Determinants of Marriage and A Classification Approach by using kNN, Classification Tree and Naïve Bays method

Presented By

Suchunya Suwanwathana 6405077

Pattarawut Ariyapurck 6405305

Present to

Dr. Pannapa Changpetch

This report is part of the Data Mining subject.

Department of Mathematics

Faculty of Science, Mahidol University

Table of Contents

Introduction

Objective

To classify the married by using k-NN, Classification Tree and Naïve Bayes.

To find the best method for Determinants of Wages Data (CPS 1985) dataset by comparing the performance of each method.

Data source

Name of Dataset: Determinants of Wages Data (CPS 1985)

Dataset Collaborators: Avik Das

Sources: https://www.kaggle.com/datasets/avikdas2021/determinants-of-wages-data-cps-1985

Number of columns: 12

Number of quantitative variables: 5

Number of categorical variables: 7

Study population

Cross-section data originating from the May 1985 Current Population Survey by the US Census Bureau (random sample drawn for Berndt 1991), totaling 534 records.

Dataset Description

Variable Descriptions

X: Row index from the original dataset. (Not used)

wage: Hourly wage (in US dollars).

education: Years of education.

experience: Years of work experience.

age: Age of the individual.

ethnicity: Ethnic background of the individual; includes 3 levels: “hispanic”, “cauc” and “other”.

region: Indicates whether the individual lives in the South; Includes 2 levels: “south” and “other”

gender: Gender of the individual; includes 2 levels: “male” and “female”.

occupation: Type of job or occupation; includes 6 levels: "worker", "technical", "services", "office", "sales" and "management".

sector: Sector of employment; includes 3 levels: "manufacturing", “construction” and “other”.

union: Union membership status: yes (member) or no (non-member).

married: Marital status: yes = married, no = not married.

Variable Type

```
#Import Data
data = read.csv("C:/Users/BOSS/Documents/Data Mining/Project/Determinants of Wages Data (CPS 1985).csv")

#Cleaning Data
data = data[, -1]

#check data
str(data)
```

Wage : num (numeric)

Education : int (integer)

Experience : int (integer)

Age : int (integer)

Ethnicity : chr (character)

Region : chr (character)

Gender : chr (character)

Occupation : chr (character)

Sector : chr (character)

Union : chr (character)

Married : chr (character)

Preliminary Data Analysis

Descriptive Statistics

An initial data analysis was performed by categorizing the variables into Continuous and categorical types. Descriptive statistics for the numerical variables are presented in Table 1, while the distribution of categorical variables is shown in Table 2 by using function summary()

```
data[,c(5:11)] = lapply(data[,c(5:11)], as.factor)
str(data)
summary(data)
```

Variable				
Continuous				
	Min	**Max**	**Median**	**Mean**
wage	1.0000	44.5000	7.7800	9.0240
education	2.0000	18.0000	12.0000	13.0200
experience	0.0000	55.0000	15.0000	17.8200
age	18.0000	64.0000	35.0000	36.8300

Table 1 summary of continuous variables

Variable						
Categorical						
ethnicity	**region**	**gender**	**occupation**	**sector**	**union**	**married**
cauc: 440 (82.4%)	south: 156 (29.21%)	male: 289 (54.12%)	management: 55 (10.29%)	construction: 24 (4.49%)	yes: 96 (17.98%)	yes: 350 (65.54%)
hispanic: 27 (5.06%)	other: 378 (70.79%)	female: 245 (45.88%)	office: 97 (18.16%)	manufacturing: 99 (18.53%)	no: 438 (82.02%)	no: 184 (34.46%)
other: 67 (12.54%)			sales: 38 (7.12%)	other: 411 (76.97%)		
			services: 83 (15.54%)			
			technical: 105 (19.66%)			
			worker: 156 (29.21%)			

Table 2 summary of Categorical Variables

Data Visualization

To make the analysis easier and faster to understand, we use various types of charts such as histograms, bar charts, box plots, and stacked charts. These visual tools help reveal trends, patterns, relationships, and unusual data points that are harder to see in raw number tables.

For the purpose of this study, the data is categorized into two main types

continuous variables: Wage, Education, Experience and Age

categorical variables: Ethnicity, Region, Gender, Occupation, Sector, Union and Married

Continuous variables

For the continuous variables, we used histograms to observe the distribution of each variable. In addition, box plots were created to compare the distribution of each independent variable across the categories of the dependent variable.

```
hist(data$wage, main = "Distribution of wage", xlab = "wage")
hist(data$education, main = "Distribution of education", xlab = "education")
hist(data$experience, main = "Distribution of experience", xlab = "experience")
hist(data$age, main = "Distribution of age", xlab = "age")
```

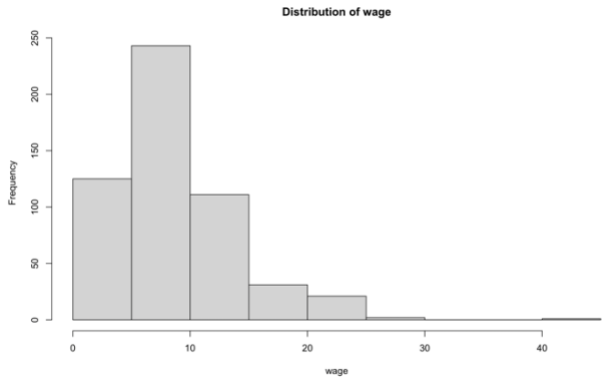

Figure 1: Histogram showing the distribution of the wage variable.

