Uncovering the Truth of Love: A Modern Approach

Kyle (Chentao) Wang
cwang556@wisc.edu

Feiyu Yue
fyue3@wisc.edu

Tz-Ruei Liu
tliu292@wisc.edu

Abstract

As machine learning becomes a truly hot topic these days, people use it to do lots of things, from gaming to stock trading. Even for those typical "human-only" fields such as painting and composing, people started doing research and tried to use machine learning to "replace" humans. This paper focuses on using naive machine learning approaches on dating and looking for soul mates, which is even more humanized. By trying different models, including Logistics Regression, Support Vector Machine(SVM), Random Forest, and XGBoost, with some model selection tricks, the best model is selected to predict if a pair of people will match after their first speed dating. The result shows that SVM gives the best prediction, while XGBoost and Random Forest also have an excellent performance. This project is based on a dataset from an experiment conducted by Columbia Business School professors Ray Fisman and Sheena Iyengar[1].

1. Introduction

Throughout the long history of human beings, love has been a super popular, exciting, and long-lasting topic in daily life. Love is said to be a kind of magic or miracle. Statistically, it actually is if you consider it in this way: given a person, try to find another one who he or she can call as his or her "soul mate" from more than 7.7 billion people living on Earth (excluding other creatures and potential aliens to reduce the "heavy" computing followed) [2]. The probability of finding such a unique match could be as low as 1.2987013e-10 (considering this as it follows a Binomial Distribution). Love is also a mystery as it cannot be simply defined or explained. People have been trying to find the thing that connects two people with a variety of different "features" together closely. There are at least thousands of poems or books talking about what love is by Literature people [3]. Neuroscience or Psychology people may define love as a property of a complex mixture of neuropeptides and neurotransmitters [4]. As Statistic, Mathematics, or Computer Science(s) people, who are always considered as indifferent to romance, models (in mathematics instead of those in real-life) are found to be sexy enough to help define love.

Figure 1: Speed dating [5]

2. Related Work

There are only a few published papers using machine learning to predict romance relationships. S. Joel, P. W. Eastwick, and E. J. Finkel published a paper based on a speed dating experiment and used Random Forest to investigate the relationship between people's self-reported traits and preferences and their desire to others [6]. They used three quantities to materialize the "romantic match": desiring other people (actor variance) and being desired by other people (partner variance), and desiring for specific partners beyond actor and partner variance (relationship variance). These three quantities were generated from a series of social-relations-model analyses (using the BLOCKO program; Kenny, 1998) and 10% was the threshold to determine if the prediction can be explained by the model or not. Their results showed that the Random Forest algorithm was only able to achieve 4% to 18% of actor variance and 7% to 27% of partner variance, but the relationship variance was not able to exceed 10% threshold, as shown in Table 1.

Even though their results are not optimistic at all, their paper still provides some insights for this project. First, random forests seems to be a good algorithm to try. Second, their approach of generating new quantities is heavily based on previous psychology works, which might not work well with modern machine learning approaches. It seems to be worthwhile to try machine learning approaches without any assumptions. Thus, this project is trying to apply a more brutal-forced way and simply use the result - ”match/not match” - as the dependent variable.

Table 2: Results from the paper ”Is Romantic Desire Predictable? Machine Learning Applied to Initial Romantic Attraction”

Quantities	Results
Actor variance	$4 - 18$ %
Partner variance	$7 - 27$ %
Relationship variance	FAILED

3. Proposed Method

This project manages predictors in two ways and checks which way is better to predict partner matching. Under each approach, four different machine learning algorithms are applied to train models.

