amadeus
a music classifier

Team
Sidwyn Koh
Eric Nelson
Krishna Parashar
Max Wolffe

Problem

Given a song, predict the year it was popular.

We used Columbia University's Million Song Dataset. After cleaning it up and exploring a subset with iPython notebook we scaled the model up using Spark to the entire dataset.

Process

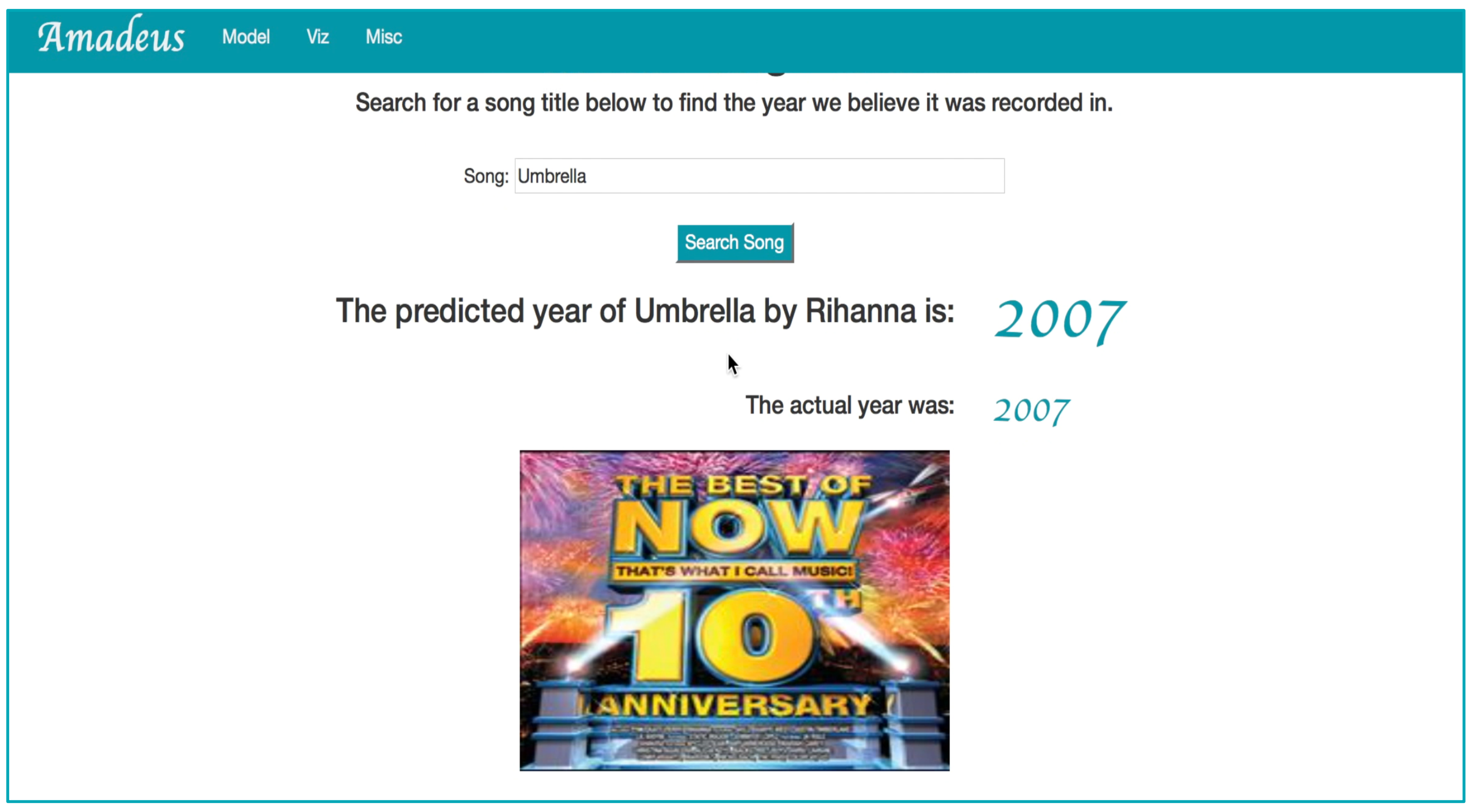
Many of the features were not present across the dataset so after selecting the most common ones, we removed missing keys and zeroed out any incomplete data. The dataset promised to not have any duplicates.

To explore our data set and do simple analysis we converted the data from *hdf5* format to *csv* we then extracted basic statistical features and visually inspected our data using `matplotlib` graphing functions.

To create our model we first used `scikit-learn` on our subset of data, and then used *Spark* and *MLLib*, running on *EC2* to train a classifier on our 200+GB of raw data.

Visualization

We wanted to make our model interactive and interesting for someone to explore the dataset so we created a *Flask* web app to run our *Pickled* Python model, checking our data against the *EchoNest API*. The app works by allowing users to search for a particular song and then our model returns the predicted year of popularity along with the actual year and includes a visualization of some of the features used by the classifier.



Search for a song title below to find the year we believe it was recorded in.

Song: Umbrella

Search Song

The predicted year of Umbrella by Rihanna is: 2007

The actual year was: 2007

Challenges & Future

In the future, there are several optimizations and improvements that we can make to our model.

We want to do regularization so that our parameters can be capped, which might allow for both forms of regression to be more competitive when compared against random forests.

