Capstone Project

Jesse Gallegos
May 14th, 2017

I. Definition

Project Overview

This project investigates a Speed Dating dataset from Kaggle. The dataset was compiled by two professors from Columbia University's Business School. Information including demographics and preferences of attraction were collected from participants during weeknights in the years 2002 through 2004. Moreover, the study aimed at reproducing a business model that Speed Dating services use and chose similar environments to hold these events (e.g. popular bars near universities).

During the events, daters had a four minute "first date" with every other participant of the opposite sex. At the end of each four minute "date", participants were asked if he, or she, would date that person again. Participants were also asked to rate his, or her, date on six attributes: Attractiveness, Sincerity, Intelligence, Fun, Ambition, and Shared Interests.

In addition, the dataset includes a questionnaire that gathered data from participants at different points in the process. These fields include: demographics, dating habits, self-perception. Other data include what features of attraction participants think people find valuable in a mate. These preferences of attraction were asked at the start of the event, half way through the event, a day after the event and three weeks after the event.

Finally, the dataset yielded a paper, Gender Differences in Mate Selection: Evidence from a Speed Dating Experiment [1], by professors Ray Fisman and Sheena Iyengar. The rest of project will explore the mentioned dataset.

Problem Statement

One finding to verify with the papers written by the Columbia professors is that men value physical attractiveness of a partner more than women do. Another is that women place a greater weight on ambition and intelligence than men do [1]. Finally, this project will do two things:

1) The first task at hand is to get an intuitive sense of the data. The goal is to build visuals that summarize key aspects of the dataset.
2) The other task is to create a model (with both genders) to predict whether a person decides to go on a second with the other person.

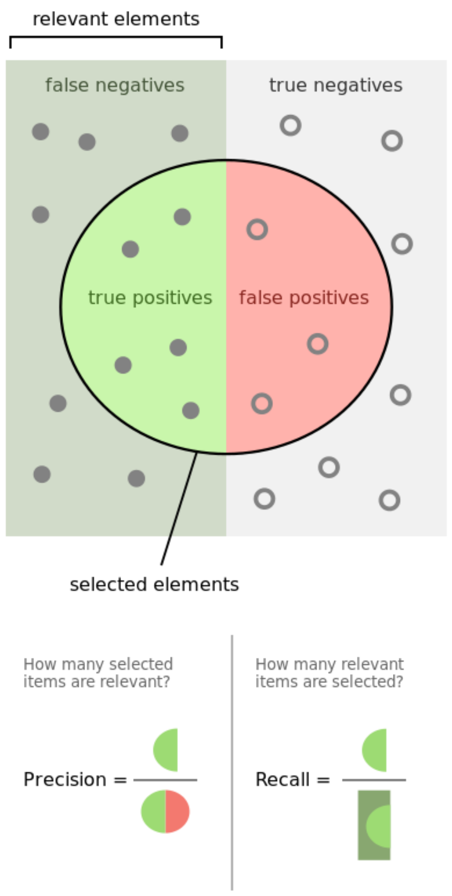
The first thing to do is look for structure in the dataset. This will require cleaning data and working with samples that have the most information possible, that is work with the least amount of NaN values. Visualization will aid in this task, especially if the number of features are reduced to a three dimensional space or lower. Once this is accomplished, choosing a learning algorithm will depend on what data exploration yields.

Metrics

The main challenge is dealing with how my model assigns a 'yes' or 'no' decision. The algorithm could incorrectly label a person saying 'no' to go out with the other person again when in fact the decision was 'yes'. The model could also incorrectly assign a 'yes' label' when the person actually decided 'no'. The two other cases are the model outputting 'yes' when a person wants a second date. Likewise, the model could output 'no' when a respondent does not want a second date.

These four possibilities make precision and recall computable. In the dating context, precision allows for a measure of the correct 'yes' decisions produced by the model. This measurement is a ratio of correct positive 'yes' to the total positive 'yes' (total means the sum of correct 'yes' and incorrect 'yes' labeling). Recall is important to the model because it measures the ratio of correct 'yes' to the sum of correct 'yes' and correct 'no'. f1score is a harmonic mean of Precision and Recall and reports accuracy for correct 'yes' responses produced by the learning model.

A visual for the metric is below and exemplifies the definition:



Example:

Set 'true positives' = 1, 'false positives' = 2, 'false negatives' = 3, 'true negatives' = 4

Precision = 'true positives' / ('true positives + 'false positives') = 1 / (1 + 2) = 1/3 = .33

Recall = 'true positives' / ('true positives + 'false negatives') = 1 / (1 + 3) = 1/4 = .25

f1_score = 2 *(Precision * Recall) / (Precision + Recall) = 2 * (1/3 * 1/4) / (1/3 + 1/4) = .29

From the visual, Precision is critical in figuring out whether a person is match (being able to correctly decide 'yes' or 'no'). Precision of as a gross filter such as the 'swiping' behavior on websites like Tinder.

Recall is after throwing away the 'no', how many of those 'yes' are good matches. What that means is that, after 'swiping' exploring what is left determines how many of those 'swipes' to the right were correct.

Note: At first, my evaluation metric was Mean Absolute Error (MAE). The goal was to benchmark with results published by Fisman and Iyengar. However, I realized that the authors of the article incorrectly used a regression algorithm for classification. Other issues are addressed later as well. However, in spite of that, the two professors did publish weights for the features that they selected. This will aid in doing sanity checks in the exploratory stages of this project.

II. Analysis

Data Exploration

The dataset file, Speed Dating Data, has 8378 data points with 195 different features. There were 551 people involved in the dating event with the population being divided into 274 women and 277 men. For the sake of brevity, the reader is invited to consult the Data Key file included with the dataset. The Key file gives the exact number of participants for each of the 21 'wave' events. Moreover, the Data Key file gives a thorough overview for what the allowed values for each field are.

The first attributes that will be explored at the start of the project include 'iid', 'gender', 'wave', 'attr', 'intel', 'sinc', 'fun', 'amb', and 'shar'. 'iid' uniquely identifies each participant in the event and is useful for querying unique members. 'gender' helps to query by gender. 'wave' number helps in querying what wave an individual participated. The features of interest are: 'attr', physical attractiveness; 'intel', intelligence; 'fun', how fun a person is; 'amb', ambition; and 'shar', shared interests. Each of these features of attractiveness are rated in one of two ways:

1) ratings based off a Likert scale ranging from 1 to 10.
2) daters are asked to distribute 100pts to the six features of attraction in the manner he or she feels is most important.

