471 Midterm

Zhoumengdi Wang   zxw534

1.Preprocessing data/Perform exploratory analyses

(1) What is missing data?

The first thing we need to do is preprocessing the data. Now we need to take care of the missing data. Hmisc::describe(census_train),this function can help us find the missing data. I use Hmisc::describe to view the whole data.

According to the form, there is no missing data, but I found:

```
workclass
       n  missing distinct
   25000        0        9

? (1404, 0.056), Federal-gov (
```

```
occupation
       n  missing distinct
   25000        0       15

? (1411, 0.056), Adm-clerical
```

```
native.country
       n  missing distinct
   25000        0       42

lowest :  ?
highest:  Thailand
```

We can see that there are three variables has the value"?". Although "?" isn't regarded as missing data, but "?" means nothing in real. In fact, "?" is the missing data.

(2) How to deal with the missing data

With the missing data, there are some ways to deal with:

1. delete
2. use the highest frequency value to replace the missing data
3. use the relationships between variables

I will combine 2 and 3 to deal with the missing data. The first class which has the missing data is workclass. I think the workclass is influenced by occupation. Because different workclass is based on what kind of job you have.   So my first step is dealing with the missing data in occupation. For the missing data in occupation, I would like to use highest frequency value to replace the missing data. For occupation, the highest frequency value is "Prof-specialty" so I use "Prof-specialty" to replace the missing data in occupation.

After dealing with the missing data in occupation, I need to find the relationship between the occupation and workclass. Show the table of these two vatiables:

	Federal-	Local-gc	Never-wc	Private	Self-emp	Self-emp	State-gc	Without-pay
Adm-cler	240	220	0	2146	19	37	186	2
Armed-Fc	6	0	0	0	0	0	0	0
Craft-re	47	112	0	2418	82	412	38	1
Exec-mar	133	162	0	2064	298	290	141	0
Farming-	5	21	0	357	33	333	13	4
Handlers	17	34	0	967	1	11	6	0
Machine-	11	9	0	1436	8	28	11	1
Other-se	27	142	0	2051	19	141	97	1
Priv-hou	0	0	0	108	0	0	0	0
Prof-spe	134	532	0	1740	120	269	316	0
Protecti	22	228	0	142	2	5	84	0
Sales	11	5	0	2202	219	284	9	0
Tech-sup	52	26	0	542	3	18	44	0
Transpor	18	91	0	963	23	88	32	1

We can find that when the occupation is “Prof-specialty”, the highest frequency value of workclass is “Private”. I use private to replace the missing data in workclass.
There is one variable which has missing date, the native_country. Because it is a US census, the native country of most people in this census is US, so the highest frequency value of native country is United_States.
After dealing with the missing data and checking there is no left missing data, group two part data.

```
census_train_new = rbind(census_train_completed,census_train_missing)
```

(3) Check the variables
Close definition variables
There are two pairs variables, which has close definition, education and education.num, marital.status and relationship. Here are the tables of these two pairs:
education and education.num,

	10th	11th	12th	1st-4th	5th-6th	7th-8th	9th	Assoc-ac	Assoc-vo	Bachelor	Doctorat	HS-grad	Masters	Preschoo	Prof-sch	Some-col
1	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0
2	0	0	0	129	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	255	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	498	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	402	0	0	0	0	0	0	0	0	0
6	726	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	897	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	355	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	8064	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5585
11	0	0	0	0	0	0	0	0	1062	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	822	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	4105	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	1304	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	450	0
16	0	0	0	0	0	0	0	0	0	0	311	0	0	0	0	0

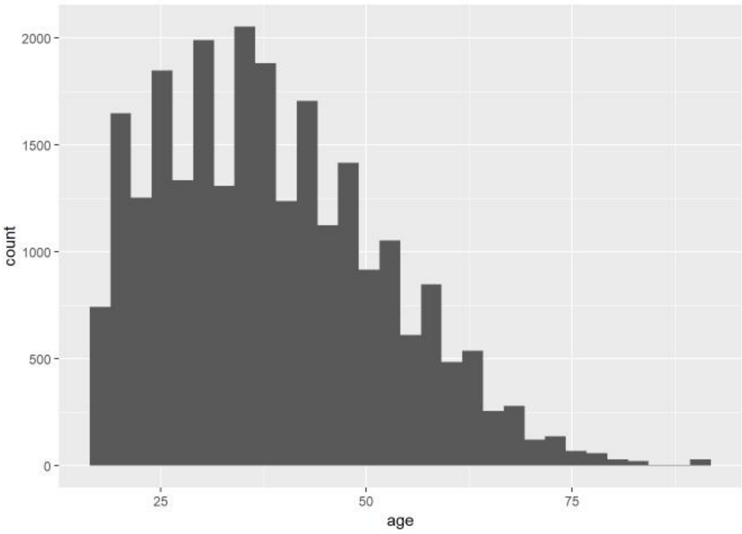
The table shows that each class of education.num matches a education level, so I think the education should be dropped.

