MSiA 420 Predictive Analysis II Final Project: Predicting “Match” in a Speed Dating Experiment

Team: Xuefei Liu, Duyun Tan, Xiaohan Wang, Zihao Zhang

1. Executive Summary

For this project, the objective is to predict whether the participant successfully matches with a partner (yes or no) and identify the important predictors when it comes to a match. Our dataset includes participant information related to demographics, dating habits, lifestyles, an attribute evaluation questionnaire taken when the participants sign up, ratings for their partners during the 4 minute interactions, and each participant's response on if they would like to match with the partner they met.

To approach the objective, we deployed a number of classification models including Simple Logistic regression, Ridge logistic regression, Neural Network, Classification Tree, Generalized Additive Models (GAM), Random Forest, Gradient Boosting Trees and Support Vector Machine. After analyzing the strengths, limitations and performances of the above models, we built another ensemble model by aggregating the prediction of best three models (Gradient Boosting Tree, Generalized Additive Models and Neural Network) in terms of predictive power.

By comparing the predicting power of all the models (including the ensemble model) we built, we chose gradient boosting trees as our final model. Our final model provides reasonable predictions for each participant by reaching a misclassification rate of 0.1420 based on 5-fold cross validation with 3 replicates on the full dataset.

2. Introduction

In today's busy world, many people have turned to speed dating as a solution that allows one to look for potential romantic partners in a short time. To research more on speed dating, Professor Ray Fishman and Sheena Lyengar from Columbia Business School compiled the famous speed dating dataset from a series of experimental speed dating events from 2002 to 2004. The dataset contains information on the participants' background as well as their responses to a few questionnaires during and after the experiment. Using this dataset, our team built a predictive classification model to predict if two participants both decided to match with each other.

This project is beneficial to the growing dating app industry, for we can get a better understanding of the influential factors in speed dating, and potentially help dating apps grow their matching rate.

3. Exploratory Data Analysis And Data Cleaning

3.1 Data Overview

The primary dataset (Fisman and Iyengar 2004) for this project was gathered from participants in experimental speed dating events from 2002-2004. During the events, the attendees would have a four minute "first date" with every other participant of the opposite sex. At the end of their four minutes, participants were asked if they would like to see their date again. They were also asked to rate their date on six attributes: Attractiveness, Sincerity, Intelligence, Fun, Ambition, and Shared Interests.

The dataset also includes questionnaire results gathered from participants at different points in the process. These fields include: demographics, dating habits, self-perception across key attributes, beliefs on what others find valuable in a mate, and lifestyle information.

The initial dataset contains 8379 observations and 195 variables on the above-mentioned information collected during the experiment.

3.2 Data Cleaning

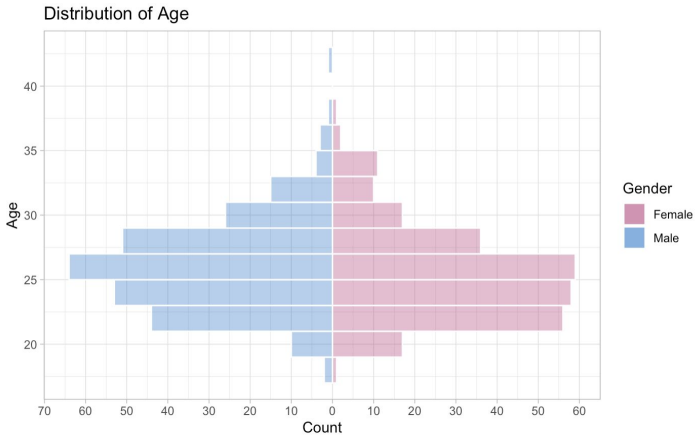
To extract the most relevant and informative variables, we dropped the variables with missing value rate higher than 0.5. For the remaining variables, we used the median imputation method to fill in the missing values with the median values of the corresponding variables. We also excluded the data gathered during the half-way-through-dating survey and the third survey, which was taken 3-4 weeks after the experiments, because we were more interested in the timely responses of the participants after their speed-dating events.

Additionally, we scaled some of the questionnaire results to make sure the survey responses are all on the same scale, since some questionnaires required participants to rate their partners with the scales different from other questionnaires. For example, in most of the events, participants were given 100 points in total to distribute to 6 attributes of their partners with priority; some groups, however, were asked to rank the importance of the attributes on a 1-10 scale. There were also cases when participants' scores did not add up to 100. In the second and third cases above, we scaled the data to form 0-100 score distributions, in order to compare the responses across questionnaires.

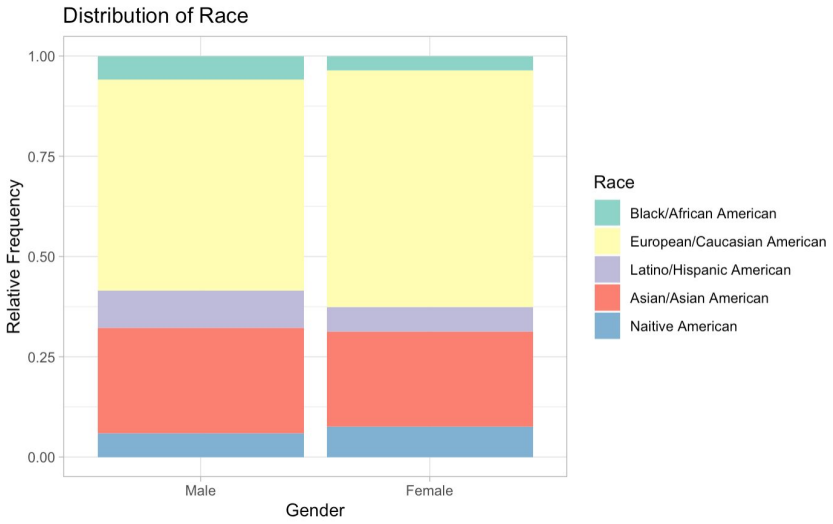
3.3 Exploratory Data Analysis

We conducted exploratory analysis on the distributions across genders of key variables, including intended participants' age, race and career.

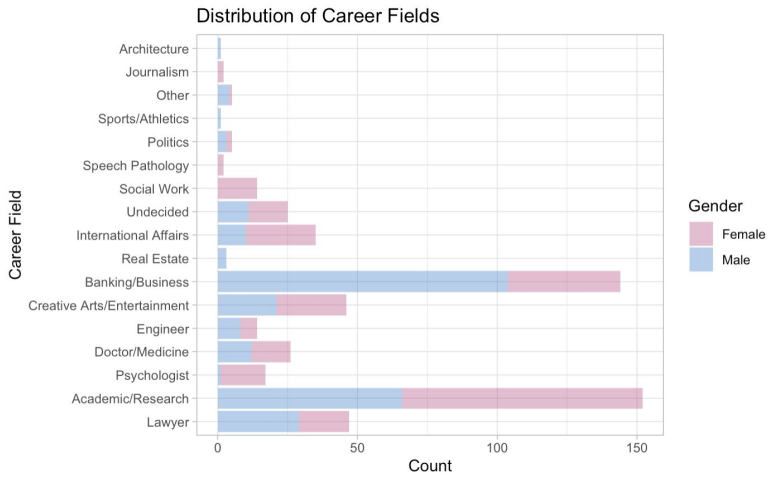
The plot below showed that nearly all of the participants were between 18 and 35 years old, which was representative for typical dating app users. The number of participants within different age buckets were relatively equal between males and females.