Other things to note: features of attraction with extensions "1_1" relate to the question asked at some point during the event; the first number pertains to the question number and the second number refers to the time of the event (e.g. 'attr1_1' is question 1 asked about 'attr' at the start, '1', of the event). Likewise "1_s" refers to question '1' asked halfway, 's', through the event. "1_2" refers to question '1' asked the day after, '2', the event. "1_3" refers to question '1' asked three weeks, '3', after the event. These extensions are important to see, if after various dates, daters change their outlook on stated preferences of attractions with respect to the beginning, middle, day after, or three weeks after the dating event.

There were various things to do with inputs of the dataset. To illustrate a subset of the

iid	gender	wave	attr1_1	intel1_1	sinc1_1	fun1_1	amb1_1	shar1_1
20	1	1	100	0	0	0	0	0

features space, here is what a sample looks like:

As noted previously, 1_1 denotes question 1 asked at the start of the event. The Dataset Key that accompanies the dataset states that question 1 is what participants look for in another person. The sample here is a male, with 'iid'= 20, who put all weight towards physical attractiveness (this participant distributed all 100 pts to physical attractiveness, 'attr'). Of all the participants in the dataset, only three people put over 90% weight on physical attractiveness: 2 males and 1 female.

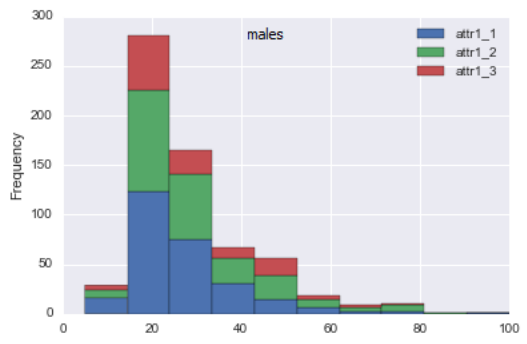
Quick things to note in this step: ratings were changed from a Likert scale of 1 - 10 to a range from 0 - 100. This was done prior to computing basic statistics on the dataset. Prior to scale changing, most ratings were on a scale of 0 - 100. With that on the record, the mean weight that males report for physical attraction in a mate is 26.97 (STD 13.39) as opposed to the mean weight of 18.02 (STD 9.93) for females. The difference between what each gender is looking for at the start of the event is 8.95 (f-test confirms as well).

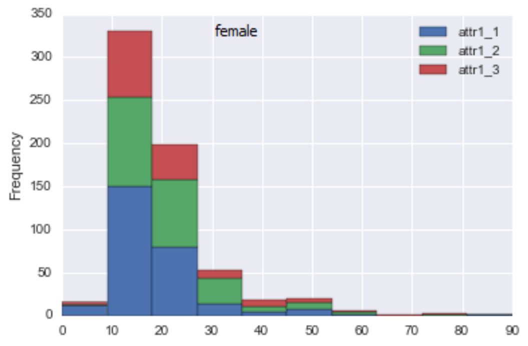
Likewise, comparing what males look for regarding intelligence, 19.43 (STD 6.81), against what females look for at 20.99 (STD 6.82) are values close to each other; STD is almost equal as well. At the exploratory stage, it is not apparent that women put more weight on intelligence than men do. Later steps will show this (f-test shows a difference in the ipynb file). However, there is disparity in ambitiousness sought by gender. Males, on average, weighted ambitiousness with 8.78 (STD 5.93) and females weighted ambitiousness at 12.82 (STD 5.63). Ambitiousness is something that females weigh more than males do f-testing shows this as well. Again, the ipynb file has more f-test statistics to pour over and the reader is invited to go explore the file.

Other things apparent from skimming through the dataset is that there are lots of NaN values once loaded into a Pandas DataFrame. Part of this is by design of the experiment; some fields were left empty as a control for the experiment. In addition, the Data Key file states that funding was an issue in gathering samples. Other explanations include some fields being left empty by respondents accidently, discomfort caused by the question, or that the question was not applicable to the person. Finally, outlier detection by Turkey's Method yielded 239 samples across 15 different features.

Exploratory Visualization

Below are histograms of what people are looking for in terms of physical attraction. The vertical represents number of respondents; the horizontal represents weight given.





These plots are intended to show whether or not there is a shift of preference over time. In this case, at the beginning of the event, a day after, and three weeks after the event. The blue bars represent what people are looking for at the start of the event. The green bars represent what people are looking for a day after the event, and the red bars represent what people are looking for three weeks after the event.

Disclaimer, the histogram used a setting that 'stacked' the bars on top of each other. Plotting them without 'stacked' yielded messy visuals. The way to interpret the visuals above is to look at the heights of each bar relative to its color. For example, the blue bar for 'attr1_1' for males peaks, on the horizontal axis, at 20 with a frequency of 125 respondents. Likewise, the green bar for the same plot has about 100 respondents and peaks at 20. Finally, the red bar in this plot also has its peak at 20 with about 50 respondents. The mean for male respondents for this features is 26.98 (STD 13.39).

The histograms also obviate that females do not weigh physical attractiveness as heavily as males. For example, attractiveness was between 10 and 20 as opposed to males who replied with numbers centered near 20. Female response rate also dissipated starting with 150 replies, falling to 100, and then to about 75. The height of bars over time is consistent relative to each color. Blue, green, and red bars have max frequency at 10 - 20 weight. The average for this feature with female respondents is 18.02 (STD 9.93).

More histogram plots are included for features in the ipynb file. Again, the reader is invited to consult these plots. Something powerful from having these frequency plots is that they are a visualization of the distribution for a feature. Both genders have a positively skewed graphs for physical attraction. This helps in computing the probability of extreme values of the three people who weighed physical attractiveness at over 90 percent.

