UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UCCD2063 Artificial Intelligence Techniques

Practical Assignment

May 2020

Supervised Classification on SpeedDating

No.	Student Name	Student Id	Year/ Semester	Course
1.	AARON KONG KAH LUN	18ACB02293	Y2S1	CS
2.	WONG JIUN LIN	19ACB01155	Y2S1	CS
3.	YAP JHENG KHIN	18ACB00224	Y2S1	CS

Effort and Contribution List

Student 1	Student 2	Student 3	Total
33.33%	33.33%	33.33%	100%

Table of Contents

1.1 Introduction

The outbreak of COVID-19 pandemic around the world at the year of 2020 has caused lots of inconvenience among the public to continue or conduct their daily works. The situation comes even worse when the Malaysia government implemented the Movement Control Order (MCO) which forces all of the citizens to stay at home unless particular necessarily methods need to be performed. The above situation has also caused the people having less chances to go out to develop their relationship or dating with others.

Therefore, the dating apps has turned to be the most popular applications for the young people to find their relationships during the MCO period. However, the dating apps still can only be placed as a platform for the young people to conduct their relationships. It is ineffective for the users to find for their relationship among the networks without preparations or filtering since the dating apps contains numerous amounts of users. The possibility for a person to keep or find a relationship still based on their own characteristics and personalities. Hence, we are going to propose a Decision Support System (DSS) based on data-driven model in order to help the users to predict which groups of users with particular characteristics would have a higher possibility to match with the partners.

Our main purpose is to train certain types of classification models and perform comparison among the models in order to obtain a best model which can estimate the users to match with their partners. This can help the users to save their time and also resources needed in order to find an appropriate partner in their relationships. Besides that, we also hope that the model can help to reduce the amount of bachelor and bachelorette in today's societies since most of the people are busy in carrying their daily work and have eventually ignored or lost interested their own relationship developments. We sincere hope that those people could regain their courage or confidence to conduct their own relationships. Furthermore, the model can also remind the users to cultivate particular behavior or characteristic so that it can help them to increase their opportunities to own a better relationship. It also helps the users to redefine the problems owned by themselves and find out the solutions to overcome with it as well as turning into a person who obtain a better personalities and characteristics.

2.1 Data Collection

The dataset we have chosen is SpeedDating (https://www.openml.org/d/40536) which focused on experimental speed dating information, which included the answers of 8,378 participants between 2002 and 2004. Each participant had a 4-minute "first date" with the opposite sex. Once they completed the short-term date, the participants were asked to rate their likelihood of seeing their partner again (between 0 and 10). Everyone is also asked to rate their partners on 6 subjective attributes. According to our preliminary analysis, these subjective attributes and ratings of self-perception, actual age and whether they match other people will form the basis for my inquiry. The attributes are Attractiveness, Sincerity, Intelligence, Fun, Ambition, and Shared Interests.

As mentioned earlier, the task of each participant is to rate themselves and rate each attribute in their partners, and their partners are responsible for rating the participants. Considering the subjective attributes mentioned above, all ratings constitute the best variable to answer our question: "Can I use these subjective ratings to improve the match between users of modern dating apps?"

2.2 Data Exploration and Problem Understanding

We have analyzed a dataset which consists of 8378 samples gathered from participants in experimental speed dating events from 2002 until 2004. The dataset output is to determine whether the given partners would match with the users. There are 122 inputs and 1 output which in summation 123 attributes in the dataset. The dataset consists of 7 attributes which datatype are in integer type and the remaining 116 attributes are all in object type.

```
dating.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8378 entries, 0 to 8377
Columns: 123 entries, has_null to match
dtypes: int64(7), object(116)
memory usage: 7.9+ MB
```

There are totally 56 preprocessed features which have undergone the data preprocessing in the dataset such as “d_d_age” which represents the various types of age difference for the users and given partners. Therefore, all the features with heading “d_” will show the particular attributes in discrete type which should be filtered out from the raw dataset while performing data preprocessing.

```
[7] dating.head()
```

	has_null	wave	gender	age	age_o	d_age	d_d_age	race	race_o	samerace	importance_same_race	importance_same_religion	d_importance_same_race	d_importance_same_religion	field	pref_o_attractive	pref_o_sincere	pref_o_inte
0	0	1	female	21	27	6	[4-6]	'Asian/Pacific Islander/Asian-American'	European/Caucasian-American	0	2	4	[2-5]	[2-5]	Law	35	20	
1	0	1	female	21	22	1	[0-1]	'Asian/Pacific Islander/Asian-American'	European/Caucasian-American	0	2	4	[2-5]	[2-5]	Law	60	0	
2	1	1	female	21	22	1	[0-1]	'Asian/Pacific Islander/Asian-American'	'Asian/Pacific Islander/Asian-American'	1	2	4	[2-5]	[2-5]	Law	19	18	
3	0	1	female	21	23	2	[2-3]	'Asian/Pacific Islander/Asian-American'	European/Caucasian-American	0	2	4	[2-5]	[2-5]	Law	30	5	
4	0	1	female	21	24	3	[2-3]	'Asian/Pacific Islander/Asian-American'	'Latino/Hispanic American'	0	2	4	[2-5]	[2-5]	Law	30	10	

5 rows × 123 columns

```
preprocessed_features = [feature for feature in dating.columns if feature.lower()[:2]=='d_']
print(f'Amount of preproccessed features: {len(preprocessed_features)}')
```

```
Amount of preproccessed features: 56
```

Besides that, the dataset also exists several irrelevant features such as “has null” which represents whether the particular sample consisting null values. Several features with “expected_” means the expectations of the users towards partners and also all the users’ self interest features such as sports, movies and others could be considered as subjective features and should also be dropped from the datasets. The fields of study of the users would also not be considered in the

dataset. Moreover, the attributes which consist of "_o" represent the opinions of the given partners which are also irrelated in the dataset. For instance, "pref_o_attractive" means the importance rated by the partners towards the attractiveness of the participants. Hence, features selection should be performed during the data preprocessing in order to filter out all irrelevant attributes.

Lastly, the training outcomes of the dataset should have high recall rate since we need to predict the possibility of the participants in the dataset are correctly matched with the given partners.

2.3 Data Analysis and Visualization

The purpose of this report is to improve the matching and grouping criteria in the online dating app, and it all starts with the user. Therefore, it is of course important to understand the distribution of people (age, race, and gender) and the frequency of matches in the data. I started by checking the race of the participants in our dataset.

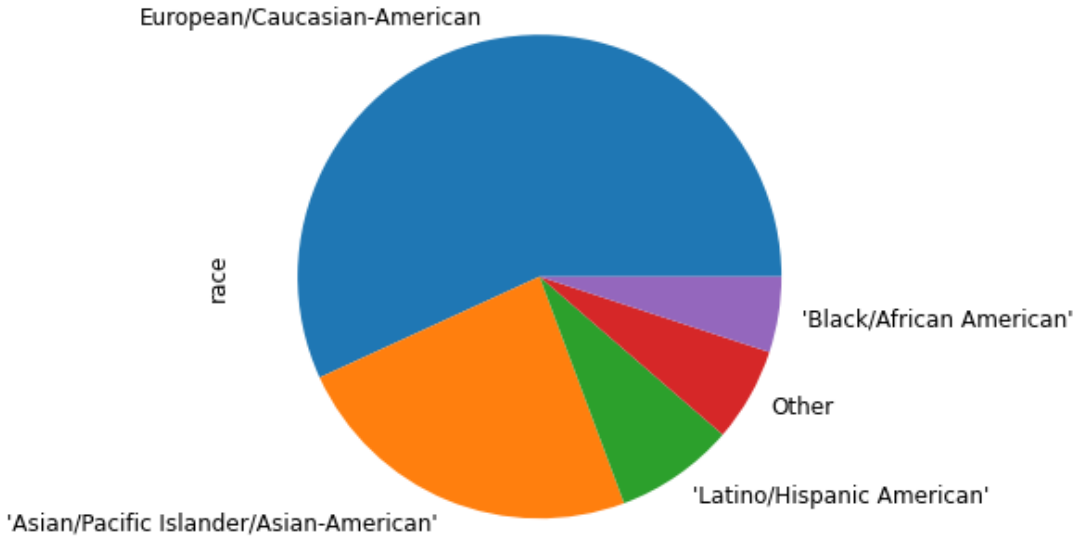


Figure 1.1: Pie chart displaying distribution of participants race

As shown in the figure (Figure 1.1), it shows the race distribution in our dataset, the data accounted for a large part of the world and consist many races.

This is the first prejudice I have encountered. It clearly shows that after the people of Europe and Asia, the rest of the race has very small representatives. Also, in the context of dating apps, it is important to understand the gender difference to point out where data quality can be improved in the future.

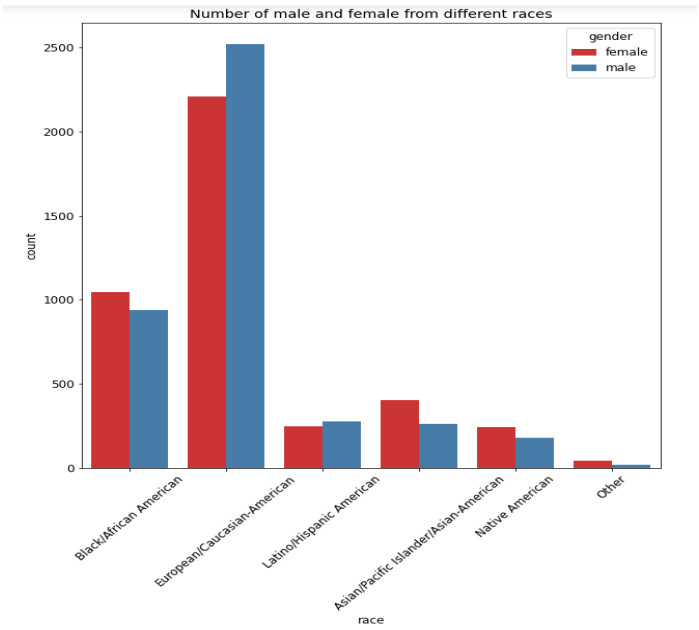


Figure 1.2: Distribution of race among the genders

In terms of the gender distribution in race, the number of men in Africa and South America is underrepresented, while the number of women in Africa and South America is relatively small.

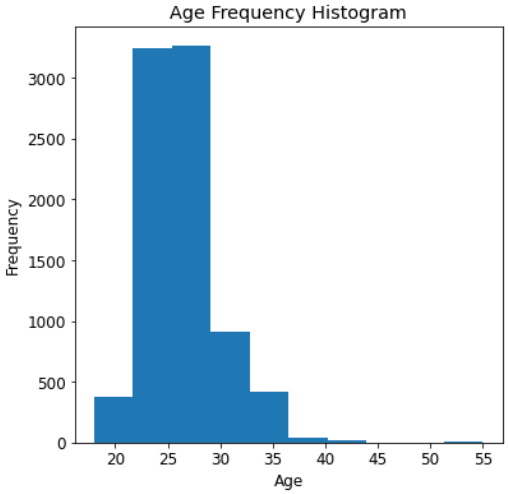


Figure 1.3: Histogram of age in dataset

Another important aspect to check before conducting an analysis is the age of the participants. This is essential for modern dating apps, because the largest proportion of users is between 18-34 so I need to know whether my data set falls within this normal distribution range. First, I plot the age frequency histogram to make sure they were within the realistic range.

After plotting the age variable, I can clearly see that my values are reasonable. The age starts at around 18 years old (in line with expectations), and within a reasonable range for a typical dating app user. In order to better show the age, I constructed a box plot.

