598: Machine Learning Project

Application of Classification Methods on Spooftify Data

Antonio Campbell, Sinta Sulistyo, Atta Ullah, Penny Wu

4/30/2021

1 Introduction

In this project we have considered a dataset of 2017 songs from Spotify. This data was provided by Spotify and posted by user GeorgeMcIntire on Kaggle ("https://www.kaggle.com/geomack/spotifyclassification"). Within the data, each song has 16 features. The feature of interest for classification is called `target` which indicates whether the creator of the dataset liked or disliked a song. A song is labeled "1" if it is liked and "0" when it is disliked. The other features include `acousticness`, `danceability`, `durationMs (duration in milliseconds)`, `energy`, `instrumentalness`, `key`, `liveness`, `loudness`, `mode`, `speechiness`, `tempo`, `time-signature`, `valence`, `songname`, `artist`.

The goal of the project is to build several classifiers for prediction that is based on the rest of the features to determine whether the individual would like a song. We begin by preparing the data to fit possible models appropriately.

2 Data exploration and feature selection:

Data Sample

X	acousticness	...	target	song_title	artist
0	0.0102	...	1	Mask Off	Future
1	0.199	...	1	Redbone	Childish Gambino
2	0.0344	...	1	Xanny Family	Future
3	0.604	...	1	Master Of None	Beach House
4	0.18	...	1	Parallel Lines	Junior Boys
5	0.00479	...	1	Sneakin'	Drake
⋮	⋮	⋮	⋮	⋮	⋮
2011	0.000586	...	0	Brightside - Borgeous Remix	Icona Pop
2012	0.00106	...	0	Like A Bitch - Kill The Noise Remix	Kill The Noise
2013	0.0877	...	0	Candy	Dillon Francis
2014	0.00857	...	0	Habit - Dack Janiels & Wenzday Remix	Rain Man
2015	0.00164	...	0	First Contact	Twin Moons
2016	0.00281	...	0	I Wanna Get Better	Bleachers

Data Preparation

We ensure the data is adequate for fitting before we begin modeling. In particular we are looking for features which have missing values, the songs that are duplicated in the dataset, the types of features (if a feature is numerical or categorical), and as well as the variables of importance for the fit.

Data Dimensions

	rows	columns
count	2017	17

We drop the first column, which is just for indexing and has no use to us here. The data has the remaining features

```
##  [1] "acousticness"     "danceability"     "duration_ms"      "energy"
##  [5] "instrumentalness" "key"              "liveness"         "loudness"
##  [9] "mode"             "speechiness"      "tempo"            "time_signature"
## [13] "valence"          "target"           "song_title"       "artist"
```

Next we check if there are any missing values within the data and we find that there are 0 records with missing data in them; however, there are 5 duplicate records:

	acousticness	danceability	...	song_title	artist
268	0.096200	0.654	...	River	Ibeyi
509	0.024600	0.586	...	Her Fantasy	Matthew Dear
895	0.000334	0.907	...	Jack	Breach
928	0.934000	0.440	...	Episode I - Duel of The Fates	John Williams
982	0.036900	0.448	...	Myth	Beach House

