Project 6

APIs + Random Forests

Project Summary

- Hired by Netflix to examine what factors lead to certain ratings for movies
- Use IMDB to predict these factors
- Give Netflix a detailed report of findings with recommendations for the next steps
- Mission : Collect the data and construct a random forest to understand what factors contribute to ratings.

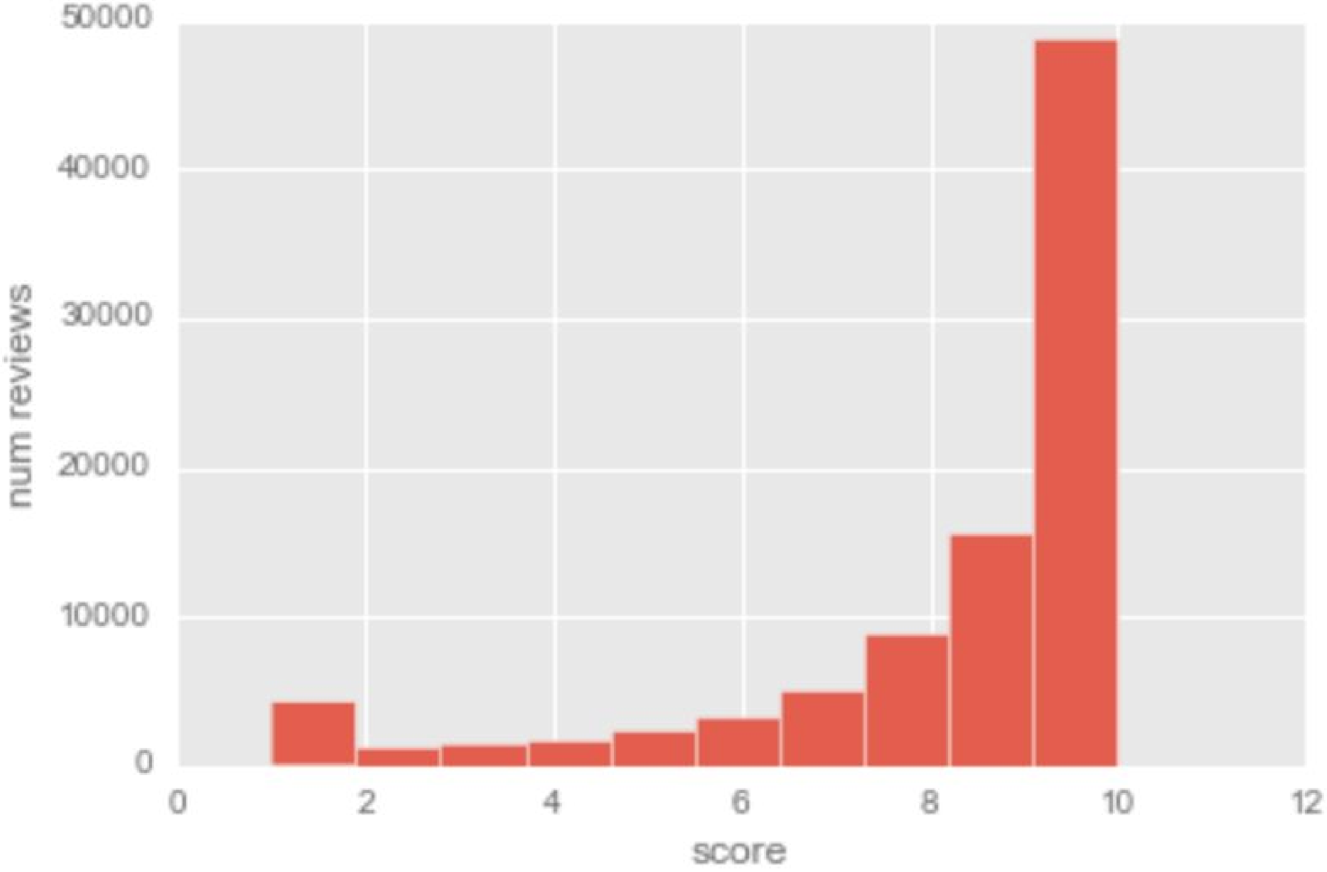
WHAT I DID

- Imported movies from IMDB API and scrapped the reviews
- Created two CSV files: reviews and top 100 movies
- Imported and Combined tables on Postgres
- Data Visualisation
- Built several models to predict the numeric rating of the review

