Predicting release year of songs

Rasmus Salmon & Haider al Saadi
50/50 effort
2020 Machine Learning

Working problem:

- **Goal:** Find out if music is different today compared to earlier

- **In ML:** Predicting the release year of a given song based on objective features
  - Classification of release decade
  - Regression of release year

“Music was much better, when I was young”

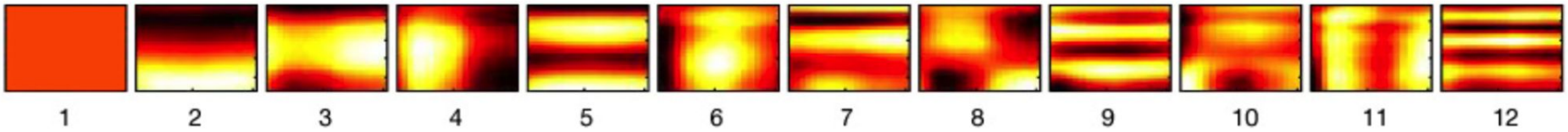
-Parents

Data:

- Use data from the “million song database”
  - Contain derived features from 1 million popular songs ⇒ No audio
  - Created in 2011 ⇒ No newer songs
  - 280 GB of raw data ⇒ Use statistical data from curated subset of the dataset
  - Created by Columbia University and The Echo Nest(MIT startup, now Spotify) ⇒ High quality

Our subset

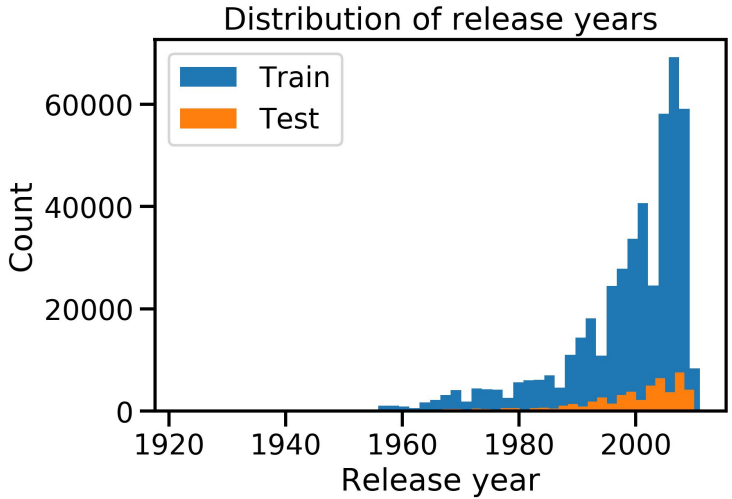
- **No. of observations:** 515.345 songs
- **Target:** Year/decade of release (1920 - 2011)
- **Features:** 90 attributes outlining the statistical sound profile of each song (Mean and covariance of a 12-dimensional timbre analysis)
- **Train-test-split:** Done on an artist level (463.715 in train and 51.630 test)



12 basis functions for the timbre vector: x = time, y = frequency, z = amplitude

Data exploration:

Balance: Both train and test data is imbalanced.

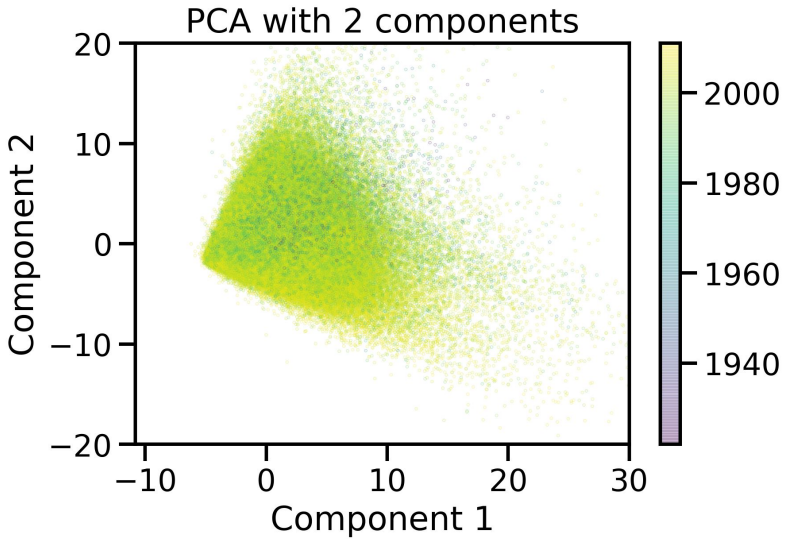


Clusters: No clear clusters in low dimensional representation.