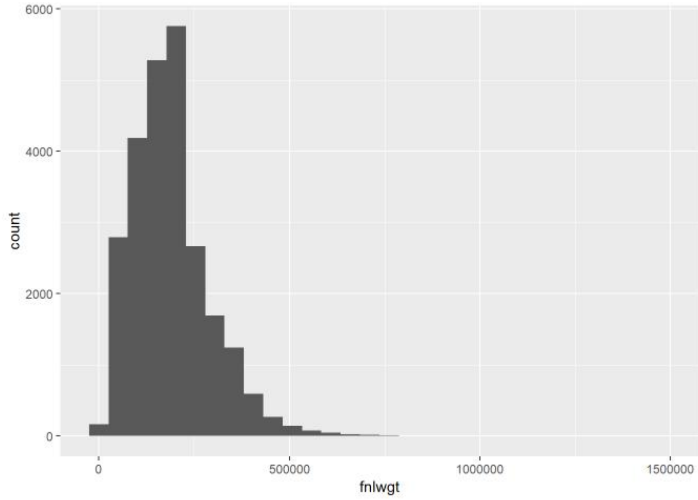
marital.status and relationship

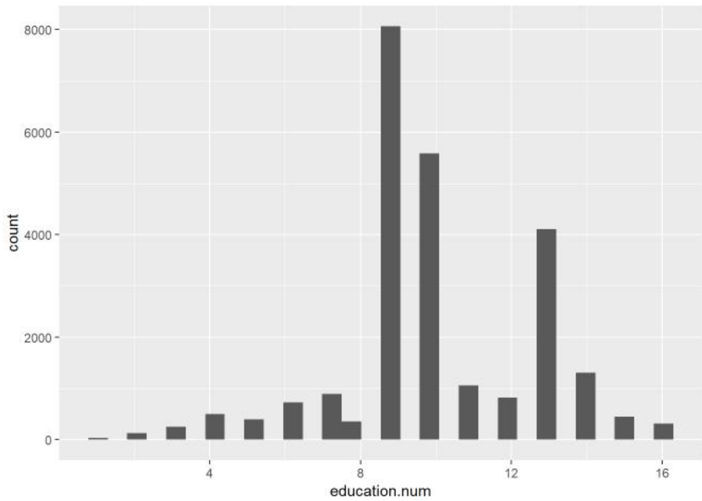
	Husband	Not-in-f	Other-re	Own-chil	Unmarrie	Wife
Divorced	0	1843	90	256	1236	0
Married-	7	0	1	1	0	9
Married-	10110	16	88	64	0	1215
Married-	0	163	26	31	107	0
Never-ma	0	3607	487	3446	656	0
Separate	0	321	44	78	347	0
Widowed	0	420	40	11	280	0

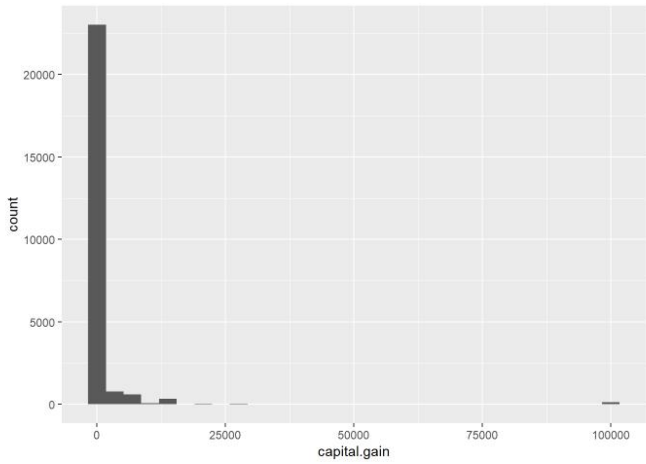
I think marital.status is more clear than the relationship, so I decide to drop relationship. And I will combine “Married-AF-spouse” and “Married-civ-spouse” which seem like similar into a single category called “Married”.

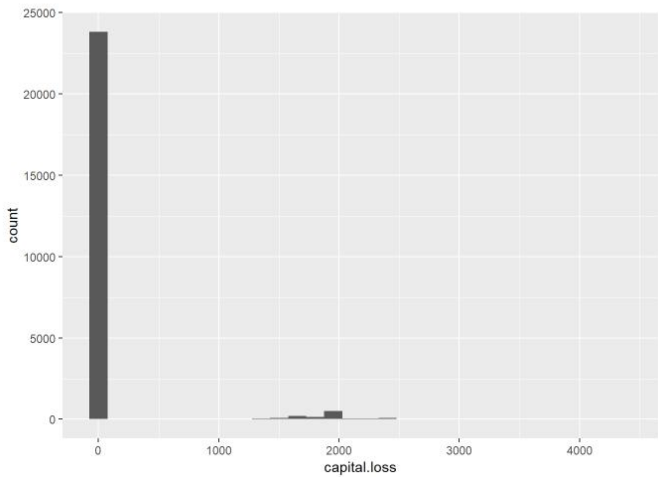
Check quantitative data

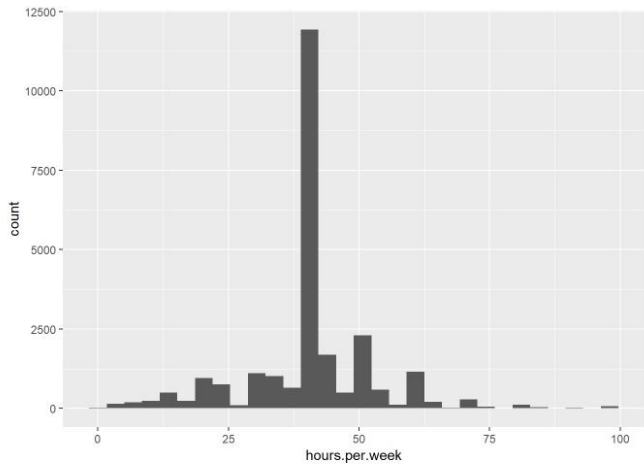






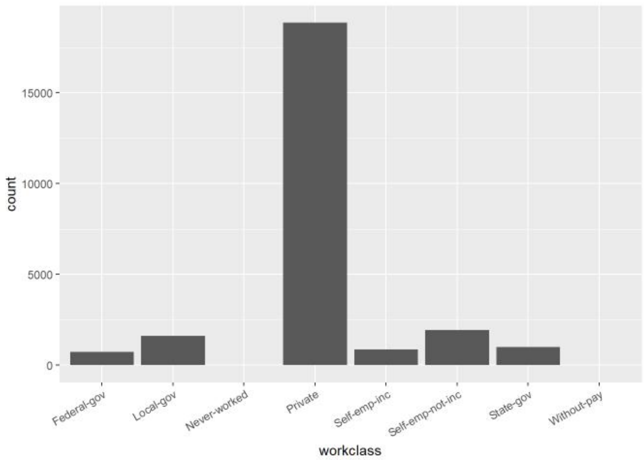


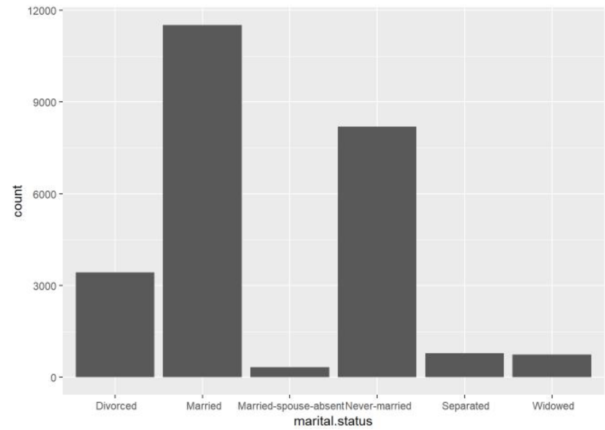


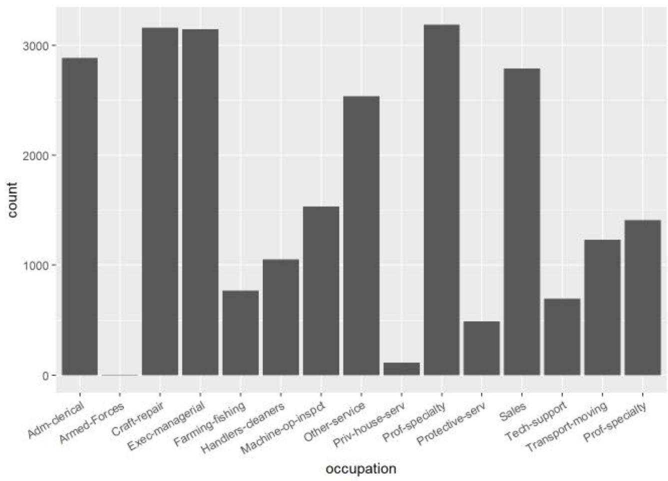


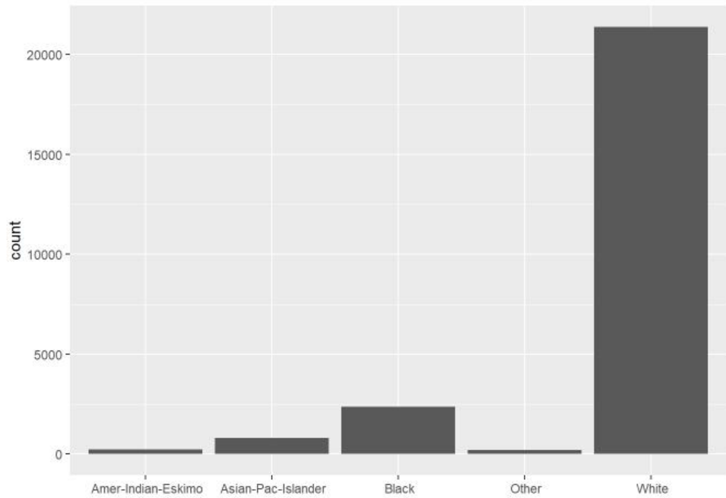
Distribution of most quantitative data is fine, however, capital.gain and capital.loss of most people is zero. I may delete them in adjusting models.

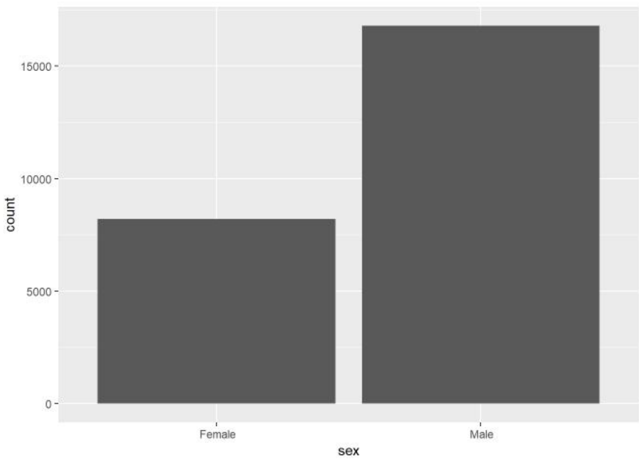
Watch qualitative data



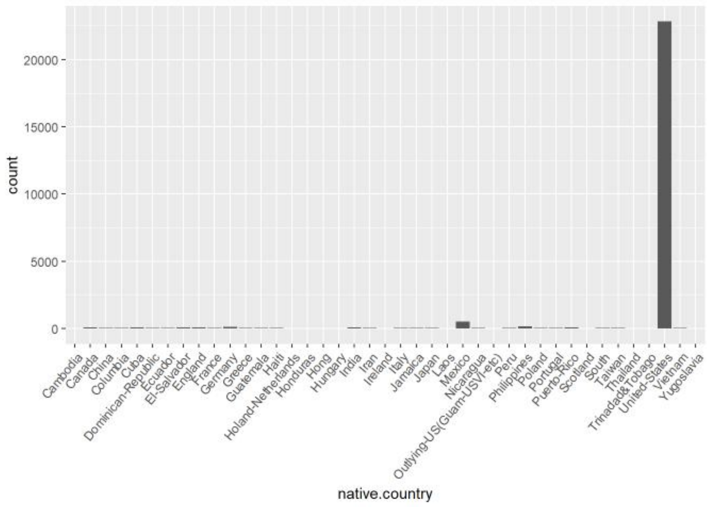








nati



For native.country, almost all people's native.country is United-Staste. So I will combine other countries into a new label"other country"

```
## [1] " Other Country" " United-States"
```

Check collinearity to find if there is some relationships between quantitative variables.

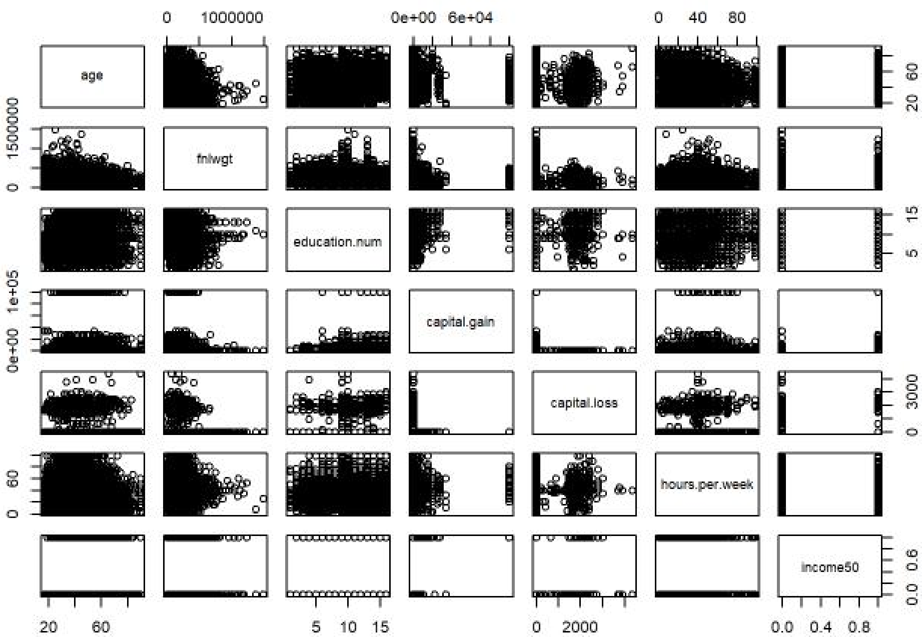
```
cor(census_train_new_quantitative)
```

```
##                        age        fnlwgt education.num  capital.gain
## age             1.00000000 -0.0754229958    0.03745462  0.0803364007
## fnlwgt         -0.07542300  1.0000000000   -0.04596356 -0.0004977309
## education.num   0.03745462 -0.0459635576    1.00000000  0.1299508553
## capital.gain    0.08033640 -0.0004977309    0.12995086  1.0000000000
## capital.loss    0.06110263 -0.0109322145    0.08314796 -0.0318820464
## hours.per.week  0.06864841 -0.0153756410    0.15357666  0.0850860033
##                capital.loss hours.per.week
## age              0.06110263     0.06864841
## fnlwgt          -0.01093221    -0.01537564
## education.num    0.08314796     0.15357666
## capital.gain    -0.03188205     0.08508600
## capital.loss     1.00000000     0.05688347
## hours.per.week   0.05688347     1.00000000
```

Most variables do not have strong collinearity, but the collinearity values of education.num-capital.gain and education.num-hours.per.week is higher than 0.1. According to ISLR, I can delete some variables which have collinearity. But,I don't want to delete education.num-hours.per.week. Because I just choose education.num rather than education, and hours.per.week is obviously related to income. How much time you work, how much money you can get.

check variables and income

Find whether there is some variable can determinate income directly or has some determinative role.



No quantitative variable is the determinative role of income.

```
table(census_train_new$workclass, census_train_new$income)
```

```
##
##                     <=50K  >50K
##   Federal-gov         447   287
##   Local-gov          1141   465
##   Never-worked          7     0
##   Private           14888  3963
##   Self-emp-inc        381   475
##   Self-emp-not-inc   1396   547
##   State-gov           732   261
##   Without-pay          10     0
```

```
table(census_train_new$marital.status, census_train_new$income)
```

```
##
##                          <=50K  >50K
##   Divorced                3070   355
##   Married                 6361  5150
##   Married-spouse-absent    302    25
##   Never-married           7844   352
##   Separated                735    55
##   Widowed                  690    61
```

```
table(census_train_new$occupation, census_train_new$income)
```

```
##
##                      <=50K  >50K
##   Adm-clerical        2503   382
##   Armed-Forces           5     1
##   Craft-repair        2441   719
##   Exec-managerial     1628  1518
##   Farming-fishing      676    94
##   Handlers-cleaners    981    70
##   Machine-op-inspct   1354   180
##   Other-service       2435   101
##   Priv-house-serv      113     1
##   Prof-specialty      1771  1415
##   Protective-serv      330   157
##   Sales               2043   743
##   Tech-support         486   211
##   Transport-moving     977   254
##  Prof-specialty       1259   152
```

```
table(census_train_new$race, census_train_new$income)
```

```
##
##                         <=50K  >50K
##   Amer-Indian-Eskimo      214    25
##   Asian-Pac-Islander      592   214
##   Black                  2077   292
##   Other                   197    19
##   White                 15922  5448
```

```
table(census_train_new$sex, census_train_new$income)
```

```
##
##           <=50K  >50K
##   Female   7310   903
##   Male    11692  5095
```

```
table(census_train_new$native.country, census_train_new$income)
```

```
##
##                  <=50K  >50K
##   Other Country   1773   396
##   United-States  17229  5602
```

No qualitative variable has the determinative role of income.

2. Creating models

In this part, I will use 10 fold cross-validation to find a better model.

```
folds <- createFolds(y=census_train_new$income50,k=10)
```

(1) Logistic Regression / LDA / KNN

```
min=100
num=0
for(i in 1:10){

  fold_test <- census_train_new[folds[[i]],]
  fold_train <- census_train_new[-folds[[i]],]

  log.reg = glm((income50 == 1) ~
                # Quantitative
                age + fnlwgt  + hours.per.week + education.num + capital.gain +
                 capital.loss+

                # Qualitative
                sex + race + marital.status + native.country + workclass + occupation,
               data=fold_train,family=binomial)

  fold_predict <- predict(log.reg,type='response',newdata=fold_test)
  fold_predict = ifelse(fold_predict>0.5,1,0)
  fold_accuracy = mean(fold_predict != fold_test$income50)


  if(fold_accuracy<min)
    {
    min=fold_accuracy
    num=i
    }

}
```

```
## [1] 0.1392
```