First, a Logistic Regression model is used since this project obtains a binary outcome: either two partners match or unmatched. Then, Support-Vector Machine (SVM) approach is applied to see how much the model can be improved compared to Logistic Regression. Random Forest is also tested to check whether the model obtains better result compared with related work mentioned above. Finally, this project tries to implement the XGBoost algorithm. Accuracy is used as the performance metric throughout the four types of models, and is defined as

$$ACC = \frac{\#\ Correct\ Predictions}{\#\ Total\ Predictions}$$

3.1. Two Prediction Approaches

This project interprets the original data in two approaches. The first approach treats the difference in two participants’ features as predictors. For continuous variables like age, the predictor would be the absolute difference between two persons’ age. For categorical variables like race, the predictor would be whether they are under the same race (Yes/No). This approach is denoted as *Diff*. The second approach directly treats both partners’ features as predictors. Unlike the first approach, there is no pre-processing needed. This approach is denoted as Concat.

3.2. Logistic Regression

Logistic Regression is first chosen as the baseline model. This model is relevant to the project because the response variable, match, falls into the Yes/No framework. For logistic regression with binary response variable, the distribution is binomial distribution.

$$Y_i | X_{1p}, ..., X_{ip} \sim Binomial(1, p_i)$$

where Y_i is the response variable match, X_{ip} are the predictors, p_i is the success rate of binomial distribution, or the probability that two partners would match [7].

To model the probability p_i based on the values of $X_{1p}, ..., X_{ip}$, the logistic regression applies logistic function to transform continuous values into a probability between 0 and 1 [7]. This can be used to model the probability of whether two partners would match based on existing predictors.

$$p_i = \frac{exp(\beta_0 + \beta_1 X_{i1} + ... + \beta_p X_{ip})}{1 + exp(\beta_0 + \beta_1 X_{i1} + ... + \beta_p X_{ip})}$$

3.3. Support-Vector Machine

A support-vector machine is based on constructing a hyperplane that has the largest distance to the nearest training-data points of any class (so-called functional margin); since larger margin indicates lower generalization error for the classifier [8]. Different from Logistic Regression, only those support-vectors that are close to the hyperplane contribute to the loss.

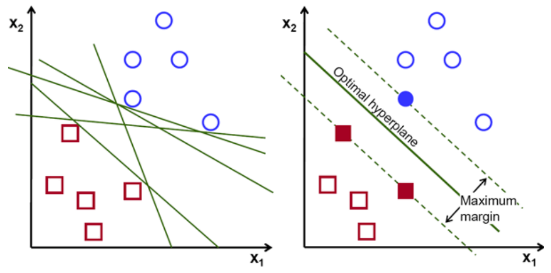

Figure 2: Hyperplane with maximum margin to support-vectors[9]

If the training data is linearly separable, two parallel hyper-planes are selected to separate the two classes of data so that the distance between them is as large as possible. In contrast, when the training data is not linearly separable, it can create nonlinear classifiers by applying the kernel trick, i.e., every dot product is replaced by a nonlinear kernel function [8]. In this case, the loss funtion of SVM is the hinge loss, which is defined as [9]:

$$\sum_{i=1}^{N} [1 - y_i(wx_i + b)]_+ + \lambda \|w\|^2$$

where

$$[z]_+ = max\{0, z\}$$

There are different kernels of SVM provided by *scikit-learn*, the most popular two are *linear* and *rbf*, which will also be applied with Grid Search for model selection through the experiment.

3.4. Random Forest

Random Forest, as used in the related work, is known as one of the most popular and easy-to-implement machine learning approaches. It can be used for both classification and regression problems [10]. This project implements Random Forest with Grid Search, which is a method to tune the hyper-parameters of models, for this classification problem, as the expected outputs are binary - Yes/No. When fitting the model implement Random Forest with training data, multiple decision trees are generated with randomly generated sub-samples from the original dataset. The sub-samples have the same sizes as the original one, but they are re-sampled with replacement.

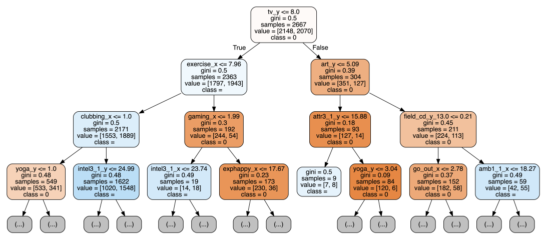

Figure 3: The top part of decision tree in the Random Forest model.