PCA: 75 principal components for 99% variance

25 components for 96% variance

-> High contribution from each variable!



A first approach:

Model: A simple decision tree

Results: Great (We thought)

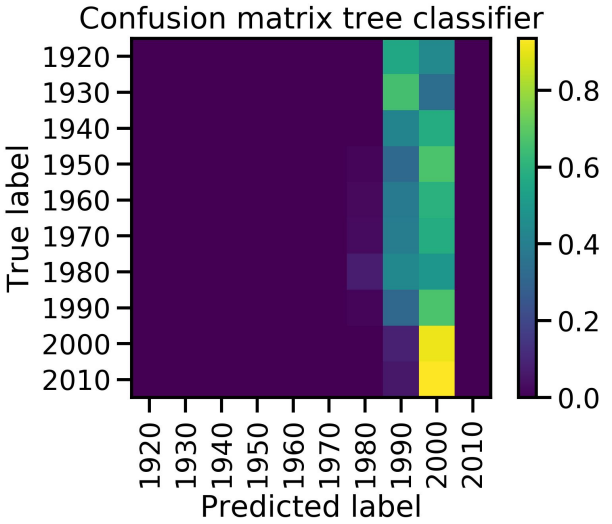
Clustering: Accuracy of 60 pct.

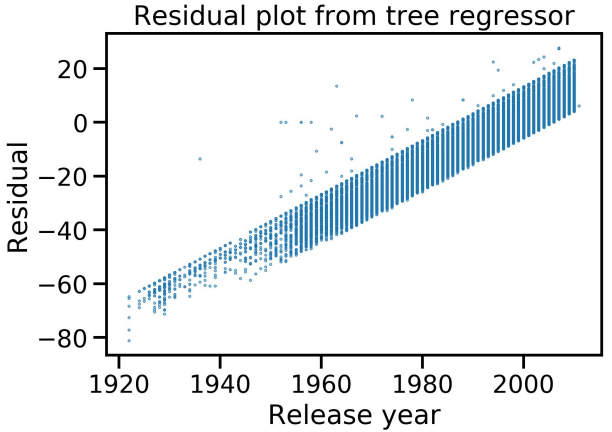
Regression: MAE of 9.1 years

Problem: Model only “used” part of data

Solution: Resample to get balanced data.

The “new” data: 1.000 songs from each year.





Xgboost model:

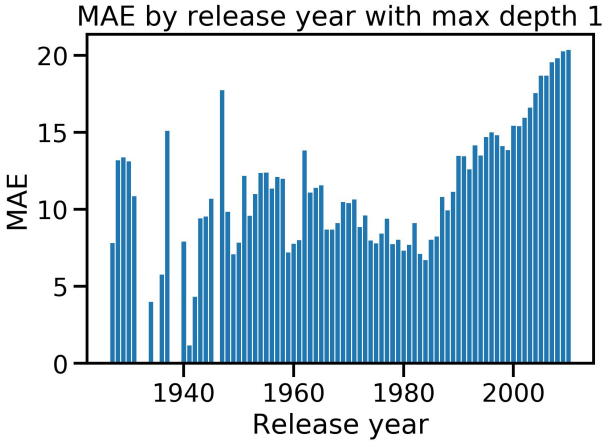
- High performance
- Extensive parameter space
- Easy to work with

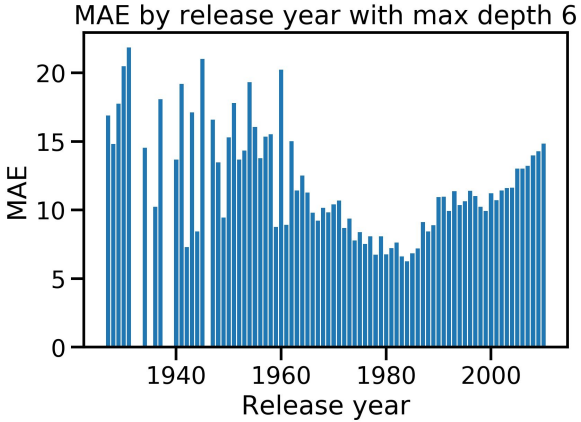
Regression for release year:

Pattern ⇒ Simpler models fit earlier songs better and complex models fit later songs better.

