Georgia State University

CSC4780 – Fundamentals of Data Science

Fall 2022

Final Project Report

Thanks, Cupid

Cupid Scientists
Cindy Thai
Lorena Burrell
Capella Edwards
Venkata Mani Mohana Rishitha Srikakulapu
Laurel Sparks

Table of Contents

1 Business Understanding

1.1 Business Problem

The purpose of dating applications is to help users find potential romantic partners to connect with. Users are connected based on the nearest location with limited filtering functionality to return a portion of neighboring users. Our app will provide premium services to allow the user to generate a more refined list of matches that have been filtered based on the preferences of both users. To sell these premium services, our model must provide useful results, meaning dates that are likely to retain interest in one another and stay together. Therefore, our app will benefit from predicting which people are most likely to match with each other. It can also stand to profit from selecting which information about its users is most critical to keep to be successful, saving data space and retaining user attention by avoiding a long 'interview quiz' portion. This will yield more profits for the business to sell more premium services, and a high success rate will draw more users to join the app which will increase the variety of dating partners for paying customers.

1.2 Dataset

The dataset that we selected recorded the results of a speed dating experiment conducted by Columbia Business School by research professors Raymond Fisman, Sheena S. Iyengar, Emir Kamenica, and Itamar Simonson during May 2006. We selected this particular dataset because it has a wide range of descriptive features with continuous and categorical aspects with over eight thousand instances. This dataset will be useful in making predictions and training our model since we have plenty of data to use as a training and testing set and allow our machine learning algorithm to predict matches between participants. The original speed dating dataset has approximately one-hundred ninety-five features and eight thousand three-hundred and seventy-nine instances. We decided to narrow down the features to a list of nineteen, many of which are derived combinations of the original raw features, for the most impactful and relevant data for selecting matches.

The features we decided to select include the target variable of whether a match was made, expected satisfaction with dating, whether both participants match race/ethnicity, average importance of matching race to both participants, and whether both participants stated the same desired outcome from dating. We also derived the difference between the participants' interest in physical hobbies, other outdoor hobbies, and indoor hobbies, along with the difference between the participants' self-rating vs. the other's desire for qualities of ambition, attractiveness, sincerity, intelligence, funniness, and ambition. Difference in income and age, along with whether both have a desired career in the same area, were also included. We marked the percentage of prospective dates a person expected to be interested in them as "confidence." The last two features indicate a ranking of how often the participants go on dates and go out in general. Based on the sample scorecard and synopsis of the data, we determined that these features were the most influential in drawing conclusions on matches. We believe that these features will help us narrow down our target feature and train our model to make better predictions on matches.

1.3 Proposed Analytics Solution

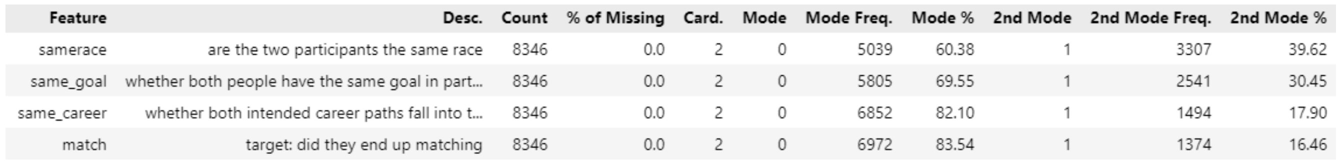
With our subset dataset, we plan to use a supervised similarity-based machine learning algorithm to predict successful matches. Our first step in accomplishing this task was to create a subset of key descriptive features from our larger dataset. We choose nineteen descriptive features, including features like happiness expectation, hobby/interest value, goal difference, and more (see table in the appendix for the complete table of key descriptive features and descriptions). Our target variable is our match feature which predicts the outcome as 0 for no match and 1 for a match between two individuals participating in speed-dating.

2 Data Exploration and Preprocessing

2.1 Data Quality Report

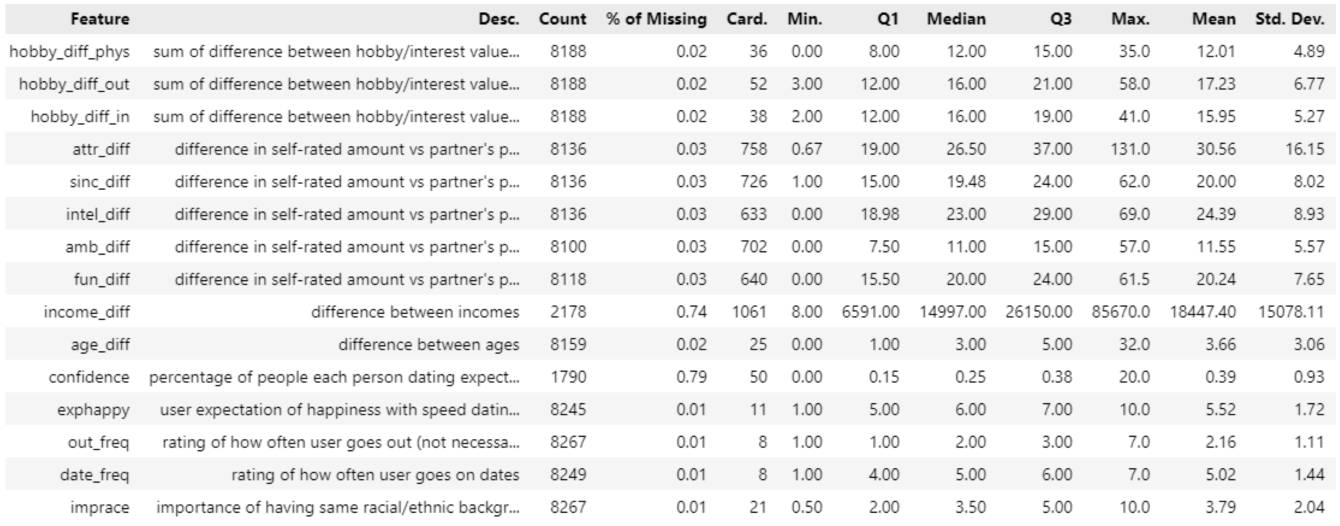
Below, we have listed the count, cardinality, and percentage of missing values for all features. For categorical features, we calculated the first and second mode, mode frequency, and mode percentage. For continuous features, we calculated the minimum, maximum, quartiles, and standard deviation.

Table 1. Data Quality Report for Categorical Features



Feature	Desc.	Count	% of Missing	Card.	Mode	Mode Freq.	Mode %	2nd Mode	2nd Mode Freq.	2nd Mode %
samerace	are the two participants the same race	8346	0.0	2	0	5039	60.38	1	3307	39.62
same_goal	whether both people have the same goal in part...	8346	0.0	2	0	5805	69.55	1	2541	30.45
same_career	whether both intended career paths fall into t...	8346	0.0	2	0	6852	82.10	1	1494	17.90
match	target: did they end up matching	8346	0.0	2	0	6972	83.54	1	1374	16.46