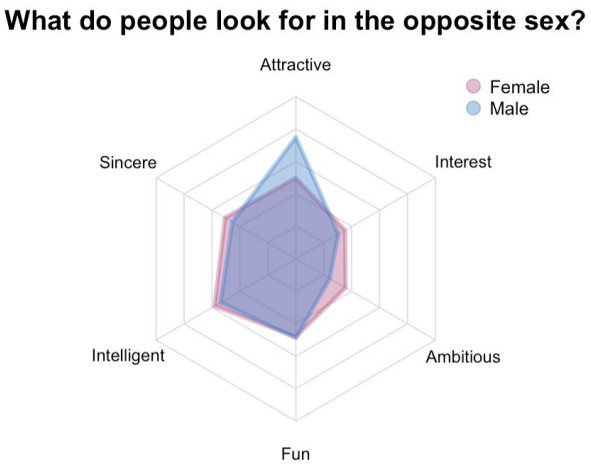
Nearly half of the participants were European/Caucasian Americans, and Asian/Asian American was the second most popular race in the experiment. The rest of the racial groups, Black/African American, Latino/Hispanic American and Native American, had smaller and similar numbers of participants.



There was a large variation within the distribution of participants across different careers. Among all the career categories of participants, the most frequent ones were Banking/Business and Academic/Research. Since the participants were all graduate students from the Columbia University at New York City, this large variation might be reasonable. However, for each of the career fields in the plot below, there was no large difference between the number of females and the number of males working in that field. The numbers of female and male participants were almost balanced across career fields.



We also explored the attributes that participants looked for in their partners, and whether there was a distinction between male and female participants. As the radar chart below indicated, the male participants gave a lot more weight on attractiveness while a lot less weight on the ambition attributes on their match. On the other hand, female participants gave each attribute almost equal weight. Compared to females, males valued attractive appearances more, and they did not value the ambition of their partners as much as women did.



4. Model Fitting

4.1 Feature Engineering

For this part, we implemented a pruned classification tree model to find out the most important factors that may determine the matching result. After the first version of the tree, we found some predictors are of high importance but not meaningful to our analysis, such as 'dec_o', dec', ‘like_o’, and ‘like.’ These predictors represent the attitude of one participant towards one of his/her specific dating partners - whether he/she likes the other participant or not. It is intuitive

that two people with good feelings for each other will end up as a good match. Thus, we excluded those post-dating feedback data from our following modeling, for our goal was to predict potential good matches and to find out the key factors leading to the matches.

After dropping the irrelevant columns, we refitted the tree model and identified the 9 most important factors: *fun*, *prob_o*, *prob*, *attr*, *shar*, *fun_o*, *attr_o*, *intel_o*, *pf_o_int* (from most important to least). The important variables are interpreted as follows:

- *fun* - importance of the "fun" attribute the participant is looking for in the opposite sex
- *prob_o* - the participant's partner's answer of the question "How probable do you think that the person will say yes for you"
- *prob* - answer of the question "How probable do you think that the person will say yes for you"
- *attr* - importance of the "attractive" attribute the participant is looking for in the opposite sex
- *shar* - importance of the "having shared interests" attribute the participant is looking for in the opposite sex.
- *fun_o* - importance of the "having shared interests" attribute the participant's partner is looking for in the opposite sex
- *attr_o* - rating by partner at the night of the event on all 6 attributes
- *intel_o* - importance of the "intellegent" attribute the participant's partner is looking for in the opposite sex
- *pf_o_int* - partner's stated preference for all 6 attributes

We then scaled all these important variables to make them all range from 0 to 1, and used them as our predictors in the classification models.

4.2 Parametric Models Fitting

4.2.1 Simple Logistic Regression

With 9 scaled variables as predictors and the matching result as the outcome, the logistic regression model suggests that all these variables are significant. According to the model result, the most important predictors are *fun_o*, *attr_o*, *intel_o* and *prob_o*, in terms of the impact on the matching result with their unit change. As the model summary shows, the increase in *fun*, *prob_o*, *prob*, *attr*, *shar*, *fun_o*, *attr_o* and *pf_o_int* would lead to the increase in the probability of matching, while the increase in *intel_o* would reduce the matching probability. In other words, if a participant and his/her partner both value more about fun, attractiveness and sharing interests, and they have higher ratings for each other, it is on average more likely that they will match. However, if the partner values more about intelligence, it is on average less likely the participant and the partner get matched in the end.

There is no multicollinearity issue among the predictors, since the variance inflation factors are all below 2.

The 5-fold misclassification rate is around 0.1472, and the confusion matrix as as follows. The model is not very good at predicting matched cases.

Prediction \ Actual	Not Matched	Matched
Not Matched	6771	1006
Matched	227	374

4.2.2 Ridge Logistic Regression

Ridge logistic regression models are different from logistic regression models in that they can differentiate “important” from “less important” predictors, and they avoid overfitting by penalizing too large predictor coefficients. From the tree model in the feature engineering process, we selected 30 most important variables as the predictors for our Ridge logistic regression model. The best lambda after hyperparameter tuning is around 0.01, and it gives 5-fold cross-validation misclassification rate 0.1468. The model result shows the same most important predictors as the logistic regression model shows. The table below shows the confusion matrix.

Prediction \ Actual	Not Matched	Matched
Not Matched	6772	1003
Matched	226	377

4.3 Non-parametric Model Fitting

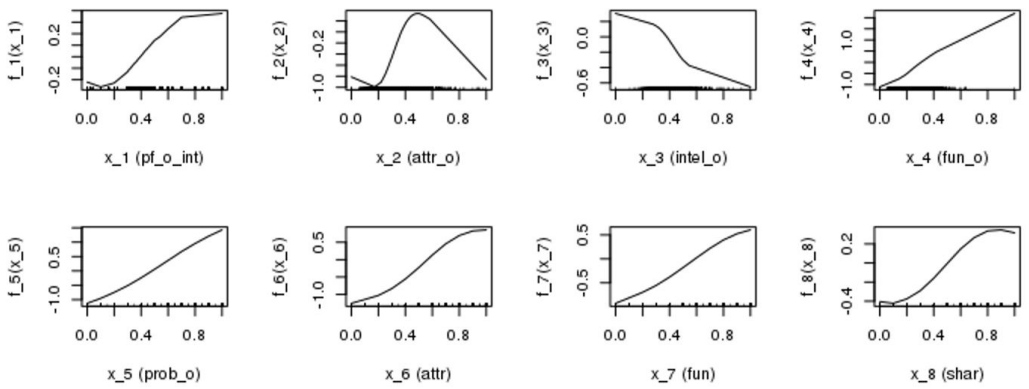
4.3.1 Neural Network

Neural Network is a model inspired by the biological neural networks that constitute animal brains and its mechanism designed to recognize patterns. It is a very flexible model, which can capture almost any nonlinear relationship with enough nodes. Also the architecture of Neural Network can be customized to tackle specific modeling problems, such as image recognition. In this course, we learnt Neural Network with only 1 hidden layer. Thus, we also used 1-hidden layer in this project and implemented CV to find the best size and lambda combination.

For the implementation of Neural Network, we first normalized the numerical predictors. Then, we used 5-fold and 3-replicate cross-validation to find the best combination of size and lambda, which is size = 3 and lambda = 0.1. The corresponding misclassification rate is 0.1450227. The confusion matrix is as follows.

Prediction \ Actual	Not Matched	Matched
Not Matched	6796	1019
Matched	202	361

Then we looked at the ALE plot to check the importance of each predator. According to the ALE plot, we can tell that the most important numerical factor is *fun_o*, followed by *attr_o* and *prob_o*. The other variables are also very important, which was generally in line with the tree model we used to select the important predictors.



