There is a wide variety of data available, suitable for building both user-based and item-based recommenders.

```
train.csv (user, event, invited, timestamp, interested, not_interested)
test.csv (user, event, invited, timestamp)
users.csv (user_id, locale, birthyear, gender, joinedAt, location,
           hometown, timezone)
events.csv (event_id, user_id, start_time, city, state, zip, country,
           lat, lng, c_1, c_2, ..., c_100, c_other)
user_friends.csv (user, friends)
event_attendees.csv (event, yes, maybe, invited, no)
```

In addition, there are a number of other recommendation factors that can come from the user's social graph. I decided to build up the following recommenders:

- User based recommender - uses the preferences of other users who have a preference for the same event and their similarity to this user to compute a user_reco score.
- Item based recommender - there are actually two of these. Both use the preference the user has expressed for other events and the similarity between that event and this one. There are two scores generated for the different measures of event similarity, one based on event metadata and one based on event content. These two recommenders return the evt_p_score and evt_c_score respectively.
- User popularity - measured by the number of friends a user has. The idea here is that people with more friends are more outgoing, and hence are more likely to attend events. This generates the user_pop score.
- Friend Influence - the idea here is that if your friends are going to an event, you are too. The score measures the number of your friends that are going to this event, and generates the frnd_infl score.
- Event Popularity - the more popular an event is, measured by the people that are going to it, the more likely the user will go to it. This produces the event_pop score.

These features replace the original train.csv and test.csv files, so now they look like this:

```
train.csv(invited, user_reco, evt_p_reco, evt_c_reco, user_pop,
          frnd_infl, evt_pop, interested, not_interested)
test.csv(invited, user_reco, evt_p_reco, evt_c_reco, user_pop,
          frnd_infl, evt_pop)
```

Now I train a Stochastic Gradient Descent classifier from Scikits-Learn with my modified training set and build a one-vs-all classifier model to predict the value of the "interested" outcome.

Running a 10-fold cross validation yields an accuracy number of 0.676043972845. Running the classifier against the modified test set yields another temporary file. The user and event columns come from the original test.csv and the outcome and dist are the predicted outcome from the classifier and the distance of the actual point from the predicted hyperplane. So for (user, event) pairs with a predicted outcome of 1, higher values of dist imply a better match.