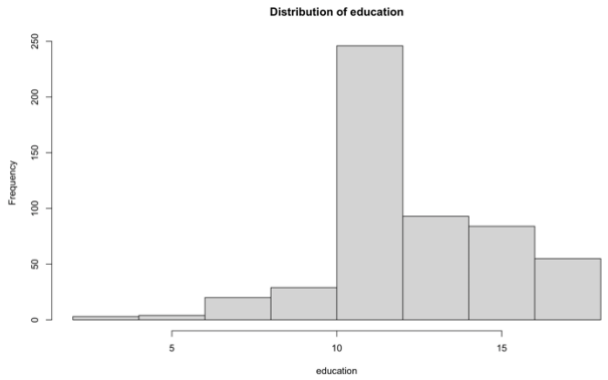

Figure 2: Histogram showing the distribution of the education variable.

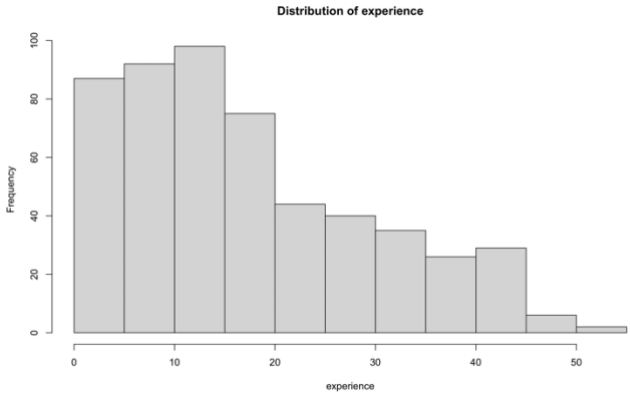

Figure 3: Histogram showing the distribution of the experience variable.

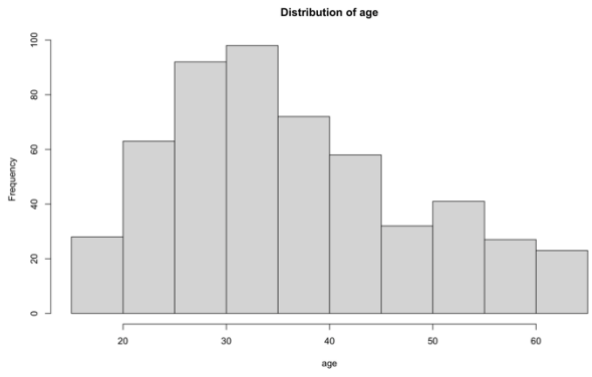

Figure 4: Histogram showing the distribution of the age variable.

The box plot was created using the function shown below:

```
qplot(x = married, y = wage, data = data, geom = 'boxplot')
qplot(x = married, y = education, data = data, geom = 'boxplot')
qplot(x = married, y = experience, data = data, geom = 'boxplot')
qplot(x = married, y = age, data = data, geom = 'boxplot')
```

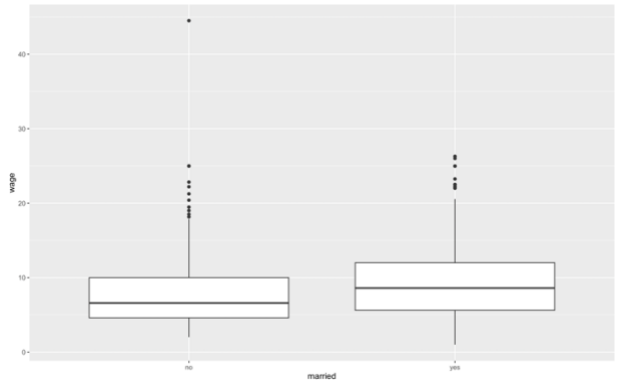

Figure 5: Box plot showing wage distribution among married and unmarried individuals.

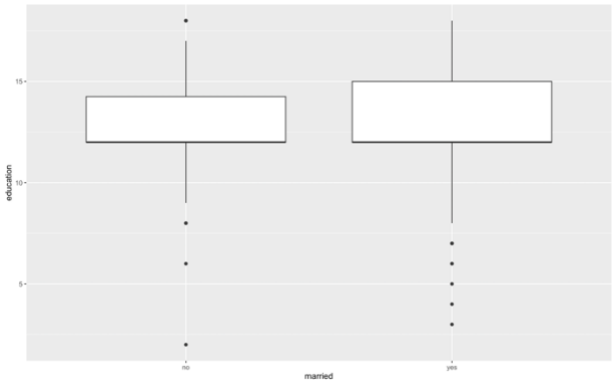

Figure 6: Box plot showing education distribution among married and unmarried individuals.

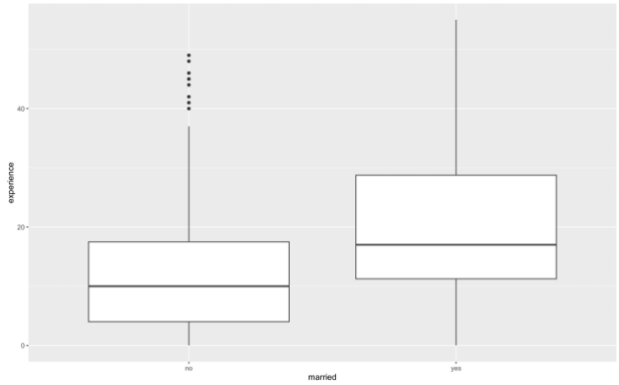

Figure 7: Box plot showing experience distribution among married and unmarried individuals.

