Predict match in speed dating using binary classification techniques

MA4270 Project Report

Wang Yanhao

1 Introduction

During recent years, speed dating has become a popular and widespread way of meeting potential partners. It is natural to ask the question of how to be more attractive to our potential partners and which attributes are the most influential ones in speed dating context. A well designed research was conducted by Ray Fisman, Sheena Iyengar and other two professors from Columbia Business School on this topic. Together with the research, a dataset[1] was compiled and a paper has been published[2].

In this report, we will make use of this dataset and try to answer the binary classification problem: *Given a list of attributes of two individuals, can we predict if they like each other?*

With the help of Python and scikit-learn, we will test with a few methods that have been introduced during the semester, such as SVM, Naive Bayes and AdaBoost, and compare the results with an additional technique named Random Forest.

2 The Dataset

The experiment conducted by the four professors was designed to simulate the real world Speed Dating scenario, where special efforts are made to ensure that the

[1] http://www.stat.columbia.edu/~gelman/arm/examples/speed.dating/

[2] Gender Differences in Mate Selection: Evidence from a Speed Dating Experiment

experiment creates a setting similar to that provided by the private firms operating in this market [3]. In addition, the dataset contains over 8k entries. Hence, this chosen dataset is relatively good for the purpose of our analysis.

Unfortunately, there are fields with large amount of missing data which might impose us a challenge in the preprocessing step as we need to find a reasonable way to simulate those entries if not skipping them. In addition, some of the fields are correlated (for instance, the median household income based on zipcode is apparently related to the geography, and SAT score is related to the participant's intelligence). Moreover, the mixture of categorical data (such as 'race') and continuous data (such as 'age') also impose us additional work to pre-process and clean up the raw data.

Based on the paper published by the professors from Columbia Business School, the most influential categories for this experiment are the participants' 'race', 'attractive', 'sincere', 'intelligent', 'fun', 'ambitious' and 'shared Interests'. Furthermore, female and male participants exhibit different valuation for those characteristics. Hence, for the purpose of this project, we have narrowed down the lists of fields in the graph below as well as next page [4].

Field Title	Type	Description
gender	categorical	male or female
age	continuous	subject's age
race	categorical	subject's race, where Black/African American = 1 European/Caucasian-American = 2 Latino/Hispanic American = 3 Asian/Pacific Islander/Asian-American = 4 Native American = 5 Other = 6
attr1_1	continuous	subject's stated preference for attractive (sum up to 100)
sinc1_1	continuous	subject's stated preference for sincere (sum up to 100)
intel1_1	continuous	subject's stated preference for intelligent (sum up to 100)
fun1_1	continuous	subject's stated preference for fun (sum up to 100)
amb1_1	continuous	subject's stated preference for ambitious (sum up to 100)
shar1_1	continuous	subject's stated preference for shared interests (sum up to 100)

[3] http://www.stat.columbia.edu/~gelman/arm/examples/speed.dating/

[4] Similar to the original paper, throughout the report, we will refer to the individual making the decision as *subject*, and the person being decided upon as *partner*.

Field Title	Type	Description
attr_o	continuous	partner's rating for subject's attractive (scale of 1-10)
sinc_o	continuous	partner's rating for subject's sincere (scale of 1-10)
intel_o	continuous	partner's rating for subject's intelligent (scale of 1-10)
fun_o	continuous	partner's rating for subject's fun (scale of 1-10)
amb_o	continuous	partner's rating for subject's ambitious (scale of 1-10)
shar_o	continuous	partner's rating for subject's shared interests (scale of 1-10)
age_o	continuous	partner's age
race_o	categorical	partner's race
pf_o_att	continuous	partner's stated preference for attractive (sum up to 100)
pf_o_sin	continuous	partner's stated preference for sincere (sum up to 100)
pf_o_int	continuous	partner's stated preference for intelligent (sum up to 100)
pf_o_fun	continuous	partner's stated preference for fun (sum up to 100)
pf_o_amb	continuous	partner's stated preference for ambitious (sum up to 100)
pf_o_sha	continuous	partner's stated preference for shared interests (sum up to 100)
attr	continuous	subject's rating for partner's attractive (scale of 1-10)
sinc	continuous	subject's rating for partner's sincere (scale of 1-10)
intel	continuous	subject's rating for partner's intelligent (scale of 1-10)
fun	continuous	subject's rating for partner's fun (scale of 1-10)
amb	continuous	subject's rating for partner's ambitious (scale of 1-10)
shar	continuous	subject's rating for partner's shared interests (scale of 1-10)

2.1 Deal with Missing Data

In practice, not all participants will complete the entire survey thoroughly. Hence, dataset with missing entries are generally unavoidable. We typically categorise missing data into two forms:

1. Data are said to be "missing at random" if the fact that they are missing is unrelated to actual values of the missing data.