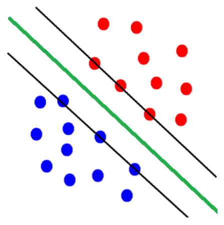
Algorithms and Techniques

Previous work in this domain is inspired by Chris McKinlay's analysis on OkCupid data [5]. McKinlay used boosting on his categorical dataset and weighed features to the website's matching questionnaire. The main finding that resulted from using a clustering algorithm is his visualization. McKinlay found categories into which females were lumped and graphed these on a three dimensional axis.

What is dissimilar with McKinlay's dataset and this one is that the dataset for this project has both categorical and numerical responses. I'll hypothesize there exists some inherent structure and use McKinlay's finding to create a visualization.

A Linear Classifier will be used to stay true to the spirit of the authors who compiled this Speed Dating Dataset. However, I am skeptical about how the authors used linear regression in a classification setting.

For the Linear Classifier, I will implement GridSearch on a Support Vector Classifier (SVC) from sklearn. A list of things that GridSearch will do for the Linear Classifier are listed below:



- Try different values of 'C'. 'C' is the separating distance, or margin width, between 'yes' and 'no' decisions that are to be predicted. The figure on the right illustrates the green separating line and the black margins.
- Do cross validation with 10 bins. Cross Validation will train on 75% of available samples and test on 25% of the remaining available samples. These sample are ones described in an earlier section, that is, samples with the least amount of NaN values.
- Use an f1_scorer to measure accuracy.

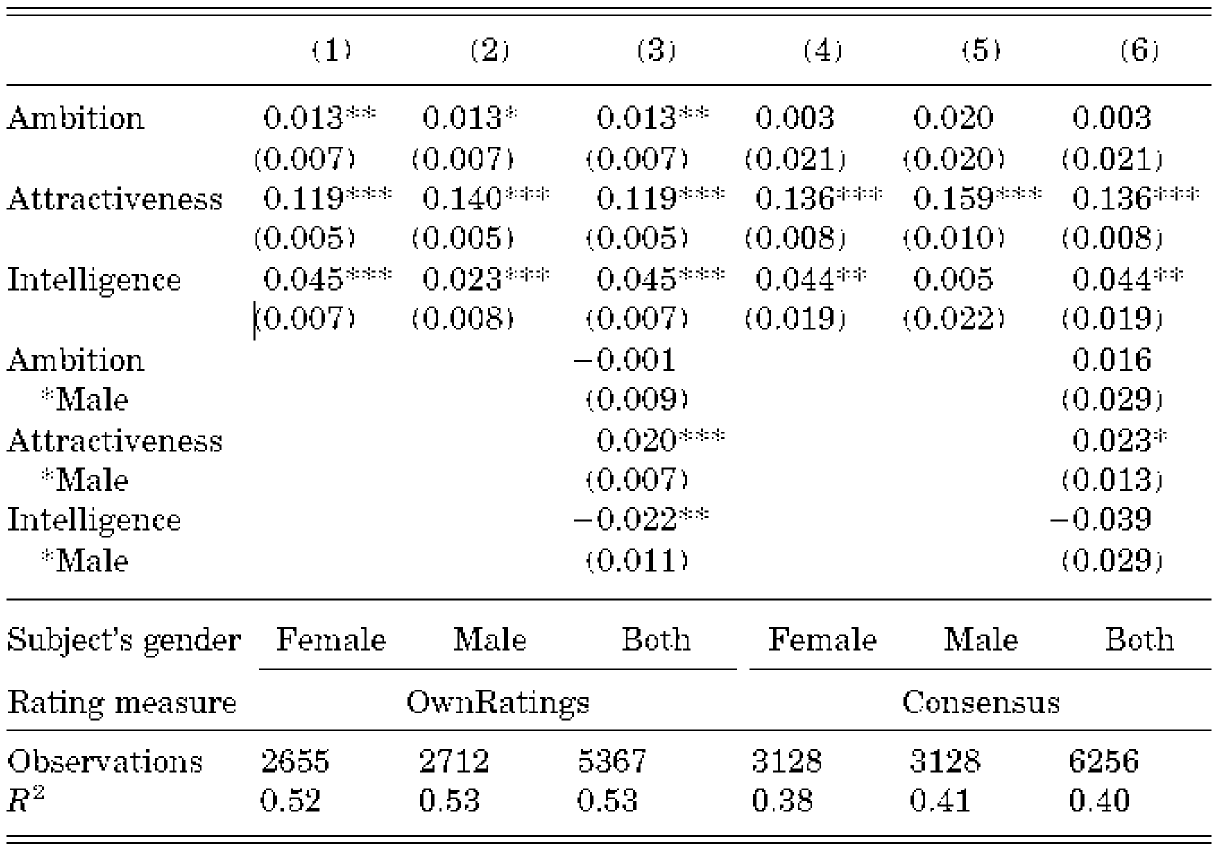
Intuition helps in thinking about how the Linear Classifier works. For instance, in a 2-D, or 3-D, feature space, samples are plotted. If the feature space is 2-D, a line is the separating boundary as shown in the figure above. For a 3-D feature, a plane is the separating boundary between decisions. Higher dimensional settings use higher dimensional hyperplanes. The biggest assumption that the authors who compiled the Dataset make, without stating, is assuming that the space is linearly separable.

A Linear Classifier in this context attempts to divide between 'yes' and 'no' decisions because it is extremely interpretable. Ideally, everything on one side of the line is 'yes' and on the other side 'no'.