Songs from 1960-1980 is generally predicted well.

The MAE for each year is in the 10-20 year range.





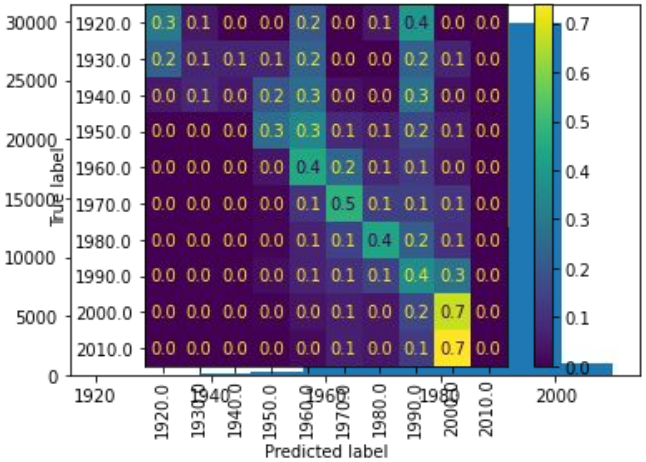
Parameter optimization part 1 XGB_classifier

Optimized number of estimators and max depth

As log-loss ratio increases, metric improves? -> metric is bad, but why?

Because test data is unbalanced too!

Also, new music might be more similar too?



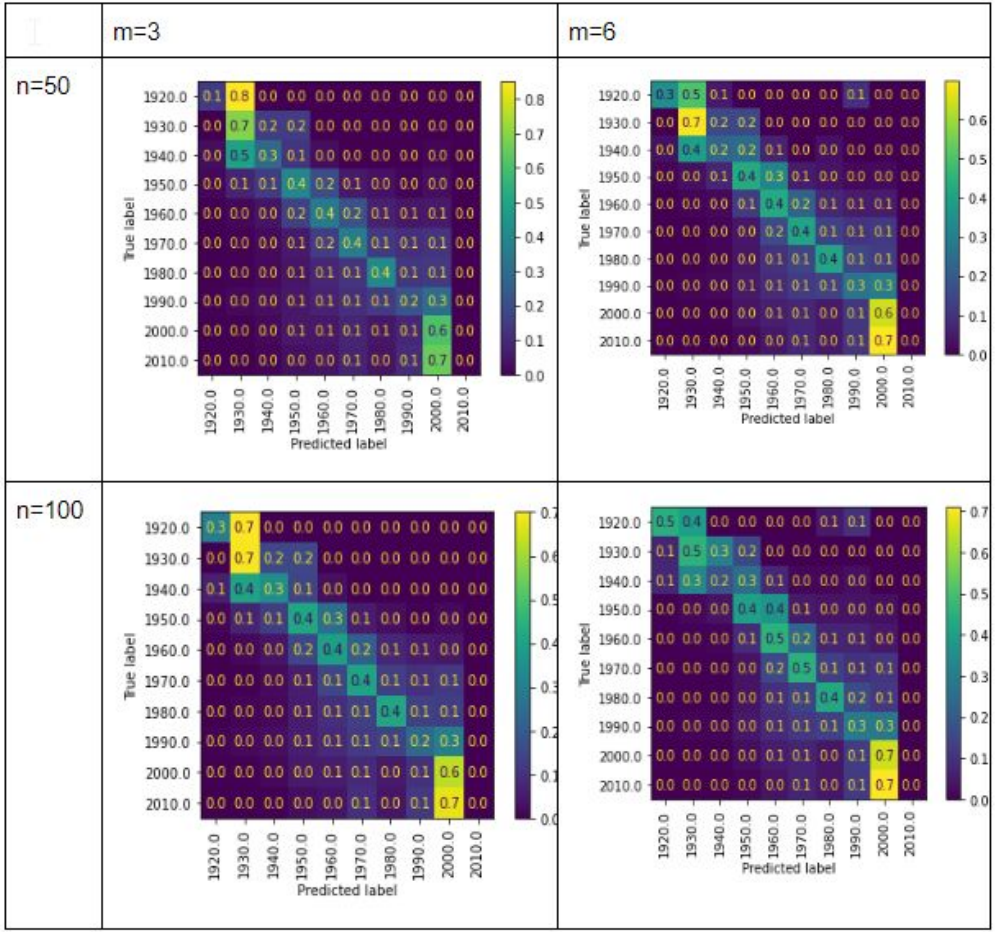
Estimator	Max depth	Log loss test	log loss train
50	2	1.695	NaN
50	3	1.605	1.34
50	4	1.53	NaN
50	6	1.43	0.8
100	3	1.49	NaN
100	6	1.33	NaN
500	6	1.31	NaN
1000	6	1.24	0.08

Parameter optimization part 2 (Visuel inspection)

Increase parameter=Better at new, worse at old.