Table 2. Data Quality Report for Continuous Features

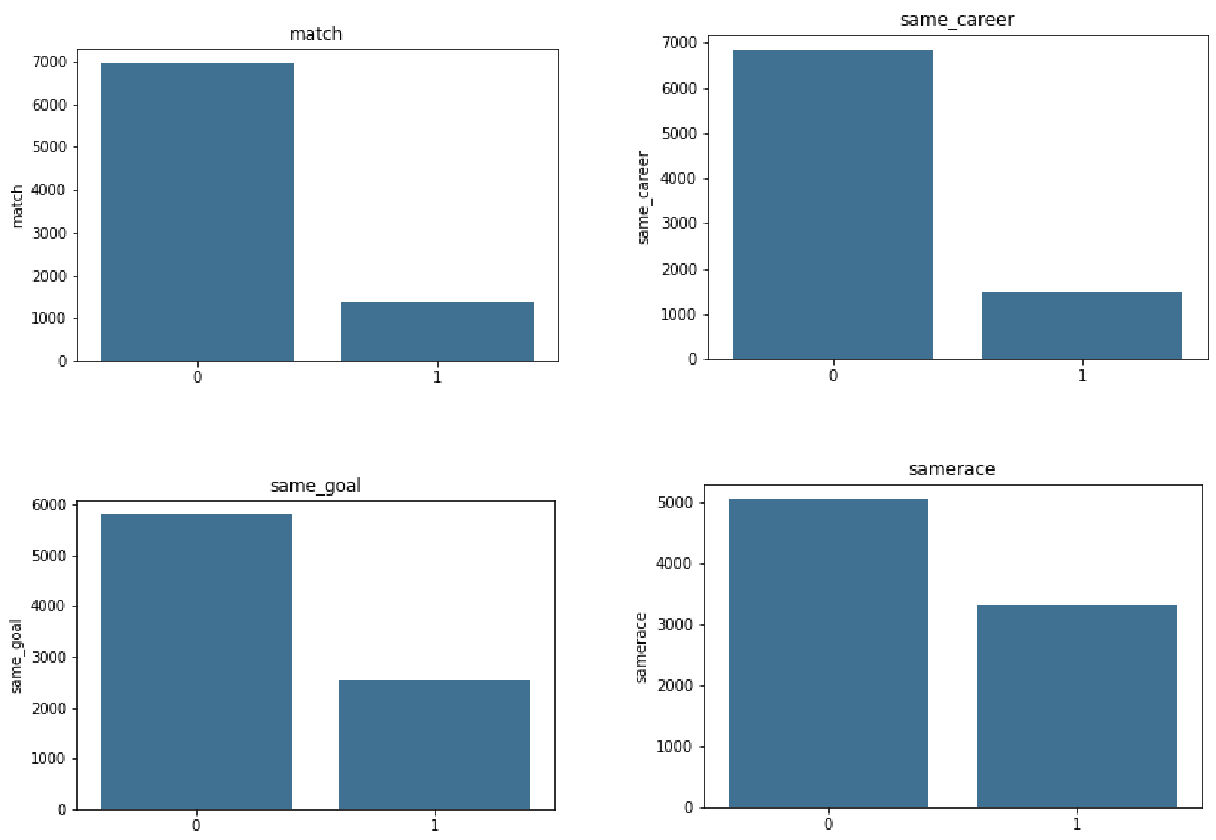


Feature	Desc.	Count	% of Missing	Card.	Min.	Q1	Median	Q3	Max.	Mean	Std. Dev.
hobby_diff_phys	sum of difference between hobby/interest value...	8188	0.02	36	0.00	8.00	12.00	15.00	35.0	12.01	4.89
hobby_diff_out	sum of difference between hobby/interest value...	8188	0.02	52	3.00	12.00	16.00	21.00	58.0	17.23	6.77
hobby_diff_in	sum of difference between hobby/interest value...	8188	0.02	38	2.00	12.00	16.00	19.00	41.0	15.95	5.27
attr_diff	difference in self-rated amount vs partner's p...	8136	0.03	758	0.67	19.00	26.50	37.00	131.0	30.56	16.15
sinc_diff	difference in self-rated amount vs partner's p...	8136	0.03	726	1.00	15.00	19.48	24.00	62.0	20.00	8.02
intel_diff	difference in self-rated amount vs partner's p...	8136	0.03	633	0.00	18.98	23.00	29.00	69.0	24.39	8.93
amb_diff	difference in self-rated amount vs partner's p...	8100	0.03	702	0.00	7.50	11.00	15.00	57.0	11.55	5.57
fun_diff	difference in self-rated amount vs partner's p...	8118	0.03	640	0.00	15.50	20.00	24.00	61.5	20.24	7.65
income_diff	difference between incomes	2178	0.74	1061	8.00	6591.00	14997.00	26150.00	85670.0	18447.40	15078.11
age_diff	difference between ages	8159	0.02	25	0.00	1.00	3.00	5.00	32.0	3.66	3.06
confidence	percentage of people each person dating expect...	1790	0.79	50	0.00	0.15	0.25	0.38	20.0	0.39	0.93
exphappy	user expectation of happiness with speed datin...	8245	0.01	11	1.00	5.00	6.00	7.00	10.0	5.52	1.72
out_freq	rating of how often user goes out (not necessa...	8267	0.01	8	1.00	1.00	2.00	3.00	7.0	2.16	1.11
date_freq	rating of how often user goes on dates	8249	0.01	8	1.00	4.00	5.00	6.00	7.0	5.02	1.44
imprace	importance of having same racial/ethnic backgr...	8267	0.01	21	0.50	2.00	3.50	5.00	10.0	3.79	2.04

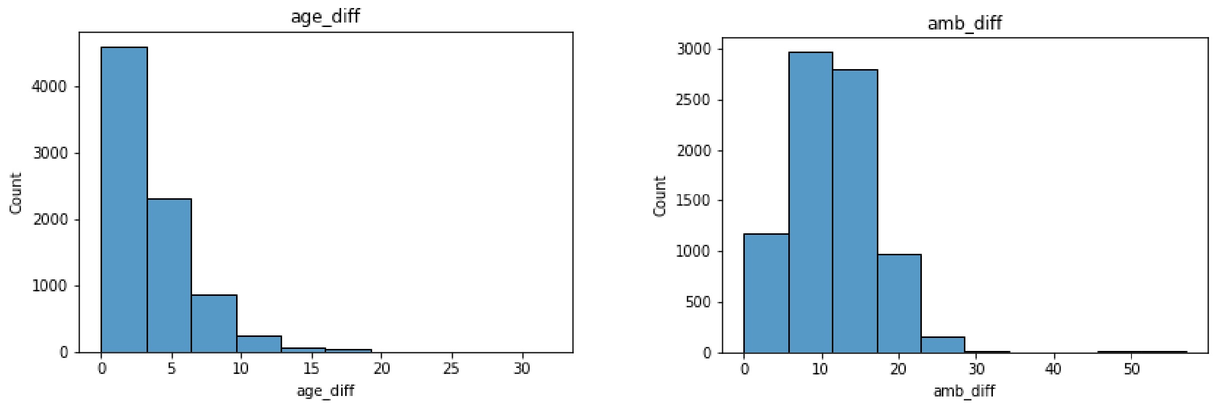
All of our categorical features are binary, and most show a heavy preference for one or the other. For this reason, we may end up employing under- or oversampling when training our model, particularly for the target feature. For continuous features, most display a healthily skewed unimodal distribution. Income difference and confidence are both exponential, and with how great the standard deviation of the income values are, it is important to handle outliers well.

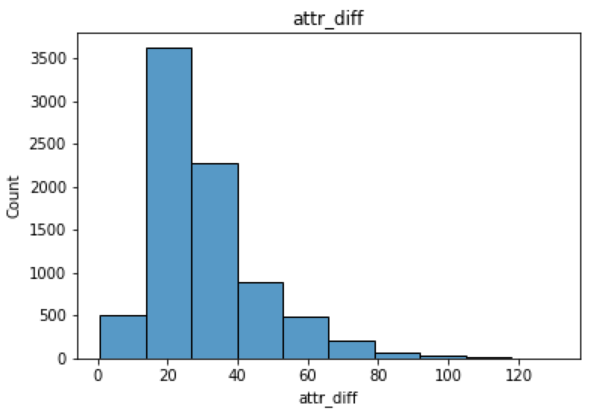
Figure 1. Visualizations of Categorical and Continuous Features in Dataset