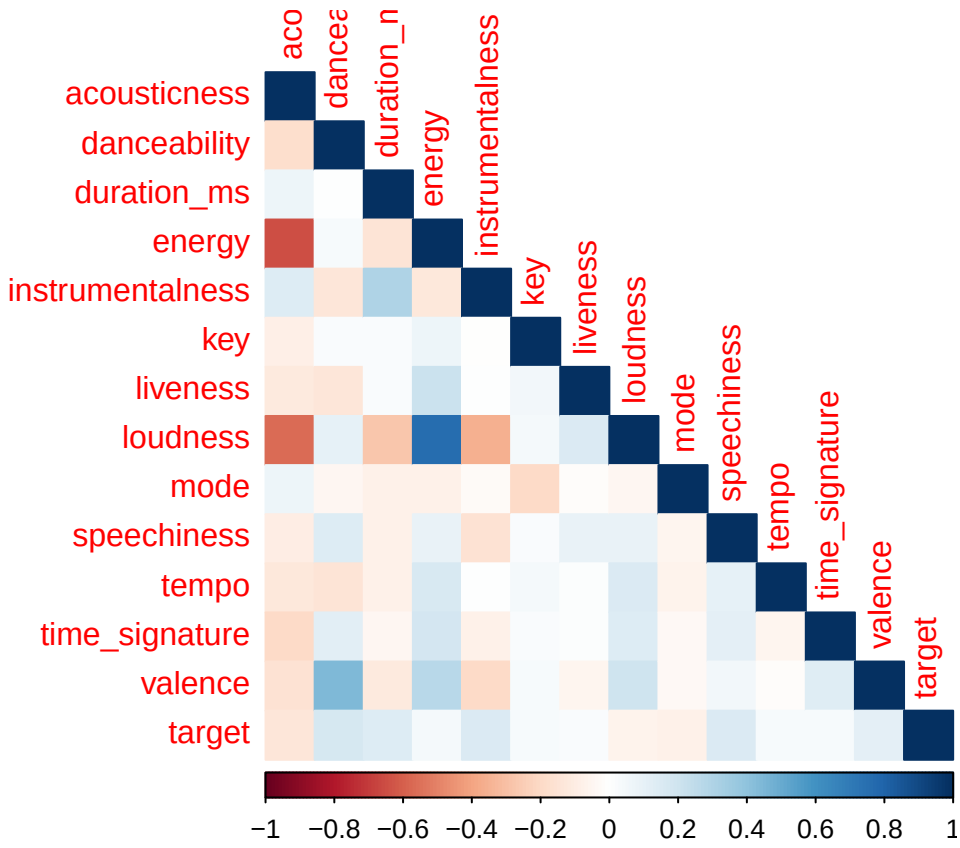
We remove the 5 remaining data points and the data that remains has the following dimensions,

Data Dimensions

	rows	columns
count	2012	16

Correlation with Target (Correlation Matrix Heatmap):

We include a correlation heat plot. For this analysis we have droped the last two features 'song_title' and 'artist_name', which we do not use in our final analysis as well as those previously dropped.



The results of this plot tells us that `instrumentalness`, `danceability`, `speechiness`, and `acousticness` are among the most correlated with the the target variable.

Correlation of other feautre with 'target':

```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```

3 Features selection based on correlations with 'target':

acousticness

danceability

duration_ms

instrumentalness

speechiness

valence

```
##   target acousticness danceability duration_ms instrumentalness speechiness
## 1      1      0.01020        0.833      204600         0.021900      0.4310
## 2      1      0.19900        0.743      326933         0.006110      0.0794
## 3      1      0.03440        0.838      185707         0.000234      0.2890
## 4      1      0.60400        0.494      199413         0.510000      0.0261
```

```
## 5      1      0.18000        0.678      392893         0.512000      0.0694
## 6      1      0.00479        0.804      251333         0.000000      0.1850
##   valence
## 1   0.286
## 2   0.588
## 3   0.173
## 4   0.230
## 5   0.904
## 6   0.264
```

```
## [1] 7
```

4 Models

The potential methods to build a calcification model for this project include:

Logistic Regression (Penny)

Naive Bayes (Sinta)

Decision Trees (Atta)

Random Forests (Sinta)

Bossting (Penny)

Test Train split