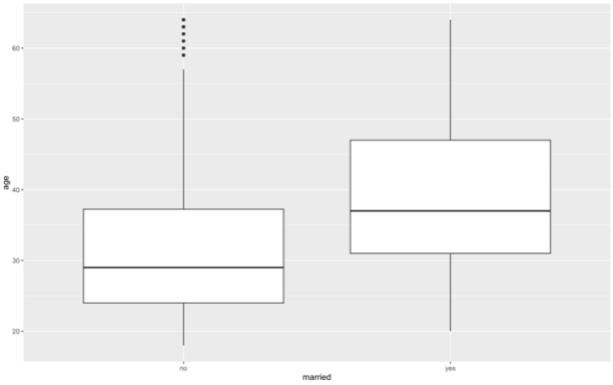

Figure 8: Box plot showing age distribution among married and unmarried individuals.

Categorical variables

For the categorical variables, bar charts were used to compare the frequencies of each category. Additionally, stacked plots were created to show the proportion of each independent variable within the subgroups of the dependent variable.

The bar charts was created using the function shown below:

```
barplot(table(data$ethnicity), main = "Bar chart of ethnicity", xlab = "ethnicity")
barplot(table(data$region), main = "Bar chart of region", xlab = "region")
barplot(table(data$gender), main = "Bar chart of gender", xlab = "gender")
barplot(table(data$occupation), main = "Bar chart of occupation", xlab = "occupation")
barplot(table(data$sector), main = "Bar chart of sector", xlab = "sector")
barplot(table(data$union), main = "Bar chart of union", xlab = "union")
barplot(table(data$married), main = "Bar chart of married", xlab = "married")
```

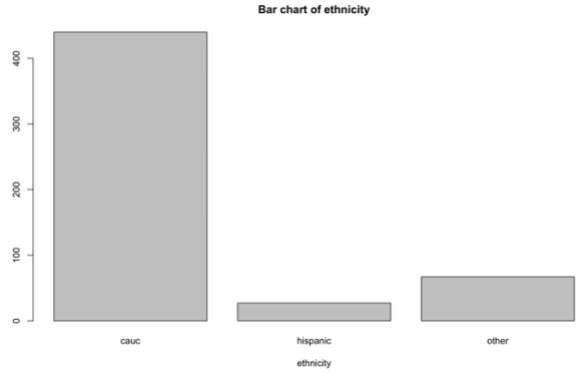

Figure 9: Bar chart showing the distribution of participants by ethnicity.

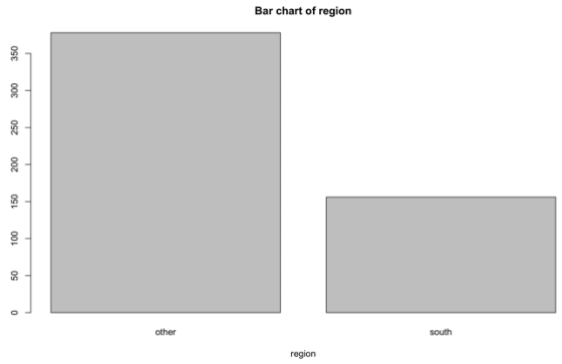

Figure 10: Bar chart showing the distribution of region.

Figure 11: Bar chart showing the distribution of gender.

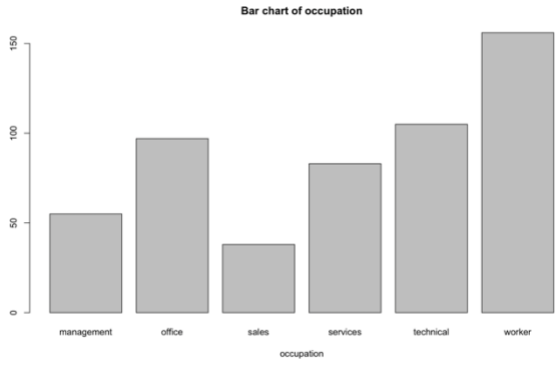

Figure 12: Bar chart showing the distribution of occupation.

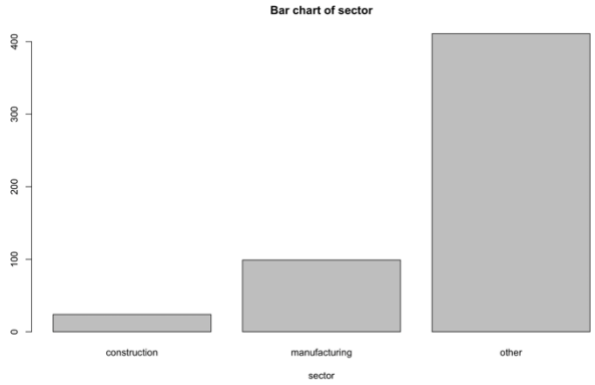

Figure 13: Bar chart showing the distribution of sector.

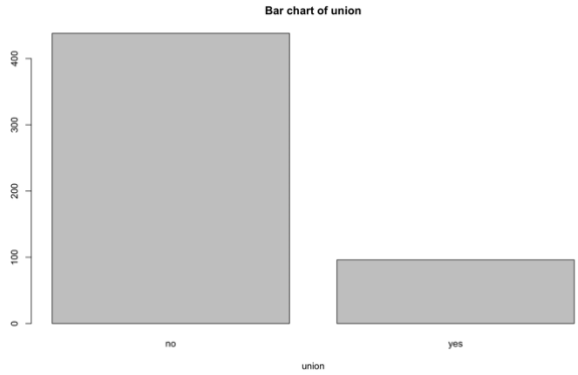

Figure 14: Bar chart showing the distribution of gender.