4.3.2 Generalized Additive Models (GAM)

Generalized Additive Model (GAM) is a generalized linear model in which the linear predictor depends linearly on unknown smooth functions of some predictor variables. Generalized additive model performs well when there is no interaction effect between the predictors (the relationship is additive). Since the generalized additive model has built in cross-validation feature, we first fit the model to the whole dataset to tune the smoothing parameters. The resulting optimal smoothing parameters are 1.041385e-02, 5.559421e-02, 4.591570e-02, 1.356581e-03, 1.432652e+05, 7.181986e-02, 1.201813e+00, 3.622384e+00, 2.348046e+04 for predictors *pf_o_int, attr_o, intel_o, fun_o, prob_o, attr, fun, shar and prob*. Then we manually performed 5-fold cross validation to access the accuracy of the optimal model.

Based on the prediction of the 5-fold cross validation, the confusion matrix is as follows:

Prediction \ Actual	Not Matched	Matched
Not Matched	6787	1012
Matched	211	368

The model has sensitivity = 0.9698 and specificity = 0.2667. The high false negative rate indicates the model is not good at predicting the actual match. The average misclassification rate is 0.145978 based on 5-fold cross-validation.

4.3.3 Classification Tree

Classification tree was another good and flexible method to intuitively classify binary outcomes. We used the same nine predictors (*pf_o_int, attr_o, intel_o, fun_o, prob_o, attr, fun, shar* and *prob*) to predict whether there was a match or not. We first tuned the model to fit different lambda with 10-fold CV. Among different lambdas, 0.00001 turned out to be the one with the least CV misclassification rate. Then, we built a classification tree with 5-fold CV and 3 replicates. *Fun* (importance: 121.126193), *prob_o* (importance: 69.913521) and *attr_o* (importance: 35.351849) are the three most important variables. The resulting pruned classification tree had a misclassification rate of 0.1542.

Prediction \ Actual	Not Matched	Matched
Not Matched	6795	1089
Matched	203	291

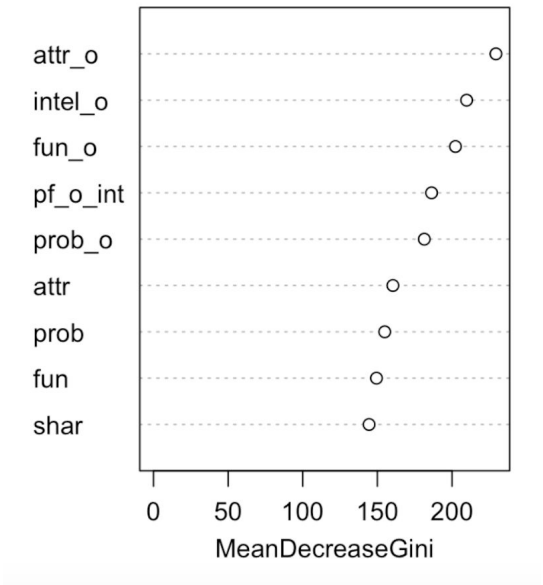
One potential problem with a classification tree was overfitting and high variance. We next performed bagging and boosting methods to improve this drawback of a tree. Another problem with a tree model was that our predictors were all numerical. There might exist linear behavior which cannot be easily handled by a tree.

4.3.4 Random Forest

In order to improve from tree models, random forest is introduced as a bagging method. We first tuned random forest to try out different numbers of predictors randomly sampled as candidates at each split with 10-fold CV. With 9 predictors fed to the model, mtry = 3 turned out to be the optimal number of predictors. From the # of trees vs error plot, we can tell 500 trees is enough for the model as the error line flattened out after 50 trees. We then built the random forest model with mtry = 3, and 500 trees on 5-folds CV and 3 replicates. The final misclassification rate of 5-fold CV was 0.1487.

Prediction \ Actual	Not Matched	Matched
Not Matched	6794	1042
Matched	204	338

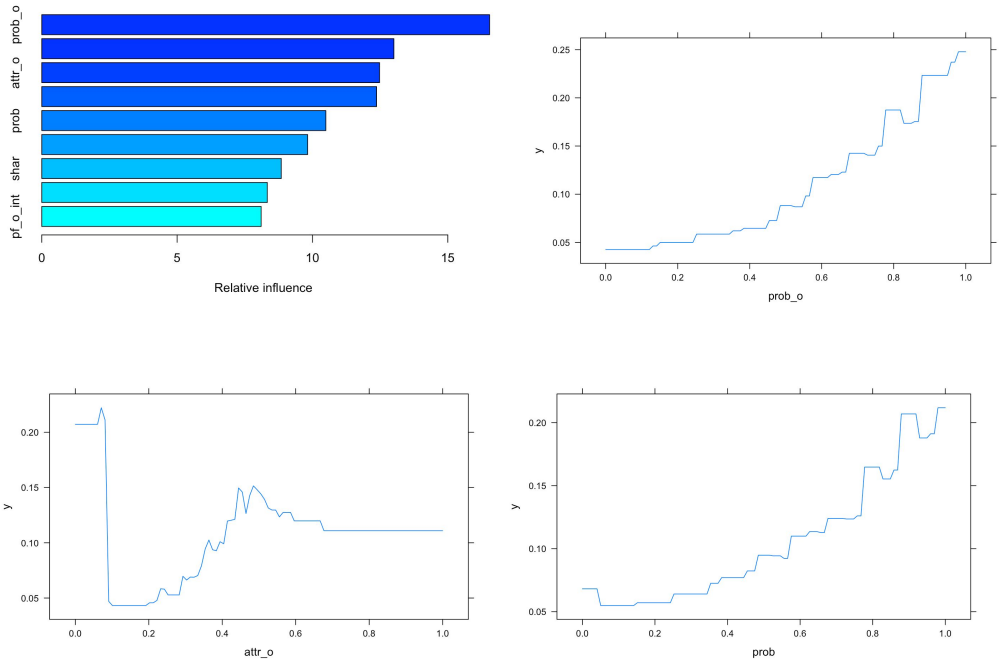
After fitting, we also checked the variable importance of the random forest model. *Attr_o*, intel_o and *fun_o* were the most important ones measured by decrease in gini. There were no influential limitations of a random forest model.



4.3.5 Gradient Boosting Trees

We last discovered our data with gradient boosting trees. First, we tuned the boosted tree with number of trees = 500, shrinkage = 0.01,0.02, 0.03, …, 0.13, 0.14, 0.15 on a 10-fold CV. The resulting optimal shrinkage is 0.06. We then manually cross validated the boosted tree model with shrinkage = 0.06 on 5 folds and 3 replicates. The best number of iteration was 310. The misclassification rate was 0.1420. *Prob_o*, *attr_o* and *prob* are the three most important predictors for a boosted tree. There was no huge drawback for a boosted tree.

Prediction \ Actual	Not Matched	Matched
Not Matched	6758	950
Matched	240	430



4.3.6 Support Vector Machine

In machine learning, support-vector machines (SVM) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. A support-vector machine classifier constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification. We used *pf_o_int*, *attr_o*, *intel_o*, *fun_o*, *prob_o*, *attr*, *fun*, *shar*, *prob* as predictors, and radial as the kernel function of the SVM, as it is one of the most commonly used kernels and is able to deal with dataset that is linearly inseparable. Then we used 5 fold and 3 replicates cross validation to find the best gamma and cost combination, which is gamma = 0.5 and cost = 10. The corresponding misclassification rate is 0.1499960. The confusion matrix is as follows.

Prediction \ Actual	Not Matched	Matched
Not Matched	6847	1075
Matched	151	305

4.4 Ensemble of Best Predictions

Generalized Additive Model (GAM), Gradient Boosting Trees and Neural Network appear to have the highest predictive power with the lowest misclassification rate.