Benchmark

The benchmark model provided with the article is flawed. Fisman and Iyengar proposed a Linear Regression Model to determine decisions of participants based off what daters believed important. Their Decision function is dependent on attractiveness, ambition, and intelligence. Below is a table summarizing the authors results:

GENDER DIFFERENCES IN SUBJECTIVE ATTRIBUTE WEIGHTS



	(1)	(2)	(3)	(4)	(5)	(6)
Ambition	0.013**	0.013*	0.013**	0.003	0.020	0.003
	(0.007)	(0.007)	(0.007)	(0.021)	(0.020)	(0.021)
Attractiveness	0.119***	0.140***	0.119***	0.136***	0.159***	0.136***
	(0.005)	(0.005)	(0.005)	(0.008)	(0.010)	(0.008)
Intelligence	0.045***	0.023***	0.045***	0.044**	0.005	0.044**
	(0.007)	(0.008)	(0.007)	(0.019)	(0.022)	(0.019)
Ambition *Male			−0.001			0.016
			(0.009)			(0.029)
Attractiveness *Male			0.020***			0.023*
			(0.007)			(0.013)
Intelligence *Male			−0.022**			−0.039
			(0.011)			(0.029)
Subject's gender	Female	Male	Both	Female	Male	Both
Rating measure	OwnRatings			Consensus		
Observations	2655	2712	5367	3128	3128	6256
R^2	0.52	0.53	0.53	0.38	0.41	0.40

For this project columns (1), (2), and (3) are explored. Columns (4), (5), and (6) are not explored at this time because the analysis of these columns adds to the features space rather than simplify it. Interpreting Columns (1), (2), and (3) is an easy feat. Column (1) corresponds to females, Column (2) corresponds to males, and Column (3) corresponds to both genders taken together in the Regression Model.

The model that the authors of the study used is below:

$$Decision_{ij} = \alpha_i + \sum_{c \in C} \beta_c * Rating_{ijc} + \varepsilon_{ij},$$

$$\text{where } C = \{\text{Attractiveness, Intelligence, Ambition}\}.$$

Note: 'C' in the equation above should not be confused for 'C' in the classifier.

The row with attractiveness gives the coefficient for females to be 0.119 and for males 0.140. Males put a greater weight on attractiveness by 0.021. Both genders lumped together in the model yielded 0.119 weight for the attractiveness features.

In a similar fashion, the coefficient for the author's Model for females gave a coefficient of 0.045 as opposed to males who ended up with a coefficient of 0.023 (a difference of 0.022). Females weigh intelligence more than males do.

The author's model put ambitiousness at a tie, but its relevance is indisputable. The authors of the study point out that females seek ambitious males because of the ability to acquire resources; males care about ambition because males do not want to date a woman more ambitious than he [1].

It will be shown later that reducing the features space helps in visualizing what sort of model to apply. It is important to keep in mind that the metric of choice for my model is f1_score and that the author's benchmark is problematic when comparing with my results.

III. Methodology

Data Preprocessing

The first thing I did before doing any preprocessing was figure out with what I was working. I ran a function I defined in a separate file called features_creator.py. The method I implemented counts the number of samples for each feature. I let my threshold of samples be 7302; the list called features_space defined in features_creator.py file includes features with the number of samples mentioned above. Features whose sample size fell below that threshold were not considered for this project. Doing so, reduces the features space from 190 to 77.

As mentioned before, this project needs data clean up. A quick look at the data set reveals that some of the features do not fall within the specified ranges laid out in the Key file. An example includes entry fields based on the Likert scale from 1 to 10. Some fields have violated these ranges and need to be fixed to conform to the range. Numbers below 1 will be raised to 1; number greater than 10 will be lowered to 10. Scale methods are defined in features_creator.py file. These methods are implemented in the ipynb file.

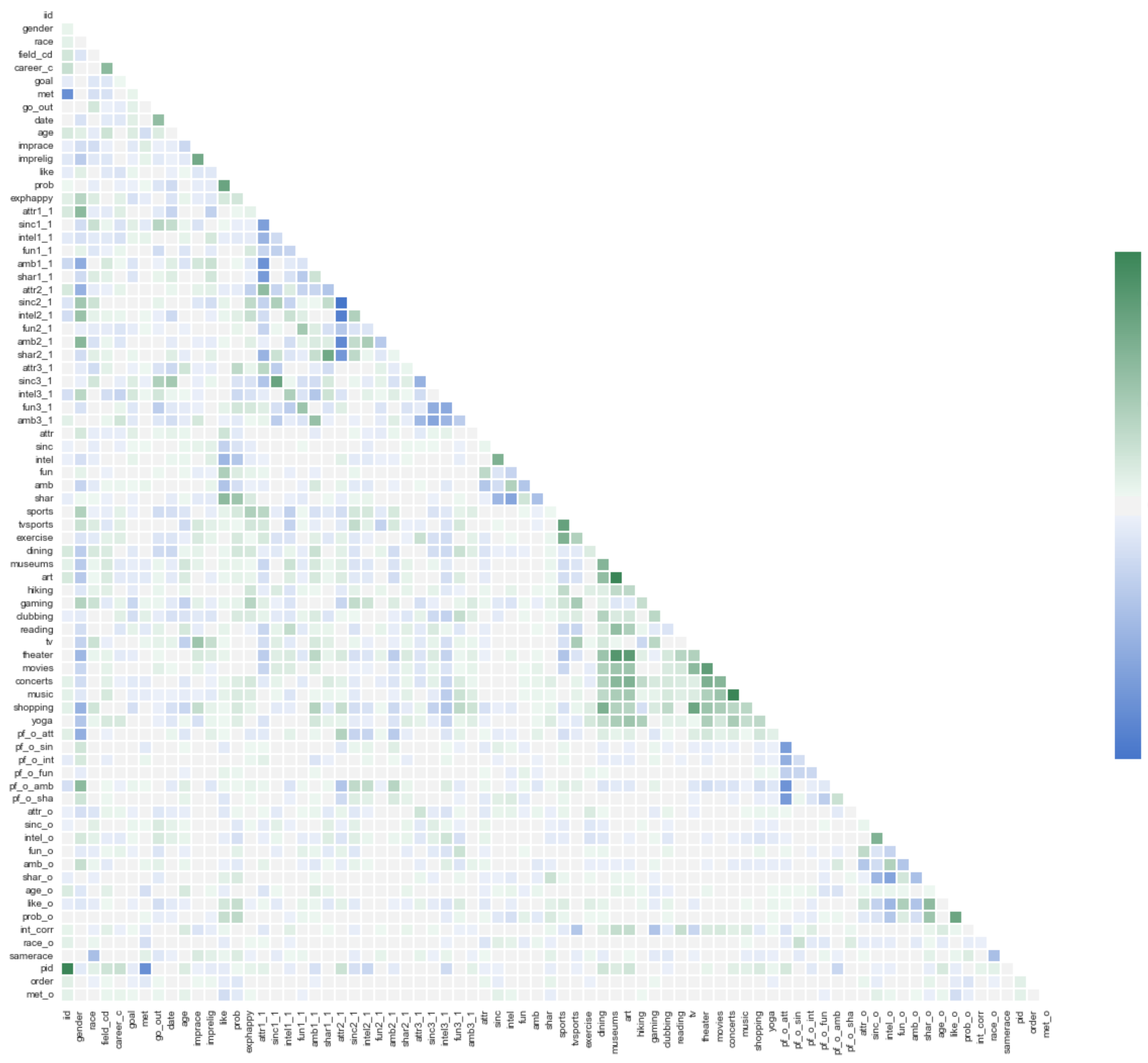
Another thing to do with Likert Scale values is to convert them to a scale of 0 - 100 and make sure that values of attraction that are being converted add up to 100. To accomplish this I added up the scores on the Likert scale and divided each feature of attraction by the sum. Afterwards, I multiplied by a hundred. This produced the desired range to compare with other ratings given to a date during the event.