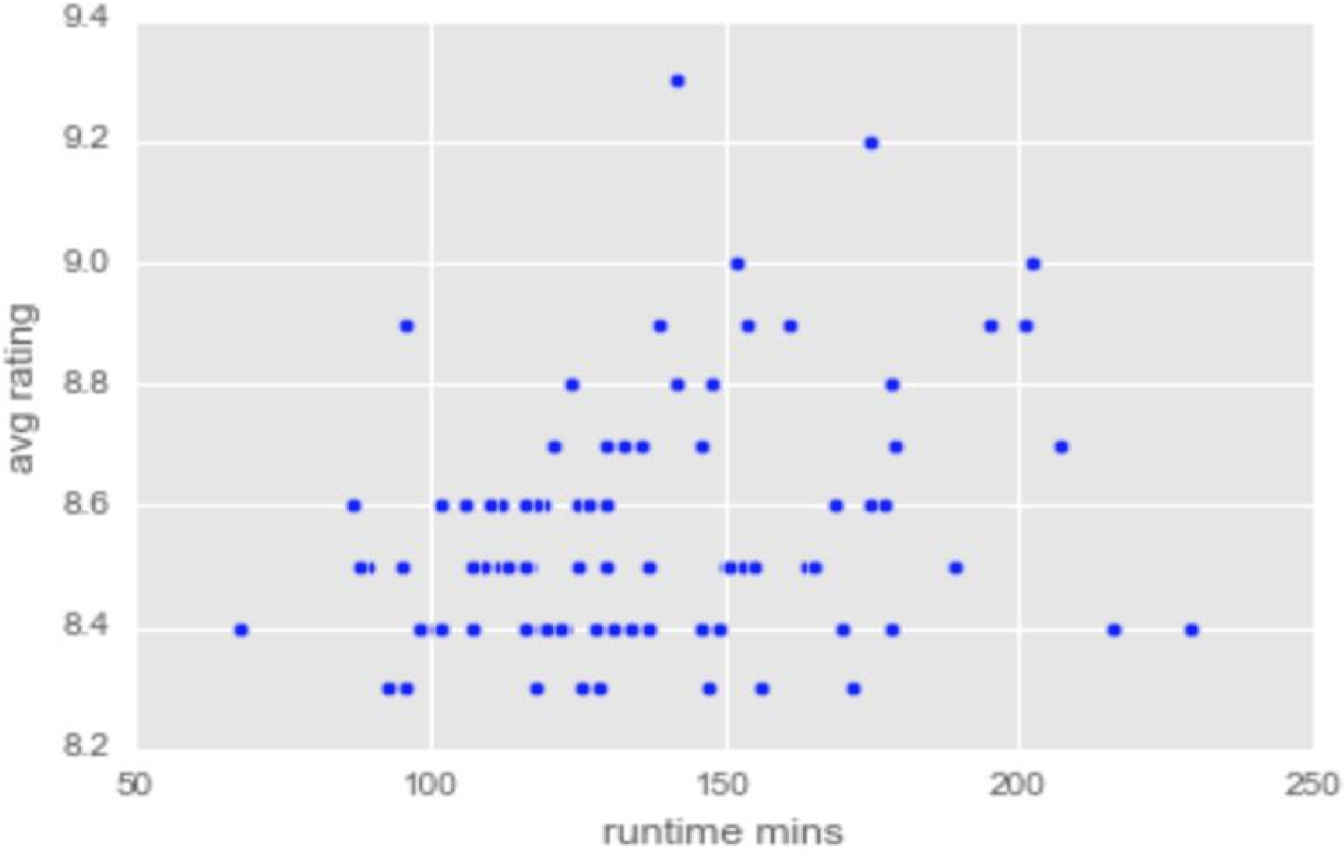
Problems

Weakness with the dataset:

