r
energy.df <- read.csv(file = "/Users/anishamara/Desktop/DA5030/Practicum2/large-scale+wave+energy+farm/
                      header = TRUE, stringsAsFactors = FALSE)

ots <- function(cols){
  col_m <- mean(cols)
  col_sd <- sd(cols)
  z_score <- (cols - col_m) / col_sd
  return(z_score)
}

energy.no.df <- energy.df

# Get the column names from energy.df
columns <- colnames(energy.df)
outliers <- numeric(0)
```

```r
for (col in columns) {
  column_data <- as.numeric(energy.df[[col]])
  z_score <- ots(column_data)

  # Identify rows with outliers
  outlier_indices <- which(abs(z_score) > 3)
  # Print out the outliers if there are any outliers in the column
  if (length(outlier_indices) > 0 ){
    print((paste0("The outliers in the column, ", col," which are at positions ",paste(outlier_indices,
  }
  # Concatenate the outliers
  outliers <- c(outliers,outlier_indices)
}
```

```
## [1] "The outliers in the column, Y1 which are at positions 3438 , 3440 , 3445 , 3447 , 3449 , 3451 ,
## [1] "The outliers in the column, Y2 which are at positions 1174 , 1175 , 1176 , 1177 , 1178 , 1179 ,
## [1] "The outliers in the column, Y3 which are at positions 3450 , 3452 , 3453 , 3455 , 3456 , 3457 ,
## [1] "The outliers in the column, Y4 which are at positions 3452 , 3453 , 3454 , 3456 , 3457 , 3458 ,
## [1] "The outliers in the column, Y5 which are at positions 3450 , 3451 , 3452 , 3453 , 3454 , 3456 ,
## [1] "The outliers in the column, Y6 which are at positions 3451 , 3454 , 3456 , 3457 , 3493 , 3495 ,
## [1] "The outliers in the column, Y7 which are at positions 6 , 3451 , 3453 , 3455 , 3458 , 3491 , 349
## [1] "The outliers in the column, Y8 which are at positions 3439 , 3448 , 3451 , 3454 , 3489 , 3490 ,
## [1] "The outliers in the column, Y9 which are at positions 1674 , 1686 , 1712 , 1716 , 1718 , 1724 ,
## [1] "The outliers in the column, Y10 which are at positions 3453 , 3454 , 3456 , 3457 , 3458 , 3489
## [1] "The outliers in the column, Y11 which are at positions 3451 , 3454 , 3455 , 3457 , 3489 , 3491
## [1] "The outliers in the column, Y12 which are at positions 3458 , 3493 , 3494 , 24577 , 24588 , 245
## [1] "The outliers in the column, Y13 which are at positions 3452 , 3454 , 3456 , 3457 , 3458 , 3494
## [1] "The outliers in the column, Y14 which are at positions 3450 , 3491 , 3494 , 3495 , 26562 , 26574
## [1] "The outliers in the column, Y15 which are at positions 3437 , 3449 , 3450 , 3453 , 3454 , 3456
## [1] "The outliers in the column, Y16 which are at positions 3449 , 3455 , 3458 , 3489 , 3490 , 3493
## [1] "The outliers in the column, Y17 which are at positions 3452 , 3489 , 3490 , 9676 , 9677 , 9678
## [1] "The outliers in the column, Y18 which are at positions 1652 , 1664 , 3450 , 3454 , 3489 , 3491
## [1] "The outliers in the column, Y19 which are at positions 3435 , 3449 , 3451 , 3453 , 3457 , 3489
## [1] "The outliers in the column, Y20 which are at positions 3439 , 3448 , 3450 , 3454 , 3457 , 3489
## [1] "The outliers in the column, Y21 which are at positions 1094 , 1095 , 1096 , 1097 , 1098 , 1099
## [1] "The outliers in the column, Y22 which are at positions 3449 , 3451 , 3452 , 3455 , 3489 , 3490
## [1] "The outliers in the column, Y23 which are at positions 3454 , 3457 , 3492 , 3494 , 3495 , 4538
## [1] "The outliers in the column, Y24 which are at positions 3453 , 3458 , 3489 , 3491 , 3493 , 3494
## [1] "The outliers in the column, Y25 which are at positions 3455 , 3457 , 3489 , 3491 , 3493 , 3494
## [1] "The outliers in the column, Y26 which are at positions 3450 , 3452 , 3453 , 3458 , 3490 , 3492
## [1] "The outliers in the column, Y27 which are at positions 3450 , 3452 , 3453 , 3456 , 3458 , 3489
## [1] "The outliers in the column, Y28 which are at positions 3442 , 3449 , 3454 , 3456 , 3457 , 3490
## [1] "The outliers in the column, Y29 which are at positions 3449 , 3450 , 3453 , 3456 , 3457 , 3489
## [1] "The outliers in the column, Y30 which are at positions 3449 , 3451 , 3452 , 3455 , 3456 , 3492
## [1] "The outliers in the column, Y31 which are at positions 3451 , 3452 , 3454 , 3455 , 3456 , 3458
## [1] "The outliers in the column, Y32 which are at positions 3449 , 3456 , 3457 , 3490 , 3492 , 3493
## [1] "The outliers in the column, Y33 which are at positions 1037 , 1038 , 1039 , 1040 , 1041 , 1042
## [1] "The outliers in the column, Y34 which are at positions 3442 , 3450 , 3454 , 3455 , 3457 , 3458
## [1] "The outliers in the column, Y35 which are at positions 3446 , 3455 , 3456 , 3492 , 3493 , 3495
## [1] "The outliers in the column, Y36 which are at positions 3449 , 3453 , 3455 , 3457 , 3491 , 3492
## [1] "The outliers in the column, Y37 which are at positions 3450 , 3495 , 20863 , 20871 , 20874 , 208
## [1] "The outliers in the column, Y38 which are at positions 3451 , 3454 , 3456 , 3493 , 3494 , 20863
## [1] "The outliers in the column, Y39 which are at positions 3451 , 3456 , 3457 , 3458 , 3489 , 3494
## [1] "The outliers in the column, Y40 which are at positions 3452 , 3457 , 3491 , 3492 , 20843 , 20863
```