```
## dimension of train data: 1509 7
```

```
## dimension of test data: 503 7
```

```
##    predict_unseen
##       0   1
##   0 166  95
##   1  64 178
```

```
## [1] "Accuracy for test data is:  0.68389662027833"
```

Neural Nets

We begin with the simplest model we can fit based on the three main predictors that we have used for the previous models, `danceability`, `speachiness`, `acousticness`. We will use a single-layer neural network to fit on the training data. For optimization we will begin by using a grid search accross of layer size. We also want to introduce regularization through weight decay so we will include those in our grid as well. The values traversed through are:

Size Grid

5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100

Decay Grid

1e-04	0.001	0.0025	0.005	0.0075	0.01	0.025	0.05	0.075	0.1	0.25	0.5

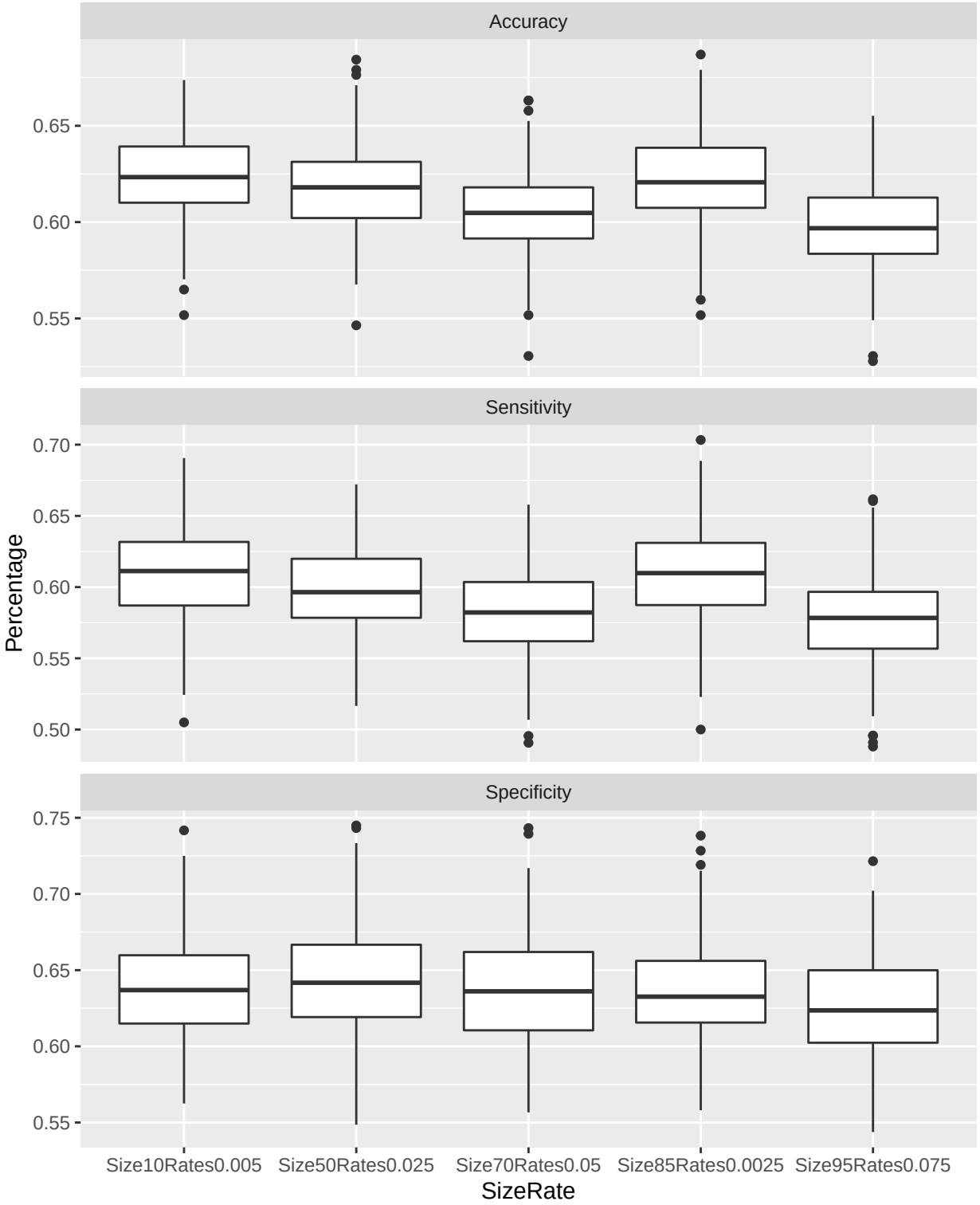
