Comparing ML Models in Predicting the Success of a joke by Comedy Robot

By Tobias Kohn

For Naomi Fitter, SHARE Lab

Processing the Data

Feature engineering, or creating features out of preexisting ones is a great way to uncover patterns in the data that may not be linear, or may not be easy to identify.

```
df['intensity_range'] = df['intensity_max'] - df['intensity_min']
```

```
df['intensity_total'] = df['length'] * df['intensity_mean']
```

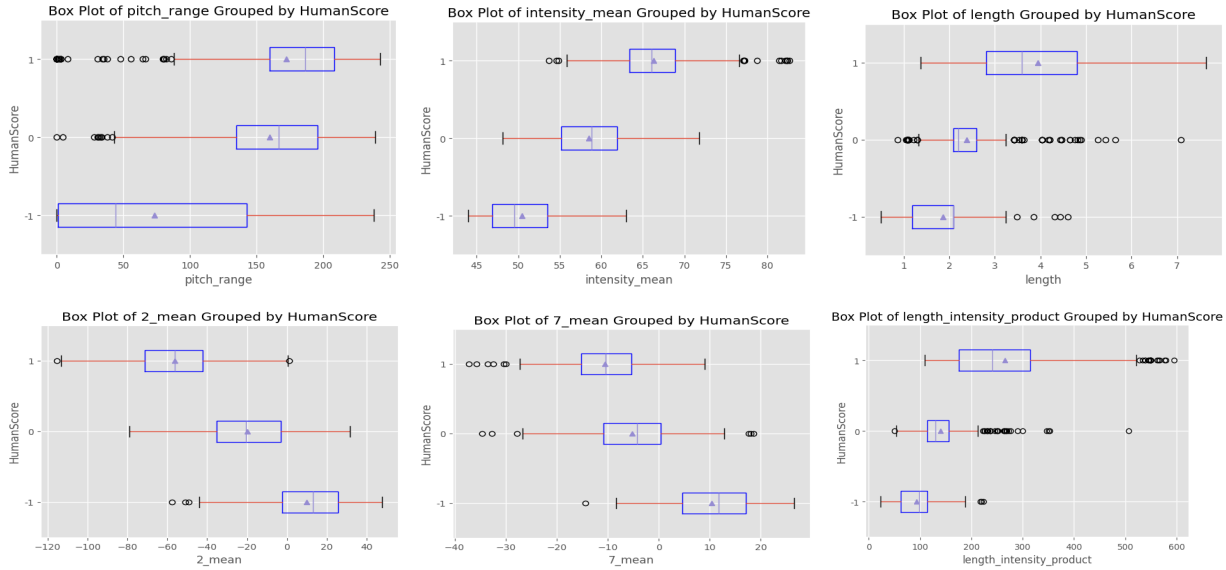
I created a range feature for every feature with a min and max value, such as intensity, pitch, and the MFCC measurements. Additionally, we calculated the total intensity.

```
df['JokeMean'] = df['JokeId'].map(df.groupby('JokeId')['HumanScore'].mean())
df['PerfMean'] = df['PerformanceId'].map(df.groupby('PerformanceId')['HumanScore'].mean())
df['WeightedHumanScore_j'] = df['HumanScore'] * df['JokeMean']
df['WeightedHumanScore_p'] = (df['HumanScore'] * df['PerfMean'] *
    df.groupby('JokeId')['HumanScore'].transform('count'))
```