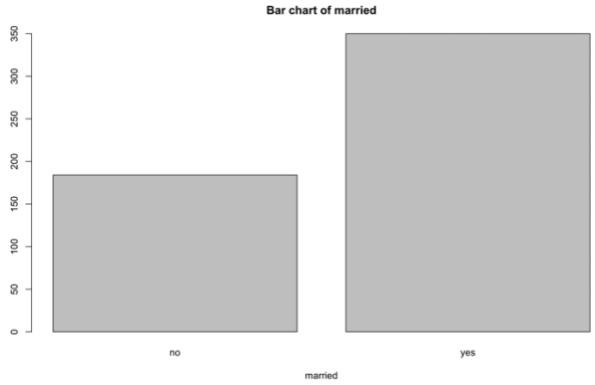

Figure 15: Bar chart showing the distribution of married.

The stacked plot was created using the function shown below:

```
attach(data)
barplot(prop.table(table(ethnicity, married), 2),
        main = "Stacked plot between married and ethnicity",
        xlab = "married", col = c("gray","lightgray", "whitesmoke"))

barplot(prop.table(table(region, married), 2),
        main = "Stacked plot between married and region",
        xlab = "married", col = c("gray","lightgray"))

barplot(prop.table(table(gender, married), 2),
        main = "Stacked plot between married and gender",
        xlab = "married", col = c("gray","lightgray"))

barplot(prop.table(table(occupation, married), 2),
        main = "Stacked plot between married and occupation",
        xlab = "married", col = c("gray","lightgray", "whitesmoke", "ghostwhite",
                                  "lavender", "lightsteelblue"))

barplot(prop.table(table(sector, married), 2),
        main = "Stacked plot between married and sector",
        xlab = "married", col = c("gray","lightgray", "whitesmoke"))

barplot(prop.table(table(union, married), 2),
        main = "Stacked plot between married and union",
        xlab = "married", col = c("gray","lightgray"))
```

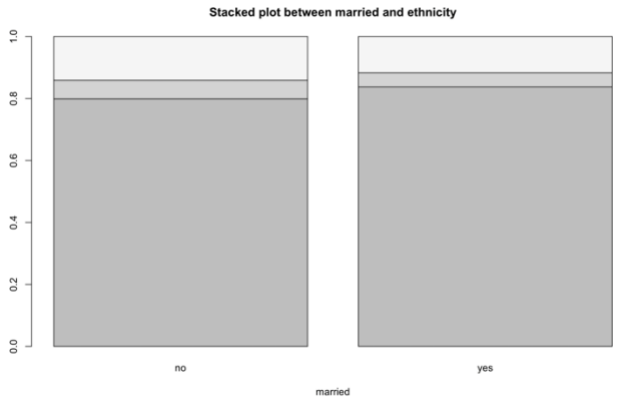

Figure 16: Stacked plot showing the distribution of ethnicity among married and unmarried individuals.

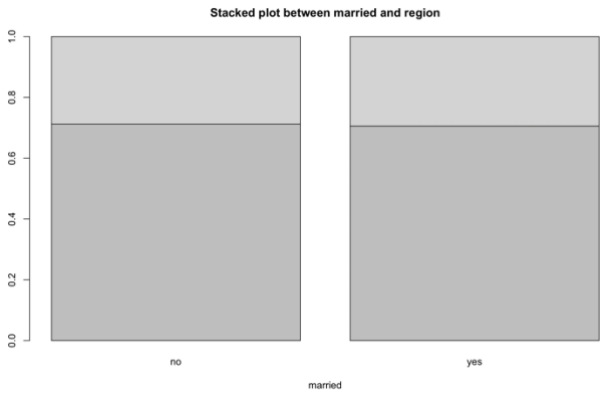

Figure 17: Stacked plot showing the distribution of regions among married and unmarried individuals.

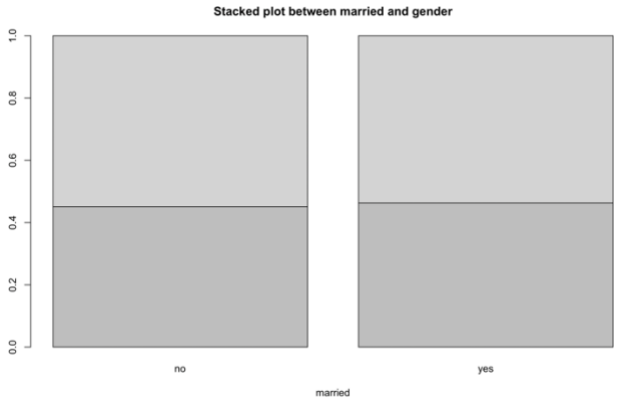

Figure 18: Stacked plot showing the distribution of gender among married and unmarried individuals.

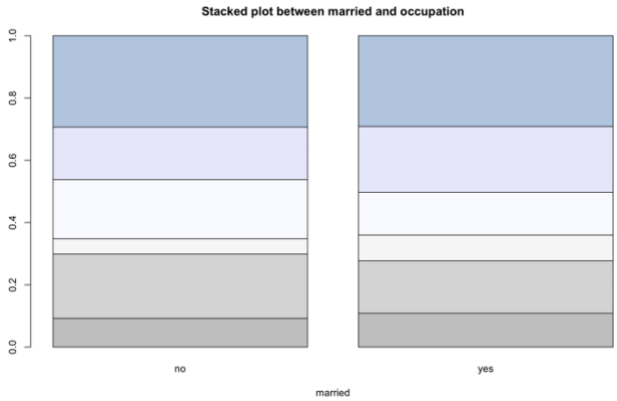

Figure 19: Stacked plot showing the distribution of occupation among married and unmarried individuals.

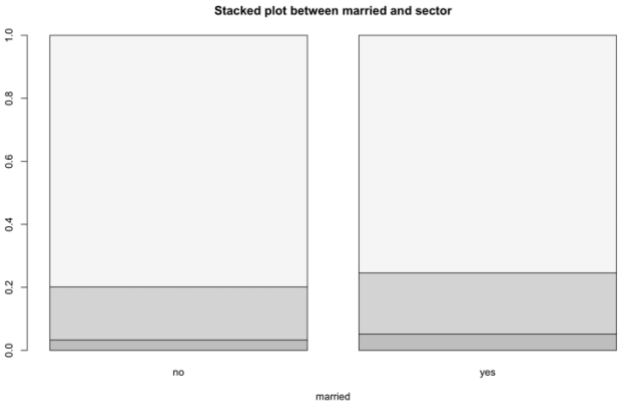

Figure 20: Stacked plot showing the distribution of sector among married and unmarried individuals**.**

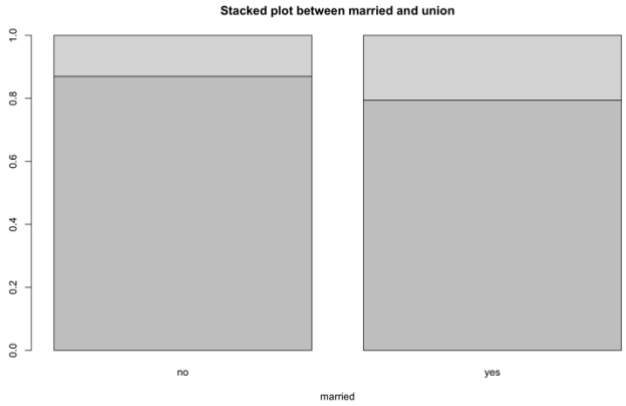

Figure 21: Stacked plot showing the distribution of union among married and unmarried individuals**.**

Methodology

k-NN with Cross validation

To perform k-NN using cross-validation, we split the dataset into 90% for training and 10% for testing in order to compare the accuracy of the model under different scenarios.

```
# 90,10
set.seed(123)
idxs = sample(1:nrow(data), as.integer(0.9*nrow(data)))
train_data = data[idxs,]
test_data = data[-idxs,]
```

For this method, four continuous variables are used, including: wage, education, experience, and age. We started by preparing the data, which involved scaling the relevant variables: wage, education, experience, and age.

```
############## kNN CV
#scale data
data = data[, -c(5:10)]
data[,c("wage","education","experience","age")] = scale(data[,c("wage","education","experience","age")])
colnames(data)
str(data)
```

Then, we ran the k-NN model by varying the value of K from 1 to 17. The model was evaluated under four different configurations: Full Model (All variables were used.)

Without education (The variable education was excluded.)

Without wage (The variable wage was excluded.)

Without education and wage (Both education and wage were excluded.)

```
#married ~ wage + education + experience + age # 90,10
#kNN  k = 1
nn1 = kNN(married ~ wage + education + experience + age, train_data, test_data,stand = FALSE, k = 1)
table(test_data[, "married"], nn1)
#kNN  k = 3
nn3 = kNN(married ~ wage + education + experience + age, train_data, test_data,stand = FALSE, k = 3)
table(test_data[, "married"], nn3)
#kNN  k = 5
nn5 = kNN(married ~ wage + education + experience + age, train_data, test_data,stand = FALSE, k = 5)
table(test_data[, "married"], nn5)
#kNN  k = 7
nn7 = kNN(married ~ wage + education + experience + age, train_data, test_data,stand = FALSE, k = 7)
table(test_data[, "married"], nn7)
#kNN  k = 9
nn9 = kNN(married ~ wage + education + experience + age, train_data, test_data,stand = FALSE, k = 9)
table(test_data[, "married"], nn9)
#kNN  k = 11
nn11 = kNN(married ~ wage + education + experience + age, train_data, test_data,stand = FALSE, k = 11)
table(test_data[, "married"], nn11)
#kNN  k = 13
nn13 = kNN(married ~ wage + education + experience + age, train_data, test_data,stand = FALSE, k = 13)
table(test_data[, "married"], nn13)
#kNN  k = 15
nn15 = kNN(married ~ wage + education + experience + age, train_data, test_data,stand = FALSE, k = 15)
table(test_data[, "married"], nn15)
#kNN  k = 17
nn17 = kNN(married ~ wage + education + experience + age, train_data, test_data,stand = FALSE, k = 17)
table(test_data[, "married"], nn17)
```

The accuracy for each model and each K value was recorded for comparison, as shown in the table below.

k-NN with Cross validation (train 90% , test 10%)

Accuracy

K	Full Model	Without education	Without wage	Without education and wage
1	0.5926	0.6667	0.6111	0.6111
3	0.6852	0.7037	0.7037	0.7037
5	0.7593	0.7037	0.7037	0.7037
7	0.7407	0.7222	0.7222	0.7222
9	0.7407	0.7222	0.7222	0.7222
11	0.7963	0.7778	0.7963	0.7963
13	0.7963	0.7778	0.7778	0.7778
15	0.7407	0.7963	0.7963	0.7963
17	0.7778	0.7593	0.7593	0.7593

Table 3 presents the classification accuracy of various k-NN models with different *k* values, using cross-validation. The data was divided into 90% training and 10% testing sets.

Next, we performed the k-NN classification using cross-validation. However, in this step, we split the dataset into 80% for training and 20% for testing in order to compare the model's accuracy.