We then ensemble the above best three models by taking the majority of the three classifications as our final ensemble model prediction. We used the same replicates and folds to test the result of the model and got 0.1418 misclassification rate. The performance of the

ensemble model is better than Neural Network but not as good as Gradient Boosting Trees, both of which are the components of the ensemble model. The confusion matrix is as follows:

Prediction \ Actual	Not Matched	Matched
Not Matched	6795	997
Matched	203	383

4.5 Model Evaluation

Based on 5-fold cross validation on the same data splits, the misclassification rates of our models are as follows.

Model	**Cross-Validation Misclassification Rate**
Simple Logistic Regression	0.1472
Ridge Logistic Regression	0.1468
Neural Network	0.1450
Classification Tree	0.1542
Generalized Additive Models (GAM)	0.1451
Random Forest	0.1487
Gradient Boosting Trees	0.1420
Support Vector Machine	0.1500
Ensemble Model	0.1432

The Gradient Boosting Trees model has the best performance. It has classification accuracy rate 0.8580, precision rate 0.3116 and sensitivity score 0.6418.

5. Conclusion

As our best classification model shows, the key factor that leads to whether participants match with their partners is how much the participants and their partners match with each other's evaluation of fun, common interests and attractiveness. The data shows that men and women put different weights on the six attributes (fun, attractiveness, common interest, intelligence, ambition and sincerity) when they look for potential romantic partners; however, the speed dating experiments indicate that gender is not a crucial factor in determining the matching result. This discrepancy shows the difference between participants' stated interest and their actual decisions. Male participants stated that they value attractiveness and do not value ambitiousness, and female participants preferred well-rounded partners. However, when it comes to speed dating, men and women all desire funny and attractive partners who share common interests. This distinction between people's stated preference and actual dating decision might be due to the lack of time to truly know a person during a speed dating event-- 4 minutes is not enough time for people to learn about other traits of their partners. The discrepancy may also be caused by participants' goals in speed dating. People who look for casual or short-term relationships might value more about fun and attractiveness of their partners, and less about intelligence, ambitiousness or career fields. Meanwhile, participants at this speed dating experiment were all Columbia graduate students, and this sample might not be representative in the real-world scenarios. It is only with larger-scale data from the real world, instead of from an experiment, that we would be able to validify the findings above and identify the true essential factors that people consider during speed dating.

6. Further Steps

First, our models are all not very good at predicting matched cases, due to the class imbalance. Therefore, given more time, we will try downsampling the unmatched cases, upsampling the matching cases and generating synthetic samples using SMOTE.

Second, we dropped some categorical predictors, such as “from” (participants’ hometown) and “zipcode,” because there are too many categories within them. Our method might leave out potential influential predictors, such as location data. Intuitively, people from the same place tend to connect and match more easily. In the future, we can text mine the messy categorical variables and group categories into several buckets, in order to include them in the models.

Third, we could build demographics-focused models to predict whether there was a match. Currently, all demographic data were dropped during the feature selecting process. However, demographic data, such as height, education, hometown, income, gender, contained important information of a unique person. There are potential interests to study what types of people are more likely to get matches as it is a heavily-discussed topic in social science.

7. Reference

Fisman, Raymond, Sheena S. Iyengar, Emir Kamenica, and Itamar Simonson. 2006. "Gender Differences in Mate Selection: Evidence from a Speed Dating Experiment." *The Quarterly Journal of Economics* 121, no. 2 (2006): 673-97.

Fisman and Iyengar. 2004. *Speed Dating Experiment*. Kaggle. https://www.kaggle.com/annavictoria/speed-dating-experiment