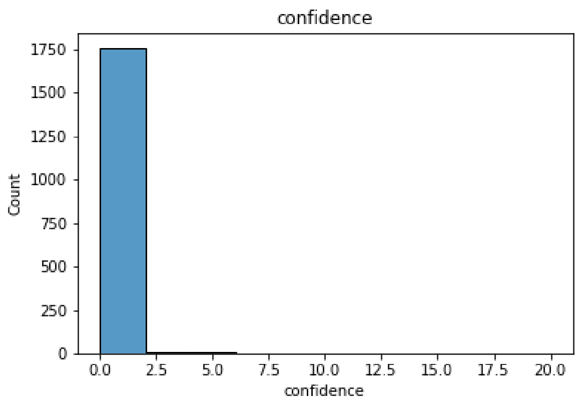
Figure 1.1 - Bar plots for categorical features:

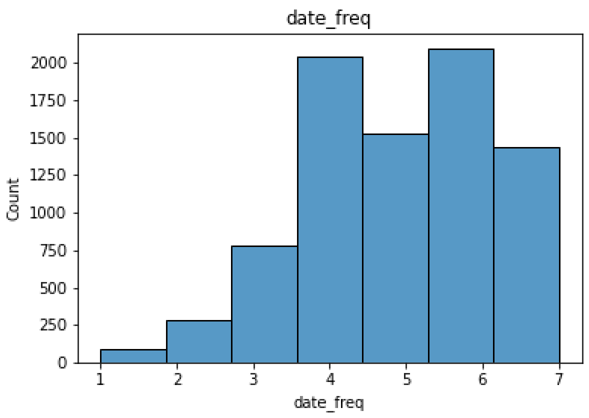


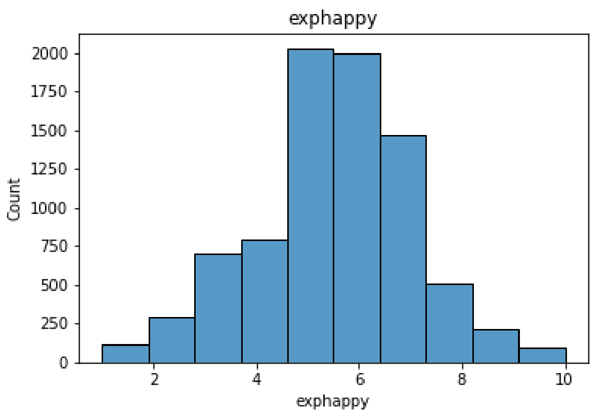
Figures 1.2 - Histograms for continuous features:

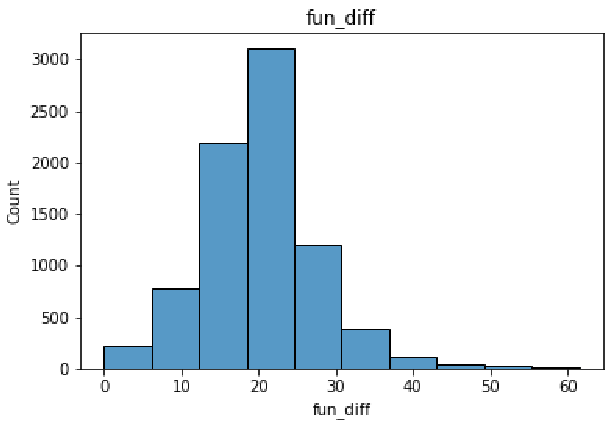


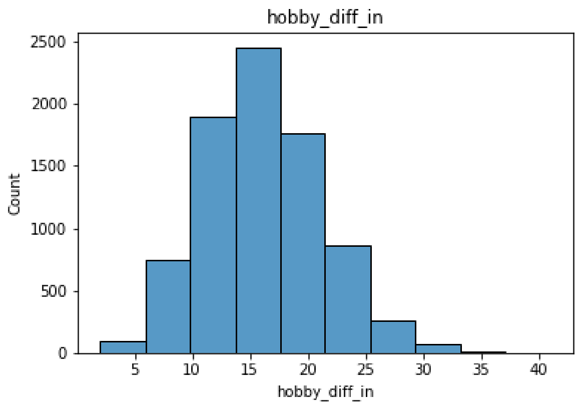

attr_diff
Count
attr_diff

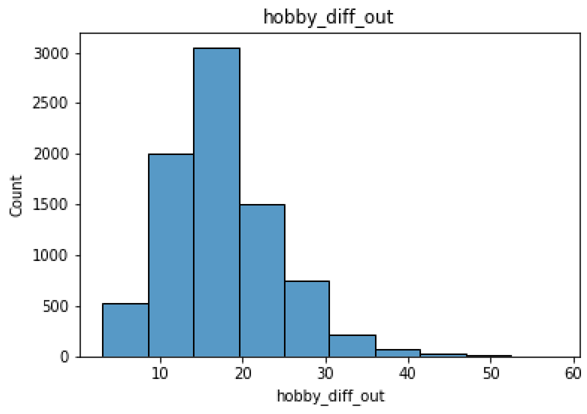

confidence
Count
confidence

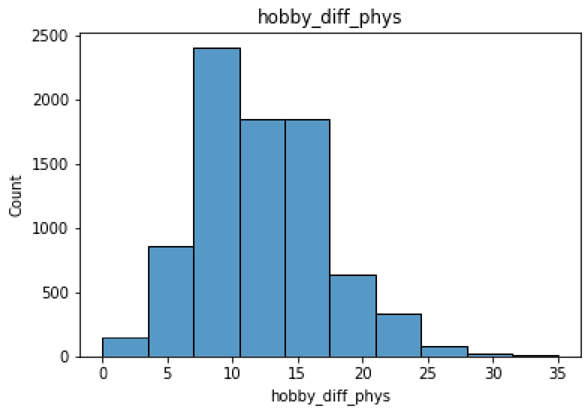

date_freq
Count
date_freq

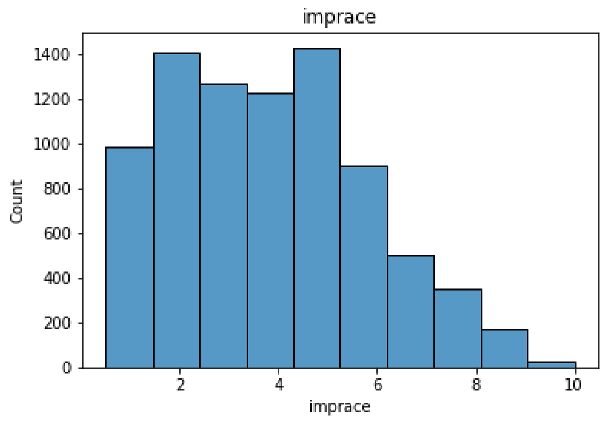

exphappy
Count
exphappy

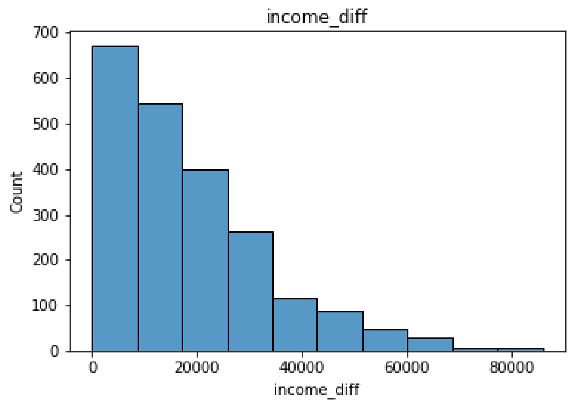

fun_diff
Count
fun_diff

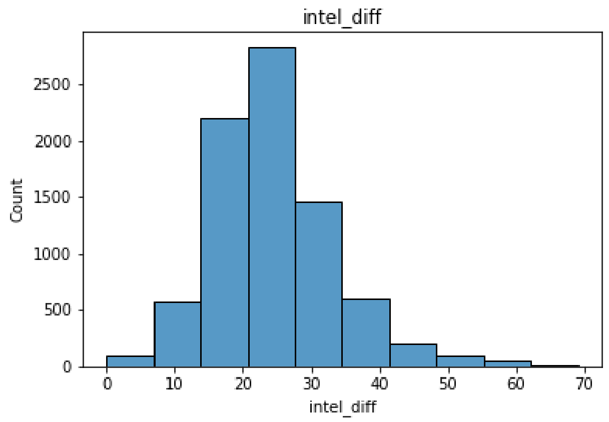

hobby_diff_in
Count
hobby_diff_in

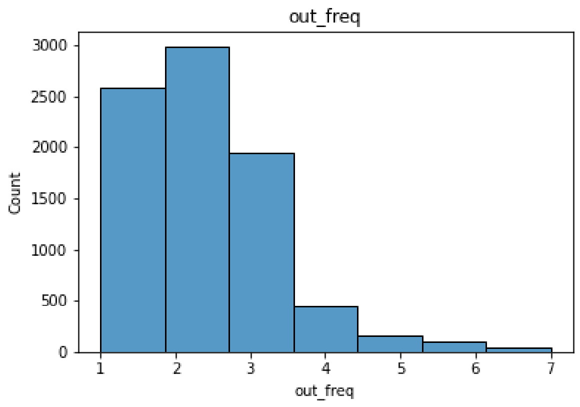

hobby_diff_out
Count
hobby_diff_out

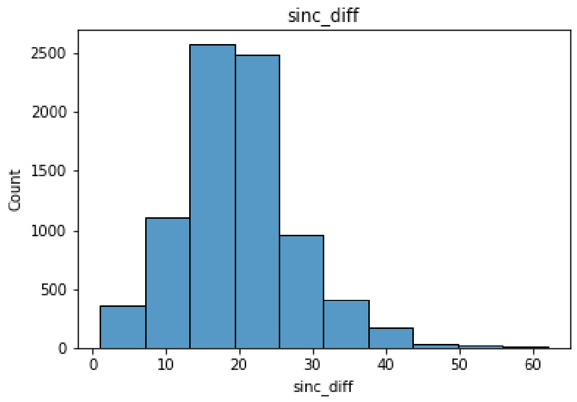

hobby_diff_phys
Count
hobby_diff_phys


imprace
Count
imprace


income_diff
Count
income_diff

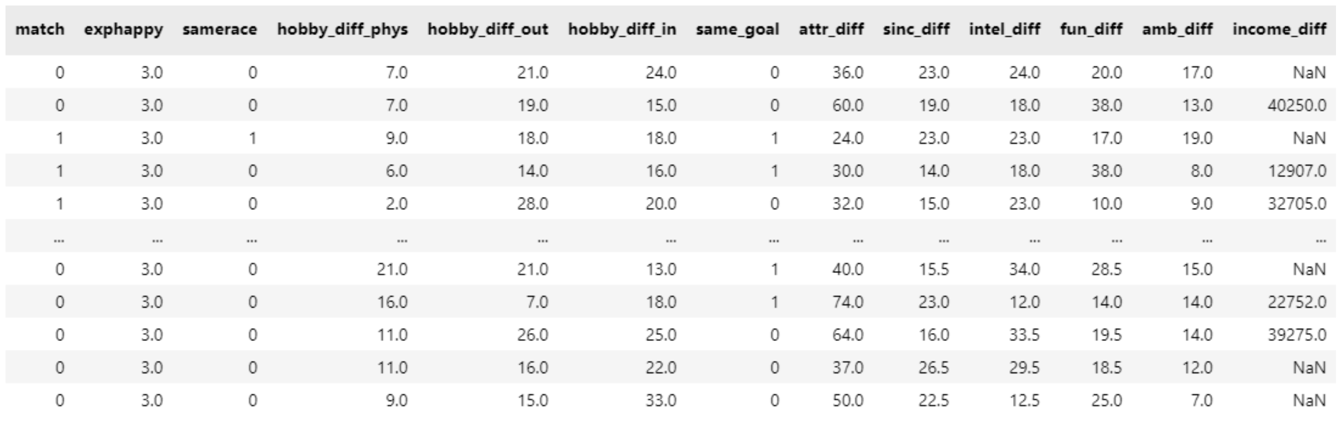

intel_diff
Count
intel_diff


out_freq
Count
out_freq



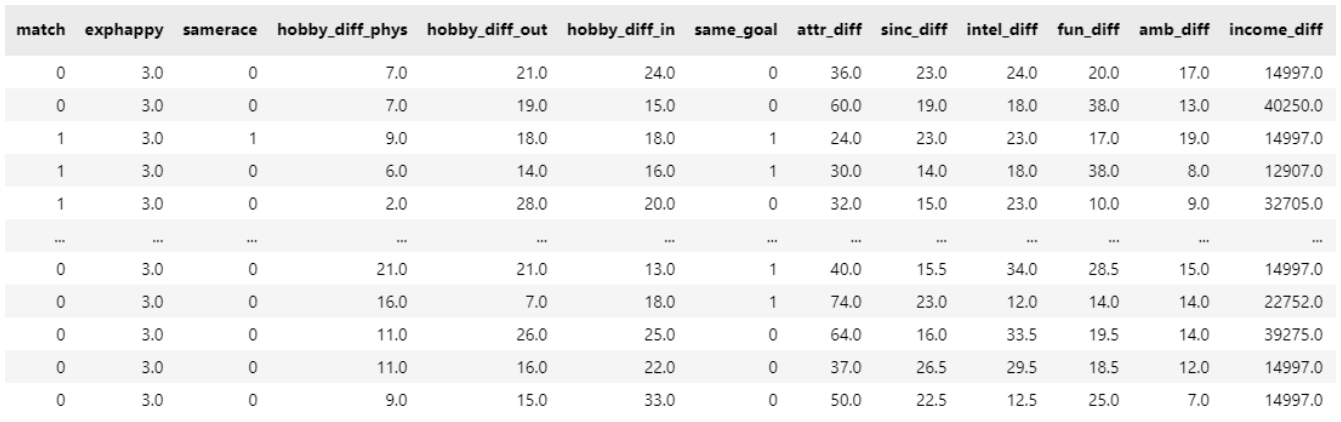
2.2 Missing Values and Outliers

Before we begin the modeling process, we must clean the data by dealing with missing values and outliers.



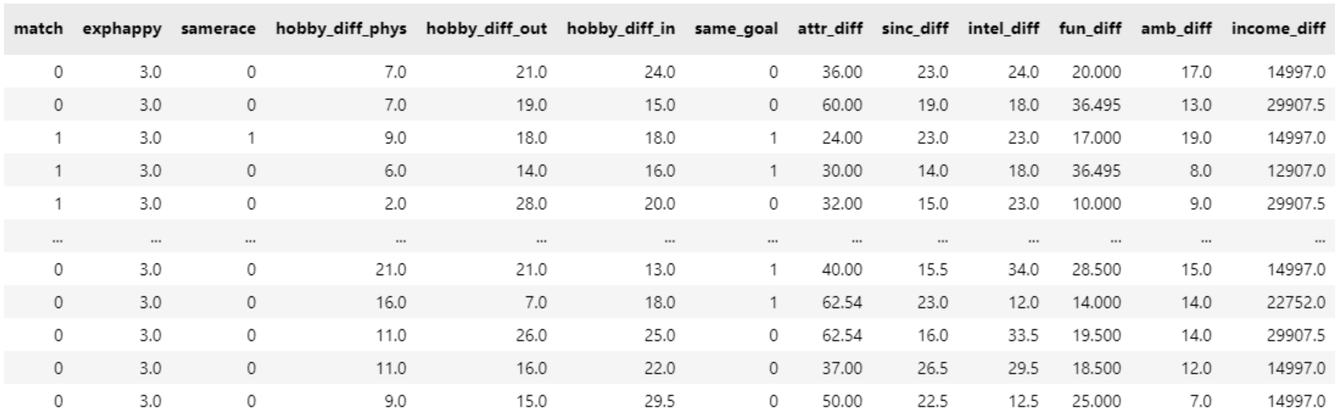
match	exphappy	samerace	hobby_diff_phys	hobby_diff_out	hobby_diff_in	same_goal	attr_diff	sinc_diff	intel_diff	fun_diff	amb_diff	income_diff
0	3.0	0	7.0	21.0	24.0	0	36.0	23.0	24.0	20.0	17.0	NaN
0	3.0	0	7.0	19.0	15.0	0	60.0	19.0	18.0	38.0	13.0	40250.0
1	3.0	1	9.0	18.0	18.0	1	24.0	23.0	23.0	17.0	19.0	NaN
1	3.0	0	6.0	14.0	16.0	1	30.0	14.0	18.0	38.0	8.0	12907.0
1	3.0	0	2.0	28.0	20.0	0	32.0	15.0	23.0	10.0	9.0	32705.0
...	...	...	...	...	...	...	...	...	...	...	...	...
0	3.0	0	21.0	21.0	13.0	1	40.0	15.5	34.0	28.5	15.0	NaN
0	3.0	0	16.0	7.0	18.0	1	74.0	23.0	12.0	14.0	14.0	22752.0
0	3.0	0	11.0	26.0	25.0	0	64.0	16.0	33.5	19.5	14.0	39275.0
0	3.0	0	11.0	16.0	22.0	0	37.0	26.5	29.5	18.5	12.0	NaN
0	3.0	0	9.0	15.0	33.0	0	50.0	22.5	12.5	25.0	7.0	NaN