We then repeated the same process as described earlier, using the same variables and the same four model variations. The results are shown in the table below.

k-NN with Cross validation (train 80% , test 20%)

Accuracy

K	Full Model	Without education	Without wage	Without education and wage
1	0.5794	0.5981	0.5981	0.5794
3	0.6822	0.6822	0.6822	0.6822
5	0.7289	0.7196	0.7289	0.7196
7	0.7196	0.7196	0.7196	0.7196
9	0.7103	0.7103	0.7103	0.7103
11	0.7383	0.7383	0.7383	0.7383
13	0.6449	0.7383	0.7383	0.7383
15	0.7477	0.7477	0.7477	0.7477
17	0.7477	0.7477	0.7477	0.7477

Table 4 presents the classification accuracy of various k-NN models with different k values, using cross-validation. The data was divided into 80% training and 20% testing sets.

k-NN with k-fold Cross validation

We then perform k-NN using k-fold validation to compare the model accuracy in different scenarios, with the same models and variables used in the above method.

We begin with 10-fold cross validation, as shown in the code below.

```
# 10 -fold married ~ wage + education + experience + age (full)
trControl <- trainControl(method  = "cv",number = 10)
fit = train(married ~ wage + education + experience + age,
               method     = "knn",
               tuneGrid   = data.frame(k=seq(1,21,by=2)),
               preProc    = c("center", "scale"),
               trControl  = trControl,
               metric     = "Accuracy",
               data       = data)
fit
```

Followed by 5-fold cross validation, as shown below.

```
# 5 -fold married ~ wage + education + experience + age
trControl <- trainControl(method  = "cv",number = 5)
fit = train(married ~ wage + education + experience + age,
            method     = "knn",
            tuneGrid   = data.frame(k=seq(1,21,by=2)),
            preProc    = c("center", "scale"),
            trControl  = trControl,
            metric     = "Accuracy",
            data       = data)
fit
```

The accuracy results from both 5-fold and 10-fold cross validation are presented in the table below.

k-NN with 10-fold Cross validation

	Accuracy			
K	Full Model	Without education	Without wage	Without education and wage
1	0.5845	0.6086	0.6481	0.6687
3	0.6519	0.6423	0.6686	0.6703
5	0.6763	0.6724	0.6759	0.6591
7	0.6576	0.6986	0.6927	0.6573
9	0.6726	0.6985	0.6963	0.6742
11	0.6912	0.6743	0.6928	0.6687
13	0.6856	0.6761	0.6964	0.6838
15	0.6819	0.6873	0.7039	0.7061
17	0.6986	0.6835	0.7096	0.7099

Table 5 presents the classification accuracy of various k-NN models with different k values, using 10-fold cross-validation.

k-NN with 5-fold Cross validation				
		Accuracy		
K	Full Model	Without education	Without wage	Without education and wage
1	0.5767	0.6292	0.6443	0.6442
3	0.6499	0.6516	0.6836	0.6554
5	0.6666	0.6796	0.6629	0.6667
7	0.6761	0.6871	0.6742	0.6780
9	0.6704	0.6721	0.6854	0.6742
11	0.6723	0.6796	0.6873	0.6666
13	0.6741	0.6853	0.6948	0.6798
15	0.6760	0.6834	0.7060	0.6966
17	0.6816	0.6853	0.7172	0.6966

Table 6 presents the classification accuracy of various k-NN models with different k values, using 5-fold cross-validation.

Classification tree method

We use the Classification tree method with k-fold cross validation to classify the data into married = yes or not married = no. This method will use married as a response and wage, education, experience, age, ethnicity, region, gender, occupation, sector and union as predictors. This method can use both quantitative and categorical predictors.

First, we prepare the dataset by adding factors to categorical predictors and then we start with using the rpart() function with all predictors to grow the tree before pruning by using minsplit = 1 and 10-fold cross validation in the first run and we will get

```
##full model
fit_model = rpart(married ~ wage + education + experience + age + ethnicity +
                    region + gender + occupation + sector + union,
                  method = "class", data = data, control = rpart.control(minsplit = 1, xval = 10))
printcp(fit_model)
rpart.plot(fit_model)
```

```
Classification tree:
  rpart(formula = married ~ wage + education + experience + age +
          ethnicity + region + gender + occupation + sector + union,
        data = data, method = "class", control = rpart.control(minsplit = 1,
                                                               xval = 10))

Variables actually used in tree construction:
  [1] age        education  experience gender     occupation wage

Root node error: 184/534 = 0.34457

n= 534

CP nsplit rel error  xerror     xstd
1 0.195652      0   1.00000 1.00000 0.059684
2 0.011957      1   0.80435 0.83152 0.056783
3 0.010870      8   0.71739 0.92935 0.058595
4 0.010000      9   0.70652 0.91848 0.058412
```

