Binary Prediction Of Income Level Based on United States Census Data

Nilay Tripathi, Katharina Dowlin, HaoYang Guan, Rajeev Atla

December 7, 2023

Contents

1 Introductory Description/Data Cleaning

The data set adult from UC Irvine's Machine Learning Repository in 1996 was extracted from the 1994 Census database, with a total of 15 variables to predict whether an individual's annual income exceeds $50k

The 15 variables were respectively: `age, workclass, fnlwgt, education, educationnum, martialstatus, occupation, relationship, race, sex, capitalgain, capitalloss, hoursperweek, nativecountry, 50k`

First, we removed variables `capitalgain, capitalloss` simply due to most observations having missing values for them, then we removed variables `fnlwgt` as we note here, `fnlwgt` being heavily determined by `age` and `sex` will produce high multicollinearity. We removed `education` for the same reason, as it's

simply categorical representations of `educationnum`. We then removed `relationship` as it also causes multicollinearity with `sex`, as husband/wife would be associated with male/female. We removed `race` as its categories were too broad for us to expect significant findings. We lastly removed `nativecountry` simply because there are too many different categorical responses, with some holding so few observations while the variable itself has many missing values.

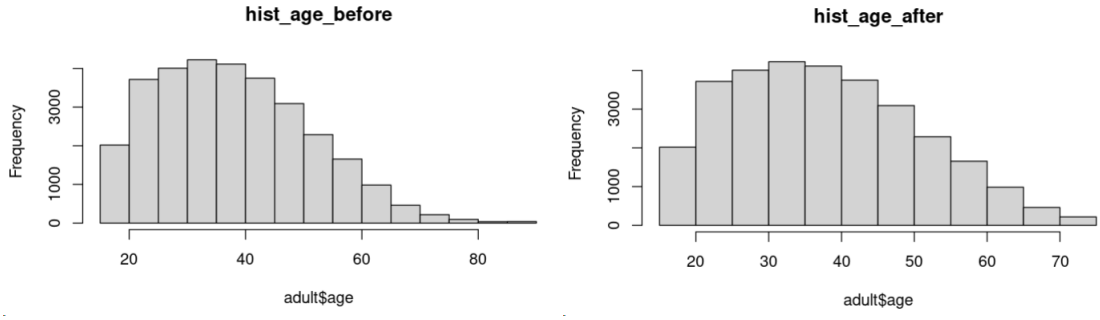
For further data cleaning, we first re-coded `sex, 50k` to binary variables, and **Federal-gov, Local-gov, State-gov** under `workclass` all to **gov**, and **Married–civ-spouse, Married-AF-spouse, Married-spouse-absent** under `maritalstatus` to **Married**, and **Divorced, Separated, Widowed** to **DSW**. We then added 3 to every observation in `educationnum`, the # of years an individual spent in education, as it was strangely counting 3 years less (for example 10th grade in `education` only having `educationnum` of 7). Lastly, we removed the observations that have missing values in `workclass, occupation`, further the 21 observations with responses **Never-worked** and **Without-pay**, the dataset still has 30703 observations after data cleaning. Lastly, we applied `as.factor` to all our categorical features.

2 Exploratory Data Analysis

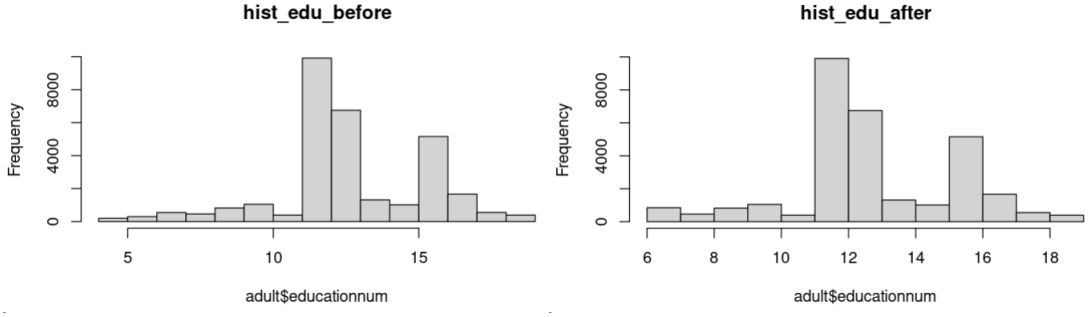
2.1 Outlier Detection

As we are predicting a binary variable, we can't easily find outliers numerical for the response variable, so we look at the 3 numerical features, `age`, `educationnum`, and `hoursperweek`. As we have more than 30000 observations from the census, we may confidently use standardized methods, we will also use the iqr test.