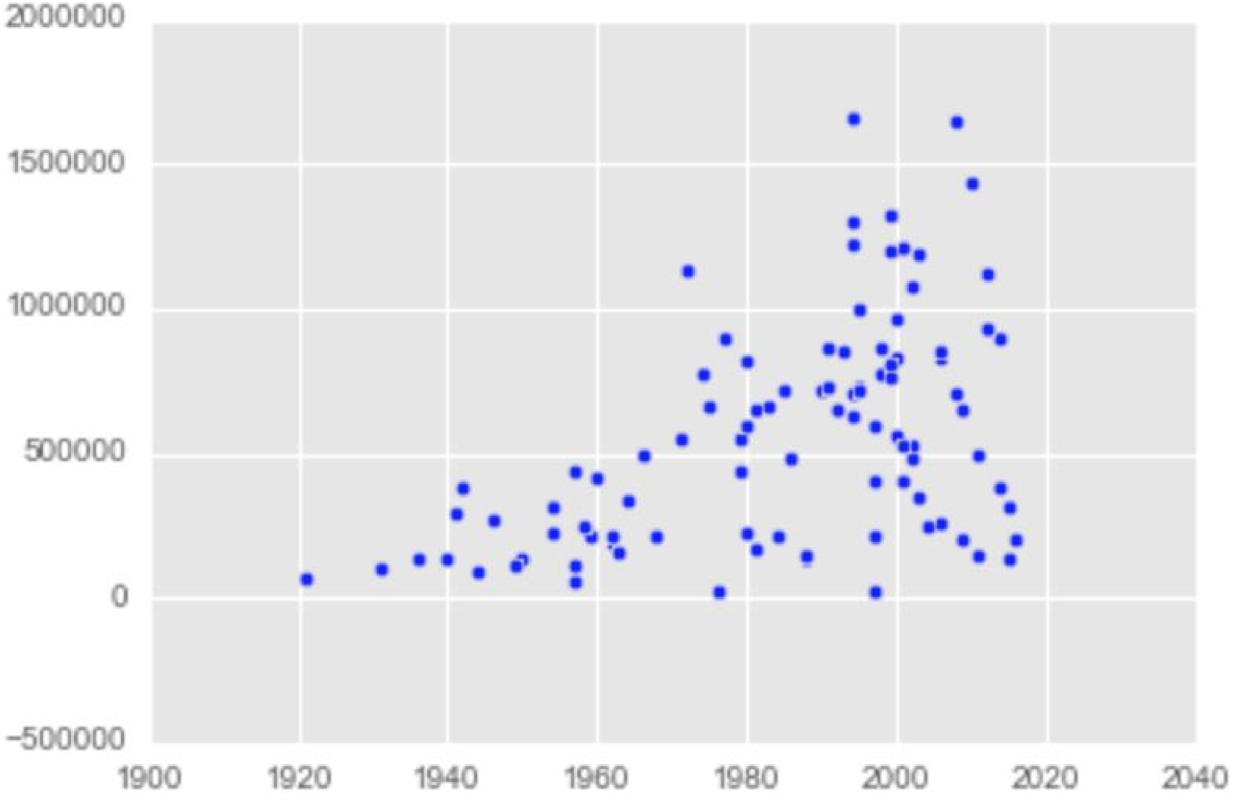
- Distribution is highly skewed towards high ratings
- Average rating of the data was 8.44 (on a scale of 1-10)
- 25th percentile had a rating of 8
- 50th percentile had a rating of 10
- Decision Trees perform poorly