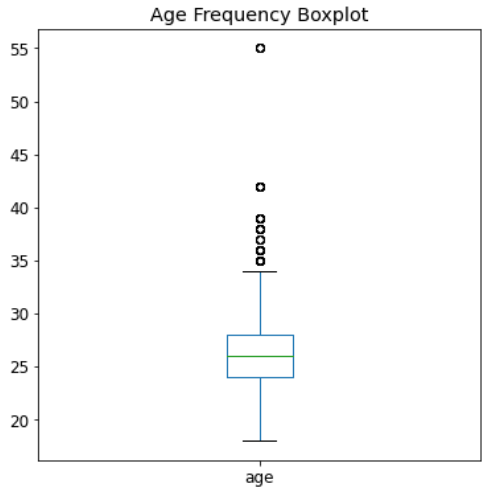


Figure 1.4: Boxplot of age in dataset

Through box plot visualization, it is easier to see that each of my age groups belongs to the normal distribution of dating app users. The outliers from 35 years old are exactly what I want to see when looking at the age range of the participants. As a further check for age differences, the density plot provides a more fine-grained check.

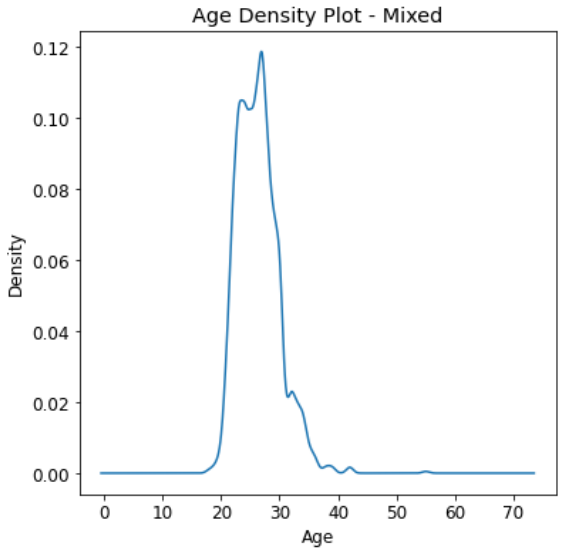


Figure 1.5: Density Plot of age in dataset

Through the above density plot, I can clearly see the mini peaks within the outliers, which gives me a better understanding of the outlier age range in the dataset.

After checking the frequency of matches within gender and checking whether the age range is within the acceptable range for users of the "regular" dating app, the next thing to check is the match itself. It is important to understand the extent to which people can successfully match without having to resort to machine learning.

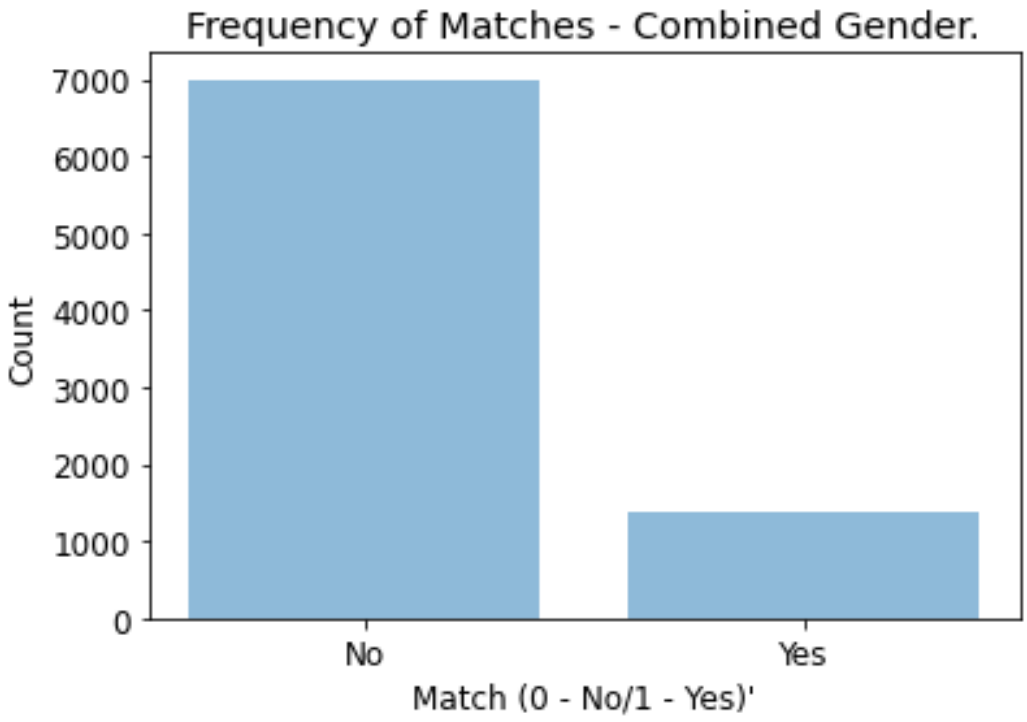


Figure 1.6 Bar chart of 'Match' frequency among genders

It can be clearly seen from Figure 1.6 that most people decide not to pair with their partners. I can only assume lack of matching down to incorrect pairing and the completely random nature of speed-dating events. We can also explain this in terms of the time each person spends with their respective partners. It seems that the time is too short to decide.

Now that I have a good overview of the data, it is time to start examining specific variables. I used a heat map to group all the variables that I thought might be important for this check.

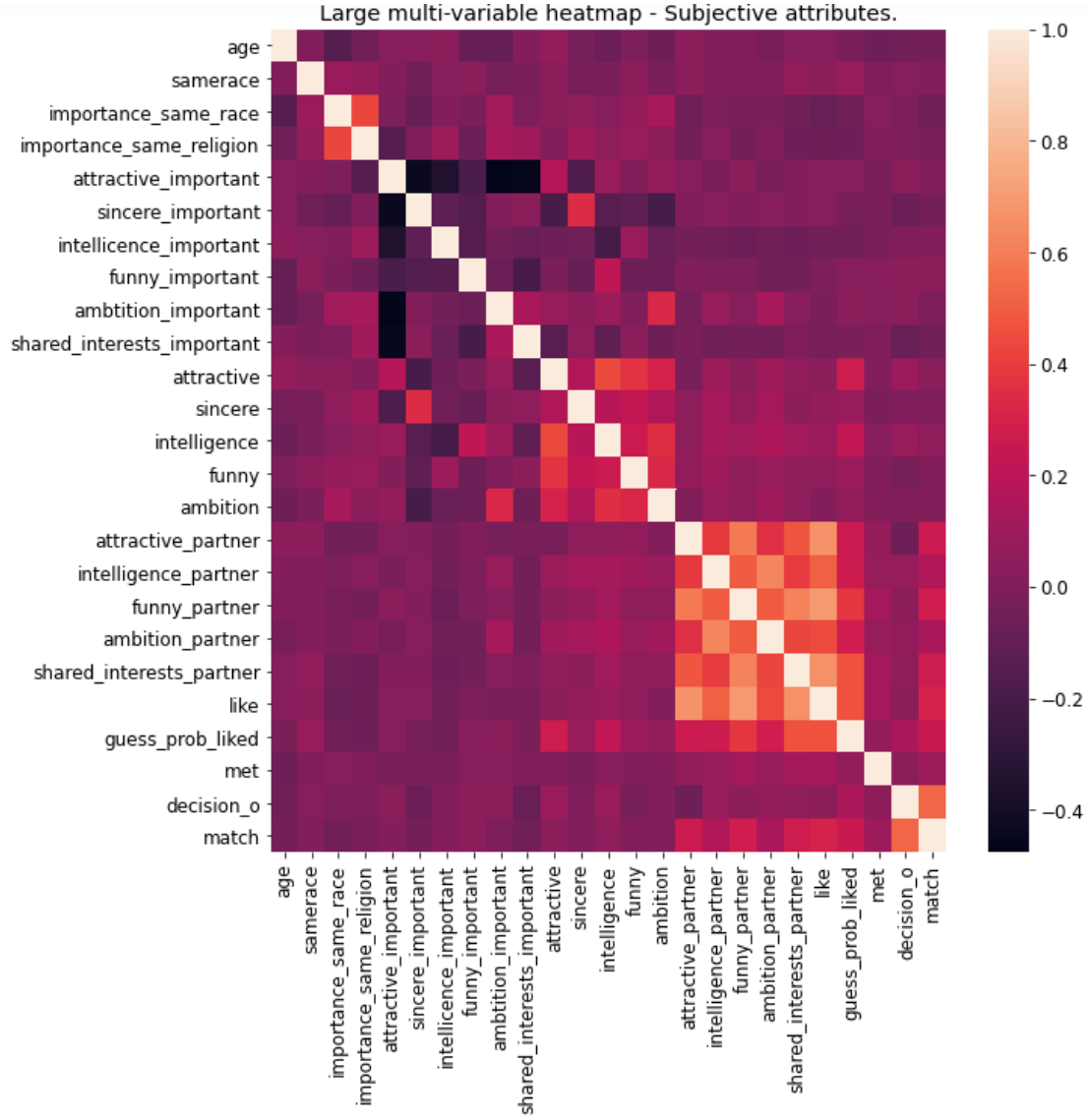


Figure 1.7: Heatmap of subjective attribute variables

This heat map confirms my suspicion that the greatest correlation can be found in the subjective rating of each participant on themselves and each other. We have the expected correlation between ‘like’ and ‘feature_patner’ how you rate your partner and evaluate their various attributes rating. However, other interesting aspects to be aware of are the correlation between the ‘importance_same_race’ and ‘importance_same_religion’ and the correlation between ‘attractive’ and ‘intelligence’. All interesting aspects can be explored in this report.

However, the main point to note from the above figure is the large square of correlation, including "match" and various rating of participants. The analysis of these variables will improve our result.