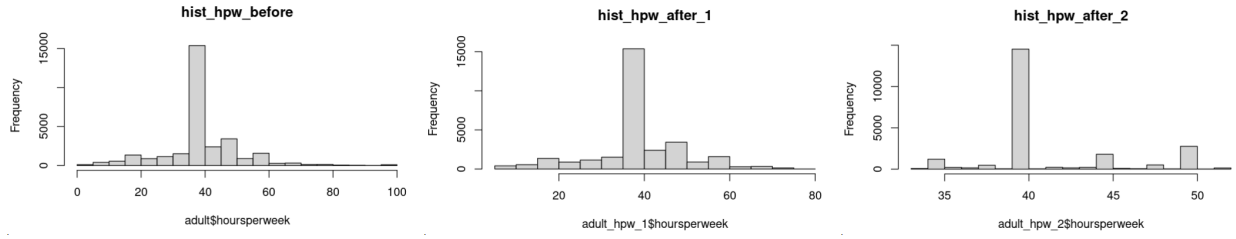
The range for `age` is 17 to 90. First, we standardize the distribution, and we find 122 observations with absolute values of z-scores larger than 3 with their ages ranging 78 to 90. Then, we find 172 observations that fall outside of the range of $(q1-1.5(\text{iqr}), q3+1.5(\text{iqr}))$ with the range being 76 to 90. As we have lots of observations, while most of the suspected outliers are overlapped, we will use the iqr test's set of suspected outliers, we then observe its histogram before and after removing the suspected outliers, here are the results:



The range for `educationnum` is 4 to 19, with 19 being the standard number of years of education for someone with a doctorate degree. We find the exact same 202 observations that have 4 or 5 years of education to be the outliers under both the standardized z test and the iqr test, we may observe the histograms from before and after removing the suspected outliers:



The range for `hoursperweek` is 1 to 99, We find 450 observations that take values in $[1, 4] \cup [77, 99]$ through the standardized z test, but through the iqr test, we find a total of 8092 observations that take value in 74 out of the original data's 94 unique values to be potential outliers. Although we wouldn't expect 450 potential outliers to heavily impact the data, 8092 is concerning. We find that $\mu \approx 41$ while $q2 = 40$, we may temporarily assign the cause of this being the distribution of `hoursperweek` having high kurtosis. We will look at the histograms after removing the different subsets of suspected outliers:



We note that distributions for both `age` and `educationnum` are seemingly more normally distributed for removing this little number of observations, thus we will keep these changes. For `hoursperweek`, we note that with the high percentage of data suspected as outliers by the iqr test, hist_hpw_after_2 may be too extreme for not too significant result as we note `hoursperweek` equals to 40 for 14522 of the observations, near half, thus removing further beyond hist_hpw_after_1 may unnecessarily erase reasonable spread useful for further analysis later.

There are still 29895 observations remaining after removing the suspected outliers, from which we may note that there are few overlaps. Further analysis will continue from here.

Lastly, the code for identifying and removing the outliers is given in Appendix A.1.

2.2 Summary Statistics

The seven variables chosen for our analysis encompass key dimensions of an individual's demographic and professional profile. These variables include age, workclass, educationnum, maritalstatus, occupation, sex, and hoursperweek. Our primary goal is to utilize this set of variables to predict whether an individual earns more or less than $50,000 annually. This prediction task involves a mix of categorical, numerical, and binary variables. Workclass, for instance, encompasses categories such as private employment, self-employment in a corporation, self-employment outside a corporation, and government employment. Occupation variables, on the other hand, group individuals into roles such as sales, farming-fishing, and handlers-cleaners. Additionally, age and hours worked per week are numerical variables, while sex is binary. Within the binary variable X50k, '1' represents individuals earning less than $50,000 a year, and '2' represents those earning $50,000 or more.