->Maybe newer decades want big trees and older want small trees?

Should we rebalance test set?



Discussion:

- What information is contained in the data?
  - Music genre, music style, choose of instruments, production technology, originates from spotify
- The time measurement?
  - Years and decades are arbitrary
- Other techniques for imbalanced data?
  - Performance metrics, algorithms, more data...
- What did this analysis of the data give us?

Future work:

- Test different types of hyperparameters in Xgboost
  - Learning rate, shrinkage, subsampling...
- Redo analysis with..
  - More balanced data.
  - audio data
  - split the set into old and new

Conclusion:

- Imbalanced data is hard to work with

- Music has changed over time
  - Qualitative difference in ML-models dependent on whether the songs are new or old
- Be very thoughtful when applying models built on this data!

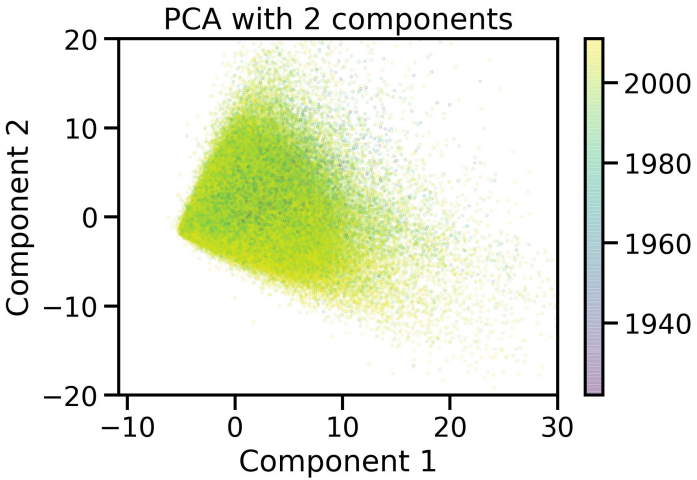
References:

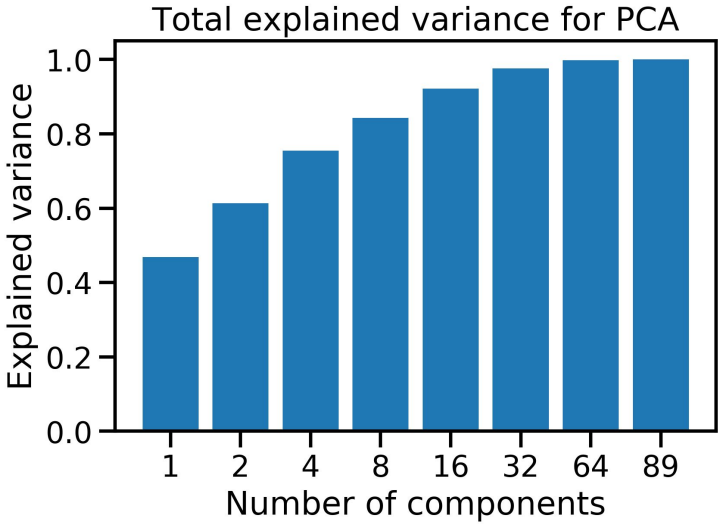
- Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.

Appendix

PCA on the data:

We performed PCA on the data. We learned that we need a lot of variables to capture the variance of the data, and that we could not spot clear clusters.