As there are multiple parameters for the Random Forest model, Grid Search is extremely helpful in this case to find the best parameter set. It implements a cross validation scheme to try different parameters for the estimator inputted and then output the parameters that give out the best score for the model. However, it should be noticed that as this method is an exhaustive way to tune the hyper-parameter. The time it takes to fully run Grid Search over a large parameter space will be quite long and unacceptable for small projects with limited computational capacities.

Besides using Grid Search to improve the performance of Random Forest model, the post-pruning method over decision tree is also used to shorten the time for training and avoid overfitting. Post-pruning is a typical trick implemented after the tree was built. It may not help much with the model's accuracy on the training set, but it is still necessary to implement as the time complexity of the whole training process can be reduced.

3.5. XGBoost

To overcome the weakness of decision tree that exhibits highly variable behavior, XGBoost is an implementation of gradient boosting created by Tianqi Chen[11]. In boosting, the trees are built sequentially as a weak learner, aiming to reduce the errors of the previous tree. Each tree learns from its predecessors and updates the residual errors for the next one. The final strong learner, which combines and averages those weak learners, reduces both the bias and variance.

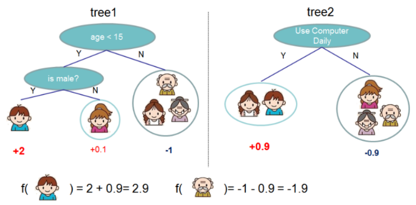

Figure 4: Tree ensemble model[12]

The regularized objective of XGBoost is given by

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

where the first term is a convex loss function and the second term $\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$ penalizes the complexity of the model by controlling the number of leaves T and weight w[11].

In contrast to Random Forest, in which trees are grown to their maximum extent, boosting builds trees with fewer splits. Parameters such as the number of trees, iterations, and tree depth, can be optimally tuned through validation. However, a large number of trees might lead to overfitting[13]. Thus, it is necessary to choose the stopping criteria for boosting carefully.

4. Experiments[1]

4.1. Dataset

The original dataset is published on Kaggle, which is a trusted online platform for data scientists and machine learners. This dataset is called "Speed Dating Experiment" and it was compiled by two Columbia Business School professors Ray Fisman and Sheena Iyengar for their paper "Gender Differences in Mate Selection: Evidence From a Speed Dating Experiment." The data was gathered from students in graduate and professional schools at Columbia University in experimental speed dating events from 2002-2004. Participants were recruited through a combination of mass email and fliers posted throughout the campus and handed out by research assistants. They generated random matching of subjects and created random variation in the number of potential partners. The original dataset contains 552 participants and in a total 8378 datings; and each row contains 195 attributes. Before the dating, each participant were asked for basic information and habits, including ethnicity, dating habits, field of study, occupation, and other

[1] All codes are available here: `https://github.com/tliu292/Uncovering-the-Truth-of-Love-A-Modern-Approach`

self-perception attributes that they find valuable in a mate. Each participant was also asked for six attributes that they think it's important for the opposite sex: Attractiveness, Sincerity, Intelligence, Fun, Ambition, and Shared Interests. During the dating, each participant had a four-minute first date with every other participant of opposite sex and were asked whether they would like to date with the participant again [1].

4.2. Data Preprocessing

First of all, related predictors are selected manually, because there are a lot of predictors within the dataset and most of them are clearly not related to this specific question. Also, there are many attributes that are meaningless and thus get removed from the dataset. The number of columns is finally reduced from 195 to 43. Each row includes 40 predictors, including age, field of study, race, etc. The remaining 3 columns are the person's id, the partner's id, and match (whether two people match). Then rows containing NA predictors are removed. The number of rows is thereby reduced from 8378 to 8124 observations.