To get a preliminary feel for the data, we compute several summary statistics for the variables. Table 1 presents a concise summary of the three numerical variables we will consider.

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
age	29895	38	13	17	28	47	75
educationnum	29895	13	2.5	6	12	16	19
hoursperweek	29895	41	11	6	40	45	76

Table 1: Summary Statistics For Numerical Variables In The Adult Data Set.

Now, we generate summary statistics for the categorical variables. We are particularly interested in the relative proportions of each level of each factor. The results are given in the four tables below

Level	%
gov	14%
private	74%
self-employed	12%

(a) Levels of `workclass`

Level	%
DSW	20%
Married	48%
Never Marrried	32%

(b) Levels of `martialstatus`

Level	%
1 (Female)	32%
2 (Male)	68%

(c) Levels of `sex`

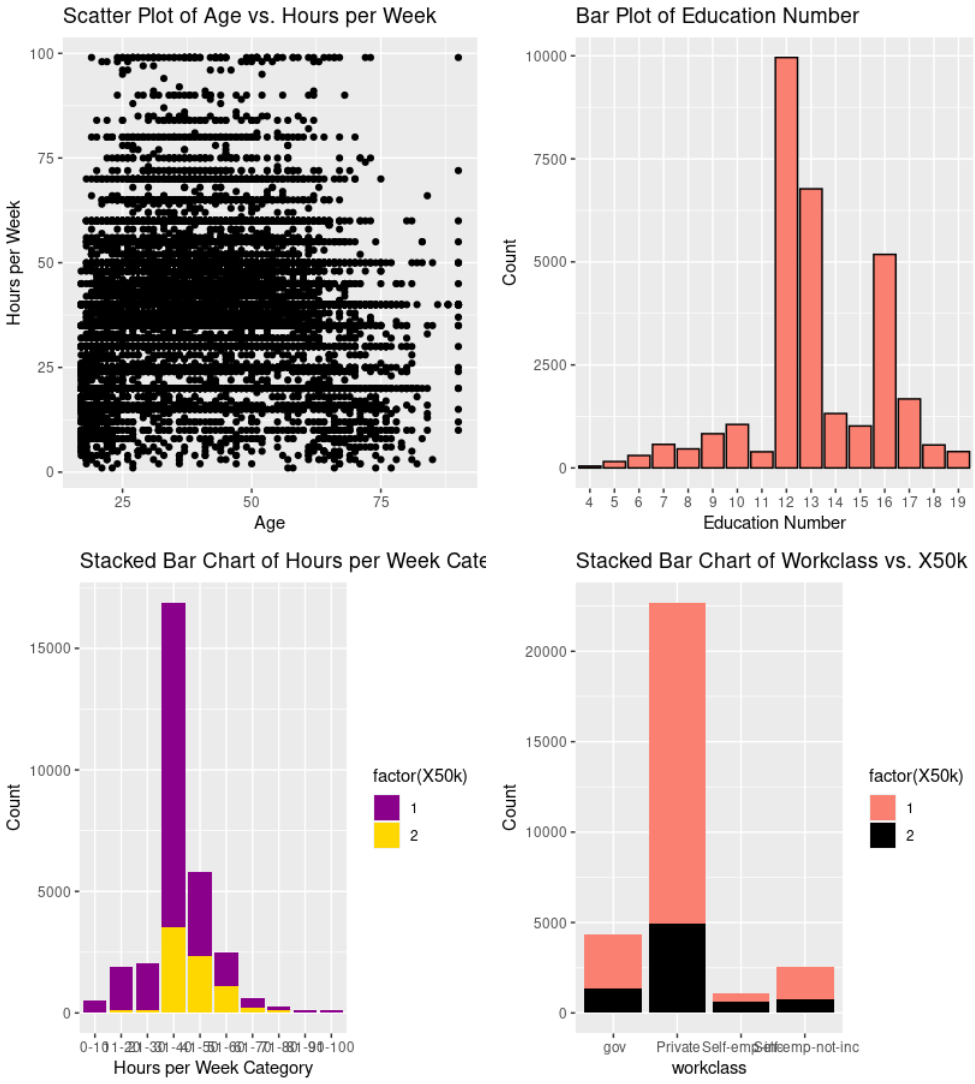
Table 2: Tables giving the relative proportions of levels in each categorical variable in the adult data set.