The dataset had other issues. The instance in mind is where daters were asked to distribute 100 points across each feature of attraction in question. Some respondents accidentally distributed more than 100 points across the six features of attraction. This was not a problem because these distributions are relative to how the these points were assigned.

After the features were scaled to 0 - 100, I ran the dating_attributes_vs_time_describe() function defined in the features_creator file. This method takes in a DataFrame and gender as input and displays the min, max, mean, and STD. I also used the dating_attributes_vs_time_hist() defined in the features_creator file. This method utilized the plot() function with parameters kind = 'hist', stacked = True, bins = 10. 'hist' tells plot() to generate a histogram with the setting 'stacked' set to True. 'stacked' places the bars in a histogram on top of other bars that fall in each respective bin. A bin is one bar along the horizontal. The histograms seen in the Exploratory Visualization section are outputs from the mentioned function.

Points of contention included categorical labels that needed to be changed to numerical values. Examples are SAT scores, zipcode, tuition, and income. These needed to be changed to numerical values for feature selection by tree method. Moreover, for PCA I needed to scale the range of the features being used to positive numbers between 0 and 1.

The best visualization that shows relationships between some of the features is the correlation scatter matrix shown below (zoom in to see).



When I went through basic statistics, I noticed that when one feature went up and other features values went down. Darker shades indicate correlation one way or the other. Specifically, green indicates positive correlation; blue indicates negative correlation. Having this graph helped me in deciding if Principal Component Analysis is useful later in my analysis. The goal is to reduce the number of features to simplify my learning model.

There is some insightful information looking at the graph above. Caveat: the graph produced above is for both genders. The reader is invited to consult the ipynb file that accompanies this report and consult the correlation matrices for males and females respectively.

The features chosen for the correlation matrix were specifically picked out of all the available features because these features had the most samples. The graph above was produced after data clean up.

That said, one feature that caught my eye was that shar2_1 is correlated with shar1_1. This result is amusing because 'shar2_2' pertains to the question of what a dater believes the person of the opposite sex is looking for in regards to shared interests. 'shar1_1' pertains to what the mentioned person is looking for in shared values. One interpretation is that a person can change the weight for what is sought based off what is believed to in demand. Another way to read things is that when dating, you are 'vibing' off each other, seeing what the other person wants, and catering to that person.

Self-perception also has its affect on what is pursued. For example, 'sinc3_1' is correlated to 'sinc1_1'. 'sinc3_1' pertains to the question of a how a person rates himself, or herself, for that feature. In plain English, this means that a sincere person looks for a sincere person. The reason why this is insightful is because of literature in the field [2]. Nate Silver, the famous statistician, wrote an article that states people look for features that they themselves have.

What stands out the most is 'attr2_1' effect on 'attr1_1' and vice-versa. This correlation implies that a person's belief on what people look for in terms of attraction affects what he, or she, wants in the opposite sex. The OkCupid Trends blog had an interesting article on the matter. The article highlights that people lie about themselves online. Moreover, males tend to pursue the unachievable, and females tend to drop their standard when immersing in online dating [3]. In fact, another work also holds that females tend to be more attractive on average than males [4]. Dating Literature helps guide that this project is heading in the right direction.

I did not discuss negative correlation, but the example to note on that would be 'attr1_1' is inversely correlated with 'sinc1_1', 'intel1_1', 'amb1_1', and 'shar1_1'. At face value, attractiveness takes weight from the mentioned features. Finally, just by exploring the correlation matrix, my hypothesis is that having information is key as to how to speed date. If a dater knows, or has some insight as to what his/her date wants, he or she can be successful in the sense of setting expectations.

Implementation

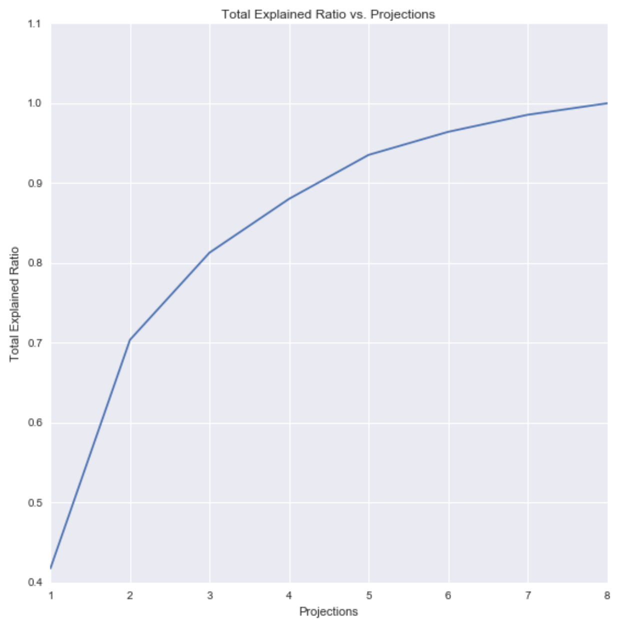
Up to this point no mention of features selection via algorithms has been discussed. However, that will change. Part of the reason is because of my internal debate whether this was part of visualization, preprocessing, or implementation.