First, we used imputation to fill in missing values in each continuous column in the dataset. None of our categorical features had any missing values, thankfully. Mean imputation was used for all features except income_diff, which had such a high number of missing values that replacing them all equally meant destroying the original distribution. Instead, we used KNN imputation to preserve a more accurate distribution.



match	exphappy	samerace	hobby_diff_phys	hobby_diff_out	hobby_diff_in	same_goal	attr_diff	sinc_diff	intel_diff	fun_diff	amb_diff	income_diff
0	3.0	0	7.0	21.0	24.0	0	36.0	23.0	24.0	20.0	17.0	14997.0
0	3.0	0	7.0	19.0	15.0	0	60.0	19.0	18.0	38.0	13.0	40250.0
1	3.0	1	9.0	18.0	18.0	1	24.0	23.0	23.0	17.0	19.0	14997.0
1	3.0	0	6.0	14.0	16.0	1	30.0	14.0	18.0	38.0	8.0	12907.0
1	3.0	0	2.0	28.0	20.0	0	32.0	15.0	23.0	10.0	9.0	32705.0
...	...	...	...	...	...	...	...	...	...	...	...	...
0	3.0	0	21.0	21.0	13.0	1	40.0	15.5	34.0	28.5	15.0	14997.0
0	3.0	0	16.0	7.0	18.0	1	74.0	23.0	12.0	14.0	14.0	22752.0
0	3.0	0	11.0	26.0	25.0	0	64.0	16.0	33.5	19.5	14.0	39275.0
0	3.0	0	11.0	16.0	22.0	0	37.0	26.5	29.5	18.5	12.0	14997.0
0	3.0	0	9.0	15.0	33.0	0	50.0	22.5	12.5	25.0	7.0	14997.0

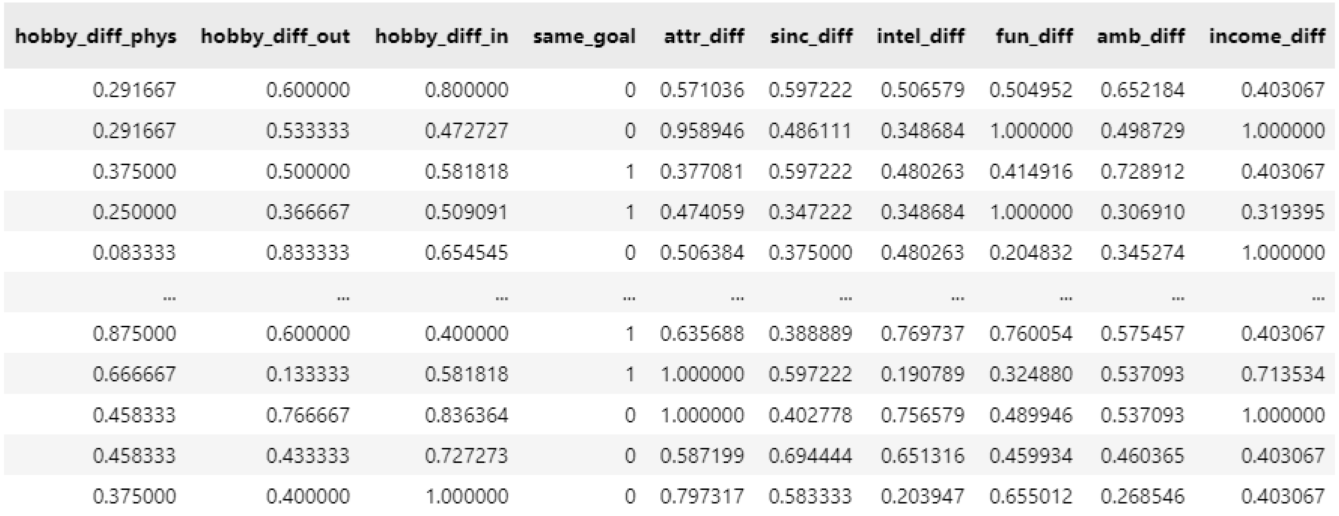
Next, we used Tukey’s Range Test or the Interquartile-Range method to create a function to clamp all outliers in the dataset. We chose not to drop all columns containing outliers, as it would remove around half the rows in the dataset. We also chose not to clamp certain features that were scales guaranteed to be in the range 1-7 or 1-10, such as imprace, out_freq, date_freq, and exphappy. We also did not clamp the confidence feature, as the high number of imputed values made the regular lower and upper bounds have microscopic difference. The same issue happened with income, but due to the high outliers, we chose to clamp this feature with percentiles 0.5 to 0.95 instead of the IQR method.



match	exphappy	samerace	hobby_diff_phys	hobby_diff_out	hobby_diff_in	same_goal	attr_diff	sinc_diff	intel_diff	fun_diff	amb_diff	income_diff
0	3.0	0	7.0	21.0	24.0	0	36.00	23.0	24.0	20.000	17.0	14997.0
0	3.0	0	7.0	19.0	15.0	0	60.00	19.0	18.0	36.495	13.0	29907.5
1	3.0	1	9.0	18.0	18.0	1	24.00	23.0	23.0	17.000	19.0	14997.0
1	3.0	0	6.0	14.0	16.0	1	30.00	14.0	18.0	36.495	8.0	12907.0
1	3.0	0	2.0	28.0	20.0	0	32.00	15.0	23.0	10.000	9.0	29907.5
...	...	...	...	...	...	...	...	...	...	...	...	...
0	3.0	0	21.0	21.0	13.0	1	40.00	15.5	34.0	28.500	15.0	14997.0
0	3.0	0	16.0	7.0	18.0	1	62.54	23.0	12.0	14.000	14.0	22752.0
0	3.0	0	11.0	26.0	25.0	0	62.54	16.0	33.5	19.500	14.0	29907.5
0	3.0	0	11.0	16.0	22.0	0	37.00	26.5	29.5	18.500	12.0	14997.0
0	3.0	0	9.0	15.0	29.5	0	50.00	22.5	12.5	25.000	7.0	14997.0

2.3 Normalization

We wanted to avoid bias and inequality between features, especially those that sum up differences between different numbers of features, such as the hobby groups, or those with particularly large values, such as income_diff. For this reason, we used range normalization to give equal footing to all continuous features. Notably, this did not affect the confidence rating, as that was already a percentage rating with some values already at 0 and 1.