Then, all variables are adjusted to the same scale. For columns evaluating the importance of attributes that one is looking for in the opposite sex (Attractiveness, Sincerity, Intelligence, Fun, Ambition, and Shared Interests), the scale of measurement is different for different waves of participants. For Waves 6-9, the importance of the attributes is measured on a scale of 1-10. However, for Waves 1-5 and 10-21, the participant is given 100 points and one can give more points to those attributes that one thinks that is more important in a potential date, and fewer points to attributes that are less important. The scale difference might skew the result; thus, all attribute scores in Waves 6-9 are adjusted from a scale of 1-10 to a scale of percentage to be consistent with the remaining dataset.

Next and most importantly, two partners are concatenated as a single pair input. In the dataset, each row represents one speed dating between two subjects. Initially, one row only contains one partner's information and the dating partner's information is represented in another row. Each row contains a column called pid that indicates which partner one is dating with. However, since this project's goal is to use models to predict whether two people would match, paired data as the training and testing set would be more relevant. Hence, two partners are merged as a single row by equaling one's pid with the other's iid. After concatenation, each row would contain duplicate attributes, since the dataset contains both the original participant's attributes and the partner's attributes. To differentiate two participants' attributes, the original participant's attributes are denoted with suffix x and the partner's attributes are denoted with suffix y. Since there are in total 40 predictors before concatenation, each row would contain 80 predictors after concatenation, in addition to iid, pid, and match. After merging two partners to a single paired input, there are in total 3943 rows and 83 columns remaining in the dataset.

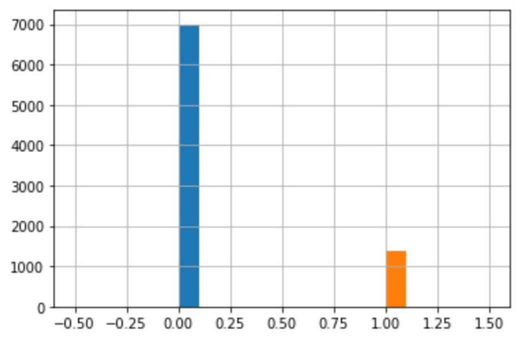

Figure 5: Imbalanced data. The original dataset contains about 7000 unmatched pairs and less than 1500 matched pairs.

However, the original dataset is very imbalanced. Figure 2 shows that there are only 20% of speed dating partners match. This imbalance would make the prediction meaningless since predicting all partners unmatched would still give us 80% prediction accuracy. Thus, adjusting for this imbalance would make the prediction more meaningful and accurate. The SMOTENC package from imbalanced-learn API is applied to offset this imbalance [14].

After data cleaning, the dataset is split into two separate parts - training set and testing set. Stratified sampling is applied over the response variable, "match", to maintain a balanced class proportion, using the function $train\_test\_split$ from scikit-learn [15]. The training set consists of 80% of the dataset, and the testing set contains the remaining 20%. The training set is further split into validation set and training set, which contain 20% and 80% of the original training dataset.

4.3. Models

There are two prediction approaches, *Diff* and *Concat*, applied over the dataset as mentioned in 3.1. Under each approach, the data is trained and tested under four different machine learning algorithms.

4.3.1 Logistic Regression

For both approaches, feature selection is applied after setting up the Logistic Regression pipeline. $SequentialFeatureSelector$ from mlxtend is used to perform forward sequential search over the entire features [16]. To run feature selection, a few parameters need to be identified, including the estimator, the number of features to select, the type of selection, scoring criteria, and the number of folds for cross validation.

For the *Diff* approach, the function $make\_pipeline$ is first called to set up the Logistic Regression model [15]. Since standardization is an important step for logistic regression, $StandardScalar$ is specified within the pipeline [15]. After setting up the pipeline, sequential feature selection is set up to select the best features. The pipeline would be the first parameter as the selector's estimator. Since there are 44 features within the *Diff* dataset, all 44 features are specified as the second parameter. Next, the hyper-parameter forward is set to true to run forward sequential search. Accuracy is then specified as the scoring criteria. The final hyper-parameter that gets tuned is cv, and 10 is chosen as the number of folds performed in cross validation. The training set is ultimately fitted to this searching model to select the best subset of features. After forward sequential search, both the training and testing set are reduced to the features selected.