Distribution of ratings



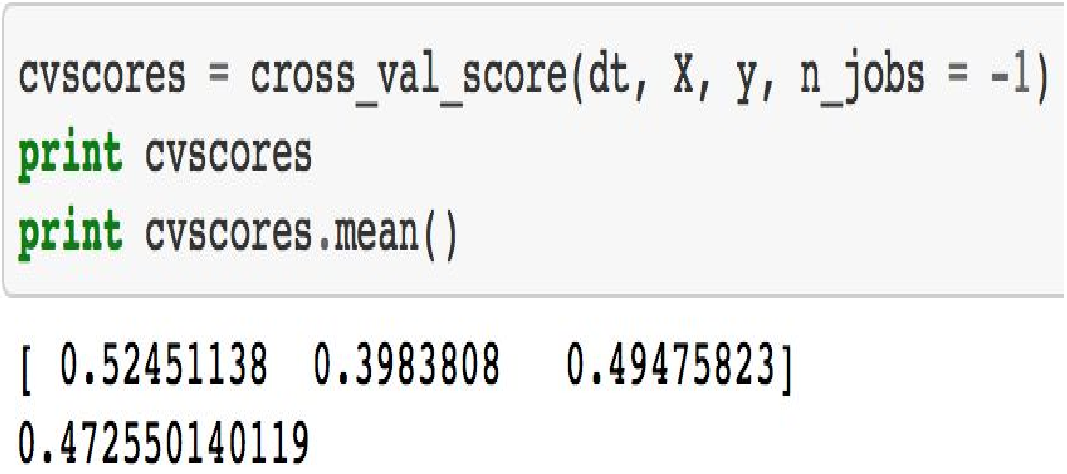
Avg. rating vs. runtime



Num of Votes VS Year

- Few votes for older movies?

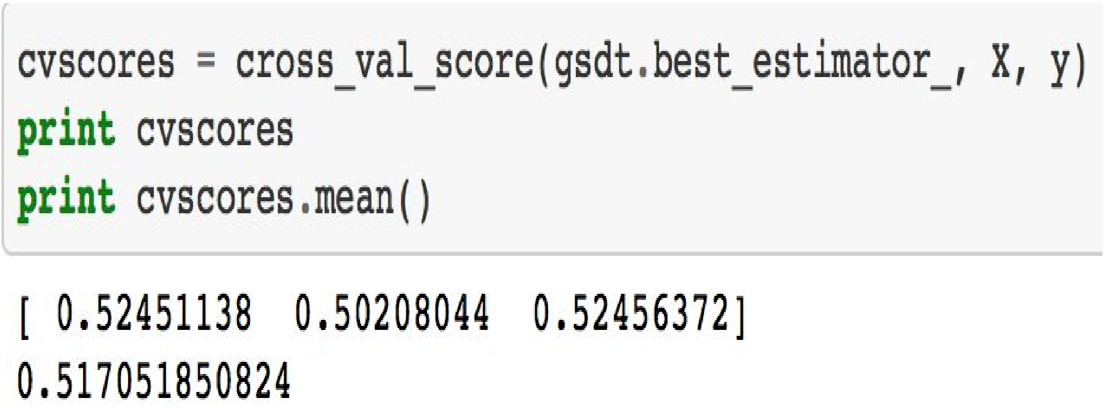
Built and Cross validated scores



```
cvscores = cross_val_score(dt, X, y, n_jobs = -1)
print cvscores
print cvscores.mean()
```