hobby_diff_phys	hobby_diff_out	hobby_diff_in	same_goal	attr_diff	sinc_diff	intel_diff	fun_diff	amb_diff	income_diff
0.291667	0.600000	0.800000	0	0.571036	0.597222	0.506579	0.504952	0.652184	0.403067
0.291667	0.533333	0.472727	0	0.958946	0.486111	0.348684	1.000000	0.498729	1.000000
0.375000	0.500000	0.581818	1	0.377081	0.597222	0.480263	0.414916	0.728912	0.403067
0.250000	0.366667	0.509091	1	0.474059	0.347222	0.348684	1.000000	0.306910	0.319395
0.083333	0.833333	0.654545	0	0.506384	0.375000	0.480263	0.204832	0.345274	1.000000
...	...	...	...	...	...	...	...	...	...
0.875000	0.600000	0.400000	1	0.635688	0.388889	0.769737	0.760054	0.575457	0.403067
0.666667	0.133333	0.581818	1	1.000000	0.597222	0.190789	0.324880	0.537093	0.713534
0.458333	0.766667	0.836364	0	1.000000	0.402778	0.756579	0.489946	0.537093	1.000000
0.458333	0.433333	0.727273	0	0.587199	0.694444	0.651316	0.459934	0.460365	0.403067
0.375000	0.400000	1.000000	0	0.797317	0.583333	0.203947	0.655012	0.268546	0.403067

2.4 Transformations

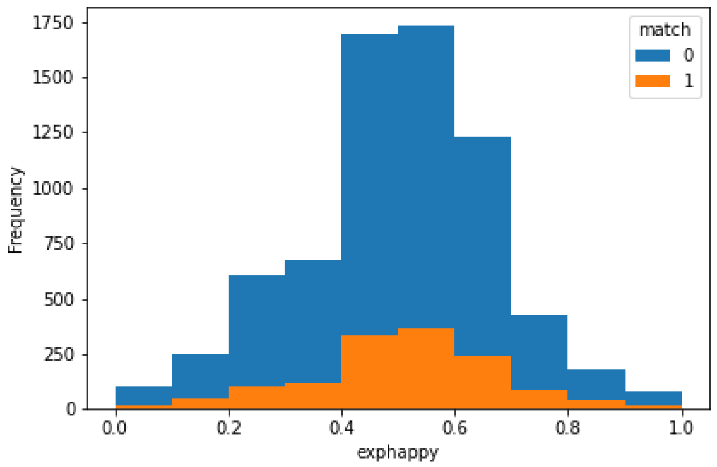
In the beginning, we had 195 features in our original dataset which were transformed using aggregations and derivations to create a dataset with lesser complexity. Now, we're using that data to identify how well the prediction of finding a match is done.

We collected metrics about the distribution of our data so that we can better understand common traits of our users and determine the importance of matching a potential couple that share these traits to improve the efficiency of our prediction. First we separate out of continuous variables and categorical variables, so that we can perform binning on the subsets. We then performed our normalization method MinMaxScaling to which has the benefit of preserving the original relationship between our original distribution. We will then generate a histogram to visualize the distributions of our data for further analysis.

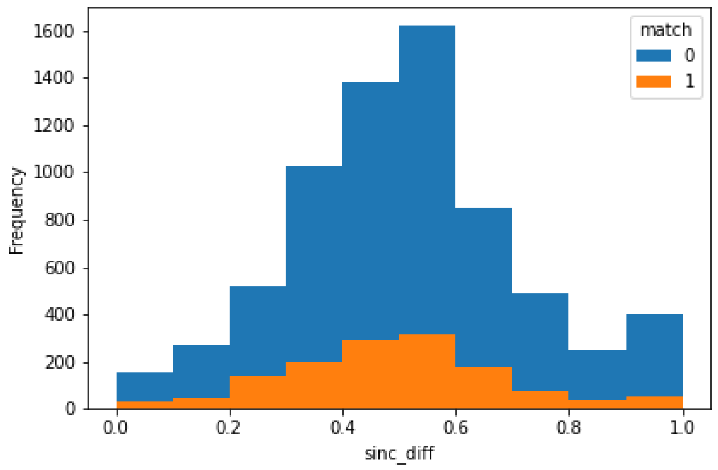
After we normalize the data we determine that intel_diff has a right-skewed unimodal distribution, in which we observe that there is a majority of potential matches with a higher correlation (lower difference) between one's self-rated intelligence and the other's desire for intelligence in a partner. The descriptive feature age_diff remained roughly the same in an exponential fashion, which indicates a smaller age gap between potential matches.

The goal of our transformations was to improve the distribution of the continuous variables and create a more uniformly distributed model. Through removing values that were rare and considered outliers, and normalizing the data we were able to achieve these goals and make some preliminary observations.

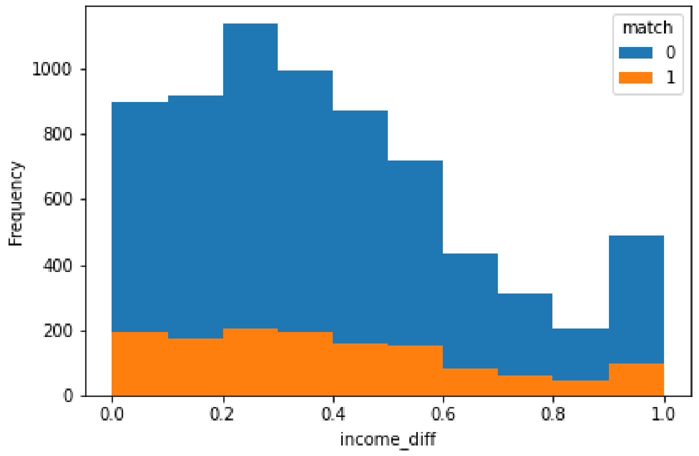
Below, we have visualized the post-transformation distributions in comparison to the target 'match' feature.



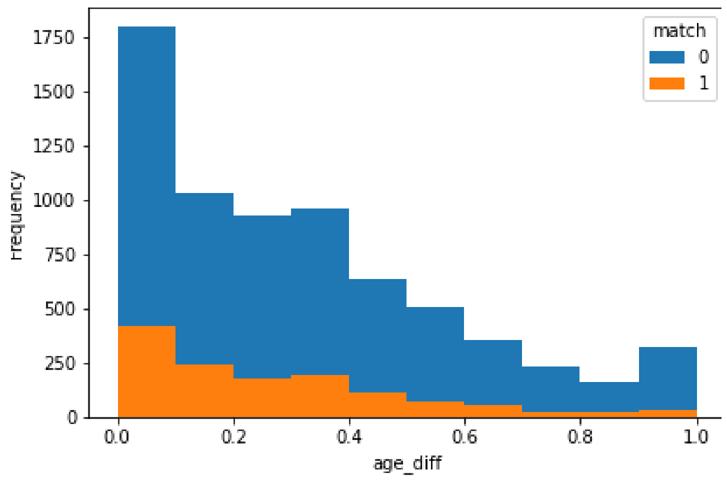
We see a symmetry or a normally distributed model for the exphappy feature. Going in with positive expectations helps. The probability of finding a match also follows the same normal distribution as the distribution of a no match.



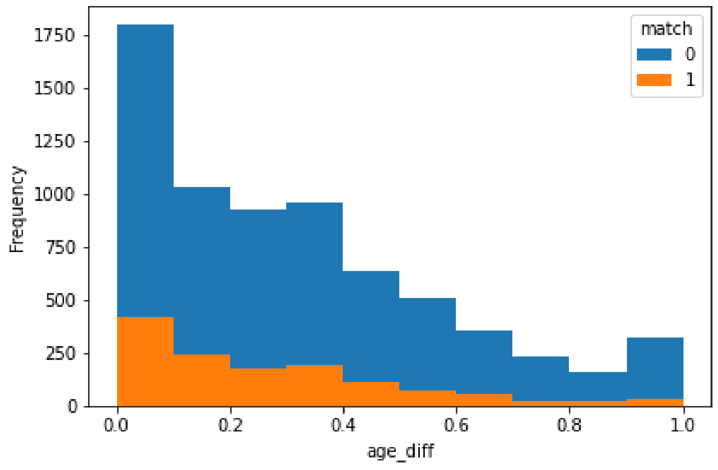
We see a similar normal distribution as exphappy for the sinc_diff feature. However, this seems to be a slightly right-skewed distribution. The corresponding match probability seems to distribute similarly.