Well, the original logistic regression model 's performance is not bad.

```r
summary(log.reg)
```

```
##
## Call:
## glm(formula = (income50 == 1) ~ age + fnlwgt + hours.per.week +
##     education.num + capital.gain + capital.loss + sex + race +
##     marital.status + native.country + workclass + occupation,
##     family = binomial, data = fold_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.2844  -0.5017  -0.2037  -0.0467   3.7888
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -1.191e+01  3.375e+01  -0.353 0.724075
## age                      2.723e-02  1.922e-03  14.168  < 2e-16 ***
## fnlwgt                   7.123e-07  2.061e-07   3.456 0.000548 ***
## hours.per.week           3.098e-02  1.940e-03  15.964  < 2e-16 ***
## education.num            2.836e-01  1.098e-02  25.829  < 2e-16 ***
## capital.gain             3.267e-04  1.247e-05  26.209  < 2e-16 ***
## capital.loss             6.677e-04  4.470e-05  14.936  < 2e-16 ***
## sex Male                 1.277e-01  6.268e-02   2.037 0.041643 *
## race Asian-Pac-Islander  5.519e-01  2.910e-01   1.896 0.057921 .
## race Black               3.947e-01  2.743e-01   1.439 0.150197
## race Other              -8.768e-02  4.133e-01  -0.212 0.831999
## race White               5.318e-01  2.611e-01   2.037 0.041677 *
## marital.status.L        -1.005e+00  1.336e-01  -7.522 5.39e-14 ***
## marital.status.Q        -6.685e-02  1.616e-01  -0.414 0.679046
## marital.status.C         1.310e+00  1.434e-01   9.135  < 2e-16 ***
## marital.status^4        -1.355e+00  1.503e-01  -9.017  < 2e-16 ***
## marital.status^5         3.646e-01  1.842e-01   1.979 0.047761 *
## native.country.L         1.679e-01  6.461e-02   2.598 0.009364 **
## workclass.L             -4.632e+00  9.517e+01  -0.049 0.961182
## workclass.Q             -4.013e+00  9.517e+01  -0.042 0.966365
## workclass.C             -9.389e+00  1.163e+02  -0.081 0.935660
## workclass^4             -1.646e+00  4.970e+01  -0.033 0.973581
## workclass^5              1.545e+00  8.524e+01   0.018 0.985542
## workclass^6             -6.181e+00  1.257e+02  -0.049 0.960786
## workclass^7              2.971e+00  8.118e+01   0.037 0.970807
## occupation.L            -1.883e-02  8.730e-01  -0.022 0.982791
## occupation.Q             1.135e+00  8.894e-01   1.277 0.201700
## occupation.C            -5.120e-01  3.511e-01  -1.458 0.144799
## occupation^4            -1.926e+00  6.775e-01  -2.843 0.004468 **
## occupation^5            -3.123e-01  9.617e-01  -0.325 0.745354
## occupation^6             1.103e+00  1.133e+00   0.973 0.330316
## occupation^7             5.391e-01  1.184e+00   0.455 0.648904
## occupation^8            -2.351e-01  8.695e-01  -0.270 0.786900
## occupation^9            -1.106e+00  9.223e-01  -1.199 0.230461
## occupation^10            1.764e-01  3.518e-01   0.501 0.616052
## occupation^11            2.231e+00  8.143e-01   2.740 0.006152 **
## occupation^12           -7.552e-02  3.619e-01  -0.209 0.834718
## occupation^13           -1.045e+00  7.795e-01  -1.340 0.180190
## occupation^14           -1.499e+00  1.045e+00  -1.435 0.151382
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 24778  on 22499  degrees of freedom
## Residual deviance: 14407  on 22461  degrees of freedom
## AIC: 14485
##
## Number of Fisher Scoring iterations: 12
```

LDA

```
library(MASS)
min=100
num=0
for(i in 1:10){
  fold_test <- census_train_new[folds[[i]],]
  fold_train <- census_train_new[-folds[[i]],]
  lda1 = lda((income50 == 1) ~ age + workclass + fnlwgt + education.num +
                  marital.status + occupation + race + sex + capital.gain +
                  capital.loss + hours.per.week + native.country,

            data=fold_train)

  fold_predict <- predict(lda1,type='response',newdata=fold_test)
  fold_predict = ifelse(fold_predict$class == "TRUE", 1,0)
  fold_accuracy = mean(fold_predict != fold_test$income50)


  if(fold_accuracy<min)
    {
    min=fold_accuracy
    num=i
    }
}

print(min)
```

```
## [1] 0.1512
```