For the *Concat* approach, similar hyper-parameter tunings are performed for pipeline and feature selection. However, there are in total 150 features for the *Concat* dataset other than 44 features for the *Diff* dataset. Since running over all 150 features is really time-expensive, 44 features is chosen again as the first parameter, while other hyper-parameters stay the same.

4.3.2 Support-Vector Machine

In this part, at first, two strategies are crafted for *Diff* and *Concat* respectively, depended on the number of features and training cost. Then an adjustment is made on *Diff* to improve the performance after analyzing the initial results.

For the *Diff* approach, similar to previous algorithm, forward sequential selection is applied within those 44 features and the scoring criteria is set as $accuracy$. Here 5-fold cross validation is chosen rather than 10-fold because both of them return similar outputs while 5-fold is cheaper on the time cost.

For the *Concat* approach, model selection is applied instead of feature selection, since there are few features contributing significantly by the result of Logistic Regression and also for the consideration of time complexity. A hyper-parameter tuning is performed before training and fitting. To be more specific, imported from $scikit-learn$ [15], the function $GridSearchCV$ [17] with $cv = 5$ is called by passing the $svm.SVC()$[18] as the estimator. Possible parameters are consisted of four combinations of $kernel$: $[linear, rbf]$ and regularization parameter C : $[0.5, 1]$. Other hyper-parameters are used by default.

Unfortunately, the first try on *Diff* approach gives a badly overfitting, therefore, $GridSearchCV$ replaces the sequential feature selection to be applied to *Diff* training set since it shows a brilliant performance on *Concat* approach. Details will be illustrated in section 5.

4.3.3 Random Forest

For both approaches, a hyper-parameter tuning is performed before training and fitting. The function $GridSearchCV$ with $cv = 5$ [17] (from scikit-learn [15]) is called again by passing the *RandomForestClassifier*[19] (also from scikit-learn) as the estimator. The parameter fields given for two approaches are different because of the long running time it takes for the exhaustive grid search to go over all parameters. As the grid search implements cross validation itself, there is no need to use the validation set again; thus, the original training set (containing 80% of the full data) is used to do the hyper-parameter tuning and the model training instead of the validation set.

For the *Diff* approach, the hyper-parameter $max\_features$ ranges from 20 to 44 (all features) and $n\_estimators$ ranges from 100 and 200. $min\_samples\_leaf$ is selected from [0.5, 1, 2] and the hyper-parameter for post-punning - ccp_alpha - is set to be either 0.001 or 0.0005. The reasons for choosing these ranges are mainly based on common knowledge, limited computational capacities and time, and also multiple runs of experiments with other sets of parameters. On the other hand, for the *Concat* approach, as the dataset has almost three times more predictors (154 in total), it will take an extremely long time to fully perform the exhaustive search over all possible parameters. As a result, the tuning for $max\_features$ is performed first to find a possible range of best parameters, which is $[90, 110]$. Another round of tuning is then performed with other parameters having the same range as those in the *Diff* approach.

It is noticed that even though random forest is known as an optimal way to avoid overfitting compared with decision tree, there is still overfitting when $min\_samples\_leaf$ is too small or ccp_alpha is not set, which means no post-pruning is performed. As a result, $[max\_features = 35, n\_estimators = 200, min\_samples\_leaf = 1]$ are chosen for the *Diff* approach and $[ccp\_alpha = 0.0005, max\_features = 90, min\_samples\_leaf = 1, n\_estimators = 200]$ are chosen for the *Concat* approach.

4.3.4 XGBoost

$GridSearchCV$ still dominates in XGBoost. $xgb.XGBClassifier(colsample\_bytree = 0.3, seed = 123)$ serves as its basic estimator [20]. For both *Diff* and *Concat* approach, scoring criteria is set to roc_auc and 5-fold cross-validation is chosen. Other tuning parameters are listed as below.

The method $.best\_estimator\_$ returns the best parameters given by $GridSearchCV$ [17], and it is specified in the next section.