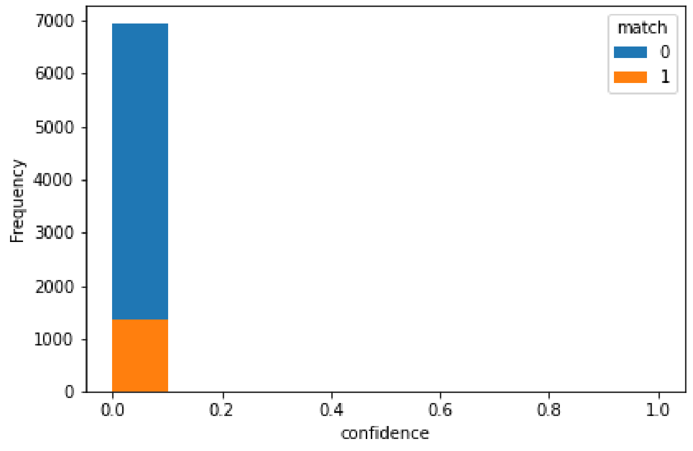
We see that the income difference of the people trying to connect tends towards being lower, though this does not appear to have an enormous impact on match likelihood. If anything, a much higher difference between incomes has a slightly higher percentage of matching.



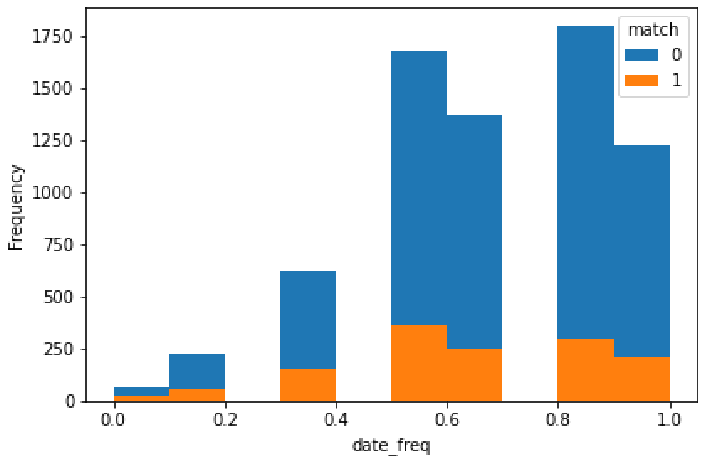
We find that age_diff follows an exponential distribution. The corresponding distribution of finding a match is similar. We can observe that the majority of couples in this experiment have a smaller age difference



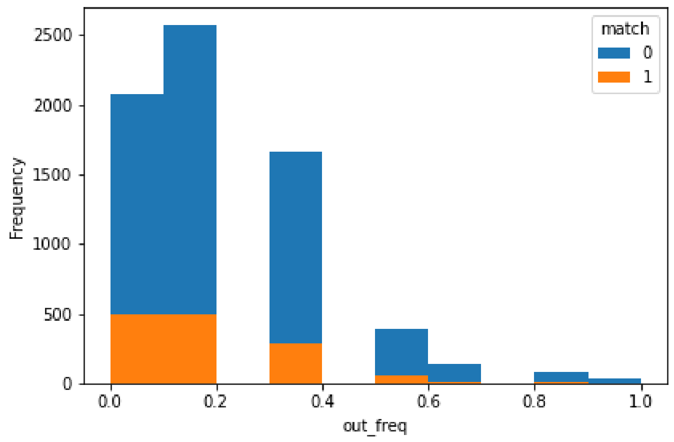
For attr_diff, it can be observed that couples skewed towards high correlation between self-judged attractiveness and the other’s value in attractiveness, although this does not appear to heavily impact the match percentage.



We can see that the confidence, percentage of the dates to show interest as expected by the person, is very low most of the time. It may be difficult to glean meaningful statistics from this feature because of how strongly it is skewed, and we may need to deal with a potential vanishing gradient.



The ‘_freq’ features are rankings, where 1 is the greatest and 7 is the lowest frequency. When we visualize getting a match vs. date_freq, we see that people who go on dates more frequently are not necessarily more likely to get a match.



When we look at out_freq, we see that many people go out on non-date outings frequently. People who go out more often seem to have better luck getting matches.

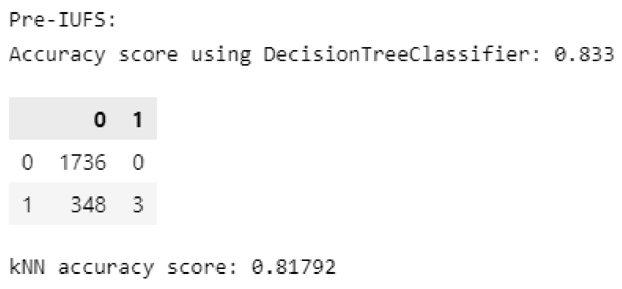
2.5 Feature Selection

To begin, we performed two feature selection methods: impurity-based univariate feature selection (IUFS), utilizing entropy, gini index, information gain, and information gain ratio. and recursive feature elimination (RFE) using a logistic regression model. We pursued IUFS first since every feature in our dataset is considered separately to one another and comparing the results before and after could give us insight to which features would impact our models' performance the greatest and which features were not crucial to our evaluation.

Based on the results from calculating our impurity, gain metrics and RFE, we decided that the features that would have the least amount of impact on our model's performance would be 'same_goal' and 'amb_diff'. 'Same_goal' refers to both participants having the same goal in entering the date, and 'amb_diff' refers to whether both participants rated themselves as similarly ambitious people.

Once IUFS had been conducted, we removed the 'same_goal' feature from our dataset which left us with seventeen descriptive features. Afterwards, we performed recursive feature elimination which utilizes logistic regression to rank our remaining seventeen features based on their KNN accuracy score and we decided to remove the 'amb_diff' feature as well to select the descriptive features we will use for model evaluation. As a result, we found that sixteen descriptive features had the highest accuracy scores for KNN classifiers. After using KNN to evaluate accuracy, we selected these sixteen features and discarded the remaining features to perform model selection and evaluation.

Pre-IUFS and RFE:

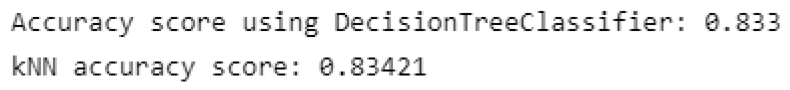


```
Pre-IUFS:
Accuracy score using DecisionTreeClassifier: 0.833
```

	0	1
0	1736	0
1	348	3

```
kNN accuracy score: 0.81792
```

Post-IUFS and RFE:



```
Accuracy score using DecisionTreeClassifier: 0.833
kNN accuracy score: 0.83421
```

Due to our KNN accuracy score improving when the selected features were removed, we concluded that our IUFS and RFE had been successful, and we proceeded to model selection and evaluation.

3. Model Selection and Evaluation

3.1 Evaluation Metrics

The evaluation metrics we included in our modeling were accuracy, precision, recall, F1 score, Hanssen-Kuipers Skill Score (TSS), and Gilbert Skill Score (GSS). Accuracy score represents the percentage of correct predictions, meaning what percentage of the speed dates stayed together afterwards. Precision score represents the percentage of predicted matches that ended up correct. Recall score represents the percent of actual matches that were correctly predicted. F1 is the harmonic mean of precision and recall, which measures our model's overall predictive performance. TSS measures how well the model separates matches from non-matches. Finally, GSS measures how well our predicted matches correspond to true matches accounting for random, lucky predictions.