8. Appendix

```
library(dplyr)
library(ggplot2)
library(caret)
library(mgcv)
library(fmsb)
library(rpart)
library(randomForest)
library(gbm)
library(e1071)

data <- read.csv("Speed Dating Data.csv")
# check NA
na_rate <- rep(1, dim(data)[2])
# drop the variables that end with _3
for (i in 1:156){
  na_rate[i] <- sum(is.na(data[,i]))/nrow(data)
}
na_columns <- colnames(data)[na_rate > 0.5]
data <- data %>%
  select(-na_columns)
data$gender <- as.factor(data$gender)
data$career_c <- as.factor(data$career_c)
data$samerace <- as.factor(data$samerace)
data$race <- as.factor(data$race)
data$dec <- as.factor(data$dec)
data$date <- as.factor(data$date)
# scale the ratings
data <- data %>%
  mutate(pf_sum_o = pf_o_att + pf_o_sin + pf_o_int + pf_o_fun + pf_o_amb + pf_o_sha,
         sum_o = attr_o + sinc_o + intel_o + fun_o + amb_o + shar_o,
         sum1_1 = attr1_1 + sinc1_1 + intel1_1 + fun1_1 + amb1_1 + shar1_1,
         sum4_1 = attr4_1 + sinc4_1 + intel4_1 + fun4_1 + amb4_1 + shar4_1,
         sum2_1 = attr2_1 + sinc2_1 + intel2_1 + fun2_1 + amb2_1 + shar2_1,
         sum3_1 = attr3_1 + sinc3_1 + intel3_1 + fun3_1 + amb3_1,
         sum5_1 = attr5_1 + sinc5_1 + intel5_1 + fun5_1 + amb5_1,
         sum1_2 = attr1_2 + sinc1_2 + intel1_2 + fun1_2 + amb1_2 + shar1_2,
         sum4_2 = attr4_2 + sinc4_2 + intel4_2 + fun4_2 + amb4_2 + shar4_2,
         sum2_2 = attr2_2 + sinc2_2 + intel2_2 + fun2_2 + amb2_2 + shar2_2,
         sum3_2 = attr3_2 + sinc3_2 + intel3_2 + fun3_2 + amb3_2,
         sum5_2 = attr5_2 + sinc5_2 + intel5_2 + fun5_2 + amb5_2) %>%
  mutate_at(c("pf_o_att", "pf_o_sin", "pf_o_int", "pf_o_fun", "pf_o_amb", "pf_o_sha"),
         funs(./pf_sum_o*100)) %>%
  mutate_at(c("attr_o", "sinc_o", "intel_o", "fun_o", "amb_o", "shar_o"),
         funs(./sum_o*100)) %>%
```

```
  mutate_at(c("attr1_1", "sinc1_1", "intel1_1", "fun1_1", "amb1_1", "shar1_1"),
            funs(./sum1_1*100)) %>%
  mutate_at(c("attr4_1", "sinc4_1", "intel4_1", "fun4_1", "amb4_1", "shar4_1"),
            funs(./sum4_1*100)) %>%
  mutate_at(c("attr2_1", "sinc2_1", "intel2_1", "fun2_1", "amb2_1", "shar2_1"),
            funs(./sum2_1*100)) %>%
  mutate_at(c("attr3_1", "sinc3_1", "fun3_1", "intel3_1", "amb3_1"),
            funs(./sum3_1*100)) %>%
  mutate_at(c("attr5_1", "sinc5_1", "fun5_1", "intel5_1", "amb5_1"),
            funs(./sum5_1*100)) %>%
  mutate_at(c("attr1_2", "sinc1_2", "intel1_2", "fun1_2", "amb1_2", "shar1_2"),
            funs(./sum1_2*100)) %>%
  mutate_at(c("attr4_2", "sinc4_2", "intel4_2", "fun4_2", "amb4_2", "shar4_2"),
            funs(./sum4_2*100)) %>%
  mutate_at(c("attr2_2", "sinc2_2", "intel2_2", "fun2_2", "amb2_2", "shar2_2"),
            funs(./sum2_2*100)) %>%
  mutate_at(c("attr3_2", "sinc3_2", "intel3_2", "fun3_2", "amb3_2"),
            funs(./sum3_2*100)) %>%
            mutate_at(c("attr5_2", "sinc5_2", "intel5_2", "fun5_2", "amb5_2"),
            funs(./sum5_2*100)) %>%
  select(-c("pf_sum_o", "sum_o", "sum1_1", "sum4_1", "sum2_1", "sum3_1",
            "sum5_1", "sum1_2", "sum4_2", "sum2_2", "sum3_2", "sum5_2"))
# career distribution plot
career_label <- c("Lawyer", "Academic/Research", "Psychologist",
                  "Doctor/Medicine", "Engineer", "Creative Arts/Entertainment",
                  "Banking/Business", "Real Estate", "International Affairs",
                  "Undecided", "Social Work", "Speech Pathology", "Politics",
                  "Sports/Athletics", "Other", "Journalism", "Architecture")
data %>%
  filter(!is.na(career_c)) %>%
  select(iid, gender, career_c) %>%
  unique(by = iid) %>%
  ggplot() +
            geom_bar(aes(career_c, fill=gender)) +
            scale_x_discrete(label = career_label) + coord_flip() +
            labs(title = "Distribution of Career Fields", x = "Career Field", y = "Count") +
            scale_fill_manual(values=c(rgb(0.7, 0.3, 0.5, 0.4), rgb(0.2, 0.5, 0.8, 0.4)),
                  "Gender", labels = c("Female", "Male")) +  theme_light()
# age distribution plot
temp_age <- data %>%
  filter(!is.na(age)) %>%
  filter(age < max(age)) %>%
  select(iid, gender, age) %>%
  unique(by = iid)
ggplot(data = temp_age, aes(x = age,fill = gender)) + coord_flip() +
  geom_histogram(data = subset(temp_age, gender == "0"), binwidth = 2, color = "white") +
  geom_histogram(data = subset(temp_age, gender == "1"),
                 aes(y = ..count.. * (-1)), binwidth = 2, color = "white") +
```

```
  scale_y_continuous(breaks = seq(-70, 70, 10), labels = abs(seq(-70, 70, 10)))+
  scale_x_continuous(breaks = seq(10, 45, 5), labels = seq(10, 45,5)) +
  labs(title = "Distribution of Age", x = "Age", y = "Count") +
  scale_fill_manual(values=c(rgb(0.7, 0.3, 0.5, 0.4), rgb(0.2, 0.5, 0.8, 0.4)),
                    "Gender", labels = c("Female", "Male")) +  theme_light()
# race distribution plot
race_label <- c("Black/African American", "European/Caucasian American",
                    "Latino/Hispanic American", "Asian/Asian American",
                    "Naitive American", "Other")
data %>%
  filter(!is.na(race)) %>%
  select(iid, gender, race) %>%
  unique(by = iid) %>%
  ggplot() +
          geom_bar(aes(x = gender, fill = race), position = "fill") +
          labs(title = "Distribution of Race", x = "Gender", y = "Relative Frequency") +
          scale_fill_brewer(palette="Set3", name="Race", labels = race_label) +
          scale_x_discrete(labels=c("0" = "Male", "1" = "Female")) + theme_light()
# what do you look for in the opposite sex
test1 <- data %>%
  filter(!is.na(attr1_1 + sinc1_1 + intel1_1 + fun1_1 + amb1_1 + shar1_1)) %>%
  select(iid, gender, attr1_1:shar1_1) %>%
  unique(by = idd) %>%
  group_by(gender) %>%
  summarise(Attractive = mean(attr1_1), Sincere = mean(sinc1_1),
          Intelligent = mean(intel1_1), Fun = mean(fun1_1),
          Ambitious = mean(amb1_1), Interest = mean(shar1_1))
test1forplot <- test1 %>%
  select(-gender)
maxmin <- data.frame(
 Attractive = c(36, 0),
 Sincere = c(36, 0),
 Intelligent = c(36, 0),
 Fun = c(36, 0),
 Ambitious = c(36, 0),
 Interest = c(36, 0))
test11 <- rbind(maxmin, test1forplot)
test11male <- test11[c(1,2,4),]; test11female <- test11[c(1,2,3),]
radarchart(test11,
          pty = 32,
          axistype = 0,
          pcol = c(rgb(0.7, 0.3, 0.5, 0.4), rgb(0.2, 0.5, 0.8, 0.4)),
          pfcol = c(rgb(0.7, 0.3, 0.5, 0.4), rgb(0.2, 0.5, 0.8, 0.4)),
          plty = 1,
          plwd = 3,
          cglty = 1,
          cglcol = "gray88",
          centerzero = TRUE,
```

```
        seg = 5,
        vlcex = 0.75,
        palcex = 0.75,
        title = "What do people look for in the opposite sex?")
legend(x = 1, y = 1.2, legend = c("Female", "Male"),
        bty = "n", pch = 20 , col = c(rgb(0.7, 0.3, 0.5, 0.4), rgb(0.2, 0.5, 0.8, 0.4)),
        text.col = "black", cex = 0.8, pt.cex = 2)
# Feature Engineering
# drop 3 or s
data <- data[, !grepl(".*_[3s]$",colnames(data))]
# drop columns with >50% null values
t = data.frame(colSums(is.na(data))/nrow(data))
colnames(t)=c("nullrate")
t <- t %>% subset(nullrate<0.5)
data <- data[,rownames(t)]
# other
data <- data[, colnames(data)!='match_es']
data <- data[, colnames(data)!='pid']
data <- data[, colnames(data)!='dec_o']
data <- data[, colnames(data)!='dec']
data <- data[, colnames(data)!='like_o']
data <- data[, colnames(data)!='like']
data <- data[, colnames(data)!='partner']
data$income <- as.numeric(data$income)
data$tuition <- as.numeric(data$tuition)
data <- data[, !(colnames(data) %in%
c('zipcode','from','career','field','undergra','mn_sat','attr5_2','attr5_1'))]
# feature selection by tree importance
library(rpart)
set.seed(420)
control <- rpart.control(minbucket = 5, cp = 0.001, maxsurrogate = 0, usesurrogate = 0, xval =
10)
date.tr <- rpart(match ~.,data, method = "class", control = control)
plotcp(date.tr) # plot of CV r^2 vs. size
date.tr1 <- prune(date.tr, cp=0.011)
date.tr1$variable.importance
# data preprocessing
selected = colnames(data) %in%
  c("fun", "prob_o", "prob", "attr", "shar", "fun_o", "attr_o", "intel_o", "pf_o_int", "match")
dataNN = data[,selected]
dataNN[,1] = as.factor(dataNN[,1])
dataNN[, -1] <- sapply(dataNN[,-1], function(x)
  (x-min(x, na.rm = TRUE))/(max(x, na.rm=TRUE) - min(x, na.rm = TRUE)))
data_clean <- dataNN %>%
  mutate_at(vars(colnames(dataNN[-1])), ~ifelse(is.na(.), median(., na.rm = TRUE), .))
# logistic regression
logistic1 <- glm(match ~., data=data_clean, family = 'binomial')
summary(logistic1)
```

```r
library(car)
vif(logistic1)
# cross validation function
CVInd <- function(n,K) {
  m <- floor(n/K) #approximate size of each part
  r<- n - m*K
  I <- sample(n,n) #random reordering of the indices
  Ind <- list() #will be list of indices for all K parts
  length(Ind) <- K
  for (k in 1:K) {
          if (k <= r) {
          kpart <- ((m+1)*(k-1)+1):((m+1)*k) }
          else { kpart <- ((m+1)*r+m*(k-r-1)+1):((m+1)*r+m*(k-r))
          } #indices for kth part of data
          Ind[[k]] <- I[kpart]
          }
  Ind
}
# misclass rate function
misclass <- function(rate = seq(0,1,0.01), true_class, predicted_value) {
  misclass_ratio <- c()
  model_class <- c()
  result_table <- data.frame()
  for (i in rate) {
          predicted_class <- as.character(ifelse(predicted_value > i, 1, 0))
          model_class <- c(model_class, predicted_class)
          temp <- mean(predicted_class != as.character(true_class))
          misclass_ratio <- c(misclass_ratio, temp)
  }
  return(c(min(misclass_ratio), rate[which(misclass_ratio == min(misclass_ratio))]))
}
K <- 5  # K-fold CV on each replicate
n <- nrow(data_clean)
Ind <- CVInd(n, K)
y <- data_clean$match
yhat <- as.numeric(y)
yhat_class <- as.character(y)
misclassCV <- c()
set.seed(2020)
for (k in 1:K) {
  out <- glm(match ~., family = 'binomial', data = data_clean[-Ind[[k]],])
  yhat[Ind[[k]]] <- predict(out, newdata = data_clean[Ind[[k]],-1], type = 'response')
  temp <- misclass(true_class = y[Ind[[k]]], predicted_value = yhat[Ind[[k]]])
  mc <- temp[1]
  threshold <- temp[2]
  yhat_class[Ind[[k]]] <- as.character(ifelse(yhat[Ind[[k]]] > threshold, 1, 0))
  misclassCV <- c(misclassCV, mc)
  }
```

```
misclass_logistic <- mean(misclassCV)
misclass_logistic
confusionMatrix(as.factor(yhat_class), y)
# Ridge logistic regression
selected_ridge <- c('match','fun','prob_o','attr','attr_o','fun_o','prob','shar','amb_o',
                    'shar_o','age_o','sinc_o','intel_o','pf_o_int','tuition',
                    'attr2_2','attr3_1','shopping','fun1_1','sinc4_1','sinc2_2','pf_o_fun',
                    'income','fun3_2','sinc5_1','sinc','theater','pf_o_sin','pf_o_amb','intel1_1')
data_ridge = data[, selected_ridge]
data_ridge[,1] = as.factor(data_ridge[,1])
data_ridge[, -1] <- sapply(data_ridge[,-1], function(x)
  (x-min(x, na.rm = TRUE))/(max(x, na.rm=TRUE) - min(x, na.rm = TRUE)))
data_ridge <- data_ridge %>%
  mutate_at(vars(colnames(data_ridge[-1])), ~ifelse(is.na(.), median(., na.rm = TRUE), .))
library(glmnet)
x <- model.matrix(match~., data_ridge)[,-1]
y <- data_ridge$match
cv.ridge <- cv.glmnet(x, y, alpha = 0, family = "binomial")
ridge1 <- glmnet(x, y, alpha = 0, family = "binomial", lambda = cv.ridge$lambda.min)
cv.ridge$lambda.min
coef(ridge1)
K <- 5  # K-fold CV on each replicate
n <- nrow(data_clean)
Ind <- CVInd(n, K)
yhat <- as.numeric(y)
misclassCV <- c()
set.seed(2020)
for (k in 1:K) {
  out <- glmnet(x[-Ind[[k]],], y[-Ind[[k]]], alpha = 0, family = "binomial",
                lambda = cv.ridge$lambda.min)
  yhat[Ind[[k]]] <- predict(out, newx = x[Ind[[k]],], type = 'response')
  temp <- misclass(true_class = y[Ind[[k]]], predicted_value = yhat[Ind[[k]]])
  misclassCV <- c(misclassCV, temp)
  }
misclass_ridge <- mean(misclassCV)
misclass_ridge


# Neural Network
```