Table 3: Tuning parameters for XGBoost

Parameter	Value
objective	['binary:logistic', 'binary:hinge']
max_depth	$range(4, 8, 1)$
n_estimators	[50,100,200]
learning_rate	[0.1, 0f.01]
reg_lambda	[0.5,0.1]
alpha	[1,5]

4.4. Software

Python is used for the entire project. For machine learning packages, Scikit-Learn, MLxtend, and Imblearn are used. For visualization packages, Matplotlib and Seaborn are used. For other packages, Pandas, Collections and Numpy are used. For the environment, JupyterLab is used for the entire project.

5. Results and Discussion

For *Diff* and *Concat* approach, all machine learning algorithms are trained to predict whether two people would match; and the models are evaluated based on the testing set. Table 4 shows the test accuracy for all four models under both approaches.

Table 4: Result for All Four Models

Model Name	Approach	Training Acc	Test Acc
LR	Diff	62.90%	63.00%
	Concat	75.44%	74.53%
SVM	Diff	100.00%	97.50%
	Concat	99.94%	97.12%
RF	Diff	99.11%	91.21%
	Concat	99.50%	92.49%
XGBoost	Diff	99.65%	94.37%
	Concat	100.00%	96.54%

5.1. Logistic Regression

For the *Diff* approach, in forward sequential search, the training set accuracy stays around 60% no matter how many features are selected. In addition, all predictors do not contribute much to the response variable, match. After fitting the training set to the pipeline, Logistic Regression gives only a test accuracy of 63.00%. Thus, treating feature difference as predictors does not seem to be relevant for Logistic Regression.

For the *Concat* approach, in forward sequential search, the training set accuracy increases as the number of predictors increases. The optimal number of predictors is 44, which is the maximum number of predictors allowed when tuning hyper-parameter. It seems that training accuracy can further increase if there are more predictors; however, to avoid overfitting and improve time efficiency, no more features are added to the model. After fitting the training set to the pipeline, Logistic Regression gives a test accuracy of 74.53%. Thus, directly treating both partners' features as predictors looks more relevant than treating feature difference as predictors; however, Logistic Regression is still not good enough for predicting love.

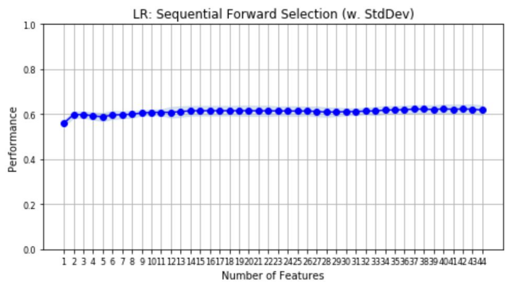

Figure 6: Sequential Feature Selection for Logistic Regression under *Diff* Approach. The training accuracy remains around 60%.

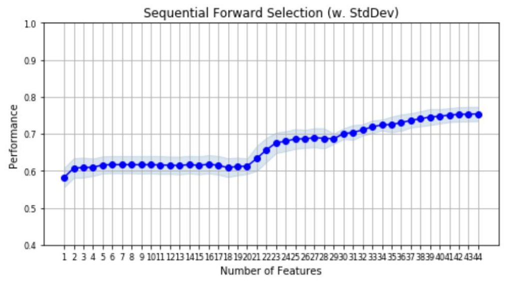

Figure 7: Sequential Forward Selection for Logistic Regression under Concat Approach. The training accuracy increases as number of predictors increases.

5.2. Support-Vector Machine

For the *Diff* approach, forward sequential feature selection gives the best number of features, 33. As figure 8 shows, the performance improves as the number of features k increases and it peaks at $k = 33$. However, the performance goes down when more features are added. Training accuracy comes with 92.74%, while test accuracy is about 86.13%.

