**PetFinder.my Animals Adoptability Analysis in R**

Work by: Xiao Li Date: 02/26/2019

**Executive Summary**

PetFinder.my has been Malaysia’s leading animal welfare platform since 2008, with a database of more than 150,000 animals. PetFinder collaborates closely with animal lovers, media, corporations, and global organizations to improve animal welfare. Animal adoption rates are strongly correlated to the metadata associated with their online profiles, such as descriptive text and photo characteristics. As one example, PetFinder is currently experimenting with a simple AI tool called the Cuteness Meter, which ranks how cute a pet is based on qualities present in their photos.

This project would like to apply R on the analysis of the dataset including PetFinder.my’s past animal profiles with adoption speed to understand the key factors could increasing the adoption speed of animals, which would be helpful to notify the organization that what kind of animals might need more advertising to increase their adoptability.

Generally, most animals have to wait at least one week for being adopted but younger animals are much easier to be adopted on the same day their profiles listed. Also, even though there no obvious difference between the adoption speed of dogs and cats, the relationship between adoption speed and detailed breeds still tell that cats are easier to be adopted on the same day their profiles listed. At the same time, the later the animal was adopted, the more videos and photos were uploaded into its profile for promoting it, which might tell that current visual materials are helpless for increasing the adoptability of the animals. And for predicting the adoption speed, the age and breed of animals play the key roles. What’s more, most descriptions about animals are too general, which is hard to effectively attract adopters and fasten the adoption speed of animals.

**Analysis Plan**

The original dataset records 14993 detailed profiles of stray animals waiting for adoption in Malaysia, which is described by 24 variables including the target variable, ‘AdoptionSpeed’. Since a few profiles in the dataset represent a group of animals instead of just one, for effective and clear analysis, this analysis only worked on those profiles including one animal, which takes 11565 rows totally in the entire dataset. After cleaning several independent variables meaningless and useless to the general classification, the final dataset has total 18 variables, without ‘Description’ also, and 11565 observations. The entire analysis consists of study about data pattern and supervised analysis. Since the target variable, ‘AdoptionSpeed’, of this analysis is a categorical variable with 5 levels, the supervised analysis here is a classification problem. Apart from that, since each animal profile has a text description about the characteristics and rescued experience, considering about the speed limitation of R Studio, a English word cloud of ‘Description’ was generated for understanding the frequent words applied to have a rough idea about the quality and efficiency of these description instead of running classification with sentiment analysis which might be worked independently later.

**Results**

After visualizing the data pattern of each variables with ggplot in R, we can see that:

* Generally, most animals have to wait at least one week to be adopted.
* There are more stray dogs than cats in the dataset; and dogs have a slightly slower adoption speed than cats.
* There are more female animals than males, whose adoption speeds have no difference.
* Most animals are very young or even just born, but easier to be adopted immediately after their profiles listed online.
* Most profiles don’t have videos as vivid introduction but have at least one photo, whose amounts have a negative relationship with adoption speed of animals.
* Most animals have a medium maturity size and short fur length. Also, extra-large size animals usually have to wait 1 week to 3 months to be adopted.
* The healthy condition of most animals could be confirmed and, not surprised, healthy animals usually have a quicker adoption speed.

Apart from that, for finding out the most accurate classification method to predict the adoption speed, these methods and their predictive accuracy are compared:

* Logistic Regression: model accuracy is 0.3451 and test accuracy is 0.3587, which is slightly overfitting and too low to tell the logistic regression model could accurately predict the adoption speed of animals.
* LDA: model accuracy is 0.3467 and test accuracy is 0.3651, which is slightly overfitting and too low to tell the model could accurately predict the adoption speed of animals.
* QDA: model accuracy is 0.3669 and test accuracy is 0.3587, which is not overfitting but still too low to tell this QDA model could accurately predict the adoption speed of animals.
* Classification Decision Tree: after applying cross validation to get the best decision tree, model accuracy is 0.3642 and test accuracy is 0.3666, which is slightly overfitting and too low to tell this tree could accurately predict the adoption speed of animals. However, by plotting this decision tree, we can know that the age and breed of animals are critical to their adoption speed.
* Bagging: by using all 17 independent variables as predictors, the model accuracy is 0.388 and the test accuracy is 0.3828, which is not overfitting but still low to tell the model accurate.
* Random Forest (with cross validation): the best model only requiring 2 predictors has model accuracy of 0.3944 and test accuracy of 0.4012, which is slightly overfitting but acceptable and also the highest-accurate method.