3.2 Models

The models we chose were probability-based Naive Bayes, information-based Decision Tree, and similarity-based K Nearest Neighbors classifiers to predict our data. First, we created training and testing splits based on our sixteen remaining descriptive features found in our feature selection, along with our target variable, 'match'. Then we conducted hyperparameter optimization using grid search with accuracy score based on the given criterion, i.e. gini, entropy, or log loss using the training and testing splits we created.

3.3 Evaluation

3.3.1 Evaluation Settings and Sampling

Before beginning any modeling, we created our training and testing dataset by splitting off 25% of the data to hold out of training, in order to test our model on that data afterwards.

3.3.2 Hyper-parameter Optimization

For our hyper-parameter optimization, we evaluated each model separately using a grid search. We based this search on accuracy, recording the scores of each model at every possible combination of the selected parameters and deciding on the best arguments to apply at the end.

For optimizing our Decision Tree Model, we searched maximum depth and purity criteria (gini, entropy, or log loss). We found that it scored best at 0.85 with depth 10 using the gini index.

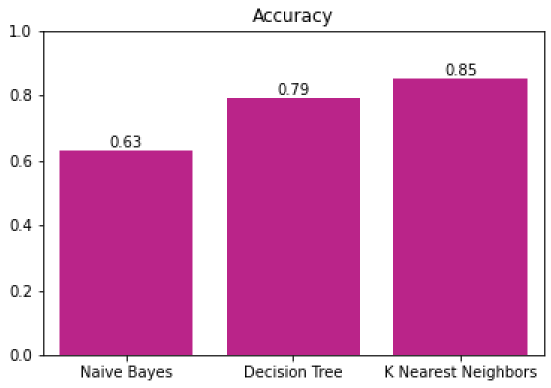
For optimizing our KNN model, we searched the weights preset, leaf sizes, and the Minkowski metric power. We found that the maximum score our KNN model achieved was 0.87 with distance-based weights, leaf size of ten, and Manhattan distance with a power parameter of 1.

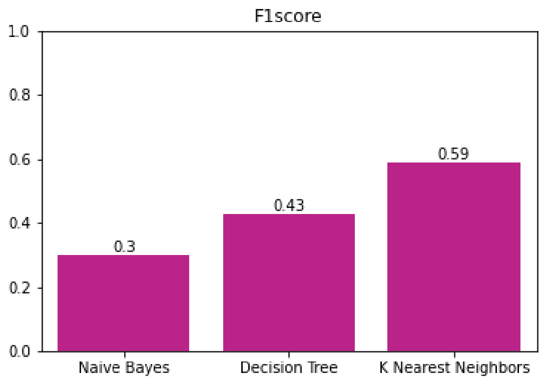
For optimizing our Naive Bayes model, we searched for the smoothing variance value. Our optimization found that the Naive Bayes model performed best with 0.84 with a smoothing variance of 0.01, meaning that portion of the largest variance was added to the variance of all features.

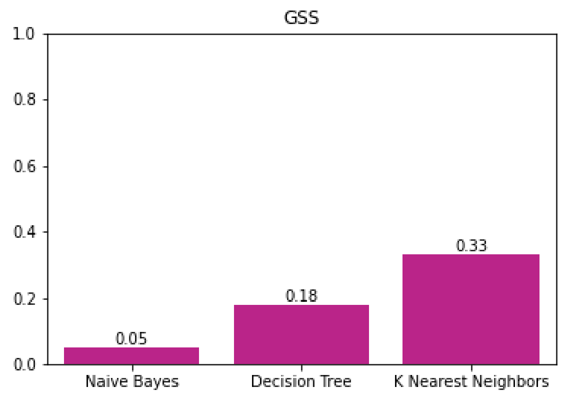
In addition, we optimized the threshold parameter for prediction. Applying the parameters found above, we ran each model at every threshold from 0 to 1 with an increment of 0.01, evaluating each based on their GSS, TSS, and F1 scores. We chose an average of these three metrics as our final thresholds. We chose not to include precision and recall in this average because they are each naturally biased towards very low and very high thresholds by definition. We also chose not to include accuracy because our dataset heavily skews towards negative matches, meaning the threshold for optimum accuracy biases towards high thresholds without actually giving us overall better results.

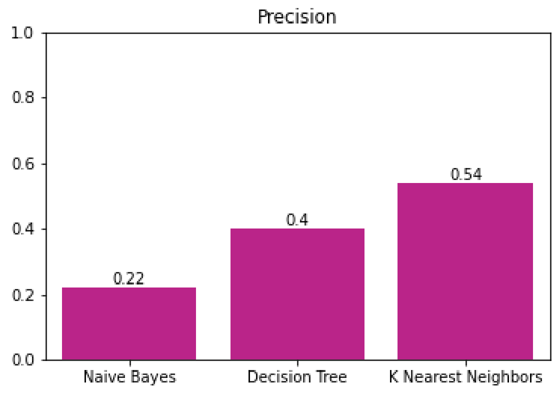
3.3.3 Evaluation

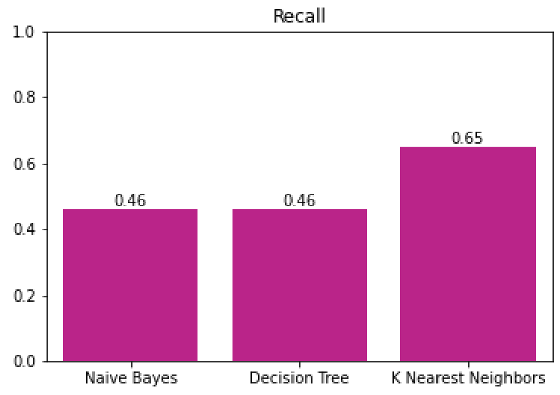
Of the three models we tested, the model that performs best is the K Nearest Neighbors classifier. With all three models optimized, it outperformed both the Decision Tree classifier and the Naive Bayes classifier in all metrics we evaluated. The classifier gave us an accuracy of approximately 0.85. Our optimal threshold for KNN is around 0.35.

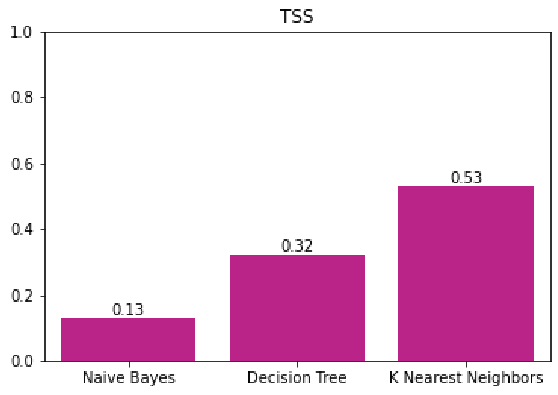












Naive Bayes confusion matrix:

	Actual Positive	**Actual Negative**
Predicted Positive	166	578
Predicted Negative	193	1150

Decision Tree confusion matrix:

	Actual Positive	**Actual Negative**
Predicted Positive	164	241
Predicted Negative	195	1487

K Nearest Neighbors confusion matrix:

	Actual Positive	**Actual Negative**
Predicted Positive	232	194
Predicted Negative	127	1534

4. Results and Conclusions

In conclusion, we recommend that our dating application utilizes the K Nearest Neighbors model with a decision threshold of 0.35, weights based on distance, a leaf size of 10, and a power parameter of 1. We found that our KNN model was not only the most accurate at 0.85, it also performed better in all other chosen metrics.

In addition, our model's optimal prediction threshold is low, meaning it is biased towards positive predictions. This is beneficial to our product, as we prefer to draw in customers with the potential of a match than have a lot of users not receive a match at all, discouraging them from using our application. In addition, given multiple potential matches, we can prioritize those with the highest probability scores. Based on the accuracy and overall performance of our KNN model, our dating application can sell these matching high-probability date services behind a premium subscription.