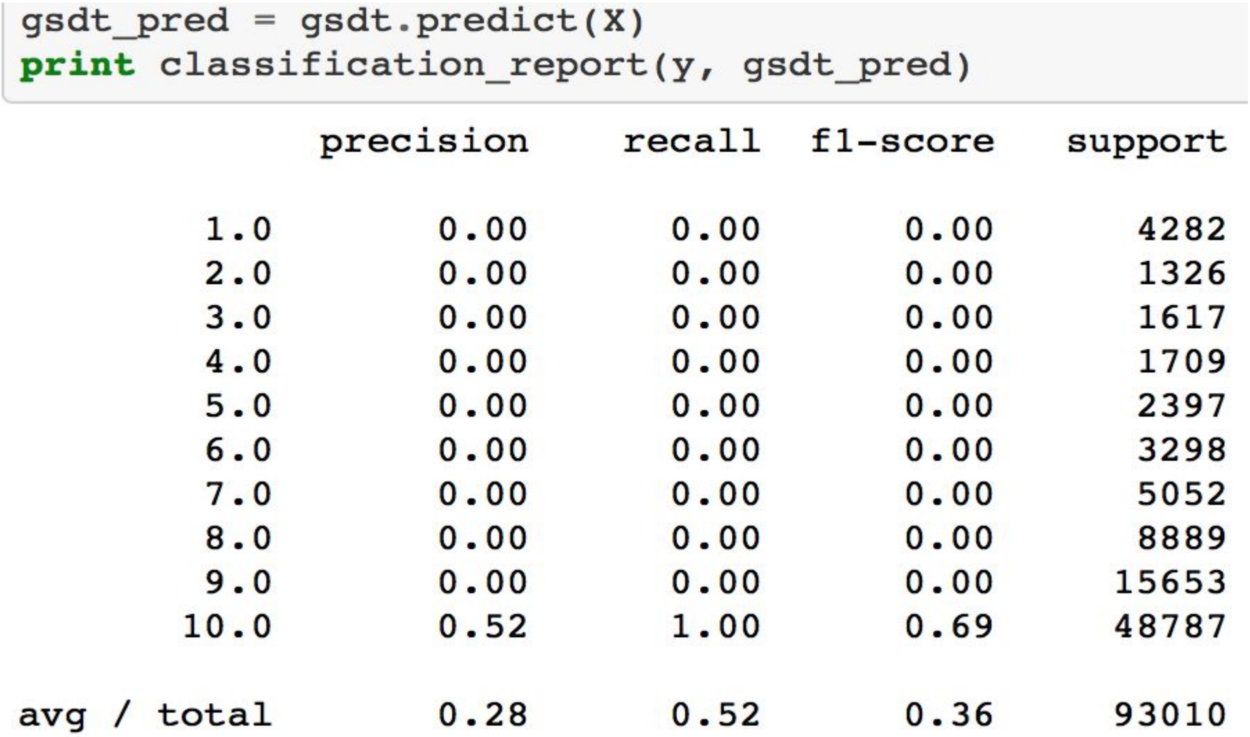
```
[ 0.52451138  0.3983808   0.49475823]
0.472550140119
```

Grid Search Results



```
cvscores = cross_val_score(gsdt.best_estimator_, X, y)
print cvscores
print cvscores.mean()
```

```
[ 0.52451138  0.50208044  0.52456372]
0.517051850824
```