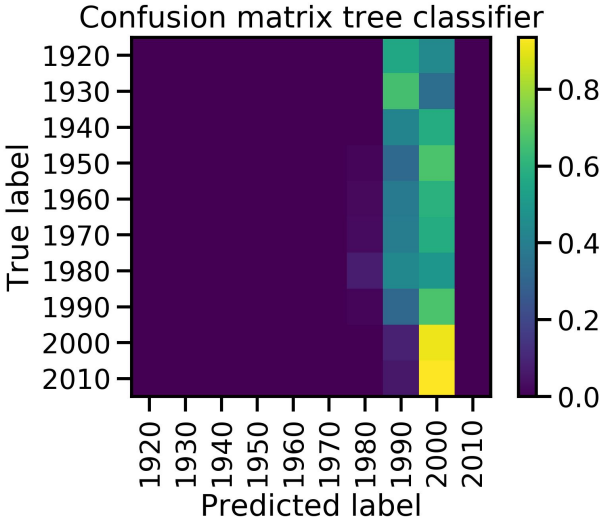


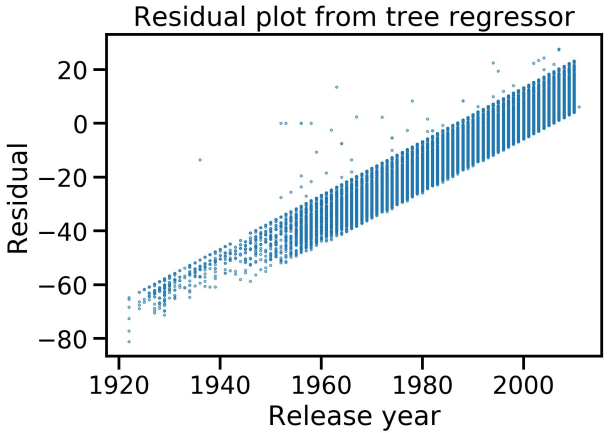
The first approach tree:

Our first approach to analysing the data was a simple decision tree with a max depth of 8.

Everything seem great intoll we plotted the result by year.

Clustering: Accuracy of 60 pct.
Regression: MAE of 9.1 years





Balancing the data(sudo and actual code):

-For each year i in the training data:

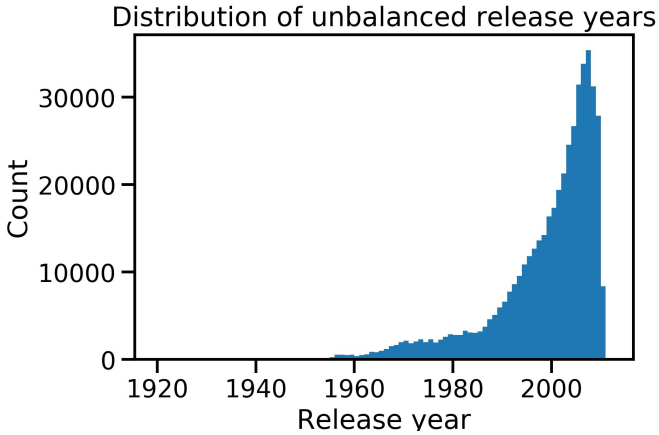
-Resample n songs with replacement from training data where realese == i

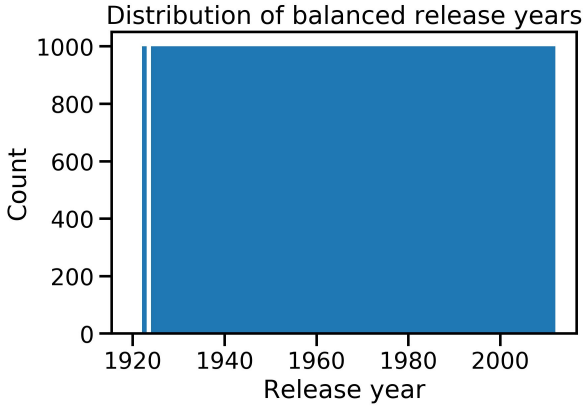
-Add resampled data for year i to combined data for all year