```
## [1] "The outliers in the column, Y41 which are at positions 3445 , 3449 , 3453 , 3457 , 3494 , 3495
## [1] "The outliers in the column, Y42 which are at positions 3440 , 3450 , 3453 , 3455 , 3456 , 3489
## [1] "The outliers in the column, Y43 which are at positions 3451 , 3453 , 3454 , 3455 , 3456 , 3492
## [1] "The outliers in the column, Y44 which are at positions 3451 , 3453 , 3456 , 3491 , 3493 , 3513
## [1] "The outliers in the column, Y45 which are at positions 3454 , 3455 , 3456 , 3492 , 3493 , 3494
## [1] "The outliers in the column, Y46 which are at positions 3449 , 3450 , 3454 , 3457 , 3458 , 3490
## [1] "The outliers in the column, Y47 which are at positions 1572 , 1579 , 1584 , 1585 , 1591 , 1596
## [1] "The outliers in the column, Y48 which are at positions 1045 , 1046 , 1047 , 1048 , 1049 , 1050
## [1] "The outliers in the column, Y49 which are at positions 1795 , 1796 , 1797 , 1798 , 1799 , 1800
## [1] "The outliers in the column, Power1 which are at positions 3515 , 12459 , 12471 , 13103 , 13117
## [1] "The outliers in the column, Power2 which are at positions 43 , 828 , 3455 , 3476 , 3512 , 3515
## [1] "The outliers in the column, Power3 which are at positions 80 , 138 , 157 , 864 , 3455 , 3459 , 3
## [1] "The outliers in the column, Power4 which are at positions 3454 , 3457 , 3478 , 3486 , 3516 , 120
## [1] "The outliers in the column, Power31 which are at positions 13105 , 13108 , 13120 , 15370 , 15378
## [1] "The outliers in the column, Power32 which are at positions 13105 , 13120 , 15553 , 33328 , 3341
## [1] "The outliers in the column, Power34 which are at positions 12783 , 14555 , 14567 , 14577 , 1458
## [1] "The outliers in the column, Power35 which are at positions 12597 , 15515 , 17392 , 17571"
## [1] "The outliers in the column, Power37 which are at positions 3450 , 17499 , 17521"
## [1] "The outliers in the column, Power38 which are at positions 303 , 926 , 3454 , 3456 , 3469 , 1263
## [1] "The outliers in the column, Power39 which are at positions 3451 , 3457 , 3458 , 3495 , 12823 , 1
## [1] "The outliers in the column, Power40 which are at positions 12897 , 12911 , 14048 , 14895 , 1970
## [1] "The outliers in the column, Power41 which are at positions 198 , 3494 , 3495 , 12913 , 12916 , 1
## [1] "The outliers in the column, Power42 which are at positions 198 , 279 , 286 , 495 , 729 , 738 , 8
## [1] "The outliers in the column, Power43 which are at positions 19 , 25 , 26 , 57 , 65 , 126 , 137 ,
## [1] "The outliers in the column, Power44 which are at positions 54 , 75 , 106 , 117 , 123 , 127 , 128
## [1] "The outliers in the column, Power45 which are at positions 20 , 38 , 52 , 54 , 75 , 80 , 83 , 21
## [1] "The outliers in the column, Power46 which are at positions 25 , 26 , 27 , 33 , 34 , 38 , 52 , 59
## [1] "The outliers in the column, Power47 which are at positions 17 , 18 , 29 , 32 , 33 , 35 , 38 , 39
## [1] "The outliers in the column, Power48 which are at positions 25 , 37 , 49 , 63 , 68 , 76 , 78 , 88
## [1] "The outliers in the column, Power49 which are at positions 1795 , 1796 , 1797 , 1798 , 1799 , 18
## [1] "The outliers in the column, qW which are at positions 24 , 59 , 12425 , 12442 , 12456 , 13094 ,
## [1] "The outliers in the column, Total_Power which are at positions 24 , 48 , 59 , 12425 , 12442 , 1
```