From these results, we see that the levels are roughly evenly distributed for most factors. We note that there is a difference in levels for `workclass`, with there being substantially more private employees than government employees or self-employed persons. Despite this difference, we didn't feel justified in combining the government and self-employed levels into one, as our prior knowledge about the job market tells us that these two categories have drastic fluctuations in income; if we were to combine these levels, these differences would be obscured. Thus, we left the variable as is. The other variables have roughly even distributions for their levels, so no recoding was necessary in these cases.

The code for cleaning the data and generating the summary statistics in the tables is given in Appendix A.2.

2.3 Data Visualization

To gain deeper insights into the data, we conducted visualizations to explore potential correlations between variables and the income level. However, no single variable emerged as highly predictive. For instance, a stacked bar chart depicting the relationship between hours worked per week and the X50k variable did not reveal a significant correlation. Similarly, visualizations of workclass did not exhibit a strong correlation with the X50k factor. Seeking to uncover potential relationships among other variables that might impact the model, we examined scatterplots of numeric variables like age and hours worked per week, which did not reveal a pronounced relationship. Furthermore, analyzing the frequency distribution of variables highlighted notable patterns, such as a substantial portion of individuals having a high school education and a considerable workforce employed in the private sector. These observations underscore the potential impact of these factors on the predictive models. As we continue our analysis, we remain mindful of the nuanced interplay between these variables and their implications for our predictive modeling efforts.



The code that was used to generate these plots is given in Appendix A.3.

3 Prediction Algorithms/Results

3.1 Support Vector Machines (SVM)

For the first prediction algorithm, we opted to use the support vector machine (SVM) algorithm to predict the binary response. The model we chose used all predictors availible from the data cleaning, as our prior experience and knowledge about income level led us to believe all predictors would be useful explanatory variables.

Prior to running the model, we standardized the three numerical variables `age`, `educationnum`, and `hoursperweek`. Since the SVM method is based partly on calculating distances, transforming all variables to a standardized scale will help ensure that the method is as accurate as possible and is not being controlled by variables which naturally take on larger values and have larger ranges. Since our data set has over 20,000 observations, computing times for cross validation and predictions became an issue while constructing our model, with many of the code segments taking several minutes to compute, due to the size of the data. Thus, we resorted to only randomly sampling 3,000 observations (roughly 10% of the total data). The size

of the sample provides enough observations to construct an accurate model while also being less demanding to compute.

First, we partitioned the sampled data into a training and test set. We opted to include more training observations than test observations, to provide a more insightful model. Specifically, we opted for a 70/30 training-test split i.e. having 70% of the sample to be training data and the remaining 30% of the sample to be the test data. Thus, we have 2100 training observations and 900 test observations.

We remark that the SVM algorithm is dependent on a hyperparameter C. This is often called the "cost" parameter and represents the leniency that we allow in our predictions. Specifically, it will control the overall number of support vectors that the model will be able to produce. Since the value of our chosen C controls the bias-variance tradeoff, it is essential to pick a good value of C. To do so, we tuned the hyperparameter C using cross-validation prior to building the entire model. We opted to use $K = 5$ folds to tune the hyperparameter. We chose $K = 5$ for computation considerations and also since using the more standard $K = 10$ folds did not yield a significant difference in accuracy in the final model. Thus, we proceeded with cross validation with 5 folds. Our set of candidate values is given by

$$\mathcal{C} = \{2^n : -7 \leq n \leq 7, n \in \mathbb{Z}\} \tag{1}$$

The choice of candidate value was governed by relevant examples we saw earlier but was also restricted by our computational limits. After running the cross validation, we find the optimal value of C to be

$$C_{\min} = 0.5 \tag{2}$$

Thus, we used the value of $C_{\min}$ to construct the final SVM model. After constructing the model and running predictions, we find that the test error is

$$\text{SVM Test Error} = 0.189 \tag{3}$$

We would like to stress that our data contains a large imbalance of values for the response. From the output in R, we see that

1	2
2280	720

Table 3: Table giving counts of the `X50k` variable in our sample.