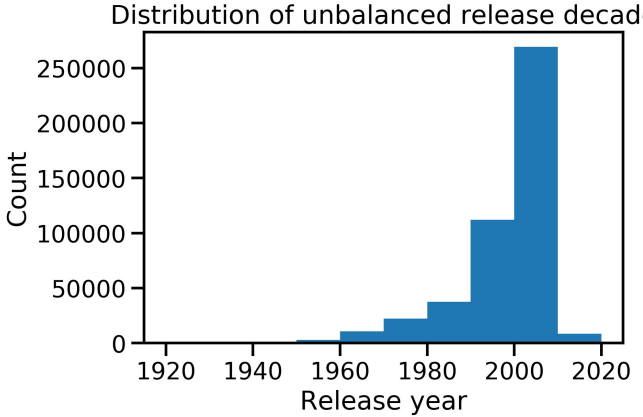
-Next i

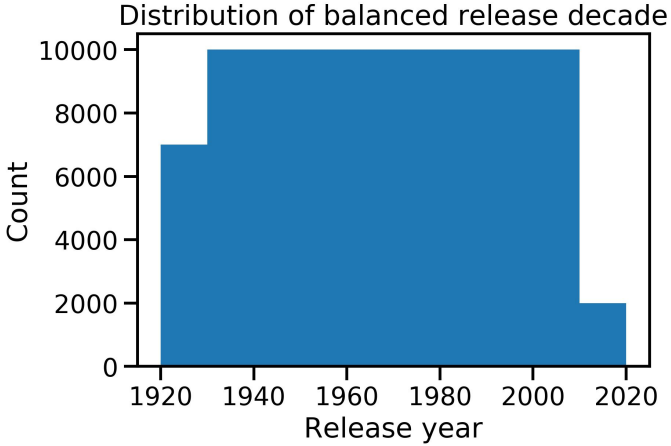
```
#Unsampled_train: The full unsampled training data, labels are in the first column
sampled_train = np.zeros([n * len(np.unique(unsampled_train[:,0])), df.shape[1]])
j = 0
for i in np.unique(unsampled_train[:,0]):
  sampled_i = resample(unsampled_train[unsampled_train[:,0] == i], replace = True, n_samples = n, random_state = 38)
  sampled_train[j*n : j*n + n ] = sampled_i
  j += 1
X_train  = sampled_train[:,1:-1]
y_train = sampled_train[:,0]
```

Before and after balancing:









Why not balance on decades for classification?

Consider the hypothetical case were the raw data contains 100 songs from 1991 and 1.000 songs from 1998. Resampling on decade would give 10 times the songs from 1998 compared to 1991 in the final dataset. Thus resampling on the decade would introduce an “invisible” imbalance to the data.

We therefore resample for the years as this is easier to control.

Why not more samples from each year?

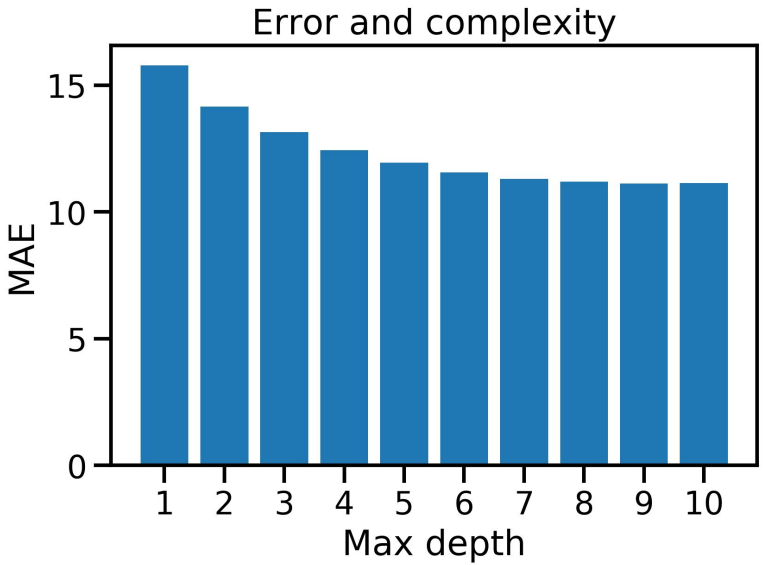
1. By resampling we can create more observations, but we can not create more variance. Taking a lot of samples from each year would result in some years being very oversampled. This would result in a final dataset where the number of observations is equally balanced but not the variance.
2. Analysing a big amount of data takes a lot of computing resources. We ran into two main issues:
    a. **Run-time**: Due to the limited time frame we wanted to reduce runtime to have more time to experiment and analyse results. We tried to use parallelism to reduce time but got problems with memory.
    b. **Memory**: Free cloud computing has a rather low RAM-limit. So trying to run big models in parallel would instantly make you hit the RAM-limit.