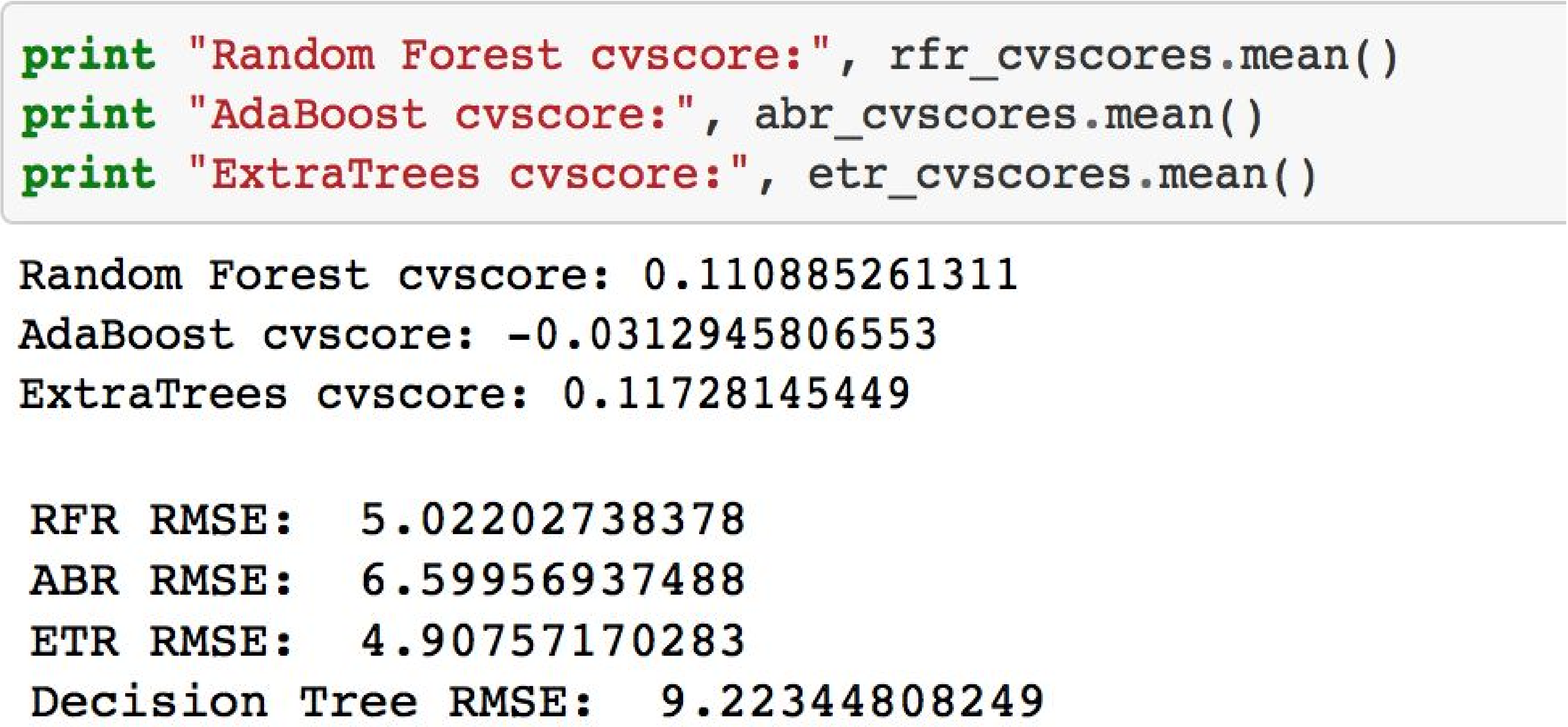
```
gsdt_pred = gsdt.predict(X)
print classification_report(y, gsdt_pred)
```

```
             precision    recall  f1-score   support

        1.0       0.00      0.00      0.00      4282
        2.0       0.00      0.00      0.00      1326
        3.0       0.00      0.00      0.00      1617
        4.0       0.00      0.00      0.00      1709
        5.0       0.00      0.00      0.00      2397
        6.0       0.00      0.00      0.00      3298
        7.0       0.00      0.00      0.00      5052
        8.0       0.00      0.00      0.00      8889
        9.0       0.00      0.00      0.00     15653
       10.0       0.52      1.00      0.69     48787

avg / total       0.28      0.52      0.36     93010
```