The feature space prior to feature selection is at 77. After doing clean up and type casting I ran four different selection algorithms. The algorithms used were ExtraTreesClassifier (ETC), RandomForestClassifier (RFC), and f_classifier. The tree classifiers choose features based off decision trees. The distinction between the tree algorithms is that one uses a log function and is computationally more expensive to do. The method forests() in the features_creator.py file prints out selected features with random_state = 0.

In a similar fashion, f_classifier computes f-values between the features and the target space. The results of the three mentioned feature selectors are as follows: ETC reduced the feature space from 77 to 22, RFC from 77 to 18, and f_classifier to 40. The ipynb file reports the feature weights and p-values. This is a huge reduction in features. Not only that, these results seem to coincide with what websites like eHarmony use (29 features).

Consistency between ETC, RFC, and f_classifier algorithms reported the same 8 features at the top. These features are 'like', 'attr', 'intel', 'sinc', 'amb', 'shar', 'fun', and 'prob'. After the eighth ranking feature there was inconsistency. There is more discrepancy with chi2 selection. At a p-value of 0.01, chi2 returned 'like' and 'prob' as the top two features. Other than that, chi2 selected different features.

For the sake of simplicity, I took the first 8 features reported by ETC, RFC, and f_classifier. The reason for such madness is that those 8 features do not have strong correlation with the other 69 features (see scatter matrix plot from previous section). With these results in mind, I have an educated guess that further feature reduction might work. I took these remaining eight features and did PCA on them. A graph of the explained variance for PCA is below:



As can be seen, explained variance goes up as the number of projections, n, increases. The question is, "what is the cut off?" At n = 1, about 42% of variance is explained; n = 2, has 70.4% of variance explained; and n =3, 81.3% explained variance. Beyond n = 3, explained variance increases by small amounts until all variance is explained at n = 8. For the sake of visualization and to simplify the model, n = 3 will be chosen for the number of projections for PCA. The method used to generate the graph above was explained_ratio_pca() from the features_creator.py file.

The table below shows the projection of features onto PCA where n =3.

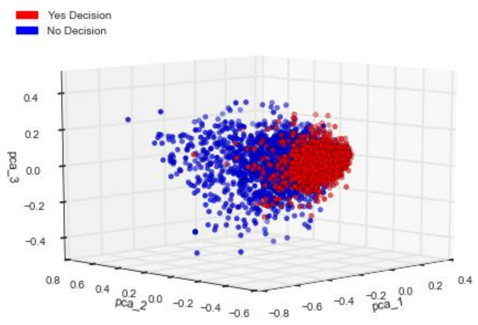
	'like'	'attr'	'intel'	'shar'	'sinc'	'amb'	'fun'	'prob'	Exp Var	Exp Ratio
PCA_1	0.18	-0.13	-0.36	-0.02	-0.21	-0.36	-0.18	0.78	0.03	0.42
PCA_2	-0.81	-0.03	0.24	-0.07	0.15	0.26	-0.03	0.44	0.02	0.29
PCA_3	0.23	-0.01	0.14	-0.92	0.10	0.16	-0.20	0.05	0.01	0.11

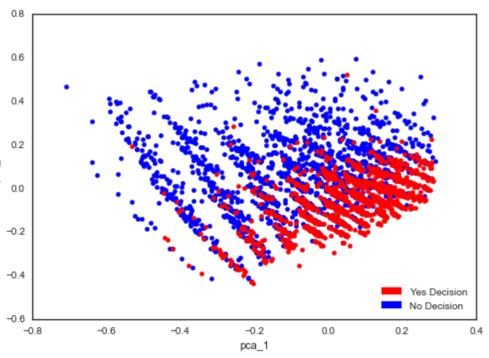
PCA_1 is formed by a large portion of the 'prob' feature and varies inversely to 'amb' and 'intel'. This vector explains 42% of the variance between the eight features listed. From the scatter matrix visualization this makes sense; 'prob' has negative correlation with 'amb' and 'intel' in the visual.

Likewise, PCA_2 is mostly a projection of 'like' and varies inversely with 'prob', 'intel', 'amb', and 'sinc'. This vector explains about 29% of the variance across eight features. Checking on the scatter matrix, once again, 'like' is negatively correlated with 'intel', 'amb', and 'sinc'.

Finally, PCA_3 is composed mostly of 'shar' feature. The percentage of variance explained by this component is 11%. The 'shar' feature did not have much correlation with other features in the scatter matrix visualization and the values making up PCA_3 are low relative to 'shar'. Hence, results for PCA_3 make sense.

The results below are pretty cool. Apparently, doing PCA, with n = 3, on the 8 features mentioned above yielded the following plots.





This is the crux of the project. I made a file with rotations of the three dimensional figure from various angles. These images are contained in the folder "Images_for_PCA." Again, the reader is invited to go through the images and see how the blue and red dots are localized. The blue dots are 'No' responses to a second date. The red dots are 'Yes' responses to a second date. The method that produced these plots is pca_plotter() from the features_creator.py file.

With these graphs I can decide on what algorithm to use. I will opt for Nearest Neighbor Algorithm for the sole reason that it is easy to explain to a board of directors. In the next section, I'll discuss the linear classifier to keep true to the spirit of the dataset as well the Nearest Neighbor implementation.

Refinement

To stay close to the intent of the authors, Linear Classification is the way to go. I will use Linear Classifier as my benchmark against my Nearest Neighbor model. Some of the issues that became apparent at the start of the project was that the computation for non-standardized features set never completed making a model. In fact, my script would time out. This made it impossible to make a direct comparison between the weights the authors reported on the table illustrated in the Benchmark section. The Linear Classification did work when I passed in a features set standardized between 0 - 1.

To create the Linear Classification model, I used GridSearchCV() method in scikit. GridSearchCV() automates parameter selection for a given classifier. For the Linear Classifier, I passed into the GridSearchCV() a list defining 'C' values in logspace from -2 to 2 with ten spacings in between. The estimator parameter was the Linear Classifier. The scoring function I passed into GridSearchCV was f1_score. The parameter for the number of cross-validation bins was 10.

The data that was passed into GridSearchCV() is the PCA transformed data. I decided to take a different direction because comparison with the published results from the authors was impossible (computational power limited the Linear Classifier model). To compensate, PCA projected 8 features, common across various selection algorithms, onto three components. Afterwards a the training set was created consisting of 75% of the selected dataset. The testing set was 25% of the remaining dataset.