2.4 Overall Framework

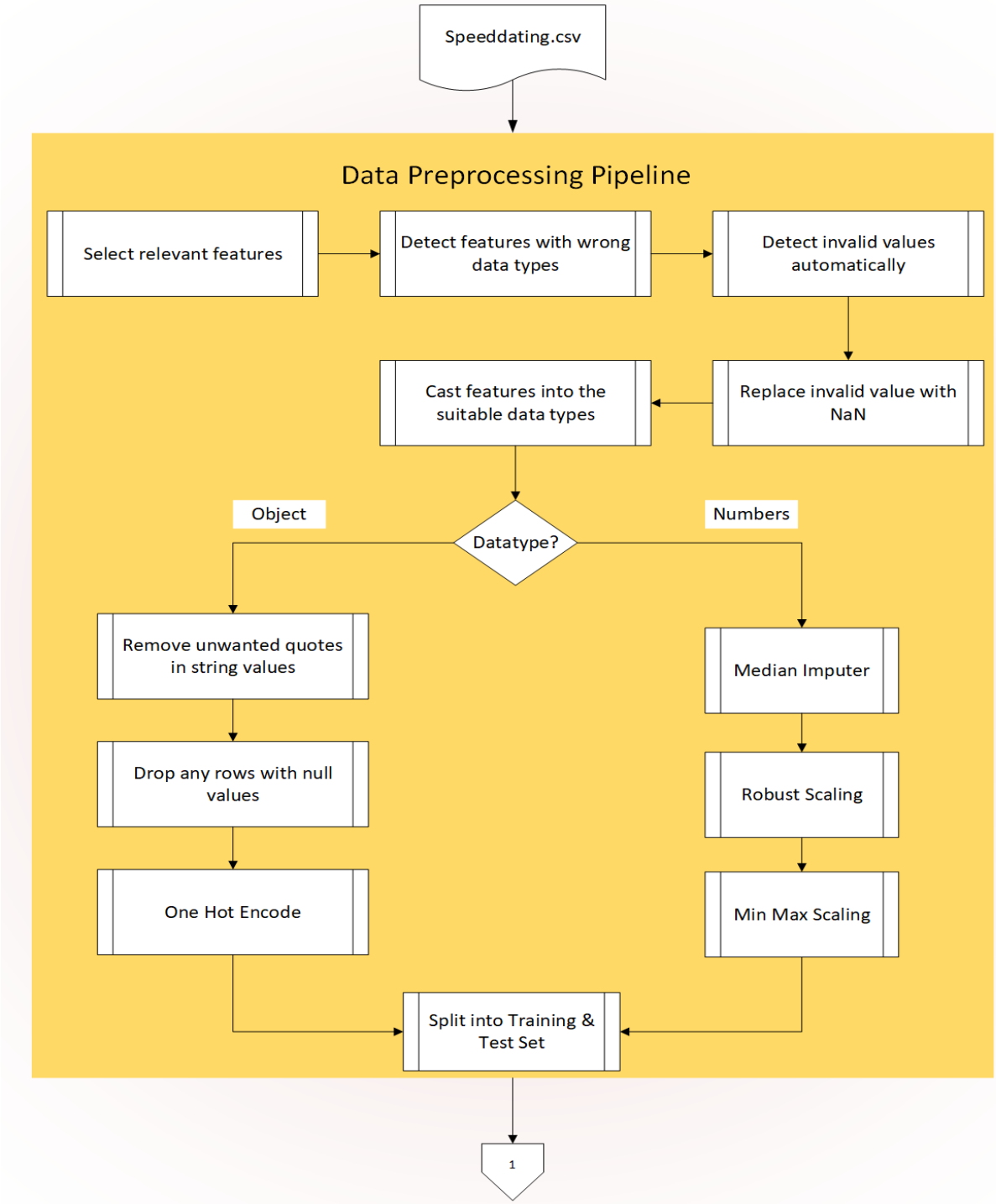


Figure 1.8: Flowchart 1

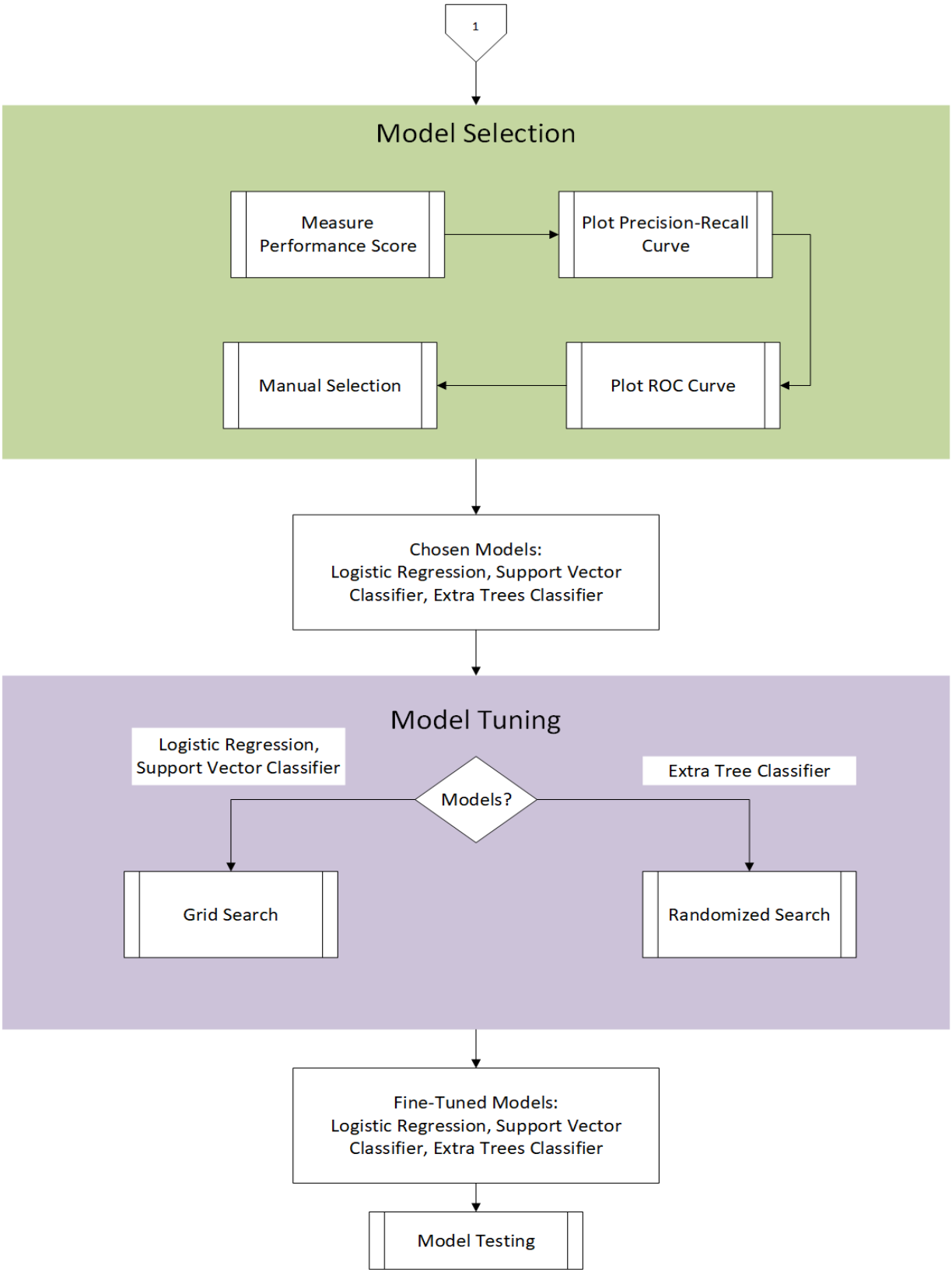


Figure 1.9: Flowchart 2

2.5 Data Preprocessing

Step 1: Drop features

```
new_dating = dating.drop(columns=preproccessed_features+irrelevant_features+self_interest_feature+partner_features, axis=1)
```

List of 94 features that are dropped

Categories	Total features	Reasons
Preprocessed features	56	Prefer raw features over preprocessed features
Self's Interest features	17	Interest features has no significant effect on 'match'
Partner's features	14	Decision made by partner, 'decision_o' feature is kept, therefore features related to partner is not importnant
Other Irrelevant features	7	These features has no significant effect on 'match' or the features meaning is not clear understood.

Note: List of dropped features, refer to Appendix B.

```
irrelevant_features = ['has_null',
                       'wave',
                       'expected_happy_with_sd_people',
                       'expected_num_interested_in_me',
                       'expected_num_matches',
                       'field',
                       'decision']

self_interest_feature = ['sports',
                         'tvsports',
                         'exercise',
                         'dining',
                         'museums',
                         'art',
                         'hiking',
                         'gaming',
                         'clubbing',
                         'reading',
                         'tv',
                         'theater',
                         'movies',
                         'concerts',
                         'music',
                         'shopping',
                         'yoga']

partner_features = ['age_o',
                    'race_o',
                    'pref_o_attractive',
                    'pref_o_sincere',
                    'pref_o_intelligence',
                    'pref_o_funny',
                    'pref_o_ambitious',
                    'pref_o_shared_interests',
                    'attractive_o',
                    'sinsere_o',
                    'intelligence_o',
                    'funny_o',
                    'ambitous_o',
                    'shared_interests_o']
```

Note: List of attributes description, refer to Appendix A.

Step 2: Creating a Pipeline by initializing Custom Transformer called DataPreprocessTransformer

2.1: Replace '?' value with NaN

'?' value must be replaced with np.NaN in order to proceed to step 2.2 successfully.

2.2 Change numerical features with 'object' data type and change to 'float64'

2.3 Remove unwanted quotes: change values like "Example" to 'Example'

	race	race_o	san
]	'Asian/Pacific Islander/Asian-American'	European/Caucasian-American	
]	'Asian/Pacific Islander/Asian-American'	European/Caucasian-American	
]	'Asian/Pacific Islander/Asian-American'	'Asian/Pacific Islander/Asian-American'	

Some samples contain duplicated open and closing single quotes, for example, 'Asian/Pacific Islander/Asian-American' in the 'race' feature instead of Asian/Pacific Islander/Asian-American.

2.4 Drop any rows with null values for categorical attribute

2.5 numerical_pipeline consists of:

A. SimpleImputer(strategy='median'),
B. RobustScaler(),
C. MinMaxScaler()

2.6 categorical_pipleline consists of OneHotEncoder (pd.dummies)

data_preprocess_pipeline consists of numerical pipeline and classification pipeline

Step 3: Split training and test set using sklearn.model_selection.StratifiedShuffleSplit

StratifiedShuffleSplit is used instead of sklearn.model_selection.train_test_split that implementing randomized splitting since the dataset is imbalanced, that is, 83.53% of negative (not match) and 15.47% of positive (match) samples, respectively. As a result, the percentage of the samples is maintained based on the proportion of 'match' after splitting.

2.6 Model Selection

Phase 1: Performance Score

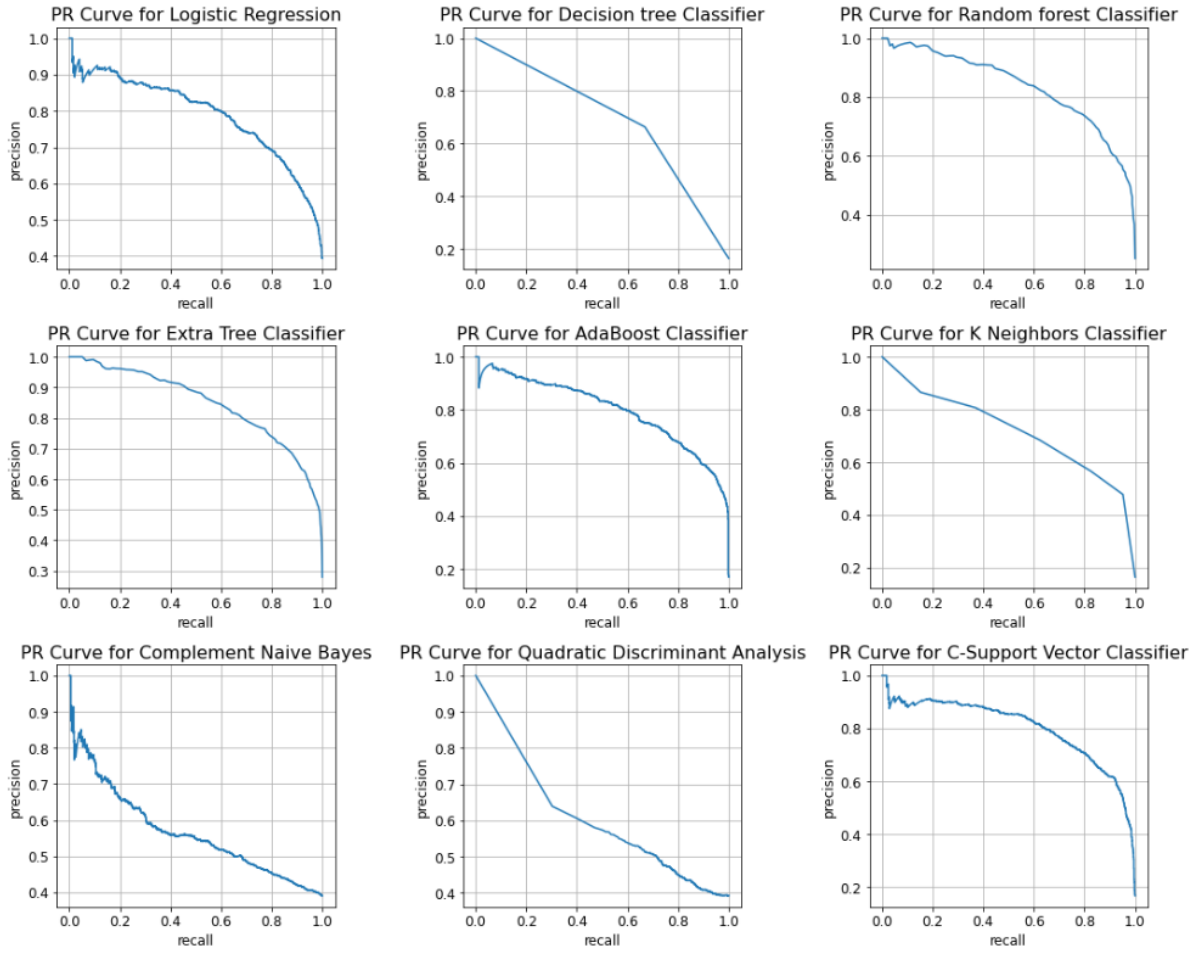
List of classifiers chosen for model selection