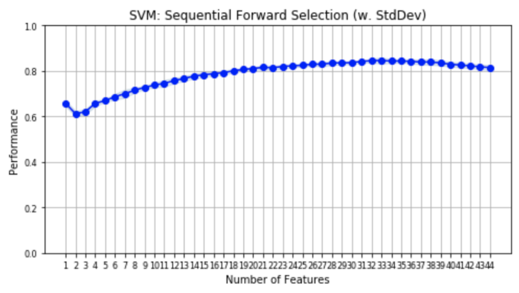

Figure 8: Sequential Feature Selection for SVM under *Diff* Approach

Note that, 11 out of 44 features are dropped by this algorithm, including features which are related to career, hometown and race. It seems that people care more about personalities instead of origins while choosing a lover. However, it is necessary to emphasize that this result may be based on the fact that the participators of the original experiment are all well-educated since they are from Columbia University.

For the *Concat* approach, the result of $GridSearchCV$ shows that the *rbf* kernel with the regularization parameter $C = 1.0$ stands out among those combinations. Training accuracy of this model is 99.94%, test accuracy is 97.12%.

Compare this two approaches, the big gap between training accuracy and test accuracy of *Diff* approach is obtrusive. Feature selection strategy doesn't perform as well as expectation. Now re-applying the same $GridSearchCV$ process of *Concat* approach to *Diff* training set, it gives the same optimal parameters as above. What makes difference is that this time, training accuracy jumps to 100% and test accuracy reaches 97.50%, which is the best result so far.

5.3. Random Forest

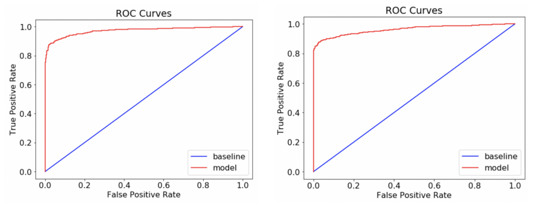

Figure 9: Left is ROC curve by *Diff* approach with Random Forest and Right is by *Concat* approach.

By using the Random Forest model with parameters selected by Grid Search (as mentioned in 4.5), both approaches achieve high performance on the training set and slightly worse performance (about 5% lower accuracy) on the test set. For the *Diff* approach, it reaches 99.11% on the training set and 91.21% on the test set. For the *Concat* approach, it reaches 99.50% on the training set and 92.49% on the test set. It seems like Random Forest gives out a slightly overfitting model but the performance on the test set is still much better than that Logistic Regression and similar to SVM. The ROC curves for both approaches are shown above.

5.4. XGBoost

The optimal parameters for *Diff* and *Concat* corresponding to the highest AUC value training by $GridSearchCV$ are shown in Table 5.

For the *Diff* approach, training accuracy of final model is 99.65%, test accuracy is 94.37%.

For the *Concat* approach, training accuracy of final model is 100.00%, test accuracy is 96.54%.

Table 5: Optimal Model Parameters for XGBoost

Parameter	Value:Diff	Value:Concat
objective	'binary:logistic'	'binary:logistic'
max_depth	7	7
n_estimators	200	100
learning_rate	0.01	0.1
reg_lambda	0.5	0.1
alpha	1	1

Here, the l_2 regularization parameter reg_lambda of logistic loss function in *Diff* is larger than that in *Concat*, which means, to some extent, a heavier penalty is needed for avoiding overfitting in *Diff* approach. This phenomenon might help to explain the result in SVM section.

6. Conclusions

Among all four algorithms, Logistic Regression is not bad. Random Forest and XGBoost give a fairly good performance. SVM gives out the best performance in general for both approaches and performs extremely well on the *Diff* approach. Among the two approaches *Diff* and *Concat*, *Concat* seems to give a better result in general. However, it is more vulnerable to overfitting and is less intuitive and interpretive than the *Diff* approach.

