IMDB Rating

In this project we were asked to make a rating system for netflix based on imdb database. They don't want to use imdb rating so I decided to categorize them as good or bad!!!
Good movies have rating of higher than 7.0 and the rest is categorized as not good. We were asked to use `'imdpie'` as a wrapper for imdb api but since it was slow and getting complex data was impossible through it I decided to download all IMDB data which eventually banned me from GA's network!
IMDB has its own ftp server which has all their data as tab separated text file which we can download them with the following command:

```
Lftp -c "open ftp://ftp.fu-berlin.de/pub/misc/movies/database/ && find . -name '*.list.gz' && exit > listing.txt
```

And then we can use the following command to download them all:

```
wget -i file-with-list-of-urls.txt
```

After that I used another imdb wrapper called `'IMDBPY'` to convert all those files to mysql database.

```
imdbpy2sql.py -d /dir/with/plainTextDataFiles/ -u mysql://user:password@host/database
```

And now we have it all.
There are around 4 million movies in that database and since it is too big and we don't need them all I'm going to use movies produced after 2010.
I extracted actors, directors, writers, budget, genres, gross, mpaa rating, movie plots, vote counts and ratings of movies from database. There are so much more data there but that would take a lot of time extracting and using them.
I used decision tree, random forest and extra tree classifiers to categorize movies and also find best features. In my algorithm I split data to train and test and found the best model based on train data and then ran my test data through those model to find the best results with lowest overfitting.

Decision Tree

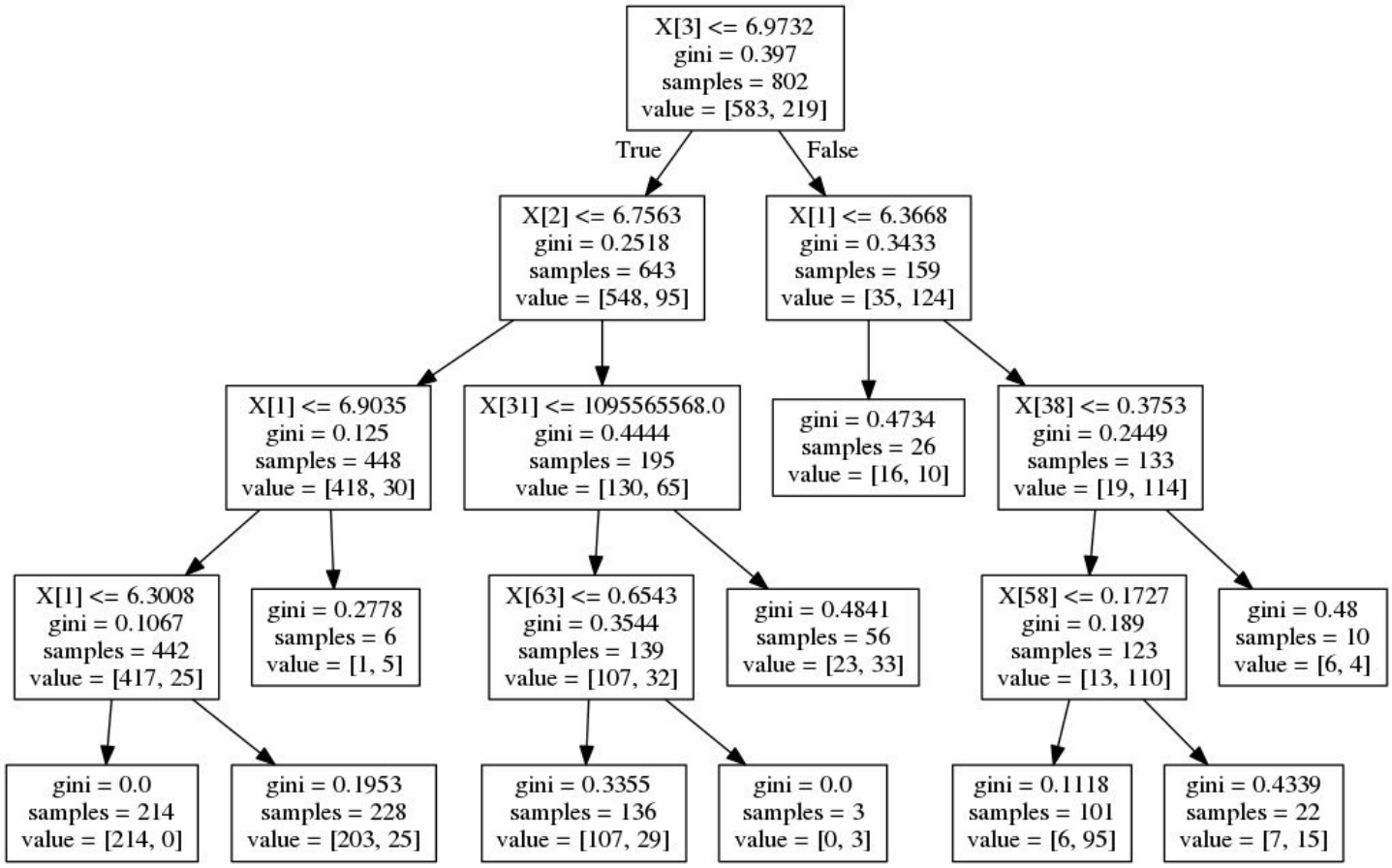
These are the results of decision tree:

```
Best Model after Grid Search:
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=4,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=100, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
Mean score of the model is: 0.821820545514
Confusion Matrix:
[[136  10]
 [ 24  31]]
Classification Report:
             precision    recall  f1-score   support

          0       0.85      0.93      0.89       146
          1       0.76      0.56      0.65        55

avg / total       0.82      0.83      0.82       201
```