Little worse than logistic regression

KNN k=1,3,5,10,50,100

```
Knn_cv <- function(n,data) {

    min=100
    num=0
for(i in 1:10){
  fold_test <- data[folds[[i]],]
  fold_train <- data[-folds[[i]],]

  train.X = cbind(fold_train$age, fold_train$fnlwgt,
              fold_train$capital.gain, fold_train$capital.loss,
              fold_train$hours.per.week,fold_train$native.country,
              fold_train$workclass,fold_train$education.num,
              fold_train$marital.status,fold_train$occupation,
              fold_train$race,fold_train$sex)

  query.X = cbind(fold_test$age,fold_test$fnlwgt,
              fold_test$capital.gain, fold_test$capital.loss,
              fold_test$hours.per.week,fold_test$native.country,
              fold_test$workclass,fold_test$education.num,
              fold_test$marital.status,fold_test$occupation,
              fold_test$race,fold_test$sex)

  set.seed(1)
  knn1 = knn(train.X, query.X, fold_train$income50, k = n)

  fold_accuracy = mean(knn1 != fold_test$income50)


  if(fold_accuracy<min)
    {
    min=fold_accuracy
    num=i
    }
}

print(min)
print(num)

}
```

```r
print("K = 1")
```

```
## [1] "K = 1"
```

```r
Knn_cv(1,census_train_new)
```

```
## [1] 0.2524
## [1] 9
```

```r
print("K = 3")
```

```
## [1] "K = 3"
```

```r
Knn_cv(3,census_train_new)
```

```
## [1] 0.2252
## [1] 9
```

```r
print("K = 5")
```

```
## [1] "K = 5"
```

```r
Knn_cv(5,census_train_new)
```

```
## [1] 0.2084
## [1] 9
```

```r
print("K = 10")
```

```
## [1] "K = 10"
```

```r
Knn_cv(10,census_train_new)
```

```
## [1] 0.2004
## [1] 9
```

```r
print("K = 50")
```

```
## [1] "K = 50"
```

```r
Knn_cv(50,census_train_new)
```

```
## [1] 0.1956
## [1] 9
```

```r
print("K = 100")
```

```
## [1] "K = 100"
```

```r
Knn_cv(100,census_train_new)
```

```
## [1] 0.2032
## [1] 9
```

With K increase, the error rate is decrease, but if K keeps increasing, the error rate will increase. So I guess there always is a threshold value of K in KNN model.

The LDA and KNN both performance worse than logistic. We decided to go with the logistic regression and fine-tune it.

(2)Adjusting Logistic Regression model /improve models' performance

delete capital.loss and capital.gain

```
min=100
num=0
for(i in 1:10){
  fold_test <- census_train_new[folds[[i]],]
  fold_train <- census_train_new[-folds[[i]],]

  log.reg1 = glm((income50 == 1) ~
                 # Quantitative
                 age + fnlwgt  + hours.per.week + education.num +

                 # Qualitative
                 sex + race + marital.status + native.country + workclass +
                   occupation, data=fold_train,family=binomial)

  log.reg.probabilities1 = predict(log.reg1, fold_test, type="response")
  log.reg.predictions1 = ifelse(log.reg.probabilities1 > 0.5, 1, 0)
  fold_accuracy = mean(log.reg.predictions1 != fold_test$income50)


  if(fold_accuracy<min)
    {
    min=fold_accuracy
    num=i
    }
}

print(min)
```

```
## [1] 0.1524
```

The model performances worse before removing capital.loss and capital.gain, so I will keep them.

use cubic splines with 5 nodes for all five quantitative predictors

```
min=100
num=0
for(i in 1:10){
  fold_test <- census_train_new[folds[[i]],]
  fold_train <- census_train_new[-folds[[i]],]

  log.reg2 = glm((income50 == 1) ~
                 # Quantitative
                 rcs(age, 5) + rcs(fnlwgt, 5) + rcs(capital.gain, 5) +
                 rcs(capital.loss, 5) + rcs(hours.per.week, 5) + rcs(education.num,5)
                 +
                 # Qualitative
                 sex + race + marital.status + native.country + workclass +
                   occupation,
                data=fold_train, family=binomial)

  log.reg.probabilities2 = predict(log.reg2, fold_test, type="response")
  log.reg.predictions2 = ifelse(log.reg.probabilities2 > 0.5, 1, 0)
  fold_accuracy = mean(log.reg.predictions2 != fold_test$income50)


  if(fold_accuracy<min)
    {
    min=fold_accuracy
    num=i
    }
}

print(min)
```

```
## [1] 0.156
```

reduce the number of nodes for fnlwgt and capital.loss from 5 to 3 and added several interaction terms

```
min=100
num=0
for(i in 1:10){
  fold_test <- census_train_new[folds[[i]],]
  fold_train <- census_train_new[-folds[[i]],]

  log.reg3 = glm((income50 == 1) ~

                 # Quantitative
                 rcs(age, 5) + rcs(fnlwgt, 3) + rcs(capital.gain, 5) +
                 rcs(capital.loss, 3) + rcs(hours.per.week, 5) + rcs(education.num,5) +

                 # Qualitative
                 sex + race + marital.status + workclass + native.country
                  + occupation +

                 # Interactions
                 marital.status * age + workclass * age +
                 education.num * age + hours.per.week * workclass +
                 hours.per.week * occupation + age * hours.per.week,
                data=fold_train, family=binomial)

  log.reg.probabilities3 = predict(log.reg3, fold_test, type="response")
  log.reg.predictions3 = ifelse(log.reg.probabilities3 > 0.5, 1, 0)
  fold_accuracy = mean(log.reg.predictions3 != fold_test$income50)


  if(fold_accuracy<min)
    {
    min=fold_accuracy
    num=i
    }
}
```

```
## [1] 0.1544
```

```
min=100
num=0
for(i in 1:10){
  fold_test <- census_train_new[folds[[i]],]
  fold_train <- census_train_new[-folds[[i]],]

  log.reg4 = glm((income50 == 1) ~

                 # Quantitative
                 rcs(age, 5) + fnlwgt + rcs(capital.gain, 5) +
                 rcs(capital.loss, 5) + rcs(hours.per.week, 5) + rcs(education.num,5) +

                 # Qualitative
                 sex + race + marital.status + workclass + native.country
                  + occupation +

                 # Interactions
                 marital.status * age + workclass * age +
                 education.num * age + hours.per.week * workclass +
                 hours.per.week * occupation + age * hours.per.week,
                data=fold_train, family=binomial)

  log.reg.probabilities4 = predict(log.reg4, fold_test, type="response")
  log.reg.predictions4 = ifelse(log.reg.probabilities4 > 0.5, 1, 0)
  fold_accuracy = mean(log.reg.predictions4 != fold_test$income50)


  if(fold_accuracy<min)
    {
    min=fold_accuracy
    num=i
    }
}
```

```
## [1] 0.1568
```