- Logistic Regression
- Decision tree
- Support Vector
- K neighbours
- Complement Naive Bayes
- Quadratic Discriminant Analysis
- Ensemble Classifiers
    - Random forest
    - Extra Trees Classifier
    - AdaBoost Classifier

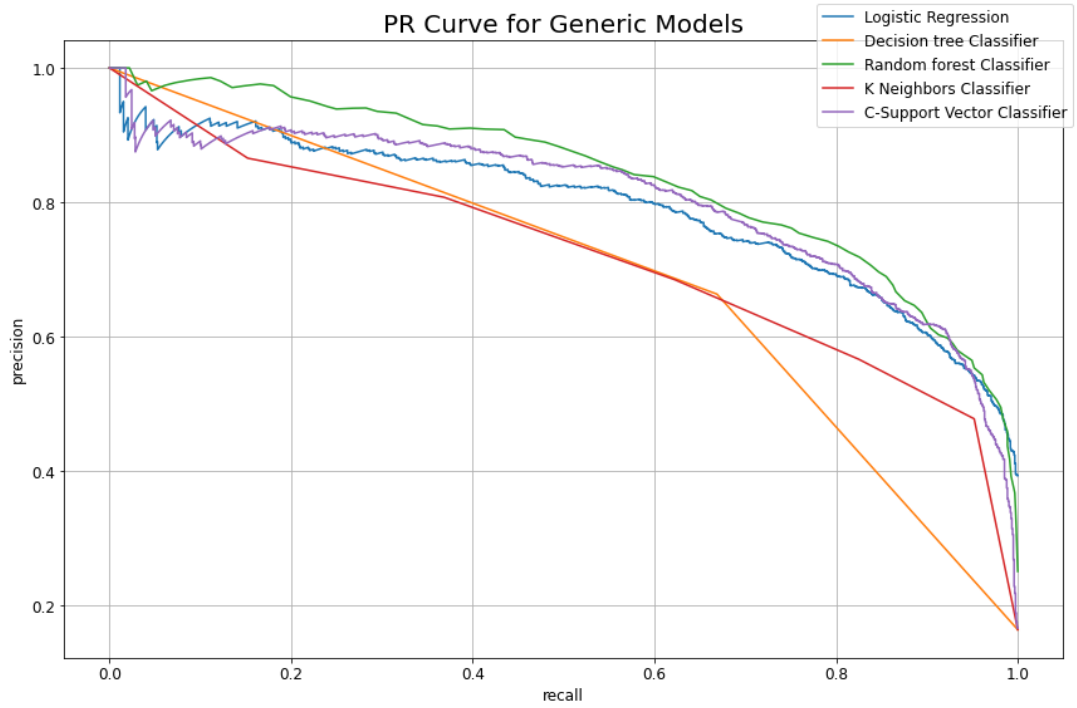
	classifier_name	duration	precision	recall	f1_score	auc_score
0	Logistic Regression	2.0780	0.753518	0.711350	0.731806	0.961143
1	Decision tree Classifier	0.3634	1.000000	1.000000	1.000000	1.000000
2	Random forest Classifier	2.4752	1.000000	1.000000	1.000000	1.000000
3	Extra Tree Classifier	2.1710	1.000000	1.000000	1.000000	1.000000
4	AdaBoost Classifier	1.8590	0.769109	0.746744	0.757748	0.967500
5	K Neighbors Classifier	6.6854	0.817422	0.745115	0.779519	0.972986
6	Complement Naive Bayes	0.2184	0.400349	0.986776	0.569572	0.899252
7	Quadratic Discriminant Analysis	0.5482	0.397856	0.965005	0.562202	0.914379
8	C-Support Vector Classifier	14.3870	0.800848	0.740032	0.769218	0.966973

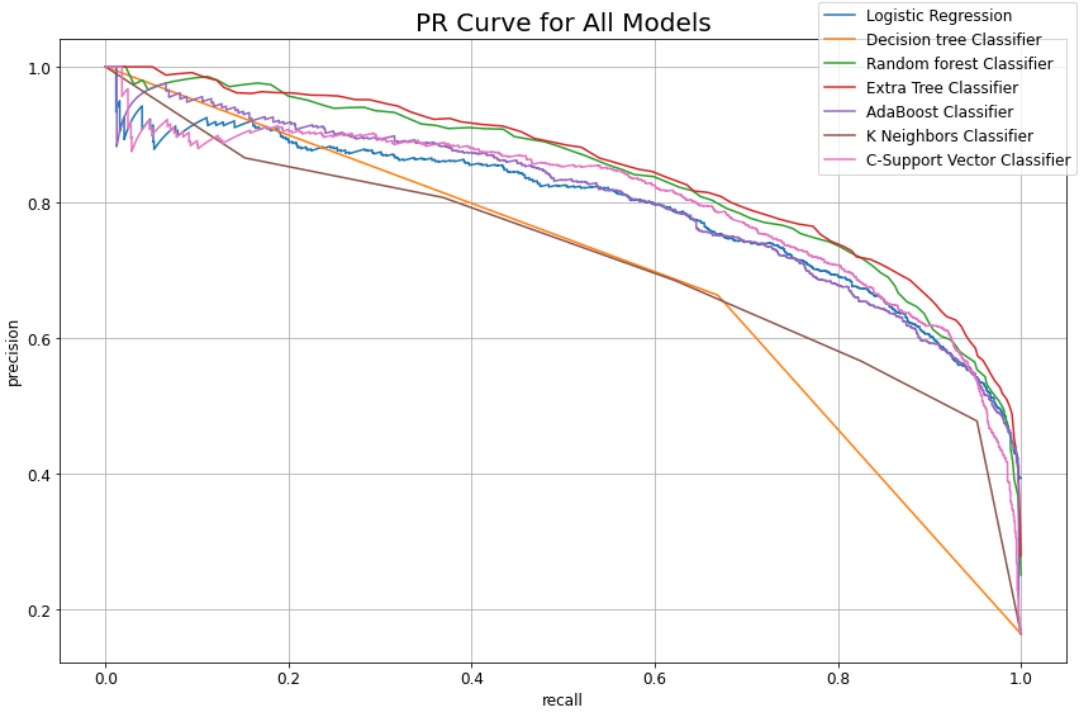
Based on the table above, all the classifiers were able to achieve more than 0.70 recall rate. Ensemble methods which are Random forest, Extra Trees Classifier, AdaBoost Classifier are clearly overfits the training set, while Complement Naive Bayes, Quadratic Discriminant Analysis performs very poor on precision.

Phase 2: Precision-Recall Curve



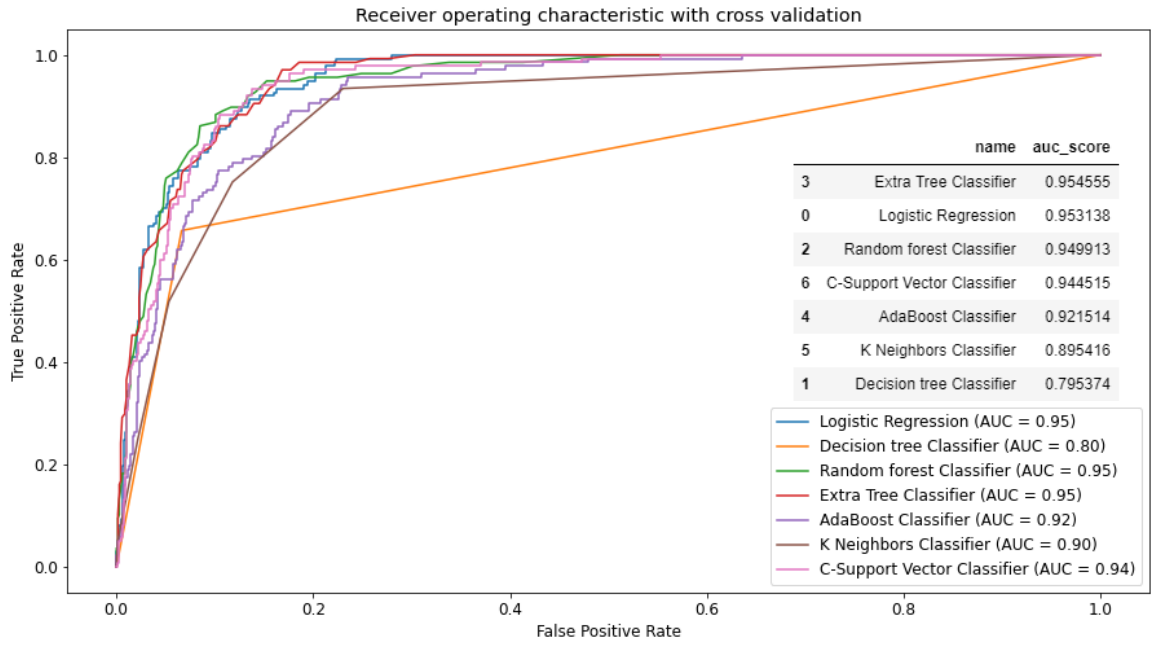
Based on the cross-validated precision-recall curve above, Complement Naive Bayes, Quadratic Discriminant Analysis are discarded since both performed even worse than a purely random dummy classifier, that is, the area under curve is less than 5.0.





Combining all cross-validated precision recall curve under one plot, we can clearly observe that Decision Tree Classifier and K Neighbours Classifier have the lowest area under curve as compared to the less of the models. Therefore, these models are later discarded in the phase 4.

Phase 3: Receiver Operating Characteristic (ROC) Curve



	name	auc_score
3	Extra Tree Classifier	0.954555
0	Logistic Regression	0.953138
2	Random forest Classifier	0.949913
6	C-Support Vector Classifier	0.944515
4	AdaBoost Classifier	0.921514
5	K Neighbors Classifier	0.895416
1	Decision tree Classifier	0.795374

Using area under curve of cross-validated ROC Curve as an indicator of overall algorithm performance, Extra Tree Classifier is the best model, followed by Logistic Regression, Random Forest Classifier, and C-Support Vector Classifier.

Phase 4: Final Selection

List of Chosen Models

1. Logistic Regression

2. Extra Tree Classifier

3. C-Support Vector Classifier

Extra Trees Classifiers, Logistic Regression, Support Vector are chosen since these models are top 4 highest in AUC score in ROC Curve. Random Forest Classifier are discarded since it is also an ensemble method which performs slightly worse than Extra Trees Classifiers.

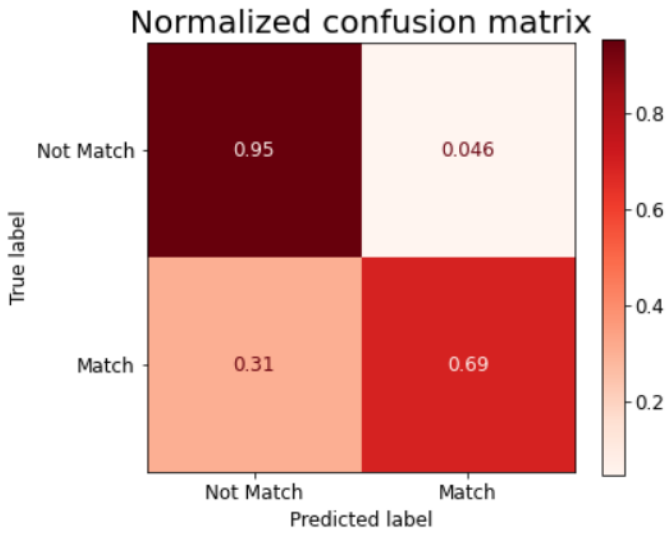
3.1 Model 1: Logistic Regression

Logistic regression is a classification algorithm which is used when the output feature or target variable is categorical in nature. It is commonly used when the binary output is found in the dataset, such as it belongs to a class or another, or is either 1 or 0. Since the target variable of our chosen dataset is determining whether the users would match with the partners given which can be considered as a binary output consisting (1-Yes) and (2-No) only. Therefore, we have chosen the Logistic Regression as one of the three models to train and perform classification in our assignment.