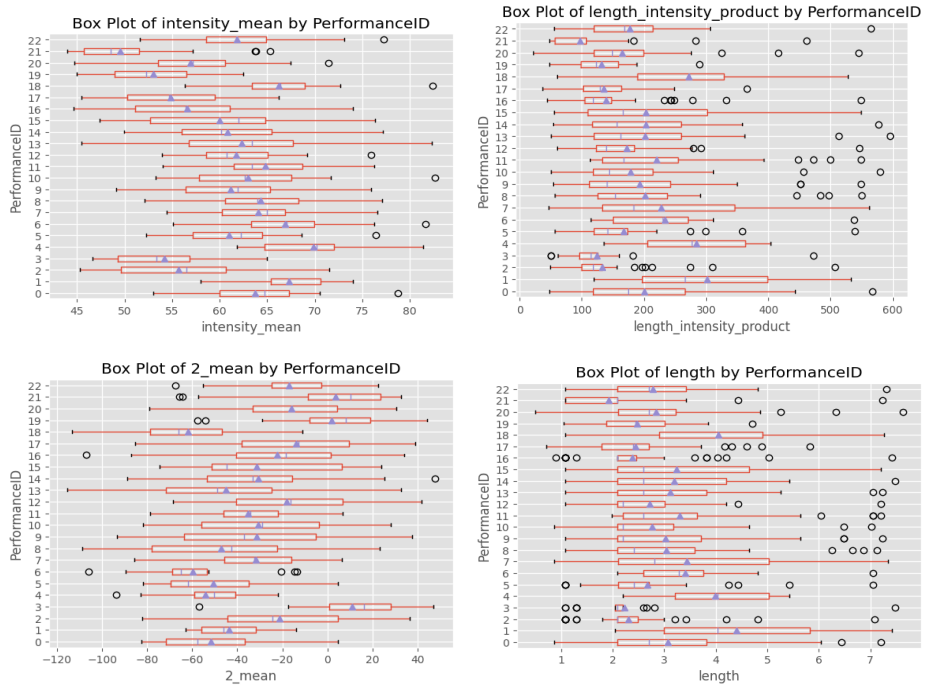
Additionally, we measured the mean HumanScore per performance and per joke, which will help us analyze biases in audiences and objective funniness of a joke. We use these to calculate the weighted HumanScore based on how a joke performed on average, which will help us understand biases in the audiences at each performance, and a weighted HumanScore based on the bias of an audience and the times a joke was told, which will help us understand which jokes are objectively better.

Analyzing Performances

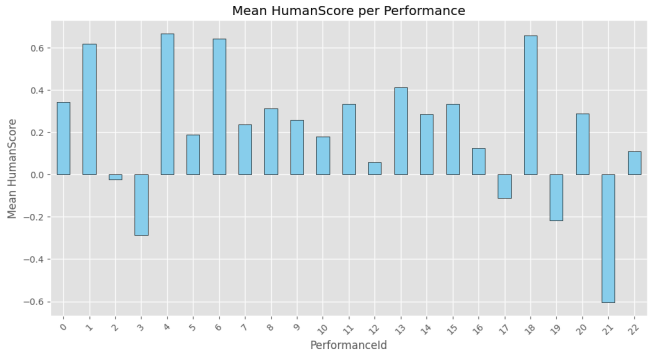
I created Box-Plots to visualize the relationship between a feature and HumanScore.



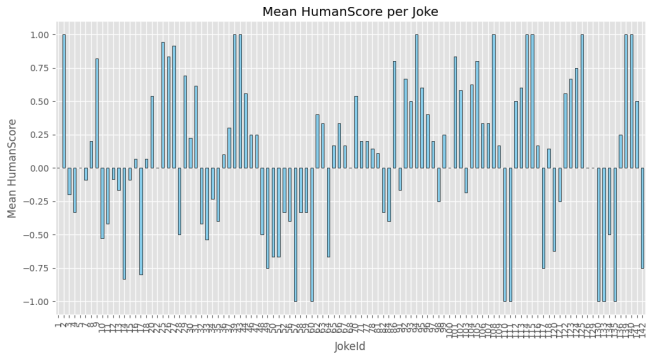
The box plots can help us understand linear relationships between features and the performance of a joke. Features known to directly affect HumanScore can be further analyzed to understand audiences and find patterns in behavior.vWe can compare performances with features that are known to influence HumanScore, and we can make predictions about potential overperforming audiences and underperforming audiences.