```r
# Get a unique list of outliers
outliers <- unique(outliers)
# Remove the indices present in outliers
energy.no.df <- energy.no.df[-outliers,]
```

Random sampling and testing for if the columns follow a normal distribution

```r
# Take a sample of 5000 rows to create a smaller dataset and check the normality distribution on that
samp_subset <- energy.no.df[sample(nrow(energy.no.df),5000,replace = FALSE),]
cols <- colnames(samp_subset)

# Apply the shapiro-wilk test on every column
for (col in cols){
  shap <- shapiro.test(samp_subset[,col])
  a <- shap$p.value
  if (a > 0.05){
    print(paste0("The column ",col," does not follow a normal distribution"))
  }
}
```

From the shapiro-wilk test all the p-values are statistically significant which means that we reject the null

hypothesis that the dataset follows a normal distribution. In other words, the dataset does not follow a normal distribution.

Performing various transforms to see if the data follows a normal distribution

Log Transform We perform the log transformation on the datasets and thus check if the columns follow a normal distribution.

```
cols <- colnames(energy.no.df)
# Apply log transform to every dataframe
energy.tx1 <- as.data.frame(lapply(energy.no.df, log))

# Check if the values follow a normal distribution
samp_subset1 <- energy.tx1[sample(nrow(energy.no.df),5000,replace = FALSE),]
cols <- colnames(samp_subset1)

# Apply shapiro-wilk to every column
for (col in cols) {
  shap <- shapiro.test(samp_subset1[, col])
  a <- shap$p.value
  if (!is.na(a) && !is.nan(a) && a > 0.05) {
    print(paste0("The column ", col, " follows normal distribution"))
  }
}
```

We can conclude that there are no columns that follow a normal distribution after performing a log transform

Square root Transform We perform the square-root transformation on the datasets and thus check if the columns follow a normal distribution.