Cross validation

```
# crossvalidation
set.seed(2020)
Nrep <- 3   # number of replicates of CV
K <- 5   # K-fold CV on each replicate
n = nrow(dataNN)
y <- dataNN$match
```

```
size = c(3,5,7,9,15)
lamda = c(0.01,0.05,0.1,0.5,1)
grid_nn = as.data.frame(crossing(size,lamda))
n.models = nrow(grid_nn)  # number of different models to fit

yhat=matrix(0,n,n.models)
CV.rate<-matrix(0,Nrep,n.models)

for (j in 1:Nrep) {
  Ind<-CVInd(n,K)
  for (k in 1:K) {
    for (m in seq(nrow(grid_nn))){
      date.nn<-nnet(match ~ . ,dataNN[-Ind[[k]],], linout=F, skip=F,size=grid_nn[m,1],
decay=grid_nn[m,2], maxit=100, trace=F, MaxNWts=10000)
      yhat[Ind[[k]],m] = predict(date.nn,dataNN[Ind[[k]],-1], type = 'class')
    }
  } #end of k loop
  CV.rate[j,] = apply(yhat,2,function(x) sum(y != x)/n)
} #end of j loop
CV.rateAve<- apply(CV.rate,2,mean) #averaged CV misclass rate
grid_nn$mis_class  = CV.rateAve
grid_nn
```

Refit Best Model

```
# crossvalidation
set.seed(2020)
Nrep <- 1   # number of replicates of CV
K <- 5   # K-fold CV on each replicate
n = nrow(dataNN)
y <- dataNN$match

size = c(3)
lamda = c(0.1)
grid_nn = as.data.frame(crossing(size,lamda))
n.models = nrow(grid_nn)  # number of different models to fit

yhat=matrix(0,n,n.models)
CV.rate<-matrix(0,Nrep,n.models)

for (j in 1:Nrep) {
  Ind<-CVInd(n,K)
  for (k in 1:K) {
    for (m in seq(nrow(grid_nn))){
      date.nn<-nnet(match ~ . ,dataNN[-Ind[[k]],], linout=F, skip=F,size=grid_nn[m,1],
decay=grid_nn[m,2], maxit=100, trace=F, MaxNWts=10000)
      yhat[Ind[[k]],m] = predict(date.nn,dataNN[Ind[[k]],-1], type = 'class')
    }
  } #end of k loop
```

```
  CV.rate[j,] = apply(yhat,2,function(x) sum(y != x)/n)
} #end of j loop

CV.rateAve<- apply(CV.rate,2,mean) #averaged CV misclass rate
grid_nn$mis_class  = CV.rateAve
grid_nn
confusionMatrix(as.factor(yhat), as.factor(y), positive = "1")
```