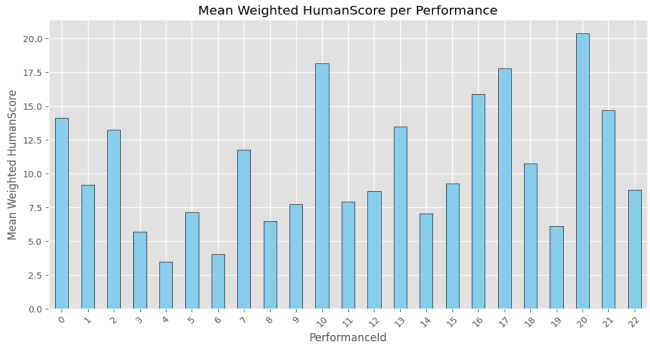
We know that there is a positive correlation between HumanScore and intensity mean, total intensity, and length. There is a negative correlation between HumanScore and 2 mean. Based on the box plots, we can infer that performances 1, 4, 18 are overachievers, and that performances 2, 3, 19, 21 are underachievers.



This is proven true by comparing the average HumanScore of each performance. The top 5 performances were 4, 18, 6, 1, and 13. The bottom 5 performances were 21, 3, 19, 17, 2.



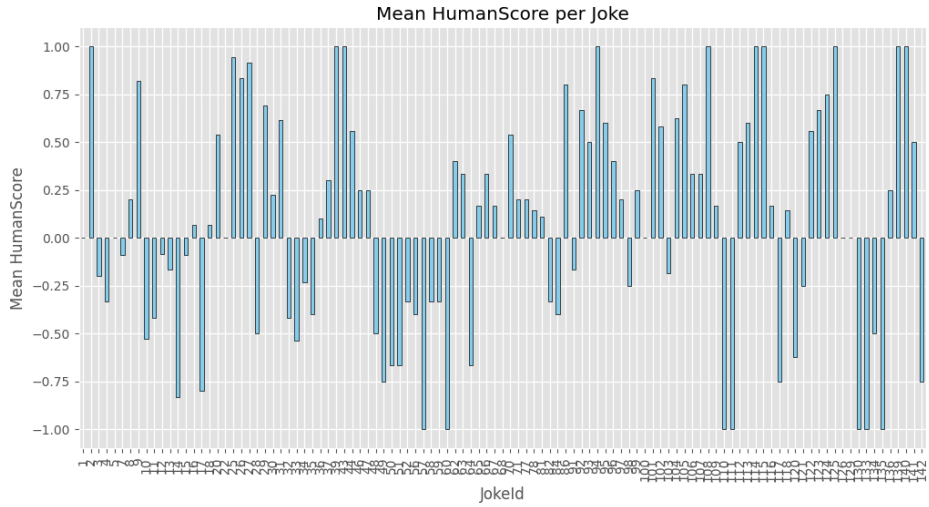
However, it can be expected that selection of jokes may bias these findings, since some jokes are objectively better than others. We can use a Weighted HumanScore to eliminate this bias, and find the audiences that truly were more or less receptive to a joke, regardless of joke quality.



This graph shows that after removing the joke bias, the top 5 performances were 17, 20, 5, 3, 8. The bottom 5 performances were 21, 12, 4, 14, 1. More importantly, however, is the fact that the performances are more similar, there are some outliers but the majority of performances scored similarly.