During the model pre-evaluation stage, we have created a Logistic Regression classifier with default settings which called “log_reg” and train the model using the “X_train” and “y_train” from the dataset which we have preprocessed.

```
# Logistics Regression with Default Settings
log_reg = LogisticRegression(penalty='elasticnet', random_state=RANDOM_SEED,
                             solver='saga', max_iter=3000, l1_ratio=0.5, C=1.0)
log_reg.fit(X_train, y_train)
```

The following image has shown the confusion matrix which shows the default performance of the Logistic Regression model after performing the cross-validated prediction of all the sample in the training sets. Based on the confusion matrix, we found that there are 69% of the actual match participants are correctly predicted as “match” by the model. 31% of actual match participants predicted as “not match”. Moreover, 95% of the actual unmatched participants are correctly predicted as “not match” by the model. There only exists 4.6% of the unmatched participants who are wrongly predicted as “match”.



Besides that, we have improved the Logistic Regression model by tuning the hyperparameters in the model using GridSearchCV in order to enable us to select the best model. The "l1_ratio" which represents the Elastic-Net mixing parameter in GridSearchCV is separated evenly into 5 sets between 0 (penalty='l2') and 1 (penalty='l1'). The "C" which represents the inverse of regularization strength in GridSearchCV is assigned with 8 positive values (0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30).

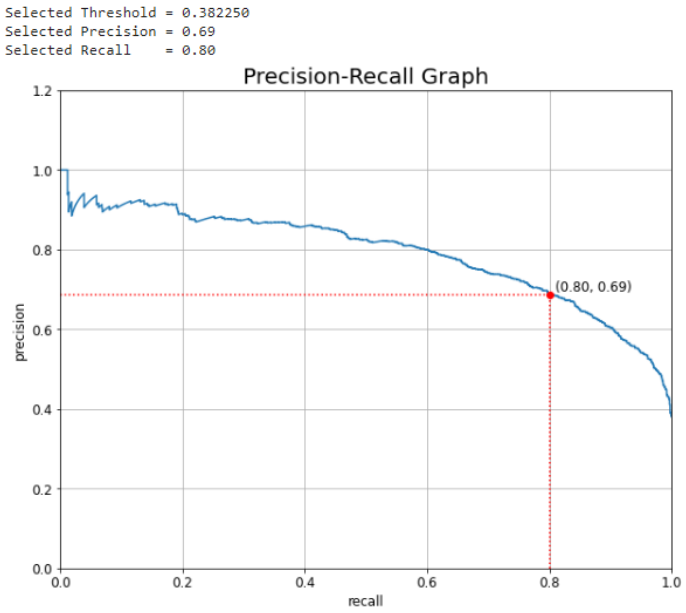
```
log_reg_param_grid = {
    'l1_ratio': np.linspace(0, 1, 5),
    'C': [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]
}
```

We have eventually come out the best parameters for the Logistic Regression classifier "tuned_log_reg", which "C" equals to 3 and "l1_ratio" is 1.0.

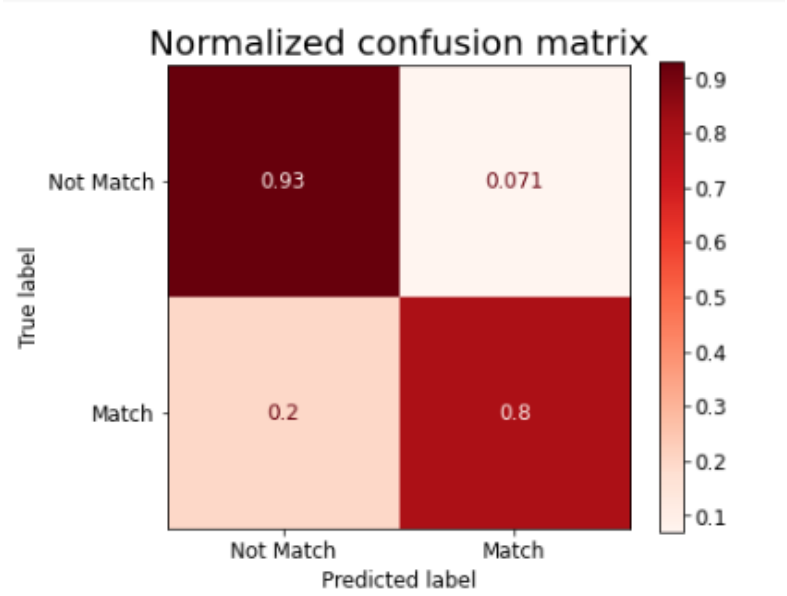
```
print(f'Best Params for Logistic Regression: {log_grid_search.best_params_}')
```

```
Best Params for Logistic Regression: {'C': 3, 'l1_ratio': 1.0}
```

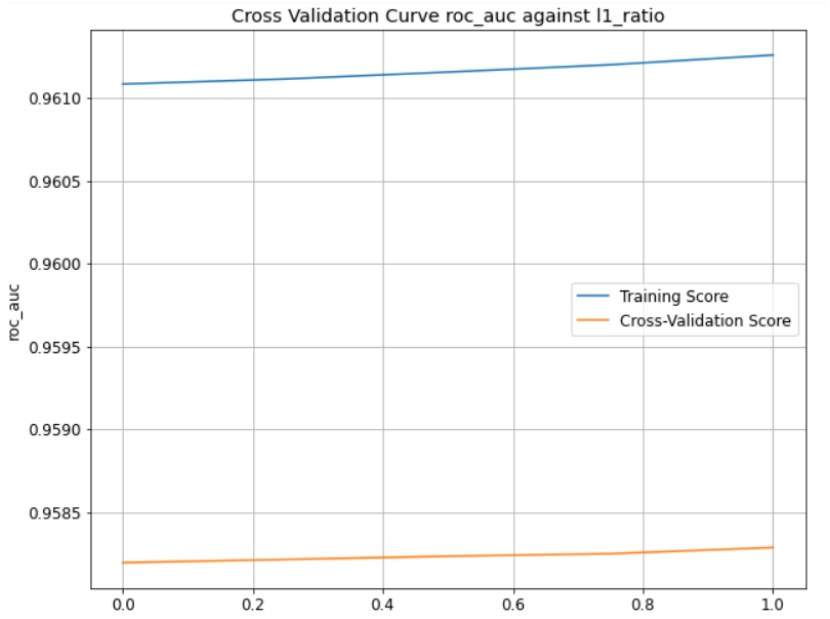
The precision-recall graph is soon plotted after we obtained the best model for the Logistic Regression. Since we need to obtain actual matched participants in the dataset that are correctly predicted. Hence, we have to choose the performance measure of the model with higher recall and can tolerate a lower precision rate. We have eventually achieved 0.8 recall and 0.69 precision rate with threshold equals to 0.382250.



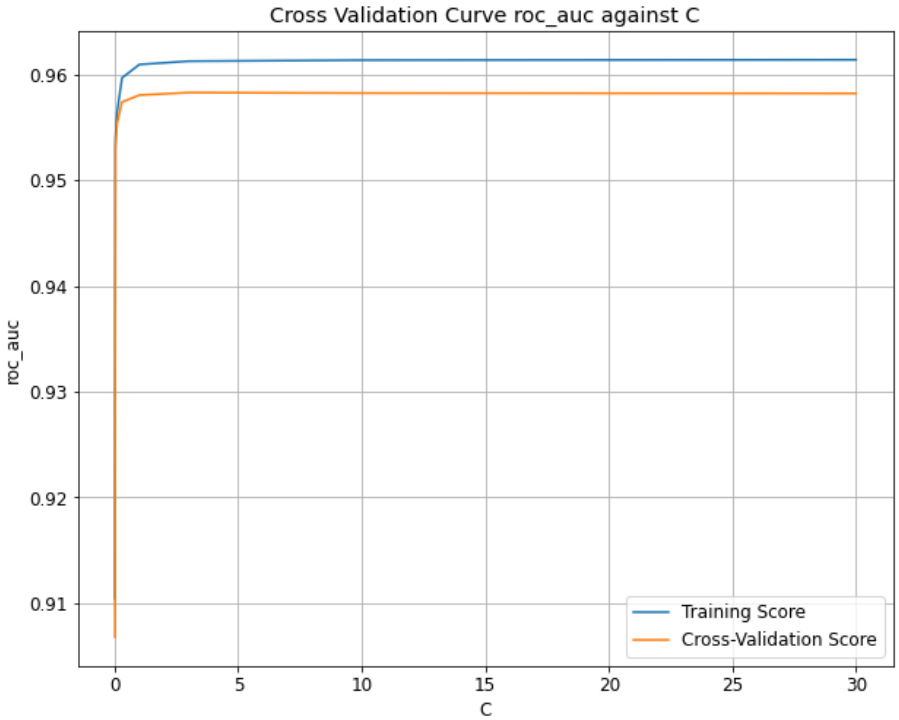
The confusion matrix is plotted by the using the best Logistic Regression model and training sets. We found that there are slightly difference results comparing to the previous default confusion matrix. The true positive value has increased from 69% to 80%. The false negative value has dropped from 31% to 20%. However, true negative has dropped from 95% to 93% and the false positive has increased from 4.6% to 7.1%.



The graph at below has shown the CV scores against difference “l1_ratio” for both the training score and cross-validate score. We have found that the l1_ratio with value 1.0 generates the highest CV score which is around 0.96.



Lastly, the graph below has shown the CV scores against difference "C" for both the training scores and cross-validate scores. We have achieved the highest CV scores around 0.96 when the value of "C" is approximately to 3.



3.2 Model 2: Support Vector Classifier

Support Vector Classifier (SVC) is a supervised learning model with learning algorithms that analyze the data used for classification. The purpose of SVC is to maximize the gap separating classes, so when new examples are mapped to the same space, depending on which side of the gap they fall on, they will be determined to belong to one of the two classes.

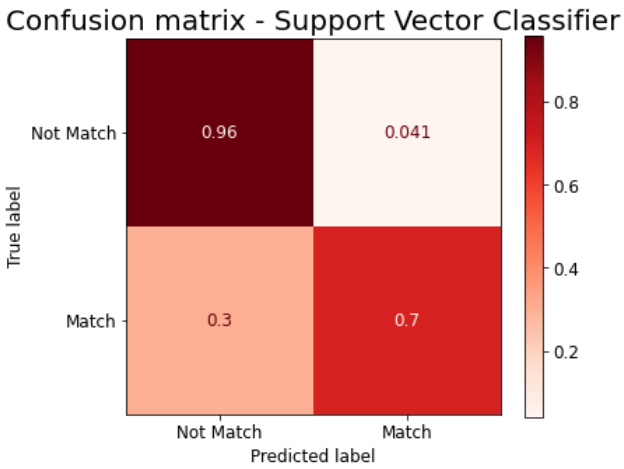
In SVC, using the correct hyperplane to maximize this margin between classes will optimize the model, and the vector points that touch this margin are called "support vectors" (hence the name "support vector machine"). Provide training examples for SVC, each example is marked as belonging to one of two categories, and then a model is built to predict new examples as belonging to one of the categories. Thus, we will use SVC (Support Vector Classifier) - a binary linear classifier as one of the models, which is great for classifying users as "match" or "not match".

During the model pre-evaluation stage, we have created an Support Vector Classifier with default settings which called “svm_model” and train the model by using the “X_train” and “y_train” which have been split from the chosen dataset.