Considerations when splitting into old and new song

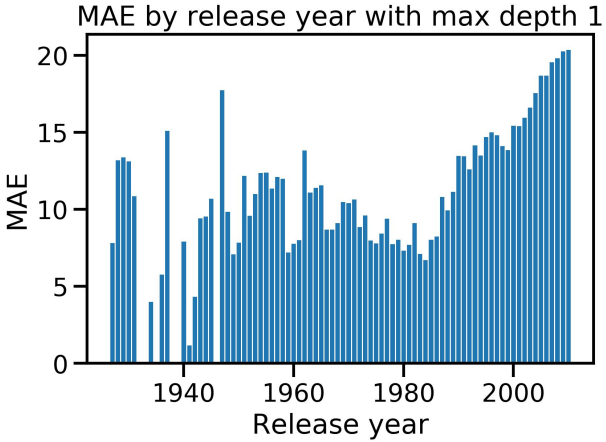
- The observation that the models optimal parameters probably significantly differ for old and new songs implies a split might be beneficial. However finding an optimal split would be prohibitively expensive time-wise, so the exact position of the split should be based on the parameters of the problem being solved and not a “naked” analysis of the data.
- Different behaviour may occur if running this based on years rather than decades. Therefore splitting subsets close to each other in years might entail different parameter choices, since the ML model that determines the decade it belongs to well is not necessarily the ML model that determines the year.

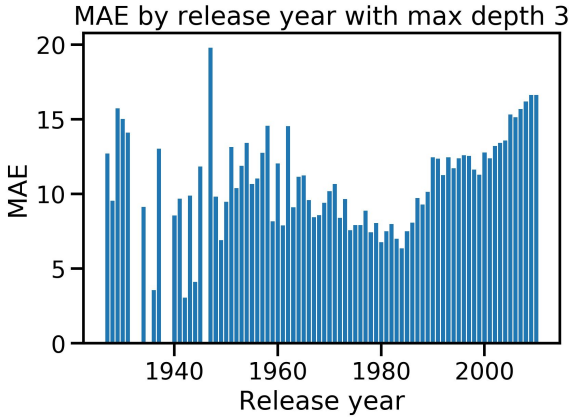
Optimizing the regression:

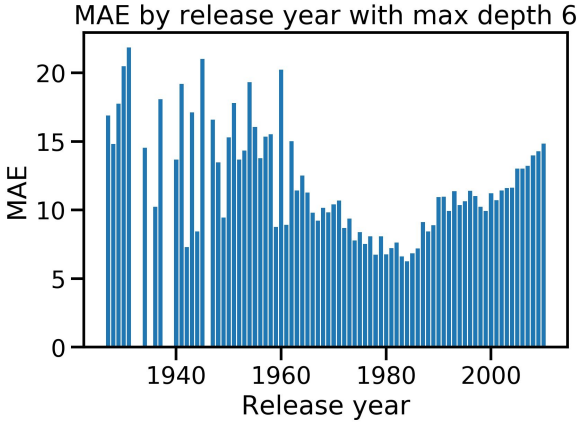
Increasing the complexity of the model decreases the error on the test data. However more complex models simply fitted more data to the newest years(see next slide). The relationship seems to be that simple models do a good job for the earlier songs, and more complex models do better on later songs. This is properly due to uneven variance across the years.

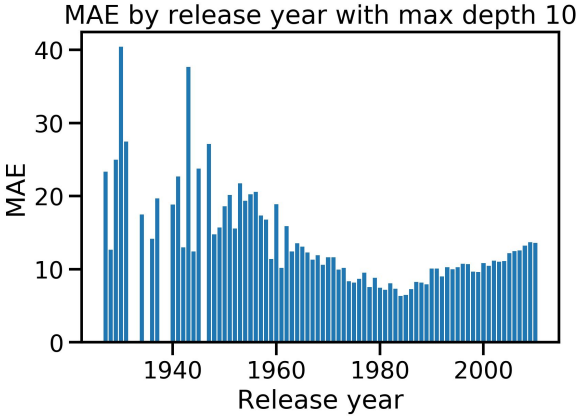


Regression results by complexity:









Optimizing using cross-validation:

One last possibility is to evaluate the models based on cross-validation.

Classification: Seems that moderately complex models do better. Might be distorted due to discontinuous nature of accuracy.

Regression: The simplest model gives the lowest cross-validation error. However looking at the results in the previous slide it seem that the simplest model does not do the best job on the test data.

Not clear if training data generalizes and therefore not clear if cross-validations is useful