Analyzing Jokes

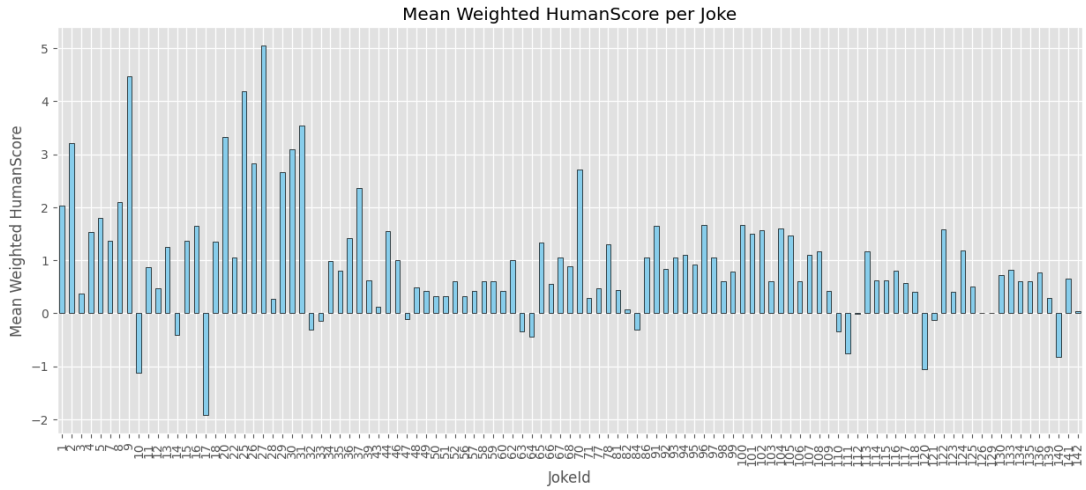
Much like performances, we know not all jokes are equal in terms of quality. We can find the best performing jokes by graphing the mean HumanScore of each joke.



The top 10 jokes are: 2, 39, 43, 94, 108, 114, 115, 125, 139, 140.
The bottom 10 jokes are: 57, 60, 110, 111, 130, 133, 135, 14, 17, 142

However, these scores may also be influenced by the overall receptivity of the audience, as well as the number of times a joke was told.

We can find a weighted HumanScore by weighing in the mean HumanScore of the performances it was told in, and the number of times the joke was told.



Once again, we can see that the joke performances are more evenly distributed now, with a few outliers. The top 5 jokes after weighing are 27, 9, 25, 31, 20. The bottom 5 jokes are 17, 10, 120, 140, 111.

Analyzing Features

Previously, we found that there are features with a linear relationship to HumanScore. However, there are potential non-linear relationships that can be hard to identify. For this, we can use models such as Random Forests and Permutation Importance to find the most significant features in the dataset.

A random forest analyzes the contribution of a specific feature to the model's prediction by measuring the reduction in Gini impurity. Permutation importance calculates the importance of a given feature by evaluating how much the model's performance decreases when a feature's values are randomly shuffled.

Feature Importances by RF

	Feature	Importance
4	intensity_max	0.090513
1	intensity_mean	0.086752
13	2_mean	0.064545
33	7_mean	0.054806
2	intensity_std	0.046804
71	intensity_total	0.044947
0	length	0.029105
9	1_mean	0.028956
15	2_min	0.025099
17	3_mean	0.024745
35	7_min	0.022802
70	pitch_range	0.020149

Feature Importance by Permutation Importance

	Feature	Importance
13	2_mean	0.005483
1	intensity_mean	0.005125
2	intensity_std	0.003099
4	intensity_max	0.002741
9	1_mean	0.002503
33	7_mean	0.002384
71	intensity_total	0.000358

Both models found the same features to be the most significant: 2_mean, intensity_mean, intensity_std, intensity_max, 1_mean , 7_mean, intensity_total, and length. Knowing what features are significant is important because it can help us curate a feature list that only contains relevant features, potentially saving processing time and increasing accuracy.

Implementing Models

Three Machine learning models were implemented to predict the HumanScore of a joke, Decision Trees, k-Nearest Neighbors, and Support Vector Machines. All 3 methods perform different calculations to predict HumanScore, so we can analyze performance metrics to decide which method is most beneficial for our dataset.

I decided to implement stratified k-fold cross validation, splitting the data into 5 folds. I chose this since this stratified cross validation ensures that each fold maintains the same proportion of these classes as the original dataset. After the model is trained, 5 performance metrics are extracted: accuracy, precision, recall, F1-score, and confusion matrices. We can later compare these to find the most optimal model for our problem.

The models were trained with different feature lists in order to find the most optimal combination. The feature lists compared were: top 6 permutation importance features, top 10 random forest features, top 50% random forest features, all features, top 50% random forest features without MFCC, and all features without MFCC.