2. Data are said to be "not missing at random" if the fact that they are missing is related to the actual missing data.

And different approaches, such as imputation of the missing values based on

mean value or most common value or any other statistical models, could be taken to handle the two scenarios differently.

As we can see from the data fields description listed in the graph in previous section, for this project, the summation of the six characteristics (attractive, sincere, intelligent, fun, ambitious, shared interest) stated by the partner and subject both equal to a fixed constant. Hence, if any of the six fields contains a missing value, it might be inaccurate to fill in the entry with value interpolated from other participants (such as mean value of this field from all participants). Therefore, it is more reasonable and easier to discard data rows with missing values in these fields.

Apart from fields related to the six characteristics, there are still two fields with missing values, namely 'age' and 'race'. For missing data in 'race' field, it could be caused by participants' carelessness; but more likely, those fields are 'not missing at random' because participants from minority races in society may be reluctant to fill in this field. Hence, it's hard to have a reasonable way to impute missing data for this field, and we will temporarily skip them. However, for 'age' field, since all participants involved in this experiment are selected from students in graduate and professional schools at Columbia University, it is reasonable to assume they are of similar age. Hence, we can use mean age to simulate missing values in 'age' and 'age_o' fields.

2.2 Preprocessing Categorical Feature Data

Many algorithms used for classification deploy the strategy of calculating euclidean distance among data from different features, therefore it is not ideal to represent categorical data with integers that have various difference between any pairs unless it is intentionally made so. Hence, we should encode the categorical data before putting them into the chosen classifier.

The algorithm employed in this case is known as *one-hot encoding* which uses binary digits (0 and 1) to encode categorical features. For example, instead of having 6 integer {1,2,3,4,5,6} for "race", we can have have 6 binary features and converting the original data with category 1 to new data <1, 0, 0, 0, 0, 0>; the original data with number 2 to new data <0, 1, 0, 0, 0, 0>; ...; the original data with number 6 to new data <0, 0, 0, 0, 0, 1>. In this way, every categorical feature with k number of possible states can be converted to k binary features.

We perform *one-hot encoding* to categories 'race' and 'race_o' before further actions.

2.3 Training and Testing Sets

After the above preprocessing steps, our dataset contains around 5800 rows in total. We then randomly divide the dataset into training and testing sets, with around 3500 (60%) rows for the training set and around 2300 (40%) rows in the testing set. After we have trained the chosen classifier with the training set, we will test the performance of the classifier with the testing set, and compare the classification result against the actual result to test for the generalisation error and effectiveness of trained models.

3 Methods

We explored the following methods for our task:

1. SVM with RBF Kernel,
2. Naive Bayes with prior being the number of different labels in each class,
3. Decision Tree
4. AdaBoost with Decision Tree being the base classifier,
5. Random Forest with Decision Tree being the base classifier

3.1 SVM

As introduced in class, the aim of SVM is to construct a hyperplane that maximise the distance between the hyperplane and the closest data point because the larger the distance is, the more accurate the classifier will be. And this hyperplane is used for the classification task later.

In class, we have also learnt that we can tune the performance of SVM by changing the parameters C and γ. Hence, we can find an optimal performance for SVM by performing a grid search through a pre-defined set of $<C, \gamma>$ combinations,

where the performance of the SVM for each set of parameters are calculated against the testing data set.

The pseudo-code for the above procedure is as following:

1. defined set of possible C values and set of possible γ values.