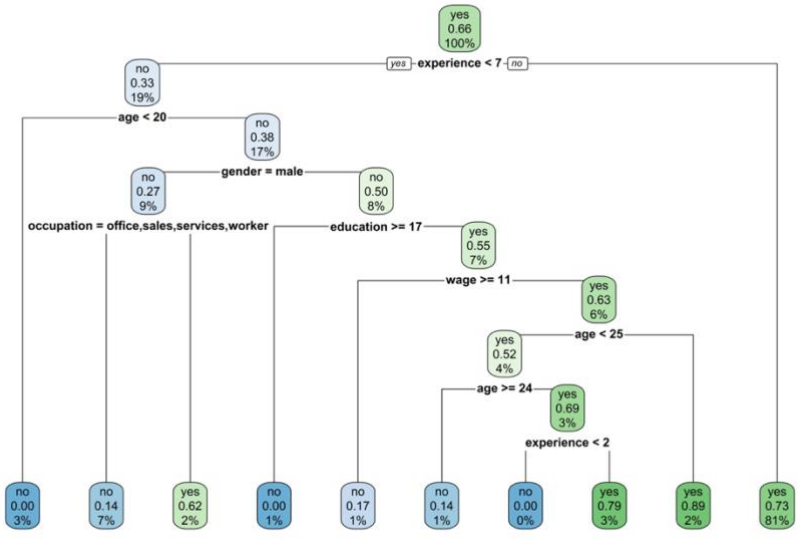

Figure 22: rpart plot showing the tree of full model before pruning

Then we pruning the tree by using the prune() function.

```
pfit_model = prune(fit_model, cp = fit_model$cptable[which.min(fit_model$cptable[,"xerror"]),"CP"])
printcp(pfit_model)
rpart.plot(pfit_model)
```

```
Classification tree:
  rpart(formula = married ~ wage + education + experience + age +
          ethnicity + region + gender + occupation + sector + union,
        data = data, method = "class", control = rpart.control(minsplit = 1,
                                                               xval = 10))

Variables actually used in tree construction:
  [1] experience

Root node error: 184/534 = 0.34457

n= 534

CP nsplit rel error  xerror     xstd
1 0.195652      0   1.00000 1.00000 0.059684
2 0.011957      1   0.80435 0.83152 0.056783
```

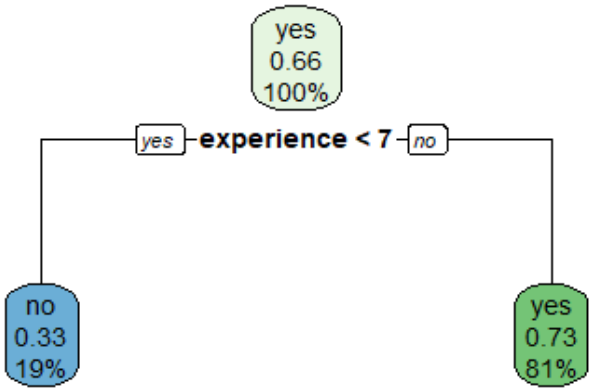

Figure 23: rpart plot showing the tree of full model after pruning

After that we repeat this process by removing some parameters. It was found that all models after pruning had only 1 branch remaining and used the experience variable to describe all data, which is an simplistic model or underfitting. Therefore, we will use all unpruned models to prevent underfitting.

Finally, we will have a table to compare the performance of the models.

Model	nsplit	xerror	Accuracy
Full model	9	0.9185	0.6835
Full model without education	6	0.8967	0.6910
Full model without wage and education	6	0.9239	0.6816
Full model without wage and age	6	0.8804	0.6966
Quantitative model with ethnicity and occupation	5	0.8967	0.6910
Quantitative model with ethnicity, region and occupation	5	0.9348	0.6779
Full model without wage, education and age	5	0.8370	0.7116
Quantitative model with occupation	4	0.8859	0.6948
Quantitative model with occupation and sector	4	0.9130	0.6854
Quantitative model with occupation and union	4	0.9511	0.6723

Table 7 Showing nsplit, xerror and accuracy of each models before pruning in Classification tree method.

Note, Full model includes all predictors of this method and Quantitative model includes: wage, education, experience and age.

Naïve Bayes method

We use the Naïve Bayes method with k-fold cross validation to classify the data into married = yes or not married = no. This method will use married as a response and ethnicity, region, gender, occupation, sector and union as a predictor. This method can use only categorical predictors. So, we will bring a quantitative predictor that includes: wage, education, experience and age to use Equal-Frequency Discretization techniques to transform quantitative variables into categorical variables includes: wage_I, education_I experience_I and age_I

```
##Intervals wage
x = classIntervals(data$wage, 4, style = 'quantile')
x
data$wage_I = cut(data$wage, breaks = unique(x$brks), include.lowest = TRUE, labels = c("Low", "Medium", "High", "Very High"))

##Intervals education
x = classIntervals(data$education, 4, style = 'quantile')
x
data$education_I = cut(data$education, breaks = unique(x$brks), include.lowest = TRUE, labels = c("Low", "Medium", "High"))

##Intervals experience
x = classIntervals(data$experience, 4, style = 'quantile')
x
data$experience_I = cut(data$experience, breaks = unique(x$brks), include.lowest = TRUE, labels = c("Low", "Medium", "High",
"Very High"))

##Intervals age
x = classIntervals(data$age, 4, style = 'quantile')
x
data$age_I = cut(data$age, breaks = unique(x$brks), include.lowest = TRUE, labels = c("Low", "Medium", "High", "Very High"))
```

```
##Intervals wage
style: quantile
[1,5.25)  [5.25,7.78) [7.78,11.25) [11.25,44.5]
131          135          128          140

##Intervals education
style: quantile
one of 560 possible partitions of this variable into 4 classes
[2,12) [12,12) [12,15) [15,18]
83       0     312     139

##Intervals experience
style: quantile
one of 20,825 possible partitions of this variable into 4 classes
[0,8)  [8,15) [15,26) [26,55]
122     137     137     138

##Intervals age
style: quantile
one of 15,180 possible partitions of this variable into 4 classes
[18,28) [28,35) [35,44) [44,64]
130     133     130     141
```