Decision Tree

Top 6 permutation importance features: Mean Accuracy: 0.7687, Mean Precision: 0.7696, Mean Recall: 0.7687, Mean F1-Score: 0.7675

Top 10 random forest features: Mean Accuracy: 0.7568, Mean Precision: 0.7590, Mean Recall: 0.7568, Mean F1-Score: 0.7563

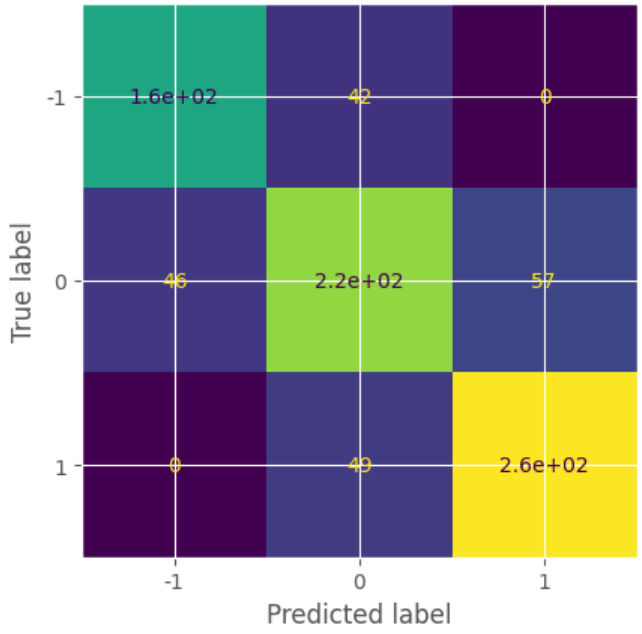
Top 50% random forest features: Mean Accuracy: 0.7210, Mean Precision: 0.7206, Mean Recall: 0.7210, Mean F1-Score: 0.7190

All features: Mean Accuracy: 0.7449, Mean Precision: 0.7466, Mean Recall: 0.7449, Mean F1-Score: 0.7444

Top 50% random forest features without MFCC: Mean Accuracy: 0.7222, Mean Precision: 0.7246, Mean Recall: 0.7222, Mean F1-Score: 0.7201

All features without MFCC: Mean Accuracy: 0.7437, Mean Precision: 0.7441, Mean Recall: 0.7437, Mean F1-Score: 0.7431

Confusion matrix for Top 6 permutation importance features :



The confusion matrix can be interpreted as the bottom left corner being a type II error (false negative), and the upper right corner being a type I error (false positive). Most importantly, no true FP or FN were predicted.

k-Nearest Neighbors

Top 6 permutation importance features: Mean Accuracy: 0.7890, Mean Precision: 0.7888, Mean Recall: 0.7890, Mean F1-Score: 0.7874

Top 10 random forest features: Mean Accuracy: 0.7950, Mean Precision: 0.7960, Mean Recall: 0.7950, Mean F1-Score: 0.7931

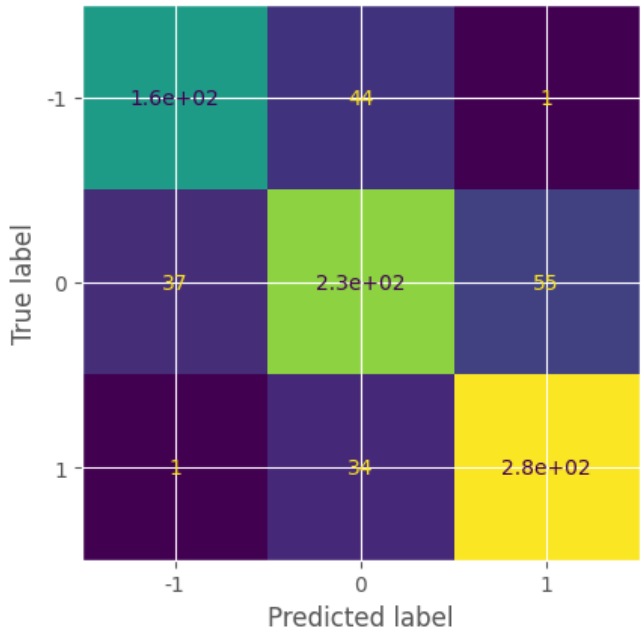
Top 50% random forest features: Mean Accuracy: 0.7604, Mean Precision: 0.7628, Mean Recall: 0.7604, Mean F1-Score: 0.7591