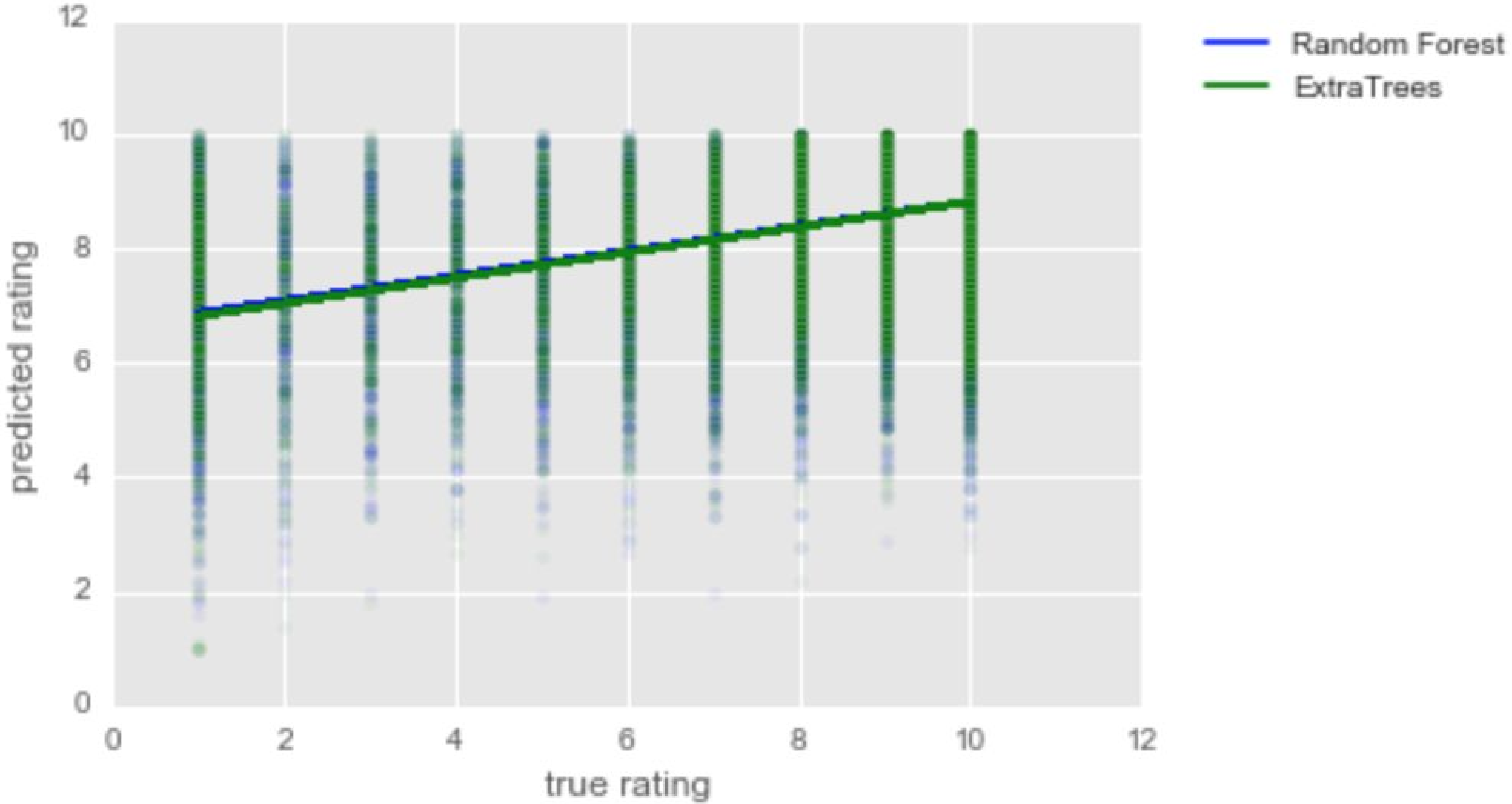
The mean cv score improved with this model, but the model is just classifying every single review as a rating of 10, which is not very helpful!