The large imbalance was difficult to correct and is an inherent flaw in the data set provided. This is something that should be noted when assessing the performance of the final model and generalizing its findings.

Lastly, the code that was used to compute the cross-validation, fit the final models, and find the test error is given Appendix B.1.

3.2 Random Forest

We employed a random forest predictive model as our chosen approach for this analysis. The initial step involved splitting our dataset into training data and testing data, opting for a 20/80 split. This division allows us to train the model on a subset of the data and evaluate its performance on unseen instances. For the random forest model, we set the number of trees (ntree) to 200, a decision made to increase the model's ability to capture complex relationships within the data.

Despite our efforts to enhance the model's performance, the variance explained remained relatively low, even after introducing additional variables, increasing the number of trees, and adjusting the mtry

parameter. The challenge of increasing the variance explained could be attributed to the complexity of the underlying patterns in the data or potential interactions between variables that are not adequately captured by the model.

Interestingly, while the variance explained posed challenges, the Mean Squared Error (MSE) of the model remained low at 0.06. The low MSE indicates that the model's predictions are accurate, suggesting that it performs well in minimizing the squared differences between predicted and actual values. This discrepancy between low variance explained and low MSE could signify that the model is making accurate predictions despite not capturing the full complexity of the underlying relationships.

If we were to continue to work with this data set we might try to explore alternative modeling techniques or consider gathering additional relevant variables to improve the model's ability to explain variance.

Lastly, the code we used to build, train, and test the random forest model is given in Appendix B.2.

4 Qualitative Discussion

The support vector machine (SVM) model training and testing presented us with unique challenges related to building intricate models with a limited amount of computing power. First, we deemed it was necessary to tune the cost hyperparameter C before building a final model. The standard method of K-fold cross validation used commonly was too computationally expensive to compute; we therefore reduced the number of folds and number of candidate values to consider while also limiting the scope of the model to a random sample of the available data. Desipte the computational challenges of the SVM method, the final model gave us a satisfactory 0.189 value for the test error. Even though we had to consider a small subset of the data, the randomness involved in the sampling leaves us optimistic that the predictive power of the model may generalize to similar data as well.

The findings from our initial random forest analysis provide valuable insights into the predictive modeling task. Despite the inclusion of seven pertinent variables in our random forest model, we encountered challenges in achieving a high variance explained. The low variance explained may suggest that the model struggled to capture intricate relationships within the data, potentially due to unobserved complexities or interactions among variables. However, the model exhibited a commendable performance with a low Mean Squared Error (MSE) of 0.06, indicating accurate predictions despite the limited variance explained. Furthermore, a detailed examination of influential variables and their potential impact on the model is warranted. Visualization revealed no strong correlations between individual variables and the target income level, but careful feature selection and engineering may unearth meaningful relationships. In the future, refining the model architecture and considering more advanced strategies like hyperparameter tuning could optimize predictive accuracy. Overall, a meticulous and iterative approach, incorporating a variety of techniques, will be key to advancing the model's predictive capabilities and unraveling deeper insights from the dataset. As for future directions, though our data set still had many observations by the start of algorithms, due to extreme imbalanced distribution of data in categories such as `educationnum` and `workclass`, more observations would certainly help our algorithms. And if we had more features, which we thought we had plenty of before we started data cleaning, especially more numerical features, we would be allowed to not only provide better results with our current algorithms, but also perform other predictive algorithms, as many of them proved to perform poorly for a data set with this many categorical features.

5 References

- Adult Dataset, `https://archive.ics.uci.edu/dataset/2/adult`. *UCI Machine Learning Repository.* Also known as "Census" Data set and also uploaded to Kaggle.