2. for each C in the set of possible C values:

2.1. for each γ in the set of possible γ values:

2.1.1. perform k-fold cross-validation to find the classifier's accuracy

2.2. end inner for loop

3. end outer for loop

4. train the SVM classifier with C and γ pair corresponding to the best performance

3.2 Naive Bayes

As introduced in the class, Naive Bayes are build upon the Bayes' theorem with a very simple (or "Naive") assumption that every pair of features is independence. This set of supervised learning algorithms make use of Maximum A Posteriori (MAP) estimation and can be implemented with the help of different distributions.

In this project, we assume the prior to be the number of different labels in each class of the training sample. Meanwhile, Gaussian Naive Bayes algorithm is used such that the likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

3.3 Decision Tree

The decision tree is probably one of the easiest and most intuitive supervised learning algorithms. Unlike SVM which requires data normalisation in order to obtain best performance parameters, decision tree classifier requires little data preparation. Moreover, the prediction of future data is extremely fast because it takes only logarithmic time complexity in the number of data points used to train the decision tree.

Despite the advantages, decision tree is not robust to noise, and is prone to overfitting because overly complicated trees could be produced which are not representative to the whole dataset.

Hence, we will use decision tree classifier as a benchmark against other state of the art classification methods.

3.4 AdaBoost

As introduced in the class, AdaBoost is an ensemble method where several simple base classifiers (decision stumps) are combined together to obtain better performance. We will use AdaBoost-SAMME algorithm with decision tree being the base estimator. For binary classification, SAMME (Stage-wise Additive Modelling using a Multi-class Exponential loss function) is equivalent to the AdaBoost algorithm introduced in the class.

3.5 Random Forest

Random Forest is a tree ensemble method which constructs many decision trees that will be used to classify a new instance by the majority vote. Each decision tree node uses a subset of attributes randomly selected from the whole original set of attributes.

As mentioned in Section 3.3, overfitting often occur for decision tree classifier, however, since random forest use averaging to improve the prediction accuracy, overfitting is under control.

Although the precision of predicting of classifying testing data generally increases as the number of trees used in Random Forest increases, sometimes, a large number of trees will only increase the computational cost and bring little significant perfor-

mance gain [5]. Hence, we use a number around 100 to generate following test results for Random Forest.

4 Experiments

As explained in Section 2, we have split the dataset into around 3500 entries for the training set and around 2300 entries for the testing set after cleanup of the data. For each of the five classification algorithms outlined in Section 3, we train the classifier on the same training set and evaluate their performance on the same testing set. The performance details are shown in the graph in Section 4.1.

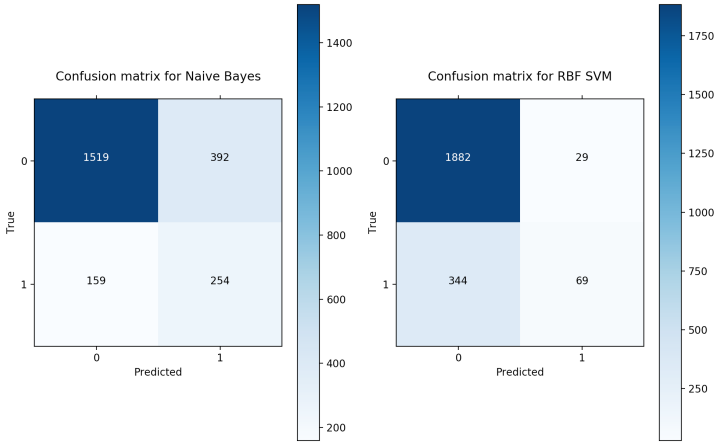
4.1 Algorithm Performances

Algorithm	Training time (5 d.p.)	Accuracy (3 s.f.)
SVM	1.14351	84.0%
Naive Bayes	0.00577	76.2%
Decision Tree	0.03022	77.2%
AdaBoost	0.27971	84.0%
Random Forest	0.69739	84.6%