Pre-Evaluation

```
[ ]  # C-Support Vector Classifier with Default Settings
     svm_model = SVC(random_state=RANDOM_SEED, probability=True)
     svm_model.fit(X_train, y_train)
```

The figure below has shown the confusion matrix which shows the default performance of the Support Vector Classifier model after performing the cross-validated prediction of all training samples. Based on the confusion matrix, we can see that there are 70% of the actual match participants are correctly predicted as “match” by the model. 30% of actual match participants predicted as “not match”. Moreover, 96% of the actual unmatched participants are correctly predicted as “not match” by the model. There only exists 4.1% of the unmatched participants who are wrongly predicted as “match”.



Futhermore, we have improved the Support Vector Classifier model by tuning the hyperparameters inside the model using GridSearchCV in order to make us be able to select the best model. The “C” is a parameter of the SVC learner and is the penalty for misclassifying a data point in GridSearchCV is assigned with 6 positive values (0.1,1,10,100,1000,10000). The “gamma” is a parameter for nonlinear hyperplanes, the higher the gamma value it tries to exactly fit the training data set in GridSearchCV is assigned with 5 positive values (1, 0.1, 0.01, 0.001, 0.0001).

Grid search

```
[ ] #List of parameters and different combinations to produce a best result
    svm_param_grid = {'C': [0.1, 1, 10, 100, 1000, 10000], 'gamma': [1, 0.1, 0.01, 0.001, 0.0001]}
```

```
    svm_model = clone(svm_model)

    svm_grid_search = GridSearchCV(svm_model, svm_param_grid, cv=3, scoring='roc_auc', return_train_score=True, verbose=10)
    svm_grid_search.fit(X_train, y_train)
    results = svm_grid_search.cv_results_
```

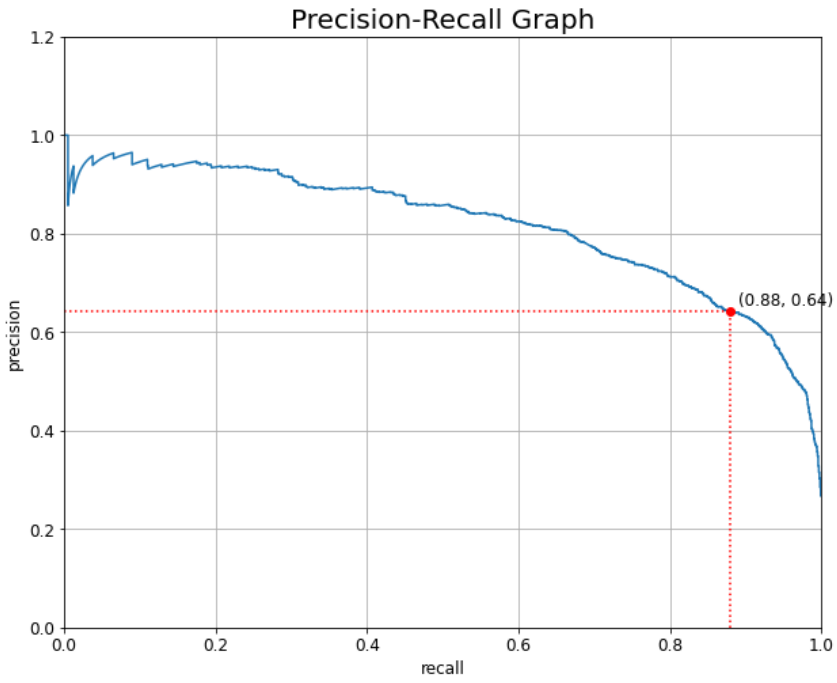
We have the best parameters result for the Support Vector Classifier model “tuned_svm_model”, which “C” is 100 and “gamma” is 0.1.

```
[ ] print(f'Best Params for Support Vector Classifier: {svm_grid_search.best_params_}')
```

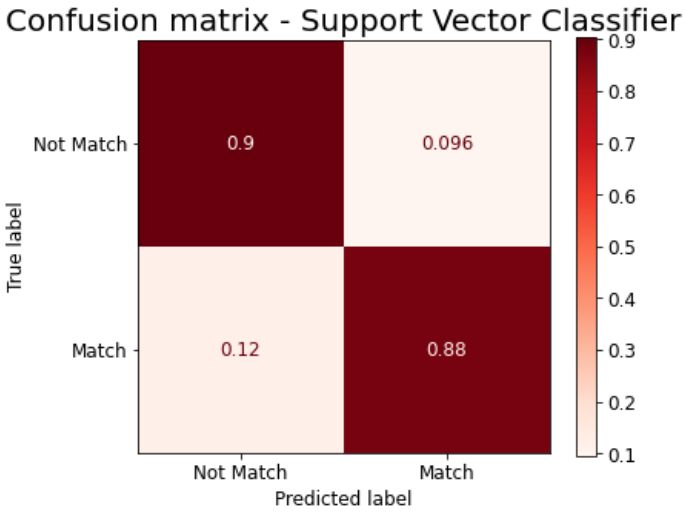
```
Best Params for Support Vector Classifier: {'C': 100, 'gamma': 0.1}
```

The precision-recall graph is plotted after we obtained the best model for the Support Vector Classifier. In order to reach to the accurate prediction on the participants correctly predicted as matched with the partners. We have chosen a higher recall rate which around 0.88 and a lower

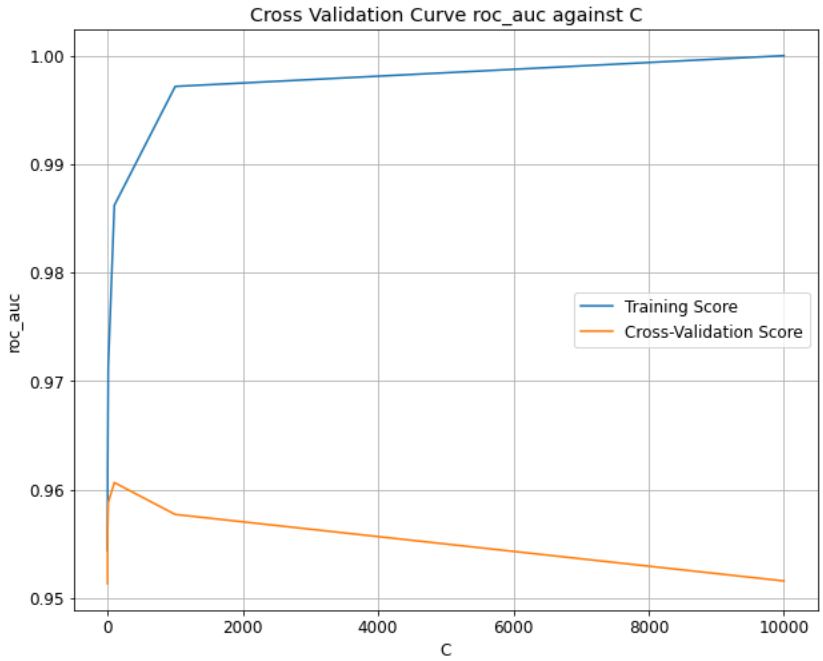
precision rate which is around 0.64. The threshold value for both chosen recall and precision is 0.25.



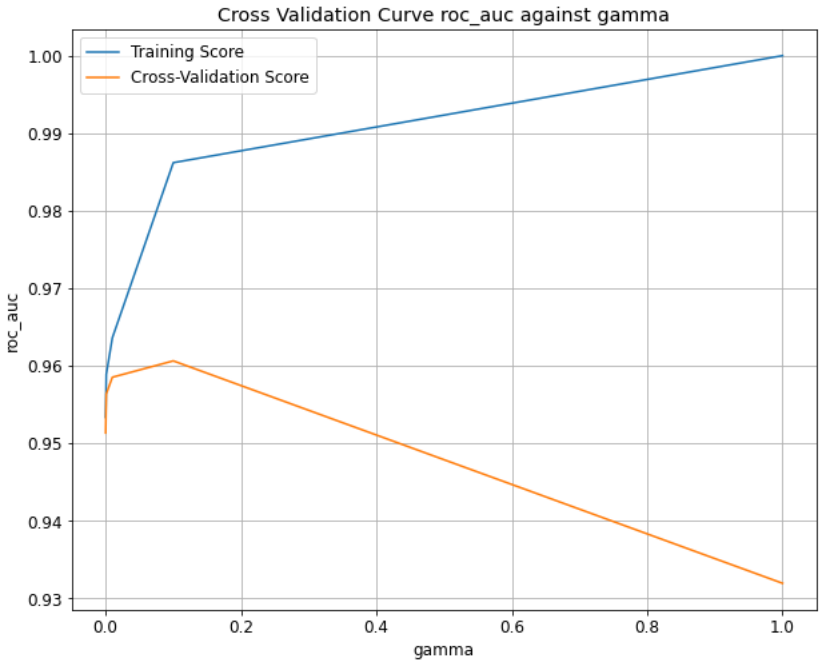
The confusion matrix is plotted by the using the best Support Vector Classifier model and training sets. We found that there are slightly difference results comparing to the previous default confusion matrix. The true positive value has increased from 70% to 88%. The false negative value has dropped from 30% to 12%. However, true negative has dropped from 96% to 90% and the false positive value has also increased from 4.1% to 9.6%.



The graph below has shown the CV scores against difference “C” for both the training score and cross-validate score. We have found that the “C” with value 100 generates the highest CV score which is around 0.96.



Lastly, the graph below has shown the CV scores against difference “gamma” for both the training scores and cross-validate scores. We have achieved the highest CV scores around 0.96 when the value of “gamma” with value 0.1.



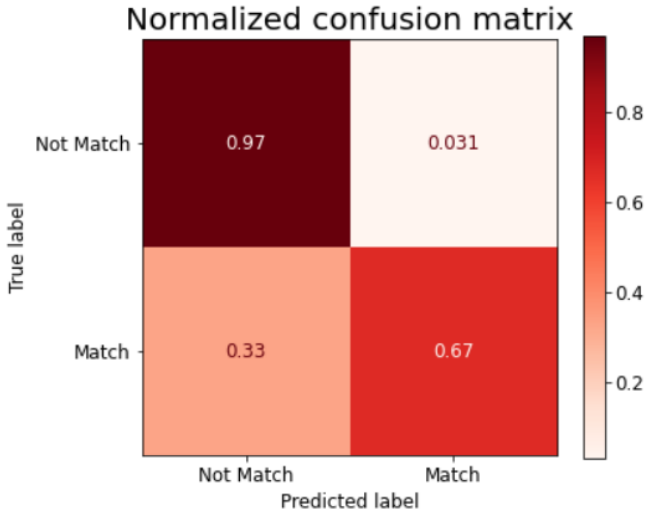
3.3 Model 3: Extra Trees Classifier

Extremely Randomized Tress, known as Extra Trees, is an ensemble machine learning algorithm which aggregates the result of multiple de-correlated decision trees collected in the forest in order to perform classification for the inputs. The Extra Trees will randomly sample all the features at each split point of decision tree which is similar like the random forest algorithm. However, it selects split point at random without using the greedy algorithm unlike random forest. Hence, we have chosen the Extra Tree Classifier as one of the models used to perform classification method towards in our assignment.

During the model pre-evaluation stage, we have created an Extra Trees model with default settings which called "extra_trees_model" and train the model by using the training sets which have been split from the chosen dataset.

```
[ ]  # C-Support Vector Classifier with Default Settings
     extra_trees_model = ExtraTreesClassifier(random_state=RANDOM_SEED)
     extra_trees_model.fit(X_train, y_train)
```

The following confusion matrix shows the default performance of the Extra Trees model after performing cross-validated prediction of all the training samples. Based on the confusion matrix, we can find that there exists 67% of the actual matched participants are correctly predicted by the model. There are 33% of the participants who actual matched but predicted as "not match" by the model. Moreover, there are 97% of the participants are correctly predicted as "not match" and the remaining 3.1% predicted as wrongly "match".