For an initial grid optimization based on the expansion of the grid over sizes/decay rates, we hold out 25% of the data training data for a validation set and compare the methods based on accuracy after fitting on what remains in the training set. This gives us a basis of what parameters will work well, but we will see that these can be unrelaible and very biased to the data. We sort the fits based on their accuracy, and then use the following candidates as parameters from the following fits for comparison: the first two best fits, the 25th/50th/75th percentiles, and the worst fit. These are:

Various Parameters Considered

size	decay	acc	sens	spec
75	0.005	0.6206897	0.5471698	0.7151515
65	0.005	0.5941645	0.5227273	0.6942675
75	0.025	0.5782493	0.5086957	0.6870748
60	0.500	0.5596817	0.4923858	0.6333333
85	0.075	0.5411141	0.4789916	0.6474820

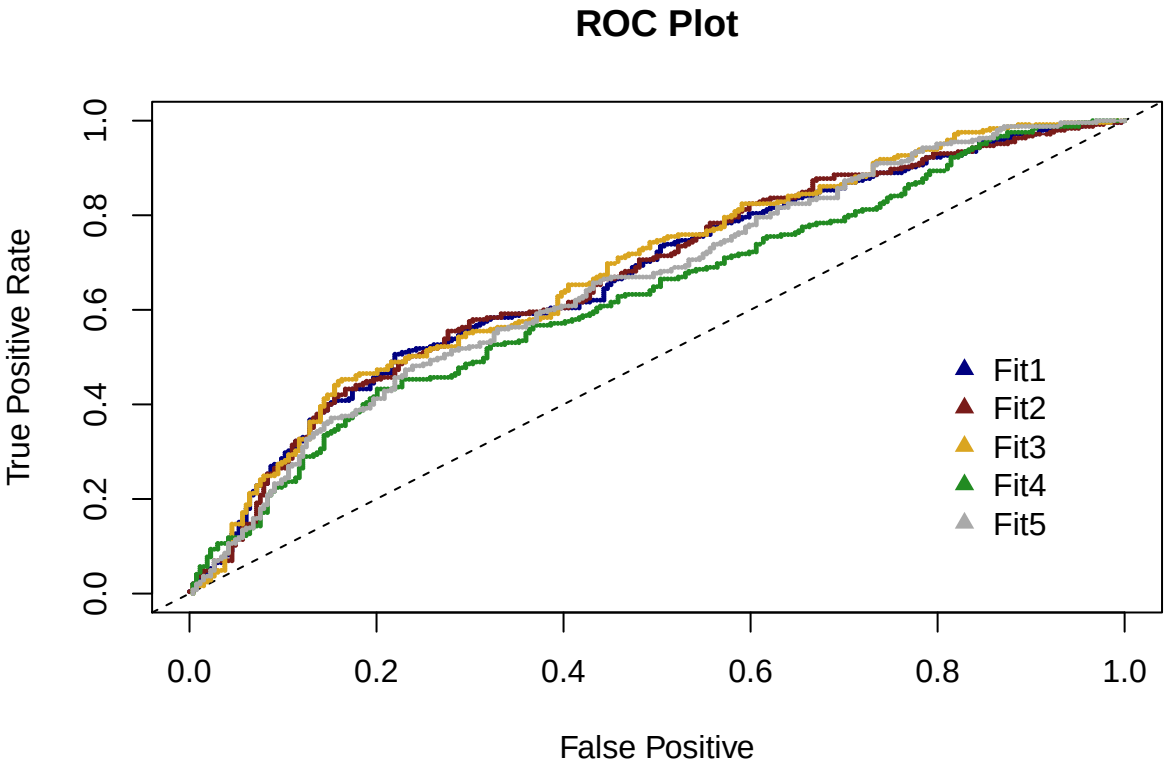
The following box plots show how these parameters fair against eachother over various random splits of the train/validation splits based on accuracy, specificity, and sensitiviy. We iterate serveral times to obtain these distributions.