- Github Repository containing the code for analysis and report: `https://github.com/NilayT321/Census-Data-Project`

A Exploratory Analysis Code Segments

A.1 Outlier Detection

```
#age
z_age <- scale(adult$age)
outliers_age_1 <- abs(z_age) > 3
outliers_obs_age_1 <- adult[outliers_age_1, ]

q1_age <- quantile(adult$age, 0.25)
q3_age <- quantile(adult$age, 0.75)
iqr_age <- q3_age - q1_age
outliers_age_2 <- (adult$age < (q1_age - 1.5 * iqr_age)) | (adult$age > (q3_age + 1.5 *
↪  iqr_age))
outliers_obs_age_2 <- adult[outliers_age_2, ]

hist_age_before <- hist(adult$age, main = "hist_age_before")

adult_age_2 <- adult %>%
  anti_join(outliers_obs_age_2, by = NULL)

hist_age_after <- hist(adult_age_2$age, main = "hist_age_after")

#educationnum
z_edu <- scale(adult$educationnum)
outliers_edu_1 <- abs(z_edu) > 3
outliers_obs_edu_1 <- adult[outliers_edu_1, ]

q1_edu <- quantile(adult$educationnum, 0.25)
q3_edu <- quantile(adult$educationnum, 0.75)
iqr_edu <- q3_edu - q1_edu
outliers_edu_2 <- (adult$educationnum < (q1_edu - 1.5 * iqr_edu)) | (adult$educationnum >
↪  (q3_edu + 1.5 * iqr_edu))
outliers_obs_edu_2 <- adult[outliers_edu_2, ]

hist_edu_before <- hist(adult$educationnum, main = "hist_edu_before")

adult_edu <- adult %>%
  anti_join(outliers_obs_edu_1, by = NULL)

hist_edu_after <- hist(adult_edu$educationnum, main = "hist_edu_after")

#hoursperweek
z_hpw <- scale(adult$hoursperweek)
outliers_hpw_1 <- abs(z_hpw) > 3
```

```
outliers_obs_hpw_1 <- adult[outliers_hpw_1, ]

q1_hpw <- quantile(adult$hoursperweek, 0.25)
q3_hpw <- quantile(adult$hoursperweek, 0.75)
iqr_hpw <- q3_hpw - q1_hpw
outliers_hpw_2 <- (adult$hoursperweek < (q1_hpw - 1.5 * iqr_hpw)) | (adult$hoursperweek >
↪  (q3_hpw + 1.5 * iqr_hpw))
outliers_obs_hpw_2 <- adult[outliers_hpw_2, ]

hist_hpw_before <- hist(adult$hoursperweek, main = "hist_hpw_before")

adult_hpw_1 <- adult %>%
  anti_join(outliers_obs_hpw_1, by = NULL)

hist_hpw_after_1 <- hist(adult_hpw_1$hoursperweek, main = "hist_hpw_after_1")

adult_hpw_2 <- adult %>%
  anti_join(outliers_obs_hpw_2, by = NULL)

hist_hpw_after_2 <- hist(adult_hpw_2$hoursperweek, main = "hist_hpw_after_2")

#remove outliers
adult <- adult %>%
  anti_join(outliers_obs_age_2, by = NULL)

adult <- adult %>%
  anti_join(outliers_obs_edu_2, by = NULL)

adult <- adult %>%
  anti_join(outliers_obs_hpw_1, by = NULL)
```