All features: Mean Accuracy: 0.7092, Mean Precision: 0.7085, Mean Recall: 0.7092, Mean F1-Score: 0.7061

Top 50% random forest features without MFCC: Mean Accuracy: 0.7819, Mean Precision: 0.7870, Mean Recall: 0.7819, Mean F1-Score: 0.7816

All features without MFCC: Mean Accuracy: 0.7628, Mean Precision: 0.7718, Mean Recall: 0.7628, Mean F1-Score: 0.7634

Confusion matrix for Top 10 random forest features :



The confusion matrix can be interpreted as the bottom left corner being a type II error (false negative), and the upper right corner being a type I error (false positive). Most importantly, only 1 true FP and 1 true FN were predicted.

Support Vector Machine

Top 6 permutation importance features: Mean Accuracy: 0.8307, Mean Precision: 0.8346, Mean Recall: 0.8307, Mean F1-Score: 0.8310

Top 10 random forest features: Mean Accuracy: 0.8533, Mean Precision: 0.8542, Mean Recall: 0.8533, Mean F1-Score: 0.8530

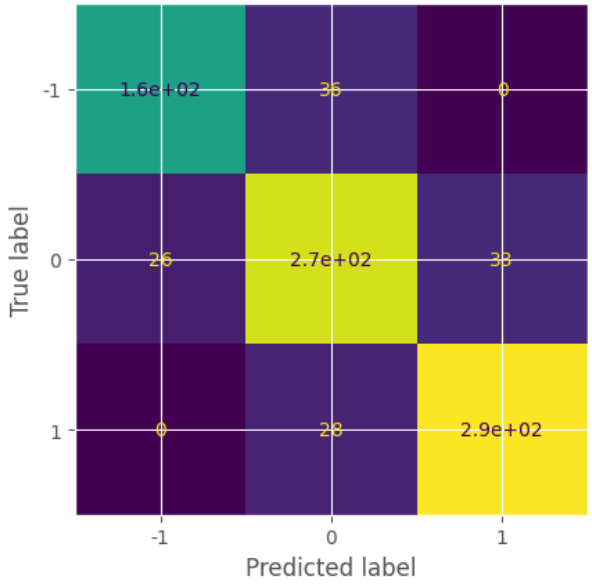
Top 50% random forest features: Mean Accuracy: Mean Accuracy: 0.8188, Mean Precision: 0.8208, Mean Recall: 0.8188, Mean F1-Score: 0.8189

All features: Mean Accuracy: 0.7926, Mean Precision: 0.7930, Mean Recall: 0.7926, Mean F1-Score: 0.7921

Top 50% random forest features without MFCC: Mean Accuracy: 0.8176, Mean Precision: 0.8194, Mean Recall: 0.8176, Mean F1-Score: 0.8177

All features without MFCC: Mean Accuracy: 0.8307, Mean Precision: 0.8331, Mean Recall: 0.8307, Mean F1-Score: 0.8308

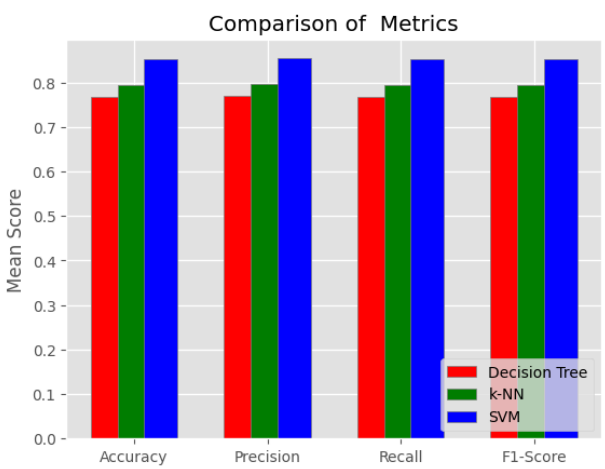
Confusion matrix for Top 10 random forest features :