Implementing the Nearest Neighbor model had a similar mechanism to it. I passed my PCA data to the GridSearchCV() method with a list defining values for the n_neighbors parameters from 1 to 100. The classifier was the KNeighborsClassifier() from scikit learn with an f1_scorer scoring function. cross-validation bins were set to 10. The data passed into GridSearchCV() was partitioned in the same 75% - 25% fashion as the Linear Classifier.

The reason I used Nearest Neighbors is because things are noisy at the boundary and classifying a 'yes' or 'no' based off a line is too simple. The next interpretable model is one based off data that are similar to each other based off their relative distance. On the graphs provided, samples are localized near each other and the 'blue' and 'red' coding are indicative I can base decision for unknown data based off what know samples near the unknown are.

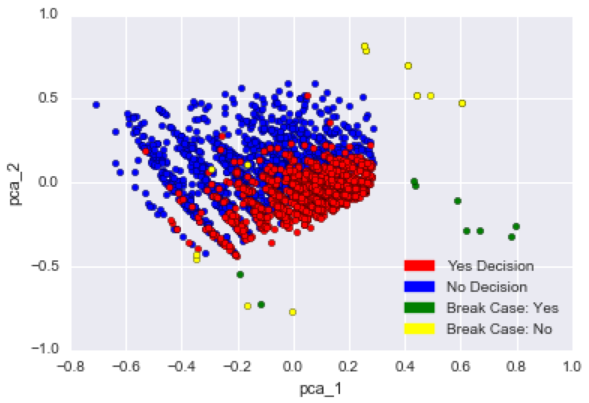
The best training result for running GridSearchCV() for the linear Classifier across 10 different logspace values between -2 and 2 had an f1_score of 0.628 with C value 35.938. The test f1_score is 0.680 at C = 35.938.

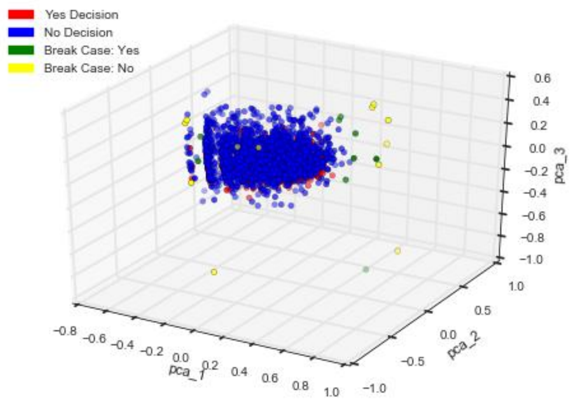
Similarly, for Nearest Neighbor model yielded a training accuracy of 0.714 with the optimal value of 63 neighbors. The test set yielded 0.752 accuracy tested with 63 neighbors.

IV. Results

Model Evaluation and Validation

To test my Nearest Neighbor model I created samples and had the algorithm predict whether a person receiving extreme ratings would return a 'yes' or 'no.' The model was trained on all data available. The following are two visuals of showing the spatial arrangement of decision making:





The ipynb file has all the test points in question. Here are three to give context to the graphs above:

Stereotype	like	attr	intel	shar	sinc	amb	fun	prob	pca_1	pca_2	pca_3	dec
all_looks_yes	1	1	0	0	0	0	0	1	0.67	-0.28	0.31	1
unlikeable_attractive_jerk_yes	0	1	0	0	0	0	0	1	0.49	0.52	0.08	0
balanced_no	0	.16	.16	.16	.16	.16	.16	0	-0.16	0.11	0.05	0

My sensitivity test cases (yellow and green dots on both plots) for the most part are extreme; this is seen by their distance from the red and blue dots. These points were arbitrarily chosen to test the model for unseen test points. Looking at the pca columns, it is easy to locate where all_look_yes falls. For all_look_yes, I located 0.67 on the pca_1 axis and then went down to -0.28 units to find a green dot indicating a 'yes' decision. all_look_yes is one of many green points that the model predicts to be a decision of 'yes'. Looking at where the green points falls in the spatial arrangement, it is plausible for these points to be 'yes' decisions since they are nearer to the red 'yes' decisions from the pca tranformed data.

Likewise, the unlikeable_attractive_jerk_yes is a yellow point located near the 'no' decisions (locate 0.49 on pca_1 axis and then go up 0.52 up the pca_2 axis). Several yellow points predicted by the model to be 'no' decisions neighbor the unlikeable_attractive_jerk. Moreover, these yellow points are near other 'no' decisions predicted by the model.

Finally, the 'balanced' points are localized near the center of the plot where pca_1 and pca_2 are near zero (things are very noisy here since at this location since red changes to blue and vice-versa). Overall, visual inspection gives a good glimpse as to whether or not the Nearest Neighbor algorithms is making good predictions. The spatial arrangement these arbitrarily chosen points help build faith that the model is yielding believable results.

Justification

A benchmark for my PCA chosen features with Nearest Neighboors Classifier is not available. The methodology used provided good results is summarized below in lieu of a benchmark model.

1) Visualization produced by projecting the most common features produced by ExtraDecisionTrees, RandomForestTrees, and the f-classifier the same top 8 features respectively. These features are 'like', 'attr', 'intel', 'sinc', 'fun', 'amb', 'shar', and 'prob'.