A.2 Summary Statistics

```
library(tidyverse)
library(vtable)

adultvars <- c("age", "workclass", "fnlwgt", "education", "educationnum",
               "maritalstatus", "occupation", "relationship", "race", "sex",
               "capitalgain", "capitalloss", "hoursperweek", "nativecountry", "50k")

adult_raw <- read.csv("adult.data", header = FALSE, col.names = adultvars , skip = 1)

#Remove unwanted variable:
adult <- subset(adult_raw, select = -c(capitalgain, capitalloss, fnlwgt,
                                       race, nativecountry, education, relationship))

#sex and 50k binary:
#Female = 1, Male = 2
```

```
adult$sex <- as.factor(adult$sex)
adult$sex<- as.numeric(adult$sex)

#<=50k = 1, >50k = 2
adult$X50k <- as.factor(adult$X50k)
adult$X50k <- as.numeric(adult$X50k)

#Re-code workclass, maritalstatus
adult <- adult %>%
  mutate(workclass = ifelse(workclass %in%
                      c(" Federal-gov", " Local-gov", " State-gov"),
                      " gov", workclass))

adult$maritalstatus = case_when(
  adult$maritalstatus %in% c(" Married-civ-spouse", " Married-AF-spouse", "
  ↪  Married-spouse-absent") ~ "Married",
  adult$maritalstatus %in% c(" Divorced", " Separated", " Widowed") ~ "DSW",
  adult$maritalstatus %in% c(" Never-married") ~ "NeverMarried",
)

#workclass, occupation filtering:
adult <- adult %>%
  filter(!(workclass %in% c(" ?", " Without-pay", " Never-worked")))

adult <- adult %>%
  filter(!(occupation %in% c(" ?")))

#Add 3 to educationnum
adult$educationnum <- adult$educationnum + 3

## =========== SUMMARY STATISTICS ===========
adult = read.csv("Adult_NoOutlier.csv")

# Get the numeric variables
adult_numeric = adult[,c(1,3,7)]

# This line gives us LaTeX code for a table that may be pasted in our report.
sumtable(adult_numeric, out="latex")
sumtable(adult)
```

A.3 Data Visualization

```
library(ggplot2)

# Example: Scatter plot of age vs. hoursperweek
ggplot(adult, aes(x = age, y = hoursperweek)) +
  geom_point() +
  labs(title = "Scatter Plot of Age vs. Hours per Week", x = "Age", y = "Hours per Week")
```

```r
# Example: Box plot of hoursperweek by workclass
ggplot(adult, aes(x = workclass, y = hoursperweek)) +
  geom_boxplot() +
  labs(title = "Box Plot of Hours per Week by Workclass", x = "Workclass", y = "Hours per
  ↪  Week")

# Example: Histogram of age
ggplot(adult, aes(x = age)) +
  geom_histogram(binwidth = 5, fill = "skyblue", color = "black", alpha = 0.7) +
  labs(title = "Histogram of Age", x = "Age", y = "Frequency")

# Example: Bar plot of educationnum
ggplot(adult, aes(x = factor(educationnum))) +
  geom_bar(fill = "salmon", color = "black") +
  labs(title = "Bar Plot of Education Number", x = "Education Number", y = "Count")

ggplot(adult, aes(x = hoursperweek, y = X50k)) +
  geom_point() +
  labs(title = "Scatter Plot of Hours per Week vs. X50k", x = "Hours per Week", y =
  ↪  "X50k")

ggplot(adult, aes(x = workclass, fill = factor(X50k))) +
  geom_bar() +
  labs(title = "Stacked Bar Chart of Workclass vs. X50k",
       x = "workclass",
       y = "Count") +
  scale_fill_manual(values = c("1" = "salmon", "2" = "black"))


library(dplyr)

# Create a new variable 'hours_per_week_category'
adult <- adult %>%
  mutate(hours_per_week_category = cut(hoursperweek, breaks = seq(0, 100, by = 10),
  ↪  labels = FALSE))

# Convert the new variable to a factor for better visualization
adult$hours_per_week_category <- factor(adult$hours_per_week_category, labels = c("0-10",
↪  "11-20", "21-30", "31-40", "41-50", "51-60", "61-70", "71-80", "81-90", "91-100"))


ggplot(adult, aes(x = hours_per_week_category, fill = factor(X50k))) +
  geom_bar() +
  labs(title = "Stacked Bar Chart of Hours per Week Category vs. X50k",
       x = "Hours per Week Category",
       y = "Count") +
  scale_fill_manual(values = c("1" = "darkmagenta", "2" = "gold1"))
```