If we also have the chance, we will also want to try feature ablation to pick the features in the future, and compare that with PCA.

Models & Results

The dataset provided us with a large number of features, but many of them were sparse across the dataset, so we carefully selected the features that were found in at least 90% of the dataset. We thus ended up using the:

timbre, segment starts, beat starts, duration, key, key_confidence, loudness, end of fade in, start of fade out, tempo, time signature and time_signature_confidence

Accuracy was defined as a +/- 10 year range of the actual year the song was produced in. This is consistent with ISMIR's accuracy statistic.

To help measure accuracy we used Mean Absolute Error – where yi is the corresponding true value, and yi_hat is the predicted value. This is an unbiased estimator.

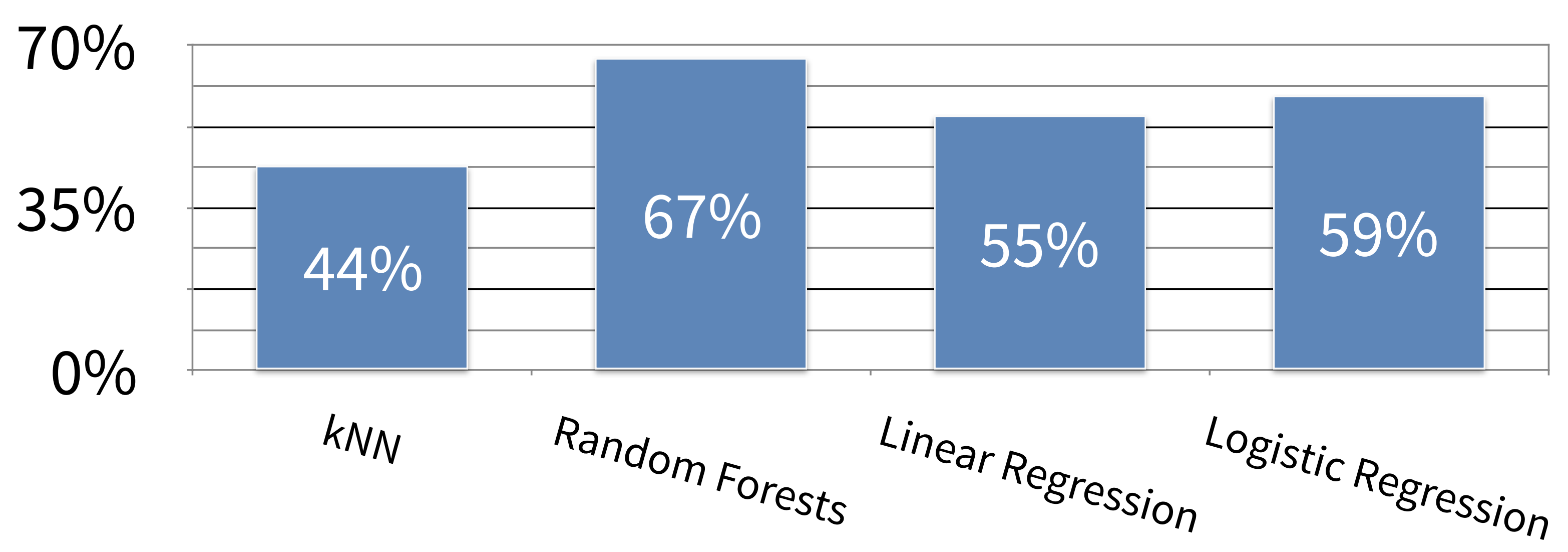
$$\text{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} |y_i - \hat{y}_i| .$$

We also used Mean Squared Error – where yi is the corresponding true value, and yi_hat is the predicted value. This is also an unbiased estimator.

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2 .$$

We used these measurements for accuracy and ran the following different models locally on a subset of the data. This resulted in the following accuracy rates (year +/- 10 years):

Accuracy on 10,000 Song Subset



Using **Random Forests** for our model proved to be the most accurate, an so using the MLLib in Spark we scaled the model to run across the entire dataset. This took about 8 hours on a Large EC2 instance and we resulted in a accuracy of **79.1% +/- 10 years**.

Model	Characteristics	Accuracy	Mean Absolute Error	Mean Squared Error
Random Forests	Trees = 5, max_depth = 8, max_bins = 32, impurity='variance',	79.1%	6.92	94.3

Using Random Forests for our model proved to be the most accurate, and so using the MLLib in Spark we scaled that model to run across the entire dataset. This took about 8 hours on a Large EC2 instance and resulted in an accuracy of 79.1% +/- 10 years. Our final result of 6.92 on Random Forests is quite a ways better than Columbia researchers' mean absolute error of 9.78. Our accuracy rate was also quite high.

There are several reasons to describe our results above. First of all, Random Forests work very well with high-dimensional data. With that, Random Forests can approximate almost any shape. Whereas on the other hand, linear and logistic regressions have the assumption to approximate functions with linear and logistic shape and might work less effectively on such high-dimensional data like ours. We also had the luxury of having more computing resources and time (with the EC2 machine) to be able to run Random Forests, which took more time.

UC Berkeley | Data Science | Fall 2015

http://github.com/kcparashar/amadeus

Thanks to John Canny & Haoyu Chen
and the Jacobs Community.