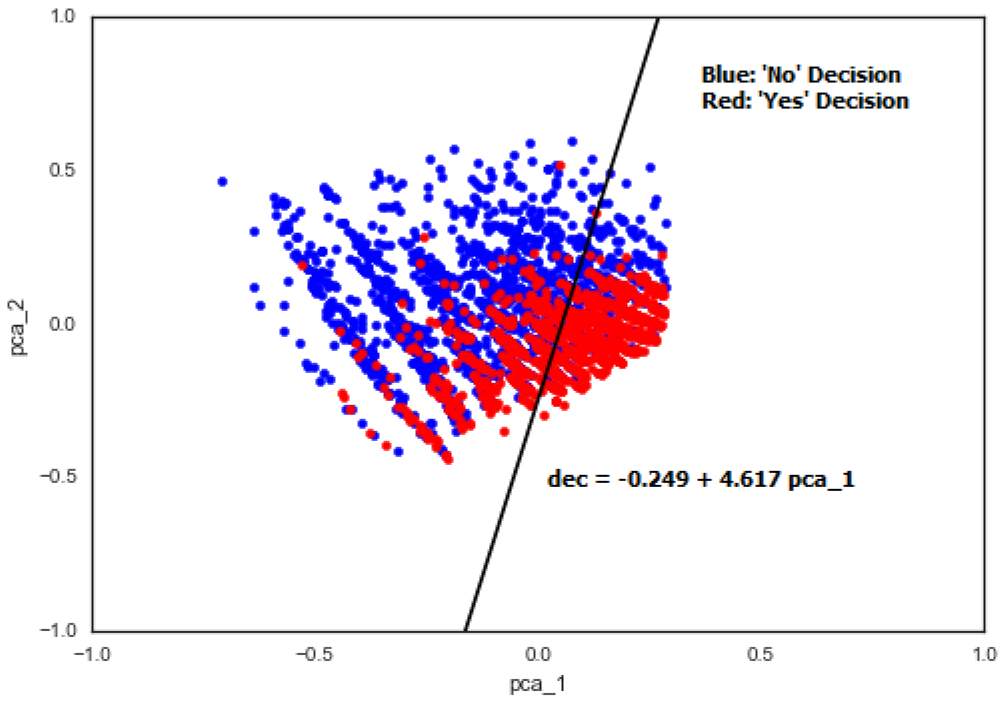
2) PCA to reduce the features space and projected variance onto three components from which plots visualized above were derived.

3) Linear Classification yielded lower accuracy (0.68) against the Nearest Neighbor Algorithm (0.75). Nearest Neighbor Algorithm was chosen because it is intuitive and interpretable.

V. Conclusion

Free-Form Visualization

If there is any evidence that a Linear algorithm is inadequate for solving a problem of this scale it is the following the plot:



The plot above was generated by running the linear_plotter() method in the features_creator.py file. The image shown is a scatter plot with a linear function superimposed on it. Things to note: the line could be angled differently to so that it more inclusive of other points. In fact, the line could be translated and rotated to improve 2-D decisions. The intercept and pca_1 were determined from Linear Classification and determine the behavior of the black line pictured above.

Note: The plot shown above is a level curve of higher dimensional, non-plottable function which consists of a linear combination of pca_1, pca_2, and pca_3. In the plot above, pca_2 and pca_3 were set to zero to generate the level curve (The black line is the projection of the Linear Classifier equation onto pca_1 vs. pca_2 space). Finally, the most obvious argument against a Linear Classification solution is that the scatter plot is noisy; blue 'No' decisions overlap into the red 'Yes' decision region. Finally, this plot is a visual representation of the limitations of Linear classification; the boundary line could have divided the 'Yes' and 'No' evenly.

Reflection

The end-to-end problem for this project is being able to predict a match when dating. Matchmaking Services such as eHarmony and Match.com are the largest providers for online dating. My original goal was to make something along the lines those companies employ. In fact, figuring out an algorithm that successfully produces 100% matches is the million dollar question. What became overwhelming was trying to directly predict whether two people are a 'match'. To reiterate, a 'match' is dependent on whether two people say 'yes' to each other. Two people saying 'yes' to each other depends on gender. Each gender weighs things differently, such as, 'attr', 'intel', or 'amb'. For this project, I put both genders together and worked with both demographics in the same model to predict 'dec'.

Qualitatively reducing the features size to predict 'dec' was a controversial decision I took on implementing my solution and gambled on whether the approach would produce results. The question at every step of the process was "how many features do I want to keep?" and "how much information am I throwing away in the process?" There was some anxiety provoked by those questions. After attempting to do this, I found out that the way to answer this aspect of the project was to predict how people are going to decide on a person. Scatter matrix visualization gave me some security and the feature selection algorithms helped in choosing the top common features that resulted with those algorithms. Finally, PCA reduced the 8 features I had from preprocessing and projected the variance from those 8 features onto 3 features. The resulting visual from PCA guided me to choose a Nearest Neighboor Model.

I am completely satisfied with the Nearest Neighbor Model for the reduced features space. It offered a huge improvement over the Linear Classifier described in a previous section. In a general setting where more features are included, Nearest Neighbors might be applicable given locality of data.

Improvement

There is room for improvement in the model without a doubt. The issue is ingenuity and having some kind of 'hunch' on how to make improvements. One way to improve this project might be to abandon interpretability and keep more features. A higher dimensional space might improve the linear classifier results. In fact, with more features, the space might become separable.

When I read McKinlay's work with the K-modes algorithm, I accidently stumbled onto the K-prototypes algorithm. K-modes works with categorical data as opposed to numerical data in the K-means algorithm. K-prototypes works with both types of data, that is, categorical and numerical. K-prototypes is applicable to this dataset, is something I want to look at, and could improve the visuals, particularly on how samples are categorized.

Finally, I'm a firm believer that there better solutions and better algorithms to use for this dataset. I admit I took 'the easy way out' in the sense that I choose the most intuitive approach and wanted to have something 'I can sell to a board of directors' who might be laypeople to the work of Machine Learning.

Works Cited

[1] Fisman, Ray. Iyengar, Sheena. "Gender Differences in Mate Selection: Evidence from a Speed Dating Experiment."

[2] Silver, Nate. "In the End People May Really Just Want to Date Themselves". https://fivethirtyeight.com/features/in-the-end-people-may-really-just-want-to-date-themselves/

[3]"Your Looks and your Inbox". OkCupid Blog. https://theblog.okcupid.com/your-looks-and-your-inbox-8715c0f1561e

[4] "Why Beatiful People Have More Daughters: From Dating, Shopping, and Praying to Going to War and Becoming a Millionaire." Miller, Alan S. Kanazawa, Satoshi.

[5] "How a Math Genius Hacked OkCupid to Find True Love". Wired. https://www.wired.com/2014/01/how-to-hack-okcupid/

[6]"eHarmony CEO : The Ingredient Online Dating Sites could be Missing about Attraction". eHarmony. http://www.cnbc.com/2017/03/31/eharmony-ceo-online-dating-is-missing-humor.html