Moreover, we have improved the Extra Trees model by tuning the hyperparameters inside the model using RandomSearchCV in order to make us be able to select the best model. The "n_estimator" represents the number of trees in the forest. The "max_depth" represents the maximum depth of the tree. Furthermore, "min_samples_split" shows the minimum number of samples required to split an internal node and the "min_samples_leaf" shows the minimum number of samples required to be at a leaf node.

Randomized search

```
[ ] tree_param_grid = {
        'n_estimators'      : np.linspace(10, 200, num=5, endpoint=True, dtype="int64"),
        'max_depth'         : np.linspace(1, 30, num=5, endpoint=True, dtype="int64"),
        'min_samples_split' : np.linspace(2, 30, num=5, endpoint=True, dtype="int64"),
        'min_samples_leaf'  : np.linspace(1, 30, num=5, endpoint=True, dtype="int64"),
    }
```

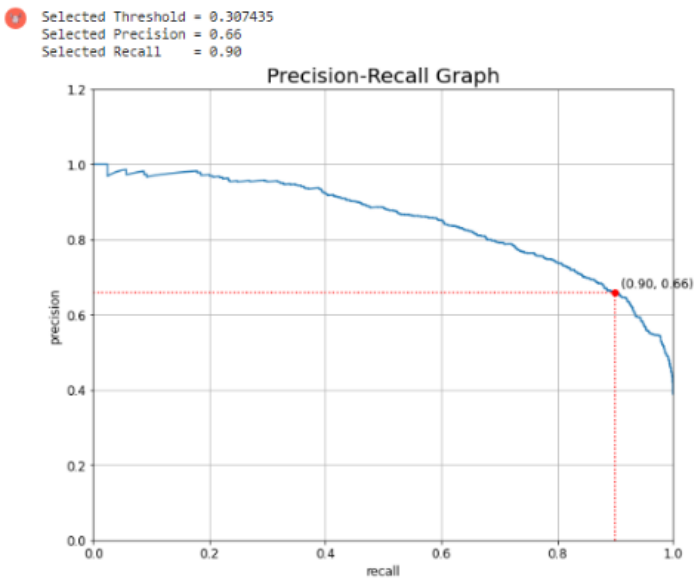
```
[ ] svm_model = clone(svm_model)

    tree_grid_search = RandomizedSearchCV(extra_trees_model, param_distributions=tree_param_grid, n_iter = 100, cv = 3, verbose=1, random_state=RANDOM_SEED)

    tree_grid_search.fit(X_train, y_train)
    results = tree_grid_search.cv_results_
```

We have found the best parameters for the Extra Trees classifier "tuned_extra_trees_model", which 'n_estimator = 200', 'min_sample_split = 9', 'min_samples_leaf = 1' and 'max_depth = 30'.

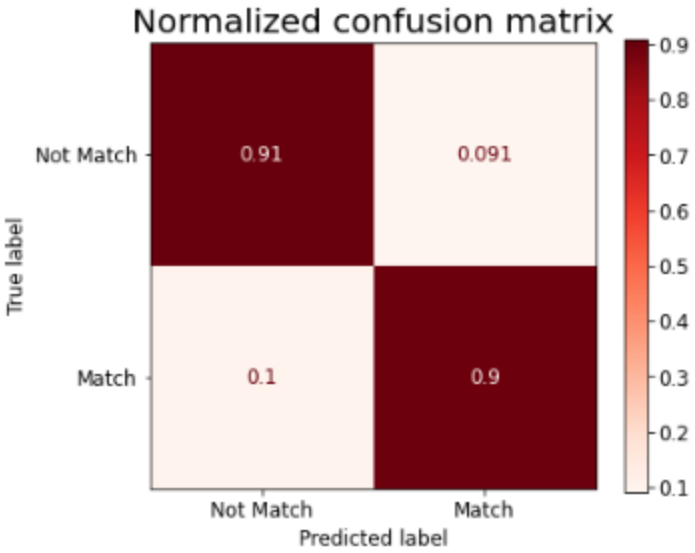
```
print(f'Best Params for Extra Trees Classifier: {tree_rand_search.best_params_}')
```

```
Best Params for Extra Trees Classifier: {'n_estimators': 200, 'min_samples_split': 9, 'min_samples_leaf': 1, 'max_depth': 30}
```

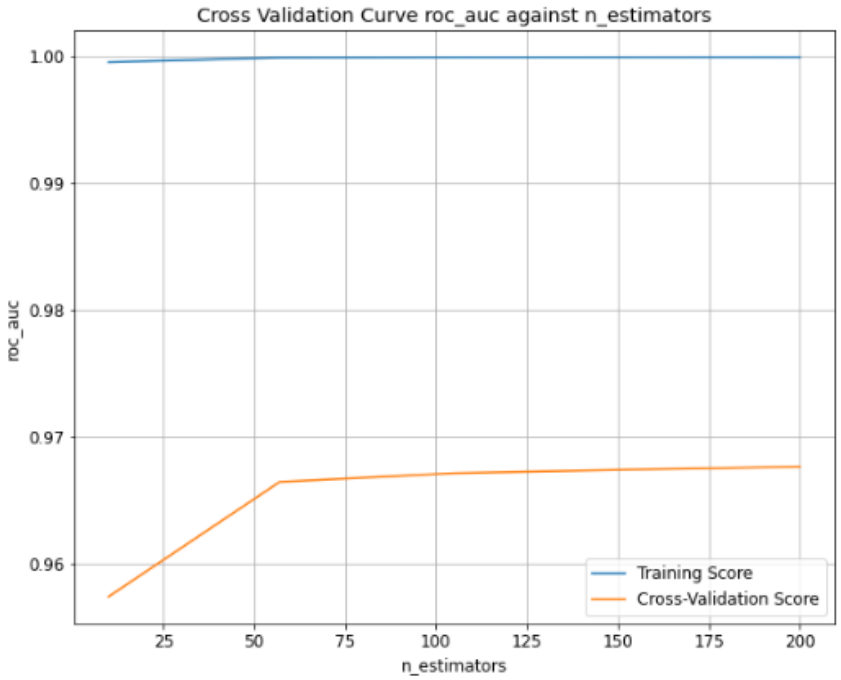
The precision-recall graph is plotted after we obtained the best model for the Extra Trees Classifier. In order to reach to the accurate prediction on the participants correctly predicted as matched with the partners. We have chosen a higher recall rate which around 0.90 and a lower precision rate which is around 0.66. The threshold value for both chosen recall and precision is equal to 0.307435.



The confusion matrix is plotted by using the best Extra Trees classifier together with the training sets. We found that there is a huge improvement for the true positive value comparing to the previous default confusion matrix. The true positive value has increased from 67% to 90% and the false negative has dropped from 33% to 10%. However, there is a slightly drop for the true negative value which is from 97% to 91% and the false positive value has also increased from 3% to 9.1%.



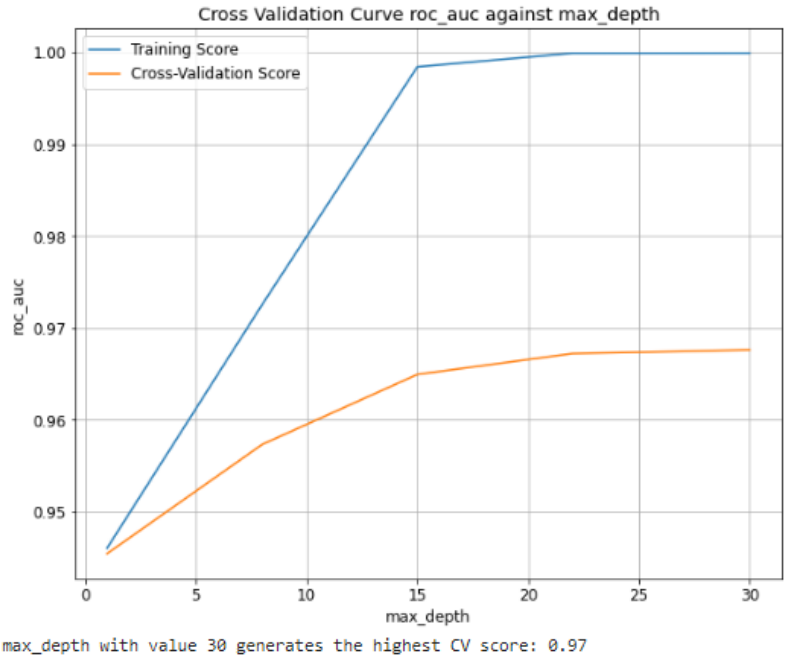
The below graph has shown the CV scores against difference “n_estimators” for both the training score and cross-validate score. We have found that the “n_estmator” with value 200 will generate the highest CV score which is around 0.97.




```
n_estimators with value 200 generates the highest CV score: 0.97
```

Secondly, the below graph has shown the correlation between "max_depth" and the CV scores for both training score and cross-validate score. We have also found that while the "max_depth" is equal to 30 will generate the highest CV score.

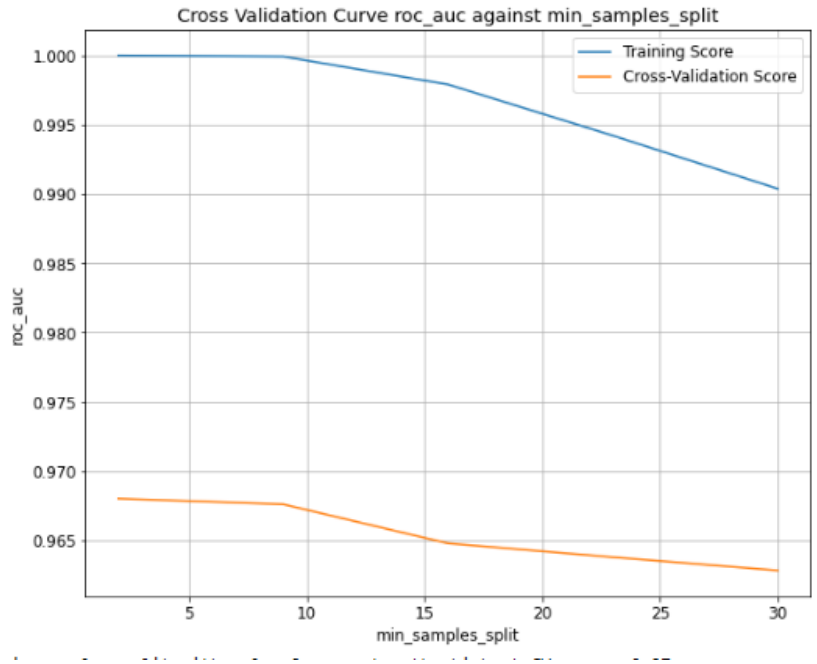
```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  15 out of  15 | elapsed:   11.4s finished
```



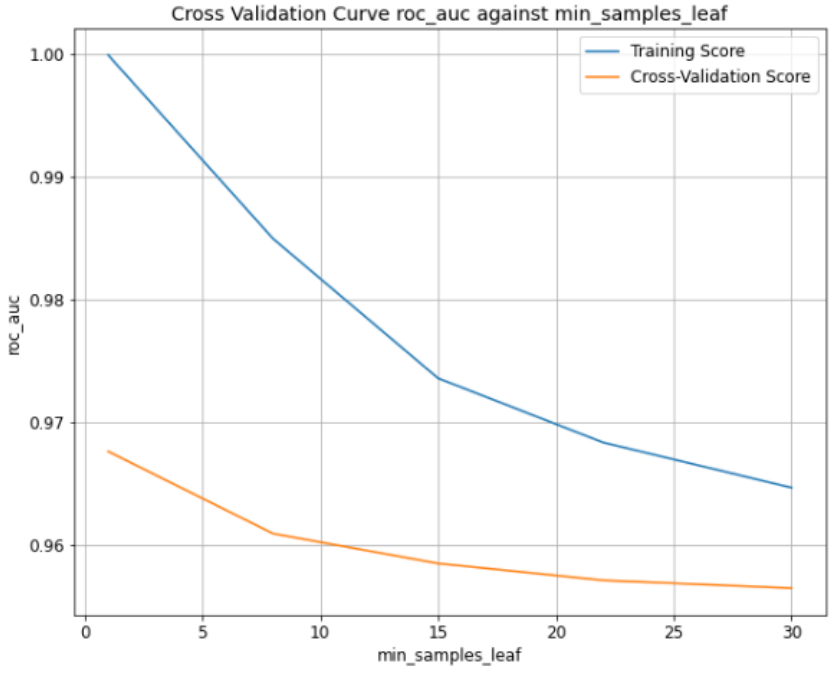

```
max_depth with value 30 generates the highest CV score: 0.97
```

Thirdly, we have also found the correlation between the “min_samples_split” and CV score. The highest CV score can be obtained while the “min_samples_split” is equal to 9.