```
cols <- colnames(energy.no.df)

# Apply square root transform to every dataframe
energy.tx2 <- as.data.frame(lapply(energy.no.df, sqrt))

# Check if the values follow a normal distribution
samp_subset2 <- energy.tx2[sample(nrow(energy.no.df),5000,replace = FALSE),]
cols <- colnames(samp_subset2)

# Apply shapiro-wilk to every column
for (col in cols) {
  shap <- shapiro.test(samp_subset2[, col])
  #print(shap)
  a <- shap$p.value
  if (!is.na(a) && !is.nan(a) && a > 0.05) {
    print(paste0("The column ", col, " follows normal distribution"))
  }
}
```

We can say that there are no columns that follow a normal distribution after performing a square root transform on the data.

Inverse transformation We perform the inverse transformation on the datasets and thus check if the columns follow a normal distribution.

```r
cols <- colnames(energy.no.df)
# Apply inverse transform to every dataframe
energy.tx3 <- as.data.frame(lapply(energy.no.df, function(x) 1/x))

# Check if the values follow a normal distribution
samp_subset3 <- energy.tx3[sample(nrow(energy.no.df),5000,replace = FALSE),]
cols <- colnames(samp_subset2)

# Apply shapiro-wilk to every column
for (col in cols) {
  shap <- shapiro.test(samp_subset3[, col])
  a <- shap$p.value
  if (!is.na(a) && !is.nan(a) && a > 0.05) {
    print(paste0("The column ", col, " follows normal distribution"))
  }
}
```

We can say that there are no columns that follow a normal distribution after performing a inverse transform on the data.

Correlation to the response variable

Correlation with each variable was given to understand how each variable is related to the target variable and in this case the Total_Power. It prints out the variables that have a strong positive or negative correlation.

```r
# Check for the correlation of Total Power with each column and print it out if the correlation is grea
for (col in cols){
  corelation <- cor.test(energy.no.df$Total_Power,energy.no.df[,col])
  if (abs(corelation$estimate) > 0.6 && col != "Total_Power"){
    print(paste0("The column ",col," has a correlation estimate of ",abs(corelation$estimate)))

  }
}
```

```
## [1] "The column Y4 has a correlation estimate of 0.623467087300243"
## [1] "The column Y45 has a correlation estimate of 0.675859385306966"
## [1] "The column Y46 has a correlation estimate of 0.63568800435773"
## [1] "The column qW has a correlation estimate of 0.994004080547564"
```

Splitting the datasets and building multiple regression models on Energy.df and Energy.no.df

The dataset was split using the caret package into 70% training and 30% validation data and thus generating different training and testing datasets.

```r
library(caret)
```

```
## Loading required package: ggplot2
## Loading required package: lattice
```

```r
set.seed(1324)

# Split the dataset into 70% training and 30% testing for both the datasets with outliers and without o
samp1 <- createDataPartition(energy.df$Total_Power,p=0.7, list = FALSE, times =1)
energy.df.training <- energy.df[samp1,]
energy.df.testing <- energy.df[-samp1,]
```

```r
set.seed(1324)
samp2 <- createDataPartition(energy.no.df$Total_Power,p=0.7, list = FALSE, times =1)
energy.no.df.training <- energy.no.df[samp2,]
energy.no.df.testing <- energy.no.df[-samp2,]
```

Backward Elimination

```r
# Automating the backward elimination process using loops
backward_elimination <- function(dataset, target, significance = 0.05) {