First, we prepare the dataset by adding factors to categorical predictors and then we start with using the naiveBayes() function with all predictors and we will get

```
##Full Model
model = naiveBayes(married ~ wage_I + education_I + experience_I + age_I + ethnicity +
                     region + gender + occupation + sector + union, data = data)
model
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
  naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
  Y
no       yes
0.3445693 0.6554307

Conditional probabilities:
  wage_I
Y           Low    Medium      High Very High
no  0.3369565 0.2608696 0.2065217 0.1956522
yes 0.2114286 0.2400000 0.2742857 0.2742857

education_I
Y           Low    Medium      High
no  0.5163043 0.2608696 0.2228261
yes 0.5914286 0.1657143 0.2428571

experience_I
Y           Low    Medium      High Very High
no  0.4510870 0.2173913 0.1902174 0.1413043
yes 0.1657143 0.2742857 0.2685714 0.2914286

age_I
Y           Low    Medium      High Very High
no  0.4728261 0.2282609 0.1630435 0.1358696
yes 0.1742857 0.2600000 0.2657143 0.3000000

ethnicity
Y           cauc   hispanic      other
no  0.79891304 0.05978261 0.14130435
yes 0.83714286 0.04571429 0.11714286

region
Y         other     south
no  0.7119565 0.2880435
yes 0.7057143 0.2942857

gender
Y        female      male
no  0.4510870 0.5489130
yes 0.4628571 0.5371429

occupation
Y     management     office      sales   services  technical     worker
no  0.09239130 0.20652174 0.04891304 0.19021739 0.16847826 0.29347826
yes 0.10857143 0.16857143 0.08285714 0.13714286 0.21142857 0.29142857

sector
Y     construction manufacturing      other
no    0.03260870    0.16847826 0.79891304
yes   0.05142857    0.19428571 0.75428571

union
Y            no       yes
no  0.8695652 0.1304348
yes 0.7942857 0.2057143
```

After this process, we will find the accuracy of this model with k-fold cross validation with k = 10 in the first run and we will get

```
#Full Model
x = data.frame(data$ethnicity, data$region, data$gender, data$occupation, data$sector, data$union,
               data$wage_I, data$education_I, data$experience_I, data$age_I)
y = data$married

model = train(x, y, "nb", trControl = trainControl(method = 'cv', number = 10))
model
```

```
Naive Bayes

534 samples
10 predictor
2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 480, 481, 481, 481, 480, 480, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
FALSE      0.7003494  0.304027
TRUE       0.7003494  0.304027

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held constant
at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = FALSE and adjust = 1.
```

After that we repeat this process by removing some parameters and we will have a table to compare the performance of the models.

Model	Accuracy
Full model	0.7003
Full model without intervals	0.6386
Full model without education_I and age_I	0.6705
Full model with intervals	0.7022
Full model with intervals and region	0.7059
Full model with intervals and gender	0.7061
Full model with intervals and sector	0.7064
Full model with intervals and union	0.6986
Full model with intervals and sector, region	0.7003
Full model with intervals and sector, gender	0.6947

Table 8 Showing the accuracy of each models in Naive Bayes method.

Note, Full model includes all predictors of this method and Model with intervals includes: wage_I, education_I experience_I and age_I

Discussion

In this study, k-Nearest Neighbors (k-NN), Classification tree and Naïve Bayes method. Starting with k-NN, Which using four variables: education, wage, experience, and age. To assess the effect of individual features on model performance, we compared four model scenarios: Full Model**,** Without education, Without wage, Without education and wage. Two cross-validation methods were used a 90/10 train-test split and a 80/20 train-test split. The results are summarized below

For the 90/10 train-test, the model's accuracy was highest at K = 11 and K = 13, with an accuracy of 0.7963. Removing education or wages had only a small effect on the accuracy, and in some cases slightly improved the performance. Excluding education and wages did not significantly reduce the model's accuracy, indicating that education and wages had little effect on the dependent variable.

For the 80/20 train-test , Overall accuracy is similar to the 90/10 train-test, but slightly lower at some values of K. The best and most consistent accuracy 0.7477 is found at K = 15 and K = 17 across all models.

Next, we applied the same k-NN method but used k-Fold Cross Validation, with 5 folds and 10 folds. We used the same variables and models as in the previous method. The results obtained are as follows . For the 10-fold cross validation model, the highest accuracy was found at K = 17, especially in models that excluded wages or both education and wages, with an accuracy 0.7099.

In terms of 5-fold cross validation, it is similar to that observed in the 10-fold validation, but the overall accuracy increases continuously as K increases, with the model excluding the wage variable performing best at K = 17, with an accuracy of 0.7172. Both 5-fold and 10-fold show that removing some features may improve model performance.

Next, In the Classification tree section, we use married as a response and wage, education, experience, age, ethnicity, region, gender, occupation, sector and union as predictors to find the model before pruning with 10-fold cross validation and we don't use model after pruning because the model is underfitting. Then we repeat this method to find the model by removing some parameters. So we get the table that shows nsplit, xerror and accuracy of the model before pruning and we get the best model that nsplit = 5, xerror = 0.8370 and accuracy = 0.7116 or 71.16% with a full model without wage, education and age predictors.

Finally, In the Naïve Bayes section, we use married as a response and ethnicity, region, gender, occupation, sector and union as a categorical predictor and we use Equal-Frequency Discretization techniques to transform

quantitative predictor into categorical predictor includes: wage_I, education_I experience_I and age_I to find the model with 10-fold cross validation. Then we repeat this method to find the model by removing some parameters. So we get the table that shows the accuracy of the model with 10-fold cross validation and we will get the best model with accuracy = 0.7064 or 70.64% with a model with intervals and sector predictors.

Conclusion

From the 3 methods consisting of k-NN, Classification tree, and Naive Bayes that we use to classify the married, we found that the best accuracy from these 3 methods: k-NN = 79.63%, Classification tree = 71.16% and Naive Bayes = 70.64%

Therefore, based on the results, we recommend using the k-NN with cross validation (train 90% , test 10%) because this method has more accuracy than the other two methods.

Suchunya Suwanwathanan 6405077

(Data visualization, k-NN are both code and report)

Pattarawut Ariyapruck 6405305

(Classification tree, Naïve Bayes are both code and report)