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  15 out of  15 | elapsed:   14.0s finished
```



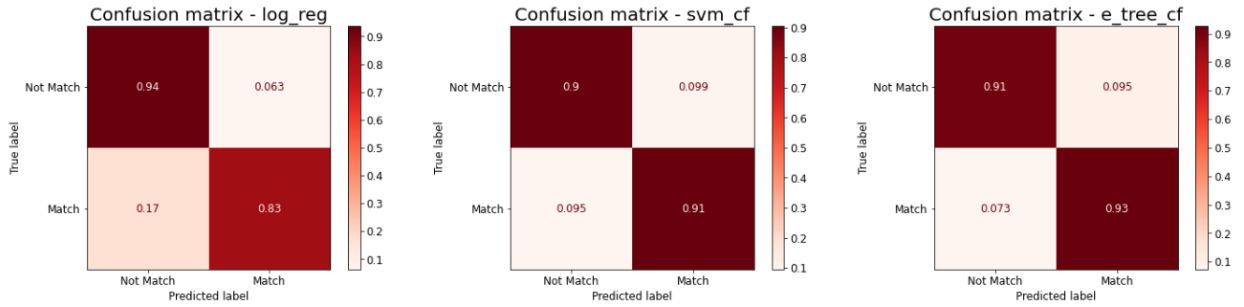
Lastly, while the “min_samples_leaf” with value 1 will generate the highest CV score.




```
min_samples_leaf with value 1 generates the highest CV score: 0.97
```

3.4 Model Comparisons

During the model testing phase, we have plotted the confusion matrices for both fine- tuned Logistic Regression classifier, Support Vector classifier and Extra Trees Classifier together with the testing sets. By comparing to the three confusion matrices, we found that the Extra Trees Classifier achieved the highest true positive value which is 93%. Then, it is followed by Support Vector Classifier which is 91%. The true positive value of the Logistic Regression classifier is the lowest among three classifiers which is 83%. For the true negative value, the Logistic Regression classifier has achieved the highest value which is 94%, then followed by Extra Trees classifier which is 91% and the Support Vector classifier obtained the lowest true negative value among three classifiers which is 90%.



4.1 Conclusion

We have chosen the Extra Trees Classifier as our model to perform the prediction towards the participants. Since our objective is to estimate whether the participants would be accurately predicted by the model as “match” with their partners.

Therefore, we need to choose the model with the highest recall rate among all the other classifiers. Based on the results we have obtained during the model testing phase; we knew the True Positive value for each classifier. The Extra Tree Classifier has achieved the highest true positive value which is 93%. Since we knew that recall = TP*(FN+TP). Hence, we can tell that the recall value is proportional to the true positive value. Therefore, the Extra Trees Classifier has been chosen as the most appropriate model for the assignment.

For the recommendation, we think that we can use the Ensemble Learning method, which is a method collects several models to work together on a single set. For instance, we can use the Voting Classifier to combine the predictions from all our machine learning algorithm. It can help us to lower the errors and reduce the over-fitting situation from occurring. The Voting Classifier works by chosen the output based on the majority vote according to the different strategies. For example, we assumed Classifier A classify the training sample as 0, Classifier B as 1, Classifier C as 1. Then based on the majority vote, the model would classify the sample as “1”.

In a nutshell, we hope that the model chosen would provide certain guidance or information for the users to know what characteristic or personalities they should be prepared and obtained in order to match with their partners.

Appendix A: List of Attributes Description

Attributes	Description
has_null	whether the sample consists of null values
wave	meaning unknown
gender	gender of self
age	age of self
age_o	age of partner
d_age	difference in age
d_d_age	preprocessed "d_age"
race	Race of self
race_o	Race of partner
samerace	Whether the two persons have same race or not
importance_same_race	How importance is it that the partner is same race?
importance_same_religion	How importance is it that partner has same religion?
d_importance_same_race	preprocessed "importance_same_race"
d_importance_same_religion	preprocessed "importance_same_religion"
field	field of study
pref_o_attractive	How importance does partner rate attractiveness?
pref_o_sincere	How importance does partner rate sincerity?
pref_o_intelligence	How importance does partner rate intelligence?
pref_o_funny	How importance does partner rate being funny?
pref_o_ambitious	How importance does partner rate ambition?
pref_o_shared_interests	How importance dos partner rate having same interests?
d_pref_o_attractive	preprocessed "pref_o_attractive"
d_pref_o_sincere	preprocessed "pref_o_sincere"
d_pref_o_intelligence	preprocessed "pref_o_intelligence"
d_pref_o_funny	preprocessed "pref_o_funny"
d_pref_o_ambitious	preprocessed "pref_o_ambitious"
d_pref_o_shared_interests	preprocessed "pref_o_shared_interests"
attractive_o	Rating by partner (about me) at night of event on attractiveness
sincere_o	Rating by partner (about me) at night of event on sincerity
intelligence_o	Rating by partner (about me) at night of event on intelligence

funny_o	Rating by partner (about me) at night of event on being funny
ambitous_o	Rating by partner (about me) at night of event on being ambitious
shared_interests_o	Rating by partner (about me) at night of event on shared interests
d_attractive_o	preprocessed “attractive_o”
d_sinsere_o	preprocessed “sinsere_o”
d_intelligence_o	preprocessed “intelligence_o”
d_funny_o	preprocessed “funny_o”
d_ambitous_o	preprocessed “ambitous_o”
d_shared_interests_o	preprocessed “shared_interests_o”
attractive_important	What do you look for in a partner – attractiveness
sincere_important	What do you look for in a partner – sincerity
intellicence_important	What do you look for in a partner – intelligence
funny_important	What do you look for in a partner – being funny
ambtition_important	What do you look for in a partner – ambition
shared_interests_important	What do you look for in a partner – shared interests
d_attractive_important	preprocessed “attractive_important”
d_sincere_important	preprocessed “sincere_important”
d_intellicence_important	preprocessed “intellicence_important”
d_funny_important	preprocessed “funny_important”
d_ambtition_important	preprocessed “ambtition_important”
d_shared_interests_important	preprocessed “shared_interests_important”
attractive	Rate yourself – attractiveness
sincere	Rate yourself – sincerity
intelligence	Rate yourself – intelligence
funny	Rate yourself – being funny
ambition	Rate yourself – ambition
d_attractive	preprocessed “attractive”
d_sincere	preprocessed “sincere”
d_intelligence	preprocessed “intelligence”
d_funny	preprocessed “funny”
d_ambition	preprocessed “ambition”
attractive_partner	Rate your partner – attractiveness
sincere_partner	Rate your partner – sincerity
intelligence_partner	Rate your partner – intelligence
funny_partner	Rate your partner – being funny

ambition_partner	Rate your partner – ambition
shared_interests_partner	Rate your partner – shared interests
d_attractive_partner	preprocessed “attractive_partner”
d_sincere_partner	preprocessed “sincere_partner”
d_intelligence_partner	preprocessed “intelligence_partner”
d_funny_partner	preprocessed “funny_partner”
d_ambition_partner	preprocessed “ambition_partner”
d_shared_interests_partner	preprocessed “shared_interests_partner”
sports	Your own interests [1-10]
tvsports	Your own interests [1-10]
exercise	Your own interests [1-10]
dining	Your own interests [1-10]
museums	Your own interests [1-10]
art	Your own interests [1-10]
hiking	Your own interests [1-10]
gaming	Your own interests [1-10]
clubbing	Your own interests [1-10]
reading	Your own interests [1-10]
tv	Your own interests [1-10]
theater	Your own interests [1-10]
movies	Your own interests [1-10]
concerts	Your own interests [1-10]
music	Your own interests [1-10]
shopping	Your own interests [1-10]
yoga	Your own interests [1-10]
d_sports	preprocessed “sports”
d_tvsports	preprocessed “tvsports”
d_exercise	preprocessed “exercise”
d_dining	preprocessed “dining”
d_museums	preprocessed “museums”
d_art	preprocessed “art”
d_hiking	preprocessed “hiking”
d_gaming	preprocessed “gaming”
d_clubbing	preprocessed “clubbing”
d_reading	preprocessed “reading”
d_tv	preprocessed “tv”
d_theater	preprocessed “theater”
d_movies	preprocessed “movies”
d_concerts	preprocessed “concerts”
d_music	preprocessed “music"
d_shopping	preprocessed “shopping”
d_yoga	preprocessed “yoga”
interests_correlate	Correlation between participant’s and partner’s ratings of interests
d_interests_correlate	preprocessed “interests_correlate”

expected_happy_with_sd_people	How happy do you expect to be with the people you meet during the speed dating event?
expected_num_interested_in_me	Out of the 20 people you will meet, how many do you expect will be interested in dating you?
expected_num_matches	How many matches do you expect to get?
d_expected_happy_with_sd_people	preprocessed "expected_happy_with_sd_people"
d_expected_num_interested_in_me	preprocessed "expected_num_interested_in_me"
d_expected_num_matches	preprocessed "expected_num_matches"
like	Did you like your partner?
guess_prob_liked	How likely do you think it is that your partner likes you?
d_like	preprocessed "like"
d_guess_prob_liked	preprocessed "guess_prob_liked"
met	Have you met your partner before?
decision	Decision at night of event
decision_o	Decision of partner at night of event
match	Match (yes/no)

Appendix B: List of Features Dropped

Categories	Feature Name
Preprocessed features	d_age, d_d_age, d_importance_same_race, d_importance_same_religion, d_pref_o_attractive, d_pref_o_sincere, d_pref_o_intelligence, d_pref_o_funny, d_pref_o_ambitious, d_pref_o_shared_interests, d_attractive_o, d_sinsere_o, d_intelligence_o, d_funny_o, d_ambitous_o, d_shared_interests_o, d_attractive_important, d_sincere_important, d_intellicence_important, d_funny_important, d_ambtition_important, d_shared_interests_important, d_attractive, d_sincere, d_intelligence, d_funny, d_ambition, d_attractive_partner, d_sincere_partner, d_intelligence_partner, d_funny_partner, d_ambition_partner, d_shared_interests_partner, d_sports, d_tvsports, d_exercise, d_dining, d_museums, d_art, d_hiking, d_gaming, d_clubbing, d_reading, d_tv,

	d_theater, d_movies, d_concerts, d_music, d_shopping, d_yoga, d_interests_correlate, d_expected_happy_with_sd_people, d_expected_num_interested_in_me, d_expected_num_matches, d_like, d_guess_prob_liked
Self's Interest features	sports, tvsports, exercise, dining, museums, art, hiking, gaming, clubbing, reading, tv, theater, movies, concerts 'music, shopping, yoga
Partner's features	age_o, race_o, pref_o_attractive, pref_o_sincere, pref_o_intelligence, pref_o_funny, pref_o_ambitious, pref_o_shared_interests, attractive_o, sinsere_o, intelligence_o, funny_o, ambitous_o, shared_interests_o
Other Irrelevant features	has_null, wave, expected_happy_with_sd_people, expected_num_interested_in_me, expected_num_matches, field, decision