B Prediction Algorithms Code Segments

B.1 Support Vector Machine (SVM)

```
library(dplyr)
library(e1071)

set.seed(2)
adult = read.csv("Adult_NoOutlier.csv")
adult = subset(adult, select = -c(X))

# Recode the response to a logical. This makes it easier to do SVM
# X50k = 1 (means <=50K, so mark it FALSE), other is TRUE
adult$X50k = ifelse(adult$X50k == "1", FALSE, TRUE)

# The code is taking too long to run. We will use a smaller sample of the data
# Only take 3000 observations from the adult data set
adult_sample = adult %>% sample_n(size = 3000)

n_obs = nrow(adult_sample)

# Use an 70/30 split for the training and test sizes
n_train = 0.7 * n_obs
n_test = n_obs - n_train

adult_train = adult_sample[1:n_train, ]
adult_test = adult_sample[(n_train+1):n_obs,]

# Find an appropriate value of C to use in the model
# The code takes a long time to run, so pick a smaller set of values
candidate_Cs = 2^seq(-7, 7, by = 1)

K = 5
err_matrix = matrix(0, K, length(candidate_Cs))

# Matrix of folds
folds = matrix(1:n_train, K)

# Loop across folds, which are rows of the matrix
for (k in 1:K) {

  # For each fold, create a validation set and a learning set
  # 20,000 obs in training set / 10 folds = 2000 in each fold


  valid_ix = folds[k,]
  learn_ix = setdiff(1:n_train, folds[k,])
```

```
  valid_fold = adult_train[valid_ix,]
  learn_fold = adult_train[learn_ix,]

  # Loop through each value in the candidate set
  for (i in 1:length(candidate_Cs)) {
    print(sprintf("Currently on fold %d, iteration %d", k, i))
    current_C = candidate_Cs[i]

    # Fit a SVM model on the learning set using the current C
    current_svm = svm(X50k ~ ., data = learn_fold, type = "C-classification",
                      kernel = "linear", cost = current_C)

    # Create predictions on the validation set
    current_preds = predict(current_svm, valid_fold[,1:7])
    current_preds = as.logical(current_preds)

    # Insert the mean test error in the matrix
    err_matrix[k, i] = mean(current_preds != valid_fold$X50k)

  }
}

# Now, get the mean error for each value of C in candidate_C
mean_errs = apply(err_matrix, 2, mean)

# Get the minimum error
min_ix = which.min(mean_errs)

C_min = candidate_Cs[min_ix]

# Fit an SVM model with this data set on the entire training set.
final_SVM = svm(X50k ~ ., data = adult_train, type = "C-classification",
                kernel = "linear", cost = C_min)

# Get predictions
final_SVM_preds = predict(final_SVM, adult_test)
final_SVM_preds = as.logical(final_SVM_preds)

# Get the mean test error
test_error = mean(final_SVM_preds != adult_test$X50k)

print(sprintf("The final test error for the SVM procedure was %.3f", test_error))
```

B.2 Random Forest

```
library(dplyr)
library(magrittr)
library(randomForest)
```

```r
set.seed(123)  # Set a seed for reproducibility
train_indices <- sample(1:nrow(adult2), 0.8 * nrow(adult2))
train_data <- adult2[train_indices, ]
test_data <- adult2[-train_indices, ]


target_variable <- "X50k"
predictor_variables <- setdiff(names(train_data), target_variable)

# Train the random forest model
rf_model <- randomForest(formula = as.formula(paste(target_variable, "~ .")),
                         data = test_data,
                         ntree = 200)

# Make predictions on the test dataset
predictions <- predict(rf_model, newdata = test_data)

# Print the predictions
print(predictions)

print(rf_model)


# Extract the target variable from the testing dataset
Y_test_actual <- test_data$X50k

# Make predictions using the random forest model
Y_test_pred <- predict(rf_model, newdata = test_data)

# Calculate the mean squared error (MSE) as the test error
test_error <- mean((Y_test_actual - Y_test_pred)^2)

# Print or use the test error as needed
print(paste("Test Error (MSE):", test_error))

adult2$predicted_class <- predict(rf_model, newdata = adult2, type = "response")
```