Proposal

Domain Background

(approx. 1-2 paragraphs)

For this project, I would like to analyze data from speed dating event(s) and create a model to predict whether two people will be interested in seeing each other again.

As a psychology professor, I'm obviously fascinated by human behavior. I became interested in machine learning and data science because I saw it as an innovative new way to understand why we make the decisions we do. And, when it comes to dating and relationships, humans are wildly unpredictable. We routinely make romantic decisions that go against our own stated preferences -- see, for instance, the work of Eastwick and Finkel (2008).

On the other hand, some general principles do in fact predict romantic relationships. Dr. David Buss is a pioneer in the field of "mate selection" -- the psychological term for choosing a partner. One of the most robust findings in his research is that of **homogamy**: people tend to pair with others that are similar to themselves in a variety of dimensions: personality, religion, political opinions, education, income, race, and so on. This principle may end up being the secret behind our machine learning model.

Problem Statement

(approx. 1 paragraph)

Given the demographic data and personal preferences of a man and woman, can we predict whether they'll be interested in one another romantically?[1]

[1] I'd happily study gay and lesbian couples as well, but our dataset is for a heterosexual speed dating event.

This problem is easily quantified:

- Our **target variable** is binary: “1” if both individuals would like to meet again, and “0” if one or both people decline.
- For **demographic** data, we can use categorical variables for information such as hometown, career, or race. We can also use U.S. census data to predict socioeconomic status based on one’s zip code.
- For **preferences, personality and hobbies**, people can give numerical answers to various statements. For instance: “How ambitious are you, from 1-10?” or “How much do you like movies, from 1-10?”

This is one of my favorite aspects of psychology. Researchers have spent the last century figuring out how to quantify our thoughts and behavior. We could measure an arbitrarily large number of features just by following the template above.

Datasets and Inputs

(approx. 2-3 paragraphs)

I’m using the Speed Dating Experiment dataset available on Kaggle. This is from a study performed by professors Ray Fisman and Sheena Iyengar for their paper Gender Differences in Mate Selection: Evidence From a Speed Dating Experiment. The data was gathered from several speed dating events conducted from 2002 to 2004 and contains 194 features in addition to the “match” target variable. Each row contains a record of one speed dating encounter: the participant’s ID, their partner’s ID, and information about the participant.

Some of the more noteworthy features include:

- Hometown
- Socioeconomic status/income (as predicted by the zip code they’re from)

- Self-rated personality traits of ambitiousness, fun-ness, & sincerity
- Importance of those same traits in a partner
- Self-rated attractiveness
- Importance of attractiveness in a partner
- Planned career path (these are college students, after all)

There are about 8,400 observations in the dataset, so we have plenty of data to learn from.

This is certainly an appropriate dataset to investigate my research question. I've been aware of it for some time, but I've avoided it because the data is messy and complicated. Therefore, it won't just be useful for investigating my question -- it'll also be a fantastic challenge for me to apply my machine learning expertise.

Solution Statement

(approx. 1 paragraph)

The solution will require us to use the features provided in order to obtain predictions for our target variable. I'll test a variety of algorithms, including logistic regression, k-nearest neighbors, naive Bayes, and various ensemble methods (including xgboost).

I expect the ensemble methods to perform the best, but naturally I'll be pragmatic in developing my model. If a simple logistic regression is the most accurate, then there's no reason not to use it! So I'll be sure to use algorithms that work very differently in order to determine which one is best-suited to the problem.

Benchmark Model

(approximately 1-2 paragraphs)

Since human behavior is "messy and complicated" (words I seem to be using a lot!) there are an infinite number of ways we could measure and benchmark a machine learning model. But since there are no agreed-upon human matchmaking algorithms, there are no publicly available benchmarks.

Instead, we'll have to compare the model's performance to a naive predictor. I plan to use sklearn's DummyClassifier to predict a "1" for each record in our dataset. If we were measuring accuracy, this would be a terrible strategy because most pairs didn't experience a love connection. But because we'll be weighting our metric towards recall (see the next section), this will provide an appropriate and useful benchmark.