ALE Plot

```
# fit the model
date.nn<-nnet(match ~ . ,dataNN, linout=F, skip=F, size=3, decay=0.01, maxit=100, trace=F,
MaxNWts=10000)
# check ALE plot
yhat <- function(X.model, newdata) {p.hat = as.numeric(predict(X.model, newdata,
type="raw")); log(p.hat/(1-p.hat))} ## to plot the log-odds
par(mfrow=c(3,4))
for (j in 1:9) {ALEPlot(dataNN[,2:10], date.nn, pred.fun=yhat, J=j, K=50, NA.plot = TRUE)
rug(dataNN[,2:10][,j]) } ## This creates main effect ALE plots for all 9 predictors
par(mfrow=c(1,1))

# Classification Tree
# Tune classification tree
set.seed(123)
Nrep <- 5
cp <- c(0.00001, 0.0001, 0.001, 0.01)
xerror <- matrix(0,Nrep,4)
for (i in 1:Nrep){
  for (j in 1:4) {
    control <- rpart.control(minbucket = 5, cp = cp[j], maxsurrogate = 0, usesurrogate = 0, xval =
10)
    tr <- rpart(match ~.,data, control = control)
    bestcp <- tr$cptable[which.min(tr$cptable[,"xerror"]),"CP"]
    tr1 <- prune(tr, cp=bestcp)
    printcp(tr1)
    xerror[i, j] <- tr1$cptable[which.min(tr1$cptable[, "xerror"]), "xerror"]
  }
}

CV.tree.compare <- rbind(0.16472*0.94493, 0.16472*0.92826, 0.16472*0.92536,
0.16472*0.92754, 0.16472*0.91232)
mean(CV.tree.compare)
# classifation tree CV
set.seed(2020)
n <- nrow(data)
K <- 5
Ind <- CVInd(n,K)
yhat = matrix(0, n, 1)
y <- data[,1]
```

```
for (k in 1:K){
  train <- as.data.frame(data[-Ind[[k]],])
  test <- as.data.frame(data[Ind[[k]],2:10])
  ytrain <- data[-Ind[[k]],1]
  control <- rpart.control(minbucket = 5, cp = 0.00001, maxsurrogate = 0, usesurrogate = 0)
  tr <- rpart(match ~.,train, control = control)
  bestcp <- tr$cptable[which.min(tr$cptable[,"xerror"]),"CP"]
  tr.final <- prune(tr, cp=bestcp)
  temp <- predict(tr.final, newdata = test, type="prob")
  l <- nrow(test)
  pred <- matrix(0, l, 1)
  for (j in 1:l) {
    if (temp[j, "0"] < temp[j, "1"]) {
      pred[j] <- 1
    }
  }
  #print(pred)
  yhat[Ind[[k]],1] <- pred
  printcp(tr.final)

}

cv = apply(yhat,2,function(x) sum(y!=x))/n
paste0("Mis classification rate is ", cv)
# classification tree confusion matrix
confusionMatrix(as.factor(yhat), y)
plot(tr.final);text(tr.final)


# GAM
set.seed(2020)
#cross validation function
CVInd <- function(n,K) {
  m <- floor(n/K) #approximate size of each part
  r<- n - m*K
  l <- sample(n,n) #random reordering of the indices
  Ind <- list() #will be list of indices for all K parts
  length(Ind) <- K
  for (k in 1:K) {
          if (k <= r) {
          kpart <- ((m+1)*(k-1)+1):((m+1)*k) }
          else { kpart <- ((m+1)*r+m*(k-r-1)+1):((m+1)*r+m*(k-r))
          } #indices for kth part of data
          Ind[[k]] <- l[kpart]
          }
  Ind
```

```
}
#tune smoothing paramters
out<-gam(match~s(pf_o_int)+s(attr_o)+s(intel_o)+s(fun_o)+s(prob_o)+s(attr)+s(fun)+s(shar)+s
(prob), data=data, family=binomial(), sp=c(-1,-1,-1,-1,-1,-1,-1,-1,-1))
sp_opt = out$sp
set.seed(2020)
K <- 5   # K-fold CV on each replicate
n = nrow(data)
y <- data$match
Nrep = 1

yhat=matrix(0,n,1)
misclass<-matrix(0,Nrep,1)
for (j in 1:Nrep) {
  Ind<-CVInd(n,K)
  for (k in 1:K) {
          train <- data[-Ind[[k]],1:10]
          test <- data[Ind[[k]],2:10]
out<-gam(match~s(pf_o_int)+s(attr_o)+s(intel_o)+s(fun_o)+s(prob_o)+s(attr)+s(fun)+s(shar)+s
(prob), data=train, family=binomial(), sp=sp_opt)
          pred <- predict(out,test,type="response")
          pred[pred>=0.5]=1
          pred[pred<0.5]=0
          yhat[Ind[[k]],1] <- pred
  } #end of k loop
   misclass[j,1]=apply(yhat,2,function(x) sum(y!=x))/n
} #end of j loop
misclass_avg<- apply(misclass,2,mean)
misclass_avg
#confusion matrix
yhat = as.factor(as.vector(yhat))
confusionMatrix(yhat, y,dnn = c("Prediction", "Actual"))
as.table(confusionMatrix(yhat, y,dnn = c("Prediction", "Actual")))


# Random Forest
# Tune random forest
set.seed(123)
Nrep <- 3
for (i in 1:Nrep){
  rForest2 <- randomForest(match~., data , mtry=2, ntree = 500, nodesize = 3, importance =
TRUE)
  rForest3 <- randomForest(match~., data , mtry=3, ntree = 500, nodesize = 3, importance =
TRUE)
  rForest4 <- randomForest(match~., data , mtry=4, ntree = 500, nodesize = 3, importance =
TRUE)
  rForest5 <- randomForest(match~., data , mtry=5, ntree = 500, nodesize = 3, importance =
TRUE)
```

```
  print(rForest2)
  print(rForest3)
  print(rForest4)
  print(rForest5)
}
# Error rate in Knit
mis.class.rate.2 <- mean(14.8,14.57,14.6)/100
mis.class.rate.3 <- mean(14.69,14.78, 14.6)/100
mis.class.rate.4 <- mean(14.82, 14.74, 14.75)/100
mis.class.rate.5 <- mean(14.73, 14.39, 14.8)/100
mis.class.rate.mtry <- rbind(mis.class.rate.2, mis.class.rate.3, mis.class.rate.4, mis.class.rate.5)

mis.class.rate.mtry
plot(rForest3)
importance(rForest2); varImpPlot(rForest2)
# Random forest CV
set.seed(2020)
n <- nrow(data)
K <- 5
Ind <- CVInd(n,K)
yhat = matrix(0, n, 1)
y <- data[,1]

for (k in 1:K){
  train <- as.data.frame(data[-Ind[[k]],])
  test <- as.data.frame(data[Ind[[k]],2:10])
  ytrain <- data[-Ind[[k]],1]
  rf <- randomForest(match~., train , mtry=3, ntree = 500, nodesize = 3, importance = TRUE)
  temp <- predict(rf, newdata = test, type="prob")
  l <- nrow(test)
  pred <- matrix(0, l, 1)
  for (j in 1:l) {
    if (temp[j, "0"] < temp[j, "1"]) {
      pred[j] <- 1
    }
  }
  yhat[Ind[[k]],1] <- pred
  print(rf)
}
cv = apply(yhat,2,function(x) sum(y!=x))/n
paste0("Mis classification rate is ", cv)
# random forest confusion matrix
confusionMatrix(as.factor(yhat), y)
plot(rf)
importance(rf); varImpPlot(rf)


# Gradient Boosting Tree
```

```
# Tune boosted tree
shrinkage <- c(0.01,0.02, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15)
n.models = length(shrinkage)   # number of different models to fit
y <- as.data.frame(data[,1])
n <- nrow(data)
Nrep <- 5
cv<-matrix(0, Nrep, n.models)
yhat<-matrix(0,n,n.models)
best.iter <- c()
set.seed(123)
for (j in 1:Nrep){

 for (i in seq(n.models)) {

   out <- gbm(match~., data=data, var.monotone=rep(0,9), distribution="bernoulli",n.trees=500,
shrinkage=shrinkage[i], interaction.depth=3, bag.fraction = .5,train.fraction = 1, n.minobsinnode
= 10, cv.folds = 10, keep.data=TRUE,verbose=FALSE)
   ntree_opt_cv <- gbm.perf(out, method = "cv", plot.it = FALSE)
   yhat[,i] <- predict(out, data[,2:10], n.trees = ntree_opt_cv, type = "response")
   #print(yhat)
   yhat[yhat>=0.5]=1
   yhat[yhat<0.5]=0
   best.iter <- cbind(best.iter, gbm.perf(out,method="cv", plot.it=FALSE))

 }
 cv[j,] = apply(as.data.frame(yhat),2,function(x) sum(y != x)/n)
}
cv.misclass.rate <- apply(cv, 2, mean); cv.misclass.rate
paste0("The best shrinkage is ", shrinkage[which.min(cv.misclass.rate)])
# boosted tree CV
set.seed(2020)
n <- nrow(data)
K <- 5
Ind <- CVInd(n,K)
yhat = matrix(0, n, 1)
y <- data[,1]

for (k in 1:K){
 train <- as.data.frame(data[-Ind[[k]],])
 test <- as.data.frame(data[Ind[[k]],2:10])
 ytrain <- data[-Ind[[k]],1]
 ytest <- data[Ind[[k]],1]
 out <- gbm(match~., data=train, var.monotone=rep(0,9), distribution="bernoulli",n.trees=500,
shrinkage=0.06, interaction.depth=3, bag.fraction = .5, train.fraction = 1, n.minobsinnode = 10,
keep.data=TRUE,verbose=FALSE)
 l <- nrow(test)
 temp <- predict(out, newdata = test, n.trees = 310, type = "response")
 temp[temp>=0.5]=1
```

```
  temp[temp<0.5]=0
  yhat[Ind[[k]],] <- temp
}
cv = apply(yhat,2,function(x) sum(y!=x))/n
paste0("Mis classification rate is ", cv)
#boosted tree confusion matrix
confusionMatrix(as.factor(yhat), as.factor(y))
# variable importance
set.seed(2020)
out2 <- gbm(match~., data=data, var.monotone=rep(0,9), distribution="bernoulli",n.trees=500,
shrinkage=0.06, interaction.depth=3, bag.fraction = .5, train.fraction = 1, n.minobsinnode = 10,
keep.data=TRUE,verbose=FALSE, cv.folds = 10)
best.iter <- gbm.perf(out2, method="cv", plot.it = FALSE)
par(mfrow=c(2,5))
plot(out2, i.var = 1, type = "response", n.trees = best.iter)
plot(out2, i.var = 2, type = "response", n.trees = best.iter)
plot(out2, i.var = 3, type = "response", n.trees = best.iter)
plot(out2, i.var = 4, type = "response", n.trees = best.iter)
plot(out2, i.var = 5, type = "response", n.trees = best.iter)
plot(out2, i.var = 6, type = "response", n.trees = best.iter)
plot(out2, i.var = 7, type = "response", n.trees = best.iter)
plot(out2, i.var = 8, type = "response", n.trees = best.iter)
plot(out2, i.var = 9, type = "response", n.trees = best.iter)

# SVM
# crossvalidation
set.seed(2020)
Nrep <- 3   # number of replicates of CV
K <- 5   # K-fold CV on each replicate
n = nrow(dataNN)
y <- dataNN$match

kernel = 'radial'
gamma = c(0.05)
cost = c(0.1,1)
grid_svm = as.data.frame(crossing(gamma, cost))
n.models = nrow(grid_svm)  # number of different models to fit

yhat=matrix(0,n,n.models)
CV.rate<-matrix(0,Nrep,n.models)

for (j in 1:Nrep) {
  Ind<-CVInd(n,K)
  for (k in 1:K) {
    for (m in seq(nrow(grid_svm))){
      date.svm = svm(match ~ . ,
                     data = dataNN[-Ind[[k]],],
                     kernel = 'radial',
```

```
            gamma=grid_svm[m,1],
            cost=grid_svm[m,2])
    yhat[Ind[[k]],m] = predict(date.svm, dataNN[Ind[[k]],-1], type = 'class')
   }
 } #end of k loop
 tmp = yhat[,]
 tmp[tmp=='1']=0
 tmp[tmp=='2']=1
 yhat[,] = tmp
 CV.rate[j,] = apply(yhat,2,function(x) sum(y != x)/n)
} #end of j loop
CV.rateAve<- apply(CV.rate,2,mean) #averaged CV misclass rate
grid_svm$mis_class  = CV.rateAve
grid_svm
```