  # Generate the initial model
  initial_model <- lm(target ~ ., data = dataset)
  # Get the max p_value
  max_p <- max(summary(initial_model)$coefficients[, 4])
  # When the max p_value is greater than significance go on eliminating the columns one by one
  while (max_p > significance) {
    max_p_index <- which.max(summary(initial_model)$coefficients[, 4])
    rem <- names(max_p_index)
    dataset <- dataset[, -which(names(dataset) %in% rem), drop = FALSE]
    new_model <- lm(target ~ ., data = dataset)
    max_p <- max(summary(new_model)$coefficients[, 4])
    initial_model <- new_model
  }
  return(initial_model)
}



## Applying the elimination to all the data frames
# 1. Applying to the model with no outliers
processed_model <- backward_elimination(energy.no.df.training,energy.no.df.training$Total_Power)
#summary(processed_model)

# 2. Applying to the raw dataset
raw_model <- backward_elimination(energy.df.training,energy.df.training$Total_Power)
#summary(raw_model)
```

Analysis

```r
# Predict the values and calculate RMSE and adjusted R-squared value
no_ots <- predict(processed_model,energy.no.df.testing[,-energy.no.df.testing$Total_Power])
sqerr_1 <- (residuals(processed_model)^2)
rmse_1 <- sqrt(mean(sqerr_1))
r1 <- summary(processed_model)$adj.r.squared

# Predict the values and calculate RMSE and adjusted R-squared value
with_ots <- predict(raw_model,energy.df.testing[,-energy.df.testing$Total_Power])
sqerr_2 <- (residuals(raw_model)^2)
rmse_2 <- sqrt(mean(sqerr_2))
r2 <- summary(raw_model)$adj.r.squared
```

Making the predictions of the total power using the respective models From the analysis, we can see that both the models are very good because of the low RMSE. The model with the outliers has an RMSE of $3.1254116 \times 10^{-12}$ and the model without the outliers has a value of $1.3292119 \times 10^{-11}$. Since the RMSEs are very close to 0, we can say that both the models are very good but the model without the outliers is better because it has a smaller RMSE value. The adjusted R-squared value for the the model with outliers is 1 and that for the model without the outliers 1.

This shows that the model can predict better when there is a good amount of data. If we have to compare the two models the one without the outliers has a slight edge over the one with the outliers. The adjusted R-squared value close to 1 means that the points are being predicted very close to the actual values with minor differences in decimal points which is mostly insignificant.

We can also say that the model without the outliers is better because in general we remove whatever has a greater variance from the mean and this can be clearly seen in the values of adjusted R-squared and the RMSE values. The errors are also lower when the models have their outliers removed. Through this, we can conclude that the model which has no outliers is better than the raw dataset.

Calculating the confidence intervals

The confidence intervals are same as the predicted values in this case because the standard residual error for the models is very small and the second term will be close to zero and this can be seen in the resulting columns as well.

```
energy.df.testing$PredictedVals <- with_ots
energy.no.df.testing$PredictedVals <- no_ots

energy.no.df.testing$CI_Lower <- energy.no.df.testing$PredictedVals - 1.96*3.138e-12
energy.no.df.testing$CI_Upper <- energy.no.df.testing$PredictedVals + 1.96*3.138e-12

energy.df.testing$CI_Lower <- energy.df.testing$PredictedVals - 1.96*1.333e-11
energy.df.testing$CI_Upper <- energy.df.testing$PredictedVals + 1.96*1.333e-11

head(energy.df.testing[,149:152],5)
```

```
##    Total_Power PredictedVals CI_Lower CI_Upper
## 3      4103680       4103680  4103680  4103680
## 4      4105661       4105661  4105661  4105661
## 5      3752649       3752649  3752649  3752649
## 6      3820015       3820015  3820015  3820015
## 11     4053880       4053880  4053880  4053880
```

```
head(energy.no.df.testing[,149:152],5)
```

```
##    Total_Power PredictedVals CI_Lower CI_Upper
## 1      4102461       4102461  4102461  4102461
## 5      3752649       3752649  3752649  3752649
## 8      3993212       3993212  3993212  3993212
## 9      4037155       4037155  4037155  4037155
## 12     4082172       4082172  4082172  4082172
```