Evaluation Metrics

(approx. 1-2 paragraphs)

I've given this some thought, and I plan to evaluate my model using the f-beta score with a beta of 1.5. F-beta is a versatile scoring metric that requires both precision and recall to obtain a high score. This means that our naive predictor is expected to perform poorly because it will be unable to identify true negatives.

I'm choosing a beta of 1.5 because I want to slightly favor recall over precision. This means that I want my model to err on the side of false positives rather than false negatives (some of its predicted matches will be incorrect).

Let's put ourselves in the shoes of a prospective dater: When it comes to dating, the cost of a false negative could be missing out on a partner who would be a great match for you. Meanwhile, the cost of a false positive is only a "wasted" evening -- and even then, many daters don't consider a pleasant conversation with someone to be a waste of time. So that's my rationale for selecting a beta of 1.5.

This is how we'll compute it:

$$F_\beta = (1 + \beta^2) \frac{Precision * Recall}{(\beta^2 . Precision) + Recall}$$

(credit: Udacity)

Where $\beta = 1.5$.

Project Design

(approx. 1 page)

I'm primarily going to use a combination of pandas, numpy, and scikit-learn inside a Jupyter notebook to tackle this problem. Here is my intended workflow:

1. Load and clean the data.

After loading the csv file into a dataframe, I'll look over the features and fix any data-entry errors. For instance, some numeric data is formatted as strings, and I need to fix this before proceeding.

Several of the features are a *little* too similar to our target variable and I'm electing to remove them. For instance, participants were asked questions such as:

- What is the likelihood of you calling this person?
- What is the likelihood of this person calling you?
- How much did you like this person?

With information such as that, we can likely predict outcomes with near-perfect accuracy! So removing them will make the project more challenging, and more like a real-world prediction scenario. Having already experimented with this, I plan to use the following code:

```
# Sort features by how correlated they are with the target variable

match_corrs = data.select_dtypes(include=[np.number])\

.corrwith(data['match'])\

.sort_values(ascending=False)


# Take the feature names of the highest correlations

match_corrs = match_corrs[match_corrs > .25].index


# Delete top correlations, but not data['match'] itself

for i in match_corrs[1:]:

       del data[i]
```

2. Dummify the categorical features.

For this step, I could also elect to label-encode these features, but I'm choosing to dummify them just in case I elect to perform a logistic regression. We have several categorical features, so I expect this to greatly increase the dimensionality of our data. So we'll have to deal with that later as well.

3. Fill in the missing values.

The way I typically handle this is I see if I can create a model to predict them using the other features (but not the target variable, because that would be looking into the future). If, through cross-validation, I get an r-squared of above, say, 0.20, I'll let the model fill in the missing values. Otherwise, I'll probably select the mean, median, or mode. Or perhaps I'll fill them in with a value of, say, -100, and let a tree-based algorithm figure out what to do with those data points.

Whatever I end up deciding, I'll be sure to cross-validate and use the strategy that results in the highest final f-beta score.

4. Engineer new features.

In order to obtain the highest f-beta score I can, I'll probably have to engineer some new features from the data. Notably, each row contains information about the participant (demographics, etc.), but not his or her partner. I plan to create several new features that make comparisons between each pair, and I think this will lead to a more effective model.

5. Transform the data

I'll almost certainly need to reduce the dimensionality of the data. I'll perform a principal components analysis (PCA) or feature selection (SelectPercentile), and use the settings that result in the best model performance.

In addition to PCA, I'll experiment with scaling the data using sklearn's MinMaxScaler or StandardScaler.

6. Test different models

I'll try a variety of models on the data, including logistic regression, naive Bayes, K-nearest neighbors, decision trees, and ensemble methods. Due to the complexity of the data, I expect ensemble methods to perform the best. If that's indeed the case, I would feel comfortable using it in a real-world scenario. My biggest issue with ensemble methods is that they can be a bit of a "black box" and it's not always clear how they're arriving at their predictions. But let's say we were using our model to create a dating app. Users wouldn't care "how" we found matches for them; only that the model works. So an ensemble-based learner is perfectly appropriate for this project.

7. Report Findings

I’ll conclude my project by sharing some final statistics on the data: how well the model performs, most important features, and ideas for further improvement.