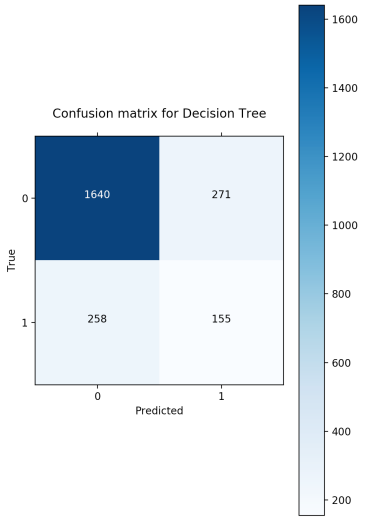
From the experiment results, we can see that simple classifiers such as Naive Bayes and Decision Tree do not have a good enough prediction accuracy, but are much better than random guessing. Meanwhile, they are extremely fast in terms of training and testing time. We observe Naive Bayes is faster than Decision Tree because intuitively, Naive Bayes has linear time complexity as it only need to maintain a map of counts, whereas although the prediction for Decision Tree has a logarithmic time complexity, the building of Decision Tree could have a polynomial time complexity up to degree 3 according to J. Kent Martin and D. S. Hirschberg [6].

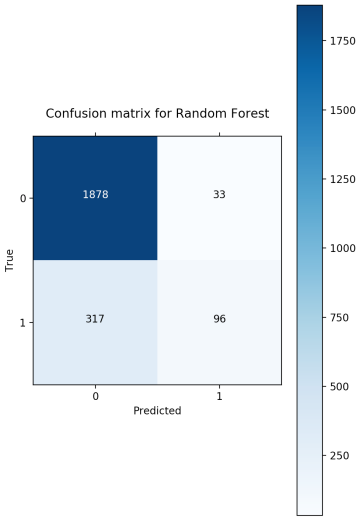
Confusion matrics for the 5 classifiers on the dataset are shown below:

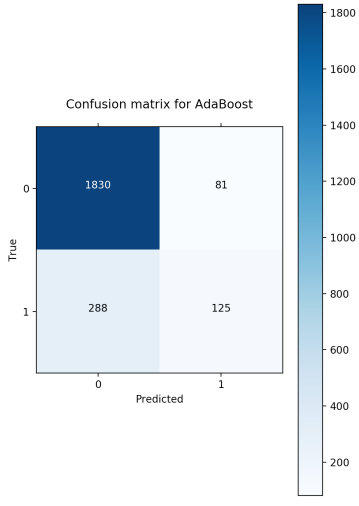
[5] Determine number of trees used in Random Forest classifier
[6] On the Complexity of Learning Decision Trees


Confusion matrix for Naive Bayes
1519
392
159
254
True
Predicted
0
1
1400
1200
1000
800
600
400
200

Confusion matrix for RBF SVM
1882
29
344
69
True
Predicted
0
1
1750
1500
1250
1000
750
500
250


Confusion matrix for Decision Tree
1640
271
258
155
True
Predicted
0
1
1600
1400
1200
1000
800
600
400
200


Confusion matrix for Random Forest
1878
33
317
96
True
Predicted
0
1
1750
1500
1250
1000
750
500
250


Confusion matrix for AdaBoost
1830
81
288
125
True
Predicted
0
1
1800
1600
1400
1200
1000
800
600
400
200

It is worth noting that although the performance is similar for SVM, AdaBoost and Random Forest, we have taken extra time to tune the parameters for SVM and have chosen a relatively large number of base classifiers for Random Forest. If we randomly choose a set of $<C, \gamma>$ combination for SVM, the performance will degenerate by around 2% for this dataset. Meanwhile, if we reduce the number of decision trees in Random Forest from 100 to 10, the training time will be reduced by almost 10 time (which is much faster than SVM and AdaBoost) whereas the prediction accuracy is reduced by 0.6%.

5 Conclusion

There is no single classifier which can out-perform others in terms of both prediction accuracy and training time. In general, the choice of classifier depends on the size, quality and nature of the dataset. But for this dataset of prediction for match in speed dating using the given fields, it seems ensemble methods such as AdaBoost and Random Forest are the best tools we should use.

6 Reference

[1] Raymond J. Fisman; Sheena Sethi Iyengar; Emir Kamenica; Itamar Simonson. Gender Differences in Mate Selection: Evidence from a Speed Dating Experiment. Quarterly Journal of Economics (2006)

[2] Thais Mayumi Oshiro, Pedro Santoro Perez, and Jos´e Augusto Baranauskas. Determine number of trees used in Random Forest classifier. Lecture Notes in Computer Science 7376 (July 2012)

[3] J. Kent Martin and D. S. Hirschberg. On the Complexity of Learning Decision Trees.