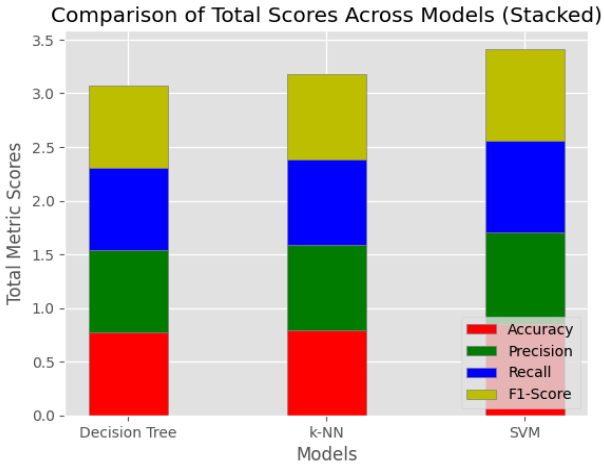


The confusion matrix can be interpreted as the bottom left corner being a type II error (false negative), and the upper right corner being a type I error (false positive). Most importantly, no true FP or FN were predicted.

Comparing the Models

All models did a good job predicting HumanScore, with ranges between 75% and 85%. Decision tree performed best using the top 6 permutation importance features, while k-Nearest Neighbors and Support Vector Machines performed best using the top 10 random forest features.





Comparing the models, it is clear that the Support Vector performed best, followed by k-Nearest Neighbors, and decision trees last.

Confusion matrices from the models show us that they all commit the same mistakes: Most often mispredicting HumanScore when the score is 0, followed by mispredicting 0 when the HumanScore is -1 or 1.

Comparing to Dr. Fitter's Models

These accuracies can be compared to the human benchmarks used by Dr. Fitter in her Laughter Classification Methods paper. The human accuracies were 88%, 72%, 91%. The accuracies of the SVM model are comparable to these benchmarks.

Additionally, we can compare the accuracies of the models developed in the paper. Our accuracies of 76%, 80% and 85% exceed those in the paper. Interestingly, the SVM model in the paper was the lowest performing model while the decision tree was the best performing. In our case, the SVM model performed the best, and decision trees the worst.

Based on these findings, I estimate that the accuracy of a model needed to be considered accurate is 80%. I believe that with careful hyperparameter testing and tuning, it is possible for a Support Vector Machine model to predict HumanScores with 90% accuracy.

Do We Need MFCC?

One of the objectives of this project is to figure out if it's worth it to calculate MFCC data, being that it is very costly to acquire. In all models, the best performing model included MFCC features. However, when comparing the models trained with all features, there is little difference between the models trained with MFCC data and not. In decision trees, the model trained on all features without MFCC had an accuracy of 74%, which is higher than the accuracy of the model trained on the top 50% features, and identical to the model trained on all features including MFCC. In k-Nearest Neighbors, the model trained on the top 50% of features without MFCC outperformed the models trained with 50% of features and all features, when including MFCC. In Support Vector Machine, the model trained with the top 50% of features without MFCC outperformed every model, except for top 10 random tree features.

Overall, specific MFCC data (1, 2, 7) is extremely useful when predicting HumanScore, and provides a different perspective than intensity and pitch. If accuracy is the top priority, then MFCC should be kept, since it maximizes accuracy of every model.

However, the models trained without MFCC perform better than many models trained with it, indicating that the majority of the MFCC data is not useful. Unless there is a way to specifically collect a particular MFCC frequency, it is very wasteful to collect MFCC, both in terms of resources and the data being used. If being cost-effective is a priority, this data should not be measured, as models trained without it are still very effective.