```
## Using SizeRate as id variables
```



Over 250 interations we have a comparison of these methods, and we see that what we though would provide parameters for the best fit do not always appear to fit just as well. While our prediction problem is not a matter of life or death, we conside measures like specificity/sensitivity can be used to choose the parameters. Intuitively, It may be the case that we can identify who likes quite a bit of music in general, and we are more concerned with classifying the songs that they wouldn't like, rather than they would. Then it would be logical to include specificity in our search for good parameters. The oppositive can be said about people who

are very picky about music. We include an ROC plot of the fit models based on their fits.



It appears that predictions for most methods look apporximately the same. The only method that falls completely off from the mark is the very last one. We do see regions where our supposed 'best fit' from our initial serach would not have the best accuracy, but the difference isn't extreme.

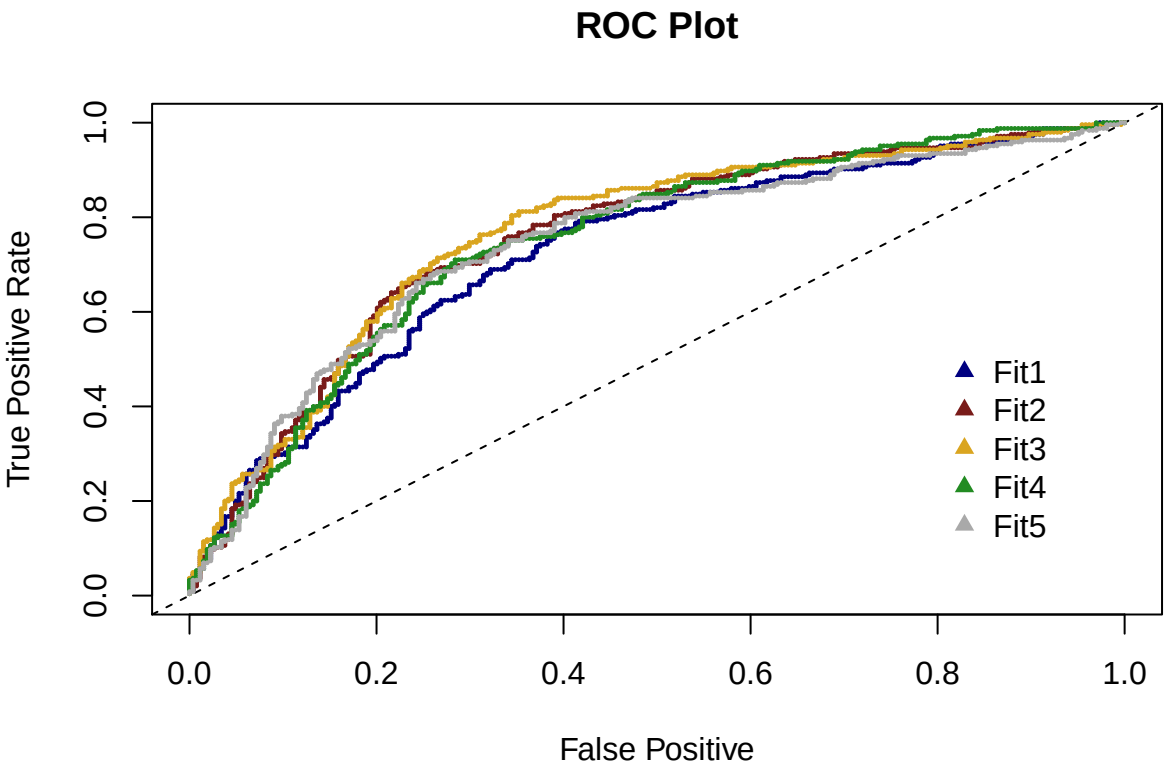
Test Accuracy

	size	decay	accuracy
135	75	0.025	0.6222664
177	85	0.075	0.6023857
73	65	0.005	0.6003976
75	75	0.005	0.5964215
232	60	0.500	0.5626243

Now we consider how neural networks compare when we use all of the predictors rather than just the three we originally looked at. We fit on various parameters again and compare different candidates based on their accuracy between random train/validation splits witihin the allocated training data. We first noticed that the fits improved immediately by including other predictors, but the same problem arose where we overfit to the data and even though our validation set