Finally, by reading the word could of English ‘Description’, most descriptions are written with general adjectives, such as ‘playful’ and ‘friendly’, and basic healthy conditions of the animals, Such words are too general to be able to attract or encourage more adopters to take actions to adopt animals in a short term after these animals’ profiles listed online.

**Conclusion**

Currently, the random forest with 2 predictors should be the best method for predicting adoption speed. Also, it’s necessary to give more thought to digging out the characteristics and advantages of each animal for writing more effective description to increase its adoption speed.

**Appendix: R Code**

---

title: "R Project"

author: "Xiao Li"

date: "2/26/2019"

output: pdf\_document

---

```{r setup, include=FALSE}

knitr::opts\_chunk$set(echo = TRUE)

```

```{r, include=FALSE}

install.packages("tm")

install.packages("wordcloud")

install.packages("RColorBrewer")

install.packages("NLP")

```

```{r libs, include=FALSE}

library(readr)

library(tidyverse)

library(gapminder)

library(dplyr)

library(ggplot2)

library(broom)

library(glmnet)

library(caret)

library(ISLR)

library(janitor)

library(rpart)

library(rpart.plot)

library(partykit)

library(nnet)

library(MASS)

library(wordcloud)

library(RColorBrewer)

library(randomForest)

library(gbm)

library(plotROC)

library(kernlab)

library(nnet)

library(plyr)

theme\_set(theme\_bw())

```

####Data Understanding

This training dataset, provided by PetFinder.my that is a Malaysia’s leading animal welfare platform, records 14993 detailed profiles of stray animals waiting for adoption in Malaysia. The data fields of this dataset are shown as below, there are 24 fields totally

```{r}

df <- read\_csv("train.csv")

str(df)

```

####Data Preparation

Since there are some variables in the dataset is useless and meaningless for predicting adoption speed, by reading first 6 rows of the original dataset to understand each variable, I would like to remove `Name`, `PetID`, `Quantity`, `RescuerID` and `State`. Also, since this project mainly focuses on predictive analytics instead of sentiment analysis, I decided to use the dataset without including the `Description` variable.

```{r}

head(df)

```

By looking at the number of data at each level of animal quantity, since the `Quantity = 1` has 11565 rows that is more than other levels' data size, I would like to remove the profile data with `Quantity > 1` for focusing on analyzing the profile data with just one animal, which could make the analysis more effective and workable.

```{r}

count(df$Quantity)

```

```{r}

df <- df%>%

filter(Quantity == "1")

head(df)

```

```{r}

df <- subset(df, select = -c(Name, PetID, Quantity, State, RescuerID))

head(df)

```

Now, our final dataset for building analysis and training models has 11565 observations and 19 variables.

###Word Cloud for `Description` of pets' profiles

Let's remove all non-English description first.

```{r}

library(stringi)

Description <- stringi::stri\_trans\_general(df$Description, "latin-ascii")

```

Then, clean our `Description`.

```{r}

library(tm)

description<-Corpus(VectorSource(Description))

```

```{r}

description <- tm\_map(description,stripWhitespace)

description <- tm\_map(description,tolower)

description <- tm\_map(description,removeNumbers)

description <- tm\_map(description,removePunctuation)

description <- tm\_map(description,removeWords, stopwords("english"))

description <- tm\_map(description, removeWords,

c("and","the","our","that","for","are","also","more","has","must","have","should","this","with","dog","cat","dogs","cats","home","adoption","can","give"))

```

```{r}

tdm\_desc <- TermDocumentMatrix (description) #Creates a TDM

TDM <- as.matrix(tdm\_desc) #Convert this into a matrix format

v <- sort(rowSums(TDM), decreasing = TRUE) #Gives you the frequencies for every word

summary(v)

```

Now, we can create the colorful word cloud to get a glance of those frequently used terms how these animals described in their profiles.

```{r}

wordcloud (description, scale=c(3.5,0.5), max.words=100, random.order=FALSE, rot.per=0.35, use.r.layout=FALSE, colors=brewer.pal(8, "Dark2"))

```

Right now, let's remove the `Description` from original dataset for the following general statistical analysis.

```{r}

df1 <- subset(df, select = -c(Description))

```

###Data Audit

```{r}

cor(df1)

```

```{r}

str(df1)

```

#####continuous variables vs. target

Let's plot the relationships between continuous variables and target variable, `AdoptionSpeed`.

- `Age(in months)` vs. `AdoptionSpeed`: by reading the scatter plot, it's obviously to see that younger animals would be more quickly to be adopted.

```{r}

p <- ggplot(df1, aes(x = Age, y = AdoptionSpeed))

p + geom\_point(alpha = .15, col = "#6e0000")

```

- `Fee` vs. `AdoptionSpeed`: by reading the plot, even though most adoptions of animals are free, we still can see such data pattern that most animals that were adopted immediately on the same day they listed have very low adoption fee.

```{r}

p <- ggplot(df1, aes(x = Fee, y = AdoptionSpeed))

p + geom\_point(alpha = .15, col = "#6e0000")

```

- `VideoAmt` vs. `AdoptionSpeed`: such plot actually might reflect that the later the animal is adopted, the more amount of videos for promoting it.

```{r}

p <- ggplot(df1, aes(x = VideoAmt, y = AdoptionSpeed))

p + geom\_point(alpha = .15, col = "#6e0000")

```

- `PhotoAmt` vs. `AdoptionSpeed`: Here we see the same relationship that the later the animal is adopted, the more photo amount uploaded for encouraging the adoption of it.

```{r}

p <- ggplot(df1, aes(x = PhotoAmt, y = AdoptionSpeed))

p + geom\_point(alpha = .15, col = "#6e0000")

```

#####continous variable

- `Age`: Most animals abandoned are very young or even baby.

```{r}

ggplot(df1, aes(x = Age)) +

geom\_histogram(fill = "dodgerblue4")

```

- `Fee`: Most adoptions are free and even the charged adoptions are still very cheap.

```{r}

ggplot(df1, aes(x = Fee)) +

geom\_histogram(fill = "dodgerblue4")

```

- `VideoAmt`: Most profiles of animals don't have videos as vivid introductions.

```{r}

ggplot(df1, aes(x = VideoAmt)) +

geom\_histogram(fill = "dodgerblue4")

```

- `PhotoAmt`: Most profiles have at least one picture of the animal.

```{r}

ggplot(df1, aes(x = PhotoAmt)) +

geom\_histogram(fill = "dodgerblue4")

```

#####categorical variables vs. target

Let's plot the relationships between categorical variables and target variable, `AdoptionSpeed`.

- `Type`(1 = Dog, 2 = Cat) vs. `AdoptionSpeed`: There are no difference between the adoption speed of dogs and cats.

```{r}

p <- ggplot(df1, aes(x = Type, y = AdoptionSpeed))

p + geom\_point(alpha = .15, col = "#6e0000")

```

- `Breed1` vs. `AdoptionSpeed`: Since the `Breed1` less than 240 and equals to 307 are dogs, others are cats. We can see that, except `Breed1` equals to 307, dogs generally have a slightly slower adoption speed than cats.

```{r}

Breed <- count(df1$Breed1)

Breed <- Breed[order(-Breed$freq),]

Breed

p <- ggplot(df1, aes(x = Breed1, y = AdoptionSpeed))

p + geom\_point(alpha = .15, col = "#6e0000")

```

- `Gender`(1 = Male, 2 = Female) vs. `AdoptionSpeed`: There are no difference between the adoption speed of different genders.

```{r}

p <- ggplot(df1, aes(x = Gender, y = AdoptionSpeed))

p + geom\_point(alpha = .15, col = "#6e0000")

```

- `Color1`&`Color2`&`Color3` vs. `AdoptionSpeed`: There are no difference among the adoption speed of different colors of animals.

```{r}

p1 <- ggplot(df1, aes(x = Color1, y = AdoptionSpeed))

p1 + geom\_point(alpha = .15, col = "#6e0000")

p2 <- ggplot(df1, aes(x = Color2, y = AdoptionSpeed))

p2 + geom\_point(alpha = .15, col = "#6e0000")

p3 <- ggplot(df1, aes(x = Color3, y = AdoptionSpeed))

p3 + geom\_point(alpha = .15, col = "#6e0000")

```

- `MaturitySize` (1 = Small, 2 = Medium, 3 = Large, 4 = Extra Large, 0 = Not Specified) vs. `AdoptionSpeed`: Animals with extra large maturity size usually have to wait 1 week to 3 months for being adopted.

```{r}

p <- ggplot(df1, aes(x = MaturitySize, y = AdoptionSpeed))

p + geom\_point(alpha = .15, col = "#6e0000")

```

- `FurLength`(1 = Short, 2 = Medium, 3 = Long, 0 = Not Specified) vs. `AdoptionSpeed`: There are no difference among the adoption speed of different fur lengths of animals.

```{r}

p <- ggplot(df1, aes(x = FurLength, y = AdoptionSpeed))

p + geom\_point(alpha = .15, col = "#6e0000")

```

- `Vaccinated`(1 = Yes, 2 = No, 3 = Not Sure) vs. `AdoptionSpeed`: There are no difference among the adoption speed of different vaccinated conditions of animals.

```{r}

p <- ggplot(df1, aes(x = Vaccinated, y = AdoptionSpeed))

p + geom\_point(alpha = .15, col = "#6e0000")

```

- `Dewormed`(1 = Yes, 2 = No, 3 = Not Sure) vs. `AdoptionSpeed`: There are no difference among the adoption speed of different dewormed conditions of animals.

```{r}

p <- ggplot(df1, aes(x = Dewormed, y = AdoptionSpeed))

p + geom\_point(alpha = .15, col = "#6e0000")

```

- `Sterilized`(1 = Yes, 2 = No, 3 = Not Sure) vs. `AdoptionSpeed`: There are no difference among the adoption speed of different sterilized conditions of animals.

```{r}

p <- ggplot(df1, aes(x = Sterilized, y = AdoptionSpeed))

p + geom\_point(alpha = .15, col = "#6e0000")

```

- `Health`(1 = Healthy, 2 = Minor Injury, 3 = Serious Injury, 0 = Not Specified) vs. `AdoptionSpeed`: It's obvious to see that seriously injured animals usually have to wait more than at least 1 month for being adopted.

```{r}

p <- ggplot(df1, aes(x = Health, y = AdoptionSpeed))

p + geom\_point(alpha = .15, col = "#6e0000")

```

#####categorical variables

Since the target is also categorical, here is the data pattern of `AdoptionSpeed`:

We can know that most adoptions happened after 1 month of the animals 'profiles created. And there still have lots of animals not adopted after even 100 days.

- The values of `AdoptionSpeed`:

0 - Pet was adopted on the same day as it was listed.

1 - Pet was adopted between 1 and 7 days (1st week) after being listed.

2 - Pet was adopted between 8 and 30 days (1st month) after being listed.

3 - Pet was adopted between 31 and 90 days (2nd & 3rd month) after being listed.

4 - No adoption after 100 days of being listed. (There are no pets in this dataset that waited between 90 and 100 days).

```{r}

p <- ggplot(df1, aes(x = AdoptionSpeed)) + geom\_bar(fill = "dodgerblue4")

p

```

- `Type`(1 = Dog, 2 = Cat): There are more dogs than cats abandoned.

```{r}

p <- ggplot(df1, aes(x = Type)) + geom\_bar(fill = "dodgerblue4")

p

```

- `Gender`(1 = Male, 2 = Female): There are more female animals than males.

```{r}

p <- ggplot(df1, aes(x = Gender)) + geom\_bar(fill = "dodgerblue4")

p

```

- `MaturitySize`(1 = Small, 2 = Medium, 3 = Large, 4 = Extra Large, 0 = Not Specified): Most animals have medium maturity size.

```{r}

p <- ggplot(df1, aes(x = MaturitySize)) + geom\_bar(fill = "dodgerblue4")

p

```

- `FurLength`(1 = Short, 2 = Medium, 3 = Long, 0 = Not Specified): Most animals are short fur.

```{r}

p <- ggplot(df1, aes(x = FurLength)) + geom\_bar(fill = "dodgerblue4")

p

```

- `Vaccinated`(1 = Yes, 2 = No, 3 = Not Sure): Most animals' vaccinated conditions could be confirmed.

```{r}

p <- ggplot(df1, aes(x = Vaccinated)) + geom\_bar(fill = "dodgerblue4")

p

```

- `Dewormed`(1 = Yes, 2 = No, 3 = Not Sure): Most animals have already been dewormed.

```{r}

p <- ggplot(df1, aes(x = Dewormed)) + geom\_bar(fill = "dodgerblue4")

p

```

- `Sterilized`(1 = Yes, 2 = No, 3 = Not Sure): Most animals haven't been sterilized yet.

```{r}

p <- ggplot(df1, aes(x = Sterilized)) + geom\_bar(fill = "dodgerblue4")

p

```

- `Health`(1 = Healthy, 2 = Minor Injury, 3 = Serious Injury, 0 = Not Specified): Most animals are healthy.

```{r}

p <- ggplot(df1, aes(x = Health)) + geom\_bar(fill = "dodgerblue4")

p

```

Data patter:

- More dogs than cats;

- Much more baby animals;

- more female animals;

- many black and browm animals;

- medium maturity is the most;

- short fur length is the most;

- most animals' vaccinated situation could be sured;

- most animals have already been dewormed;

- most animals haven't been sterilized;

- most animals are healthy;

- most animals' adoption fee is 0;

- most animals' profiles don't have video;

- most profiles include 1-5 photos;

- a few animals were adopted immediatly on the same day of being listed.

##Modeling: Supervised Analytics

####Logistic Regression

```{r}

adop\_multinom <- multinom(AdoptionSpeed ~ .,data = df1)

tidy(adop\_multinom)

```

Let's look at the predictive accuracy of this multinomial logistic regression for the testing dataset.

```{r}

set.seed(2)

inTraining <- createDataPartition(df1$AdoptionSpeed, p = .7, list = F)

training <- df1[inTraining, ]

testing <- df1[-inTraining, ]

train\_per\_multinom <- multinom(AdoptionSpeed ~ ., data = training)

multinom\_training <- training %>%

mutate(fits = predict(train\_per\_multinom)) %>%

mutate(multinom\_accuracy = if\_else(AdoptionSpeed == fits, 1, 0))

multinom\_accuracy <- sum(multinom\_training$multinom\_accuracy==1)/nrow(multinom\_training)

multinom\_accuracy

```

Test Accuracy:

```{r}

multinom\_test\_pred <- predict(train\_per\_multinom, newdata = testing)

multinom\_testing <- testing %>%

mutate(AdoptionSpeed\_pred = multinom\_test\_pred) %>%

mutate(multinom\_pred\_accuracy = if\_else(AdoptionSpeed == AdoptionSpeed\_pred, 1, 0))

head(multinom\_testing)

```

```{r}

multinom\_accuracy\_test <- sum(multinom\_testing$multinom\_pred\_accuracy==1)/nrow(multinom\_testing)

multinom\_accuracy\_test

```

The testing accuracy generated by applying this multinomial logistic regression is only `0.3587`, which tells the overfitting happened with this logistic regression model.

####LDA

```{r}

set.seed(2)

inTraining <- createDataPartition(df1$AdoptionSpeed, p = .7, list = F)

training <- df1[inTraining, ]

testing <- df1[-inTraining, ]

adop\_lda <- lda(AdoptionSpeed ~ ., data = training)

fits <- predict(adop\_lda)

confMat\_lda <- table(fits$class, training$AdoptionSpeed)

confMat\_lda

```

Model Accuracy:

```{r}

accuracy\_lda <- sum(418, 907, 316, 1166)/sum(confMat\_lda)

accuracy\_lda

```

Test Accuracy:

```{r}

test\_preds <- predict(adop\_lda, newdata = testing)

confMat\_lda\_test <- table(test\_preds$class, testing$AdoptionSpeed)

confMat\_lda\_test

```

```{r}

accuracy\_lda\_test <- sum(178, 377, 158, 553)/sum(confMat\_lda\_test)

accuracy\_lda\_test

```

Therefore, the test accuracy of this LDA model is `0.3651`, which is slightly overfitting.

####QDA

```{r}

adop\_qda <- qda(AdoptionSpeed ~ ., data = training)

fits <- predict(adop\_qda)

confMat\_qda <- table(fits$class, training$AdoptionSpeed)

confMat\_qda

```

Model Accuracy: this QDA model is better than LDA model.

```{r}

accuracy\_qda <- sum(14, 831, 840, 326, 960)/sum(confMat\_qda)

accuracy\_qda

```

Test Accuracy:

```{r}

test\_preds <- predict(adop\_qda, newdata = testing)

confMat\_qda\_test <- table(test\_preds$class, testing$AdoptionSpeed)

confMat\_qda\_test

```

```{r}

accuracy\_qda\_test <- sum(2, 342, 345, 136, 419)/sum(confMat\_qda\_test)

accuracy\_qda\_test

```

So, the test accuracy of this QDA model is `0.3587`, which is lower than the test accuracy of the LDA model but not overfitting.

####Classification Tree

First, let run a basic decision tree

```{r}

df2 <- df1 %>%

mutate(AdoptionSpeed = as.factor(AdoptionSpeed))

```

```{r}

adop\_tree <- rpart(AdoptionSpeed ~ . , df2)

prp(adop\_tree)

plot(as.party(adop\_tree))

adop\_tree

printcp(adop\_tree)

```

```{r}

set.seed(2)

inTraining <- createDataPartition(df2$AdoptionSpeed, p = .7, list = F)

training <- df2[inTraining, ]

testing <- df2[-inTraining, ]

fit\_control <- trainControl(method = "repeatedcv",

number = 10,

repeats = 10)

cv\_adop\_tree <- train(AdoptionSpeed ~ .,

data = training,

method = "rpart",

trControl = fit\_control)

plot(cv\_adop\_tree)

```

```{r}

cv\_adop\_tree #cp: method for choosing final nodes

```

```{r}

plot(as.party(cv\_adop\_tree$finalModel))

```

```{r}

tree\_test\_pred <- predict(cv\_adop\_tree, newdata = testing)

confMat\_tree <- table(tree\_test\_pred, testing$AdoptionSpeed)

confMat\_tree

```

```{r}

accuracy\_tree\_test <- sum(89, 544, 638)/sum(confMat\_tree)

accuracy\_tree\_test

```

The predictive accuracy on testing dataset of this decision tree is `0.3666`, which is higher than the logistic regression model but still too low.

Now, let's try bagging and random forest for finding the best decision tree

####Bagging

```{r}

set.seed(2)

inTraining <- createDataPartition(df2$AdoptionSpeed, p = .7, list = F)

training <- df2[inTraining, ]

testing <- df2[-inTraining, ]

```

```{r}

set.seed(10982)

adop\_bag <- randomForest(AdoptionSpeed ~ ., data = training, mtry = 17) # mtry: the number of predictors to try.

adop\_bag # default setting of the number of tree is 500, which is pretty enough.

```

```{r}

accuracy\_bag <- 1 - .612

accuracy\_bag

```

Test Accuracy:

```{r}

test\_preds <- predict(adop\_bag, newdata = testing)

test\_df\_bag <- testing %>%

mutate(y\_hat\_bag = test\_preds,

accuracy = if\_else(y\_hat\_bag==AdoptionSpeed,1,0))

accuracy\_bag\_test <- sum(test\_df\_bag$accuracy==1)/nrow(test\_df\_bag)

accuracy\_bag\_test

```

The test accuracy of this bagging model is `0.3828`.

####Random Forest

```{r}

set.seed(1982)

rf\_adop\_cv <- train(AdoptionSpeed ~ .,

data = training,

method = "rf",

ntree = 100,

importance = T, # output the 'importance'

tuneGrid = data.frame(mtry = 1:17)) # important piece for tree function training

rf\_adop\_cv

```

```{r}

plot(rf\_adop\_cv)

```

Therefore, for highest accuracy of prediction, the tree with `mtry = 2` is chosen as the best tree model.

```{r}

set.seed(1982)

rf\_adop\_2 <- randomForest(AdoptionSpeed ~.,

data = training,

mtry = 2)

rf\_adop\_2

```

```{r}

accuracy\_2 <- 1- .6056

accuracy\_2

```

Test Accuracy:

```{r}

test\_preds <- predict(rf\_adop\_2, newdata = testing)

test\_df\_rf <- testing %>%

mutate(y\_hat\_rf = test\_preds,

accuracy = if\_else(y\_hat\_rf==AdoptionSpeed,1,0))

accuracy\_rf\_2 <- sum(test\_df\_rf$accuracy==1)/nrow(test\_df\_rf)

accuracy\_rf\_2

```

After comparing the test accuracy of bagging and random forest with `mtry=2`, the random forest model get better test accuracy but is slightly overfitting which could be accepted.