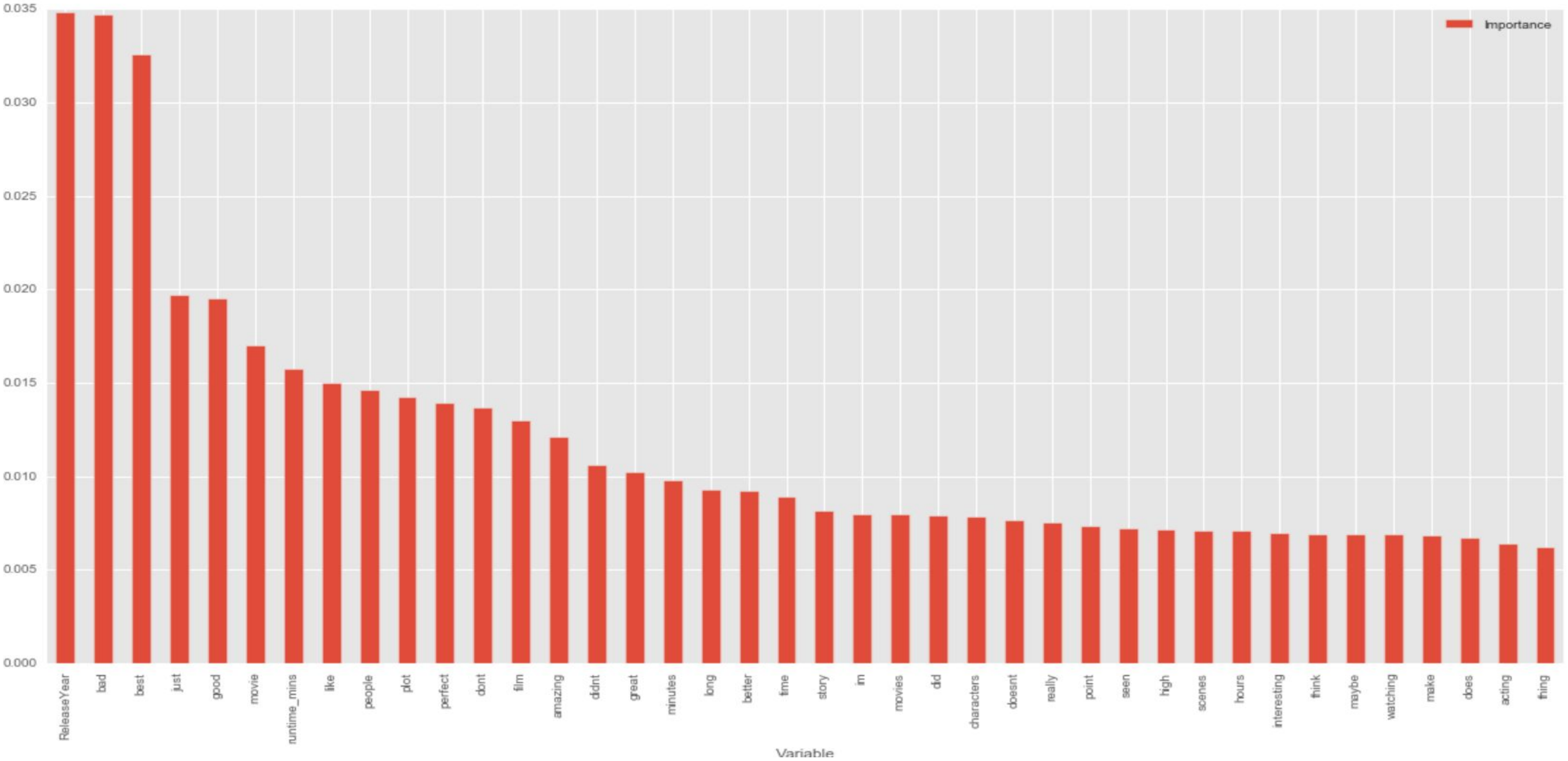


```
print "Random Forest cvscore:", rfr_cvscores.mean()
print "AdaBoost cvscore:", abr_cvscores.mean()
print "ExtraTrees cvscore:", etr_cvscores.mean()
```

```
Random Forest cvscore: 0.110885261311
AdaBoost cvscore: -0.0312945806553
ExtraTrees cvscore: 0.11728145449
```

```
RFR RMSE:  5.02202738378
ABR RMSE:  6.59956937488
ETR RMSE:  4.90757170283
Decision Tree RMSE:  9.22344808249
```


Random Forest
ExtraTrees
predicted rating
true rating
0
2
4
6
8
10
12
0
2
4
6
8
10
12



Feature Importances from Random Forest

Findings

- Random Forests Regressor and Extra Trees Regressor performed the best
- Random Forest RMSE = 5.0
- Extra Trees RSME = 4.91
- Models tended to overestimate the ratings more than underestimate
    - Not effective at predicting low ratings
    - High skew of the data set

Next Step

To get a better predictive model for a review:

- Use data with evenly distributed reviews across a sampling of movies with varying average ratings
- Maybe use Random Forests and Extra Trees since they performed the best out of the models as a place to start