After trying these model, the performance still isn't satisfactory. I'd like to try tree.

(3)Tree / RandomForest

tree

```r
min=100
num=0
for(i in 1:10){
  fold_test <- census_training[folds[[i]],]
  fold_train <- census_training[-folds[[i]],]

  library(tree)

  tree = tree(High_train~ age + fnlwgt  + hours.per.week + education.num +
                capital.gain + capital.loss+
                 # Qualitative
                 sex + race + marital.status + workclass + occupation
              +native.country,
                data=fold_train)

  tree.pred = predict(tree, fold_test, type = "class")
  score = table(tree.pred, fold_test$High_train)
  fold_accuracy = (score[1,2] + score[2,1])/(score[1,2] + score[2,1] + score[1,1] +
                                               score[2,2])


  if(fold_accuracy<min)
    {
    min=fold_accuracy
    num=i
    }
}

print(min)
```

```
## [1] 0.1468
```

```r
summary(tree)
```

```
##
## Classification tree:
## tree(formula = High_train ~ age + fnlwgt + hours.per.week + education.num +
##     capital.gain + capital.loss + sex + race + marital.status +
##     workclass + occupation + native.country, data = fold_train)
## Variables actually used in tree construction:
## [1] "marital.status" "capital.gain"   "education.num"
## Number of terminal nodes:  7
## Residual mean deviance:  0.6991 = 15730 / 22490
## Misclassification error rate: 0.156 = 3509 / 22500
```

Decision Tree performances better than LDA and KNN, but still worse than Logistic Regression

RandomForest

```
min=100
num=0
for(i in 1:10){
  fold_test <- census_training[folds[[i]],]
  fold_train <- census_training[-folds[[i]],]

  set.seed(3)
  library(randomForest)
  X = fold_train[,c(1:12)]
  y = fold_train[,15]
  rf = randomForest(X,y,mtry = 10,ntree = 100)

  rf.pred = predict(rf, fold_test, type = "class")
  score = table(rf.pred, fold_test$High_train)
  fold_accuracy = (score[1,2] + score[2,1])/(score[1,2] + score[2,1] + score[1,1] +
                                              score[2,2])


  if(fold_accuracy<min)
    {
    min=fold_accuracy
    num=i
    }
}
```

```
## [1] 0.1304
```

```
summary(rf)
```

```
##                 Length Class  Mode
## call                5  -none- call
## type                1  -none- character
## predicted       22500  factor numeric
## err.rate          300  -none- numeric
## confusion           6  -none- numeric
## votes           45000  matrix numeric
## oob.times       22500  -none- numeric
## classes             2  -none- character
## importance         12  -none- numeric
## importanceSD        0  -none- NULL
## localImportance     0  -none- NULL
## proximity           0  -none- NULL
## ntree               1  -none- numeric
## mtry                1  -none- numeric
## forest             14  -none- list
## y               22500  factor numeric
## test                0  -none- NULL
## inbag               0  -none- NULL
```

According to all models, I think the best one is randomForest, I will choose this one to do test.

3. Test Data

Do the same reprocessing as training data, and use randomForest to do the test.

```
set.seed(0)
test_rf.pred = predict(rf, census_test, type = "class")
score = table(test_rf.pred, census_test$High_train)
fold_accuracy = (score[1,2] + score[2,1])/(score[1,2] + score[2,1] + score[1,1] +
                                           score[2,2])
fold_accuracy
```

```
## [1] 0.1596338
```

The final error rate is around 0.16. Not bad!

After doing this midterm project, I think preprocessing is very important. We have to view the data by eyes firstly. If we input the data directly, some missing data won't show off. That may cause big problem in fitting data and testing data.

And I think I should do more data processing, I should continue merging some variables.

Although the best model in my project is randomForest, Logistic Regression also performances well. Logistic Regression is a basic classifier, but it is still very useful, and Logistic Regression can do many extensions.