Comparing to the results from related work, this experiment gives much more optimistic results on predicting matching or not matching after speed dating. However, it is still unknown if the final model can be applied to actually "uncover the truth of love" in reality as it requires another speed dating experiment to be performed and all those data about participants to be collected and formed a "truly" unbiased test set. With the existing dataset, a clean test set can be derived by excluding all matches with both participants on the test set from the training set. Because of the limited timing, this project fails to redo the whole training and fitting to test on the clean test set unfortunately. This may help to give a more realistic result and avoid overfitting in Random Forest and SVM algorithms. As a side note, the "clean" test set may not be derived as easily as it may sound, because the package SMOTENC that deals with imbalanced dataset will auto generate some random data.

In conclusion, this project uses two approaches to interpret information from the speed dating and finds SVM and XGBoost as two potentially well-performed algorithms to predict the result as matching or not. Machine learning models seems to successfully uncover the tip of the iceberg about the truth of love.

7. Acknowledgements

The dataset is downloaded online from Kaggle (`https://www.kaggle.com/aeshen/the-secret-to-getting-the-second-date`). Several python packages are used to perform the experiment, mainly involving scikit-learn and mlxtend. We would like to say special thanks to Professor Sebastian Raschka, who kept generously giving out extremely useful suggestions through the whole process of this project.

8. Contributions

For the data pre-processing, Tz-Ruei mainly worked on the data cleaning and the preparation for the *Diff* approach, while Feiyu and Kyle managed the preparation for the *Concat* approach. Then, for the experiment, Tz-Ruei was in charge of testing the Logistic Regression approach, Kyle was in charge of the Random Forest approach, and Feiyu worked on SVM and XGBoost. For the presentation, each person made the slides about what they had done, so was this paper.

References

[1] A. Montoya, "Speed dating experiment," 2016, (Accessed: 16 October 2019). [Online]. Available: https://www.kaggle.com/annavictoria/speed-dating-experiment

[2] "Current world population." [Online]. Available: https://www.worldometers.info/world-population/

[3] M. Popova, "What is love? famous definitions from 400 years of literary history," Mar 2016. [Online]. Available: https://www.brainpickings.org/2013/01/01/what-is-love/

[4] K. Seshadri, "The neuroendocrinology of love," *Indian Journal of Endocrinology and Metabolism*, vol. 20, no. 4, p. 558, 2016.

[5] A. Shen, "The secret to getting the second date," Aug 2018. [Online]. Available: https://www.kaggle.com/aeshen/the-secret-to-getting-the-second-date

[6] S. Joel, P. W. Eastwick, and E. J. Finkel, "Is romantic desire predictable? machine learning applied to initial romantic attraction," *Psychological Science*, vol. 28, no. 10, pp. 1478–1489, 2017, pMID: 28853645. [Online]. Available: https://doi.org/10.1177/0956797617714580

[7] "Logistic regression," Dec 2019. [Online]. Available: https://en.wikipedia.org/wiki/Logistic_regression

[8] "sklearn.svm.svc." [Online]. Available: https://en.wikipedia.org/wiki/Support-vector_machine

[9] G. Drakos, "Support vector machine vs logistic regression." [Online]. Available: https://towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f

[10] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[11] C. G. Tianqi Chen, "Xgboost: A scalable tree boosting system," 2016. [Online]. Available: arXiv:1603.02754

[12] D. Liu, "Boosting model: Analysis of xgboost theory." [Online]. Available: https://www.csuldw.com/2019/07/20/2019-07-20-xgboost-theory/

[13] R. B. Sundaram, "An end-to-end guide to understand the math behind xgboost." [Online]. Available: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

[14] "Imbalanced learn over-sampling smotenc." [Online]. Available: https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTENC.html

[15] "sklearn.model-selection train-test-split." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[16] S. Raschka. [Online]. Available: http://rasbt.github.io/mlxtend/api_subpackages/mlxtend.feature_selection/#sequentialfeatureselector

[17] "3.2. tuning the hyper-parameters of an estimator." [Online]. Available: https://scikit-learn.org/stable/modules/grid_search.html

[18] "sklearn.svm.svc." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[19] "3.2.4.3.1. sklearn.ensemble.randomforestclassifier." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[20] X. community, "Python api reference." [Online]. Available: https://xgboost.readthedocs.io/en/latest/python/python_api.html