pointed us in the direction of the top parameters, they weren't great when testing. Just choosing the worst, best, and candidates in between so we tried something else. Instead of just picking candidates only from an initial train/validation split, we start with our initial fits remove a fourth of the candidates iteratively by sorting the carndidates on accuracy on the current validation set, we split the training data once again and add the new accuracy too the current after refitting on this split. We repeat the proceedure until only 5 parameter combinations are left and then we average over the values of accuracy and sort once again for our choice of the top 5 models. The following are the results we ended up with for candidates and their ROC curves. We also include parameters chosen at th percentiles of accuracy again from the first train/validatioon split.

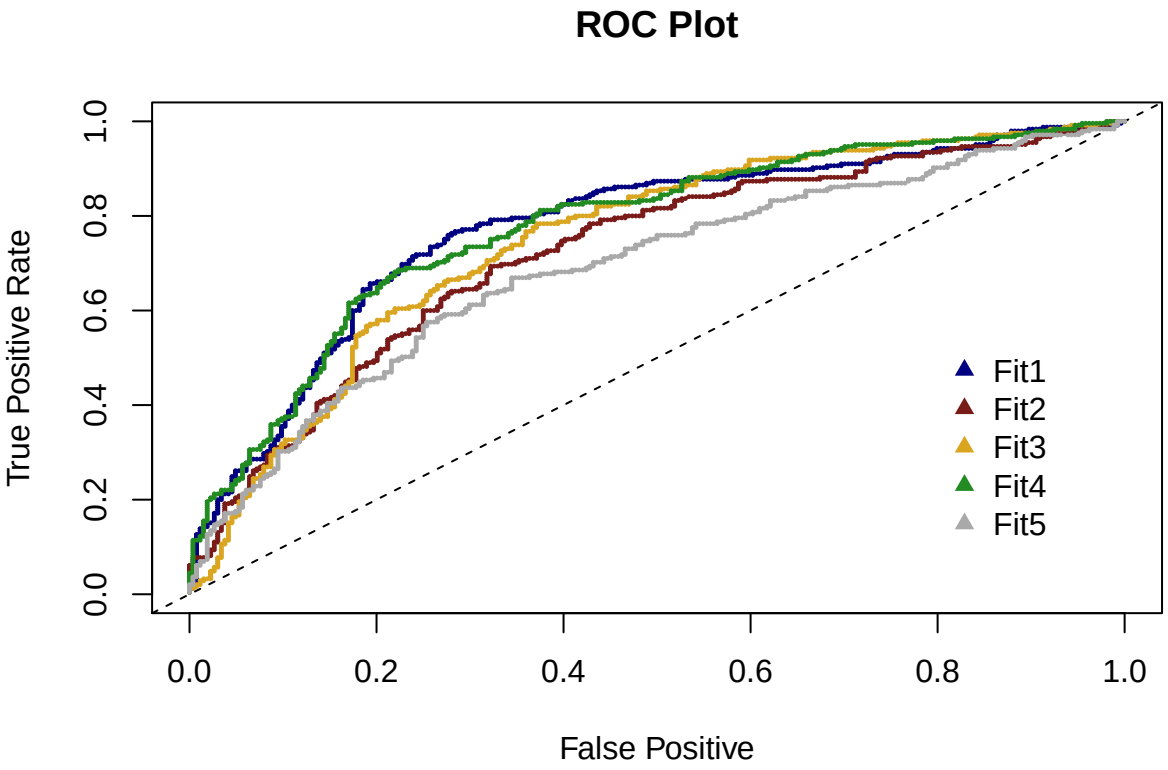
	size	decay	acc	sens	spec
109	45	0.010	0.7206012	0.6925296	0.7541079
137	85	0.025	0.7190097	0.6863147	0.7595031
132	60	0.025	0.7175950	0.6853846	0.7568818
102	10	0.010	0.7167109	0.6894583	0.7488452
111	55	0.010	0.7179487	0.6882824	0.7531451

Using candidates at the same positions before



Test Accuracy

	size	decay	accuracy
137	85	0.025	0.7137177
111	55	0.010	0.7117296
132	60	0.025	0.7077535
102	10	0.010	0.7037773
109	45	0.010	0.6759443



Test Accuracy

	size	decay	accuracy
117	85	0.0100	0.7316103
188	40	0.1000	0.7256461
49	45	0.0025	0.6938370
93	65	0.0075	0.6679920
240	100	0.5000	0.6540755

Deep Neural Nets (Antonio)

valence and possibly instrumentalness are not important

play with 2-3 layers with smallest ammount of predictors. If worse than 1 layer neural net stop there.

Summary oof metrics of thre

5 Conclusions