Refit Best Model

```
# crossvalidation
set.seed(2020)
Nrep <- 1   # number of replicates of CV
K <- 5   # K-fold CV on each replicate
n = nrow(dataNN)
y <- dataNN$match

kernel = 'radial'
gamma = c(0.5)
cost = c(1)
grid_svm = as.data.frame(crossing(gamma, cost))
n.models = nrow(grid_svm)  # number of different models to fit

yhat=matrix(0,n,n.models)
CV.rate<-matrix(0,Nrep,n.models)

for (j in 1:Nrep) {
 Ind<-CVInd(n,K)
 for (k in 1:K) {
  for (m in seq(nrow(grid_svm))){
   date.svm = svm(match ~ . ,
            data = dataNN[-Ind[[k]],],
            kernel = 'radial',
            gamma=grid_svm[m,1],
            cost=grid_svm[m,2])
   yhat[Ind[[k]],m] = predict(date.svm, dataNN[Ind[[k]],-1], type = 'class')
  }
 } #end of k loop
 tmp = yhat[,m]
 tmp[tmp=='1']=0
 tmp[tmp=='2']=1
 yhat[,m] = tmp
```

```
  CV.rate[j,] = apply(yhat,2,function(x) sum(y != x)/n)
} #end of j loop
CV.rateAve<- apply(CV.rate,2,mean) #averaged CV misclass rate
grid_svm$mis_class  = CV.rateAve
grid_svm
confusionMatrix(as.factor(yhat_ensemble), as.factor(y), positive = "1")
# Ensemble
# cross validation
set.seed(2020)
Nrep <- 1   # number of replicates of CV
K <- 5   # K-fold CV on each replicate
n = nrow(dataNN)
y <- dataNN$match

n.models = 3  # number of different models to fit

yhat=matrix(0,n,n.models)
CV.rate<-matrix(0,Nrep,n.models)

Ind<-CVInd(n,K)
for (k in 1:K) {
  date.nn<-nnet(match ~ . ,dataNN[-Ind[[k]],], linout=F, skip=F,size=3, decay=0.1, maxit=100,
trace=F, MaxNWts=10000)
  yhat[Ind[[k]],1] = predict(date.nn,dataNN[Ind[[k]],-1], type = 'class')
} #end of k loop

set.seed(2020)
for (k in 1:K) {
  train <- data[-Ind[[k]],1:10]
  test <- data[Ind[[k]],2:10]

out<-gam(match~s(pf_o_int)+s(attr_o)+s(intel_o)+s(fun_o)+s(prob_o)+s(attr)+s(fun)+s(shar)+s
(prob), data=train, family=binomial(), sp=sp_opt)
  pred <- predict(out,test,type="response")
  pred[pred>=0.5]=1
  pred[pred<0.5]=0
  yhat[Ind[[k]],2] <- pred
} #end of k loop

set.seed(2020)
for (k in 1:K) {
  out <- gbm(match~., data=data[-Ind[[k]],], var.monotone=rep(0,9),
distribution="bernoulli",n.trees=500, shrinkage=0.06, interaction.depth=3, bag.fraction = .5,
train.fraction = 1, n.minobsinnode = 10, keep.data=TRUE,verbose=FALSE)
  temp <- predict(out, newdata = data[Ind[[k]],-1], n.trees = 310, type = "response")
  temp[temp>=0.5]=1
  temp[temp<0.5]=0
  yhat[Ind[[k]],3] <- temp
```

```
} #end of k loop

CV.rate[1,] = apply(yhat,2,function(x) sum(y != x)/n)
CV.rateAve<- apply(CV.rate,2,mean) #averaged CV misclass rate
CV.rateAve
## [1] 0.1450227 0.1451420 0.1408451
yhat_ensemble=matrix(0,n,1)
for (i in seq(nrow(yhat))){
  a1=0;a0=0
  for (j in seq(ncol(yhat))) {
    if (yhat[i,j]=='1'){
      a1=a1+1
    }
    else {
      a0=a0+1
    }
  }
  if (a1>a0) {
    yhat_ensemble[i]='1'
  }
  else {
    yhat_ensemble[i]='0'
  }
}
sum(y!=yhat_ensemble)/n
confusionMatrix(as.factor(yhat_ensemble), as.factor(y), positive = '1')
```