As we can see score is high and we have decent results in precision/recall too.
This is the tree based on best model:



Random Forest

These are the results of using random forest as classifier:

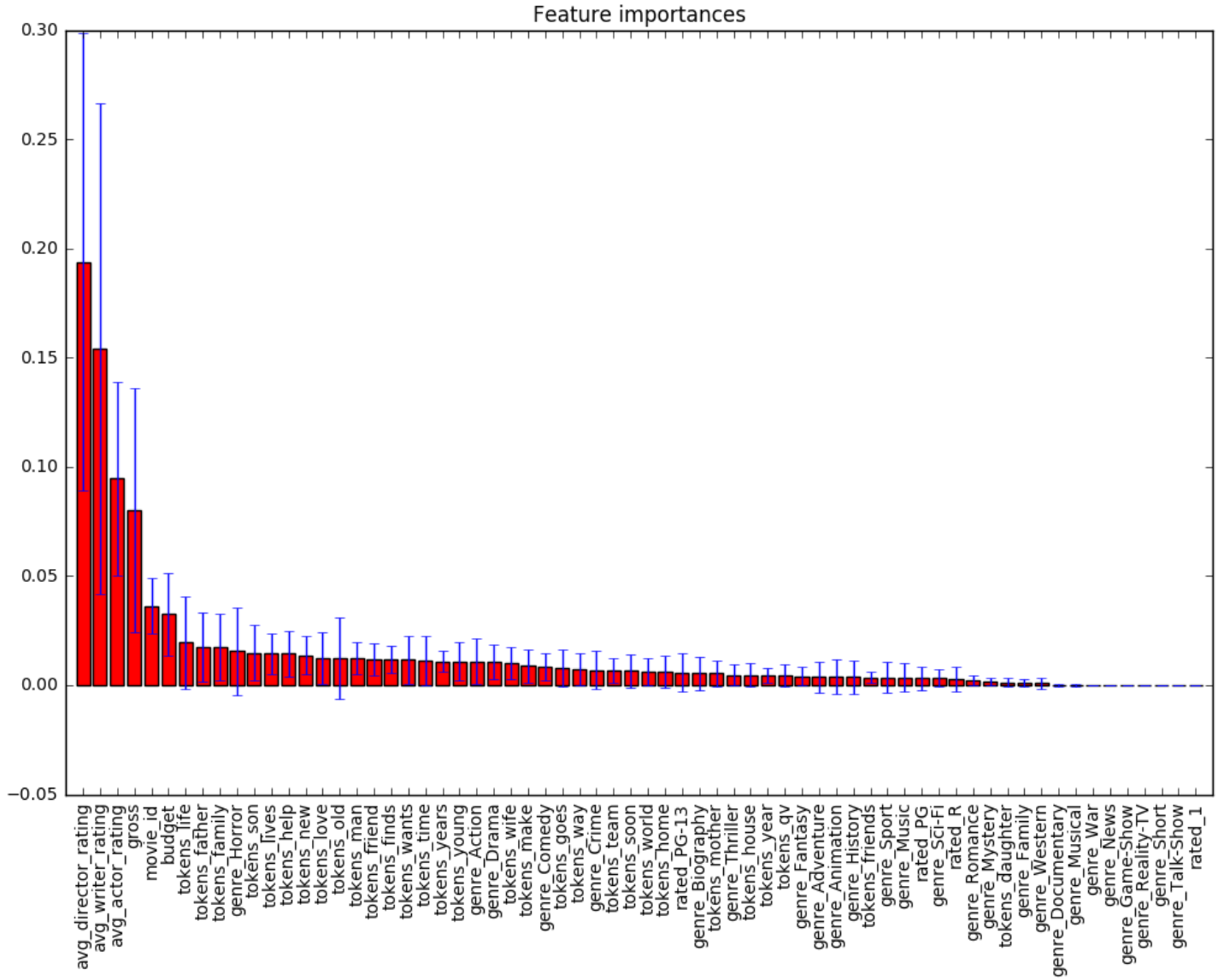
```
Mean score of the model is: 0.806995174879
Confusion Matrix:
[[140   6]
 [ 24  31]]

Classification Report:
             precision    recall  f1-score   support

          0       0.85      0.96      0.90       146
          1       0.84      0.56      0.67        55

avg / total       0.85      0.85      0.84       201
```

As we can see precision and recall has been improved which suggests lower overfitting. Also we can see feature selection as below:



As we can see i made a mistake and did not remove ‘movie_id’ but except that important features are pretty much as we would’ve guessed from a good movie.

Extra Trees

We can also use extra trees:

```
Best Model after Grid Search:
ExtraTreesClassifier(bootstrap=False, class_weight='balanced',
           criterion='gini', max_depth=None, max_features='auto',
           max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=20, n_jobs=-1,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
Mean score of the model is: 0.803695092377
Confusion Matrix:
[[140   6]
 [ 31  24]]
Classification Report:
             precision    recall  f1-score   support

          0       0.82      0.96      0.88       146
          1       0.80      0.44      0.56        55

avg / total       0.81      0.82      0.80       201
```

Score and precision recall haven't improved but they are acceptable. And here are best features: