

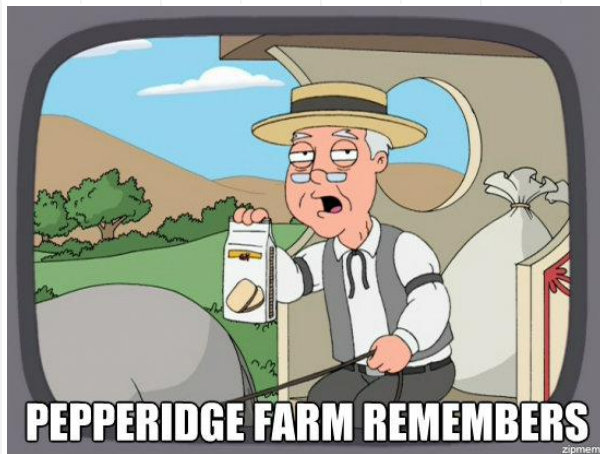


# HeatReplay

By Sabbir Ahmed

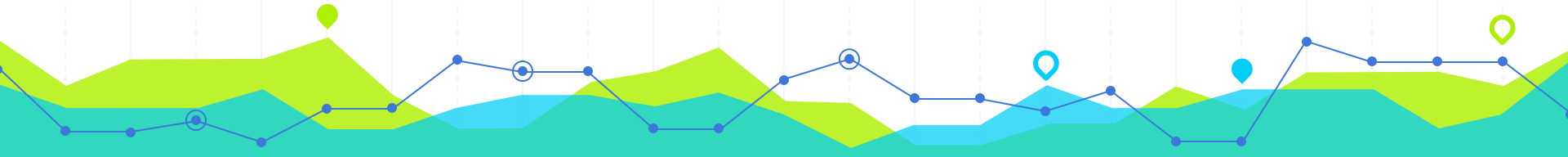
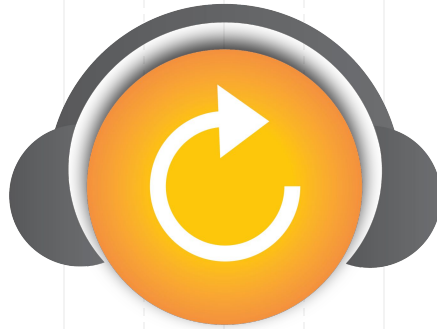
# INTRODUCTION

**Remember when songs used to have meanings?**



# QUESTION

**Can the lyrical content of a song determine if it will chart on Billboard?**



# DATA

- **Billboard Year-End Charts**
  - A cumulative measure of a single or album's performance in the United States
  - Yielded 100 songs a year starting on 1961
- **Hot Singles**, a script that scrapes all the song titles and categorized them by year
  - Compiled: **4647** charted songs
  - BUT:
    - $[1961 - 2010] * 100 \text{ songs a year} = \mathbf{4900} \text{ songs, right?}$

# DATA (CONT.)

## DATASETS/ DATA MINING

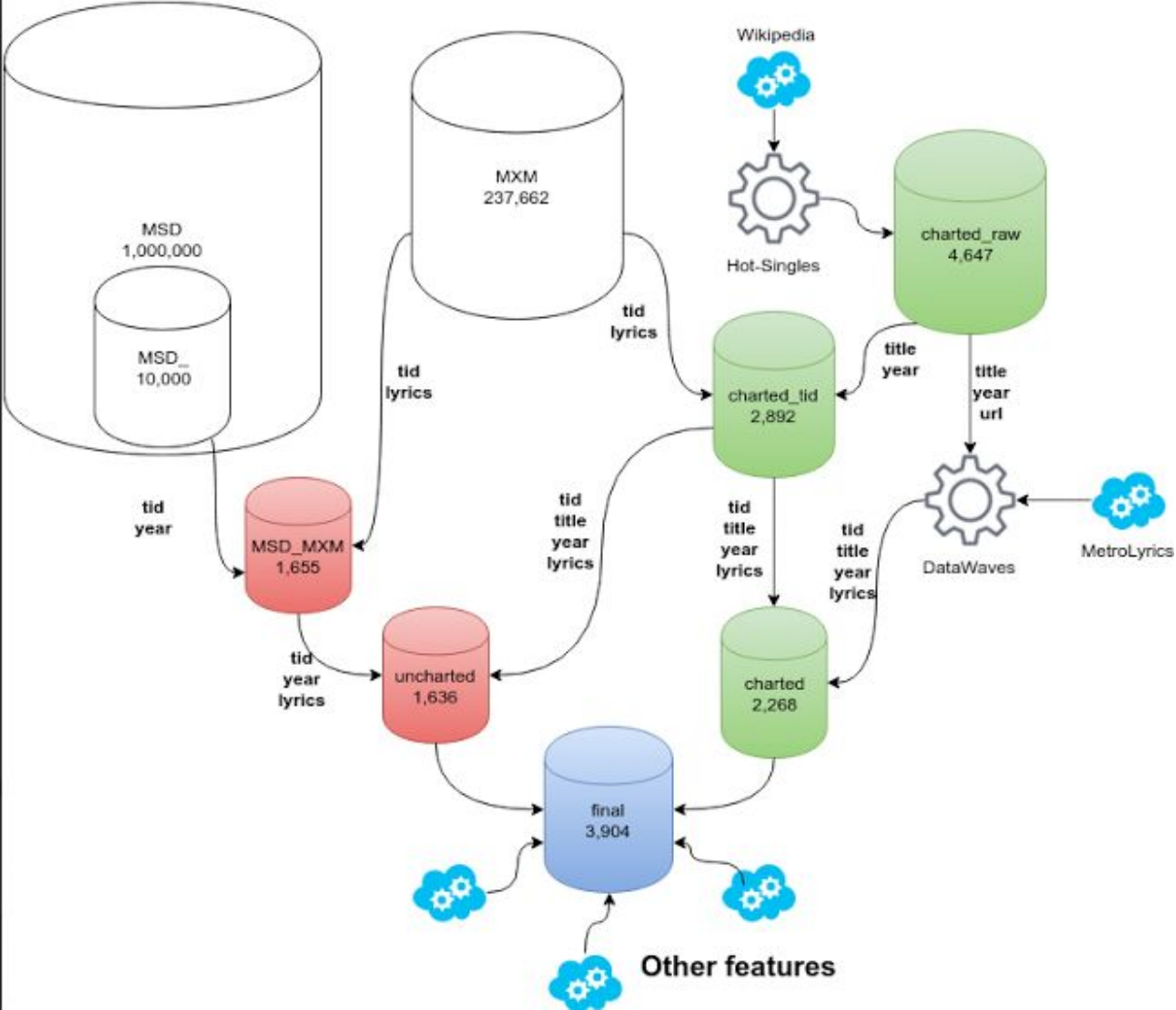
- The lyrics were obtained in 2 different ways:
  - **musiXmatch dataset**: official lyrics collection of Columbia University's **Million Song Dataset**
  - **DataWaves**: a mini-API to scrape lyrics from MetroLyrics



```
code
├── db
│   ├── analytics
│   │   ├── analytics.py
│   │   └── index_feat.py
│   └── mgmt
│       ├── DataWaves
│       │   ├── lyrics.py
│       │   └── scrapelyrics.py
│       ├── final
│       │   ├── charted_final.py
│       │   ├── concat.py
│       │   └── uncharted_final.py
│       ├── HotSingles
│       │   └── hot_singles.py
│       ├── lyrics
│       │   ├── bow.py
│       │   ├── lyrics_to_bow.py
│       │   ├── more1.py
│       │   └── sep_to_url.py
│       ├── msd
│       │   ├── display_song.py
│       │   ├── hdf5_getters.py
│       │   ├── list_msd.py
│       │   ├── map_msd.py
│       │   └── msd_mxm.py
│       └── mxm
│           ├── intersect.py
│           ├── more0.py
│           ├── mxm_combine.py
│           ├── mxm_filter.py
│           └── mxm_main.py
├── README.md
├── settings
│   ├── filemgmt.py
│   ├── paths.py
│   └── regexify.py
```

## DATA (CONT.)

### Directory tree of scripts



## DATA (CONT.) DATABASE DIAGRAM\*

\*DIAGRAM NOT TO SCALE

# DATA (CONT.)

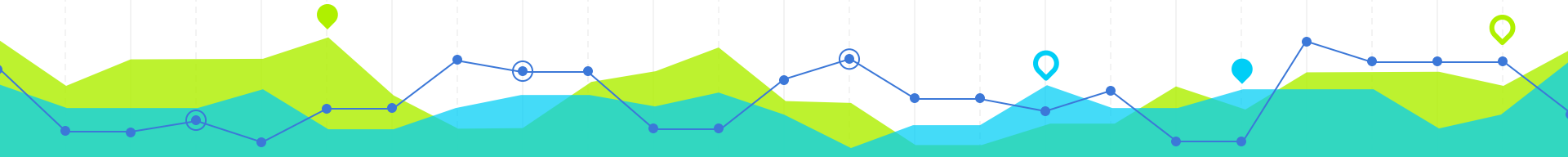
## FINAL DATA

**3904**  
**SONGS**

**58%**  
**CHARTED**

**:**


**42%**  
**UNCHARTED**





# DATA (CONT.)

## FEATURE ENGINEERING

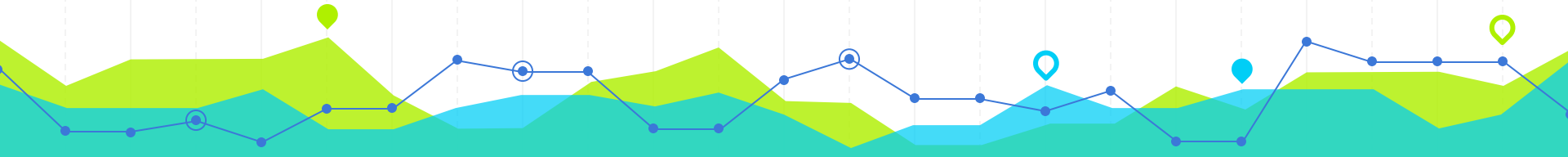
- The bag-of-words version of lyrics:
  - stripped out the concept of context
  - contained a substantial amount of stopwords
  - words were stemmed
  - only mapped the top **5000** words that appeared in all the documents
- Sentiment analysis:
  -  SentimentIntensityAnalyzer, from NLTK VADER Sentiment Analysis



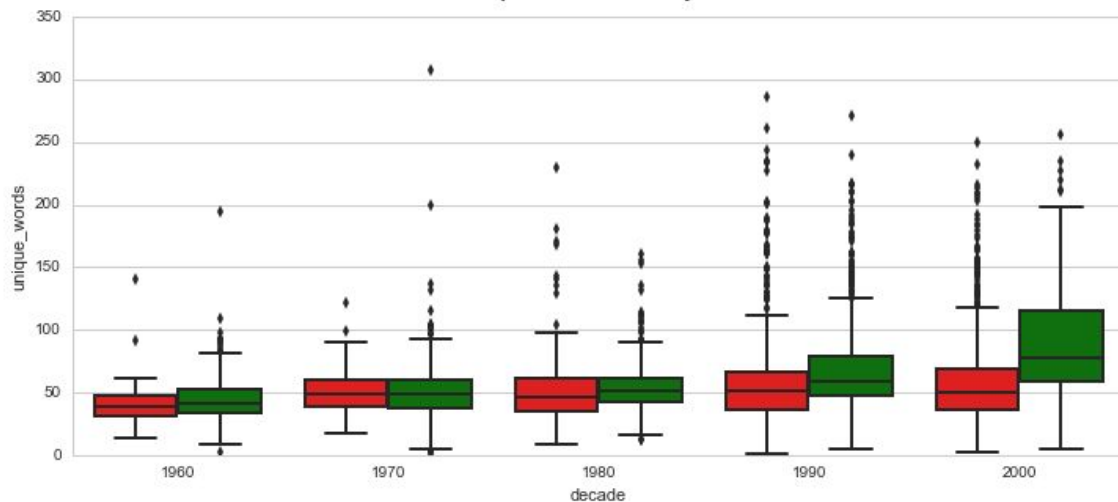
# DATA (CONT.)

## FEATURE ENGINEERING

- **Reading Scores:**
  - Flesch-Kincaid readability tests
  - Requires total number of sentences per document
- **Parts of speech tagging:**
  - Total number of verbs, nouns, and adjectives
- **Profanity**



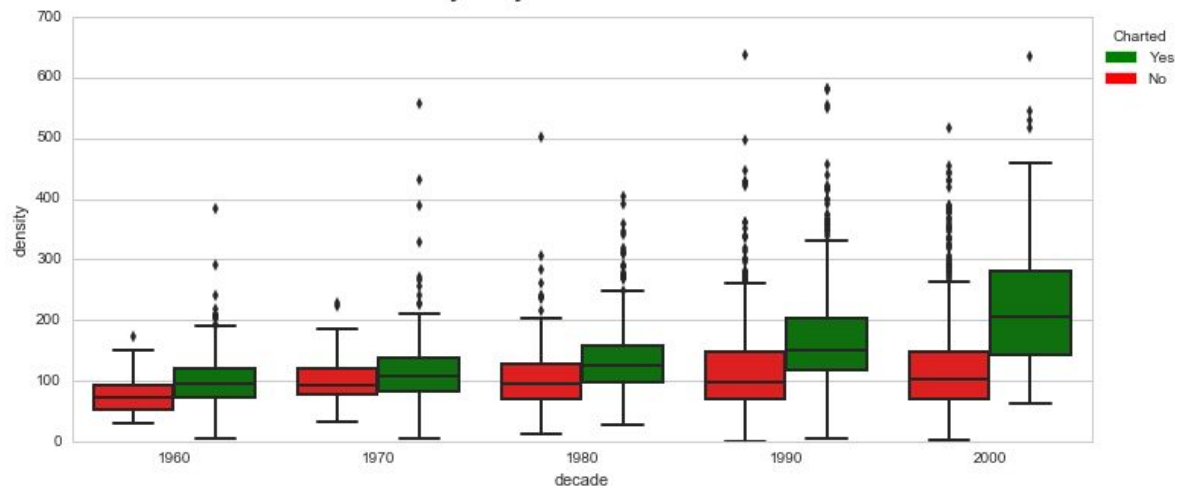
Total number of unique words in lyrics over decades



# DATA (CONT.)

## FEATURE ENGINEERING

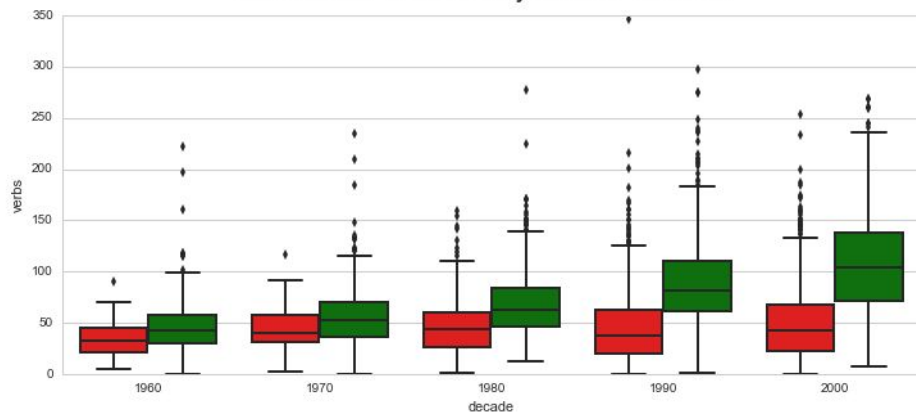
Density of lyrics over decades



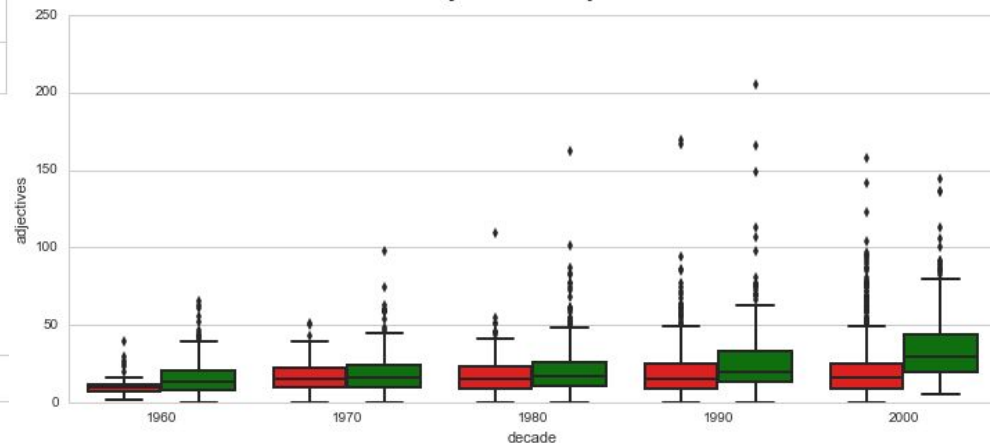
# DATA (CONT.)

## FEATURE ENGINEERING

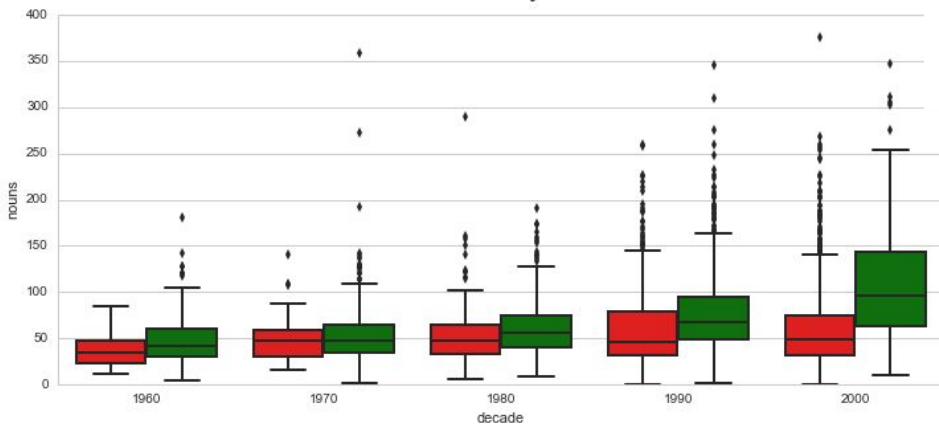
Total number of verbs in lyrics over decades



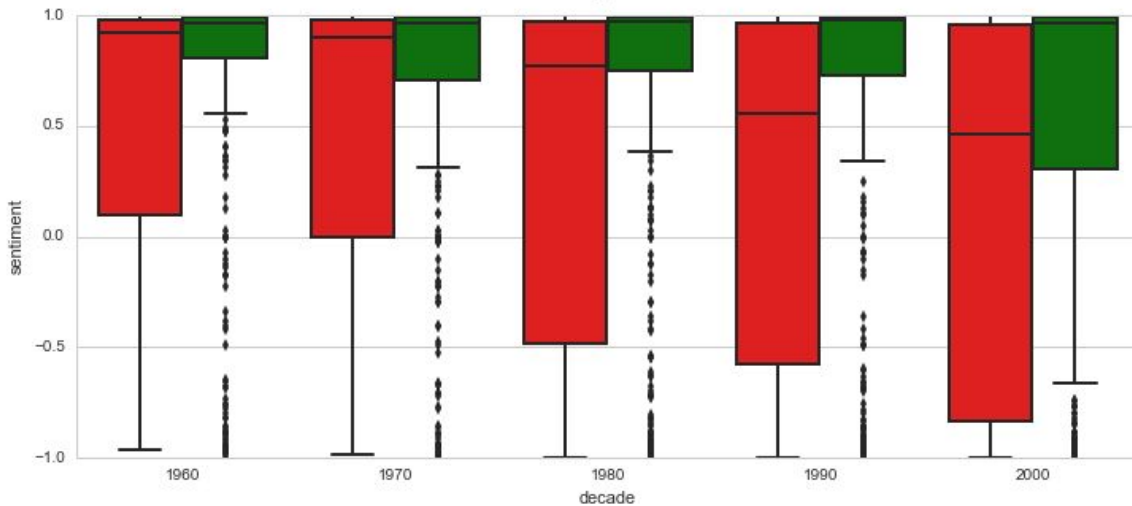
Total number of adjectives in lyrics over decades



Total number of nouns in lyrics over decades



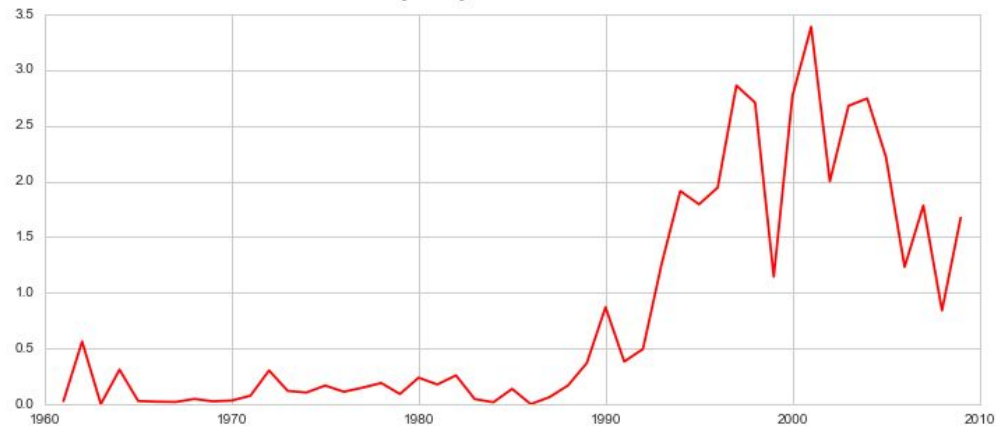
Sentiment score in lyrics over decades

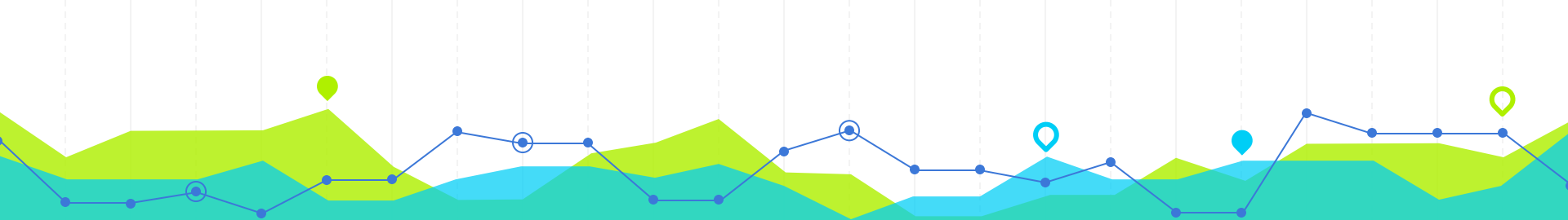


# DATA (CONT.)

## FEATURE ENGINEERING

Profanity in lyrics over decades





# THE DATA SCIENCE PART

..... FINALLY

```
model.gridSearch()
```

Grid searching for best parameters:

K Neighbors

Best Params: {'n\_neighbors': 16}

Accuracy of current clf: 0.768

Accuracy using best param: 0.793

Logistic Regression

Best Params: {'C': 256}

Accuracy of current clf: 0.728

Accuracy using best param: 0.730

Random Forest

Best Params: {'n\_estimators': 1024}

Accuracy of current clf: 0.779

Accuracy using best param: 0.811

Decision Tree

Best Params: {'max\_depth': 4}

Accuracy of current clf: 0.718

Accuracy using best param: 0.788

# MACHINE LEARNING WITH TIME

```
model.getBestParams('best_param_time.txt')
```

Best clf is Random Forest with {'n\_estimators': 1024}

```
model.gridSearch()
```

Grid searching for best parameters:

K Neighbors

Best Params: {'n\_neighbors': 64}

Accuracy of current clf: 0.664

Accuracy using best param: 0.691

Logistic Regression

Best Params: {'C': 0.03125}

Accuracy of current clf: 0.727

Accuracy using best param: 0.711

Random Forest

Best Params: {'n\_estimators': 32}

Accuracy of current clf: 0.691

Accuracy using best param: 0.710

Decision Tree

Best Params: {'max\_depth': 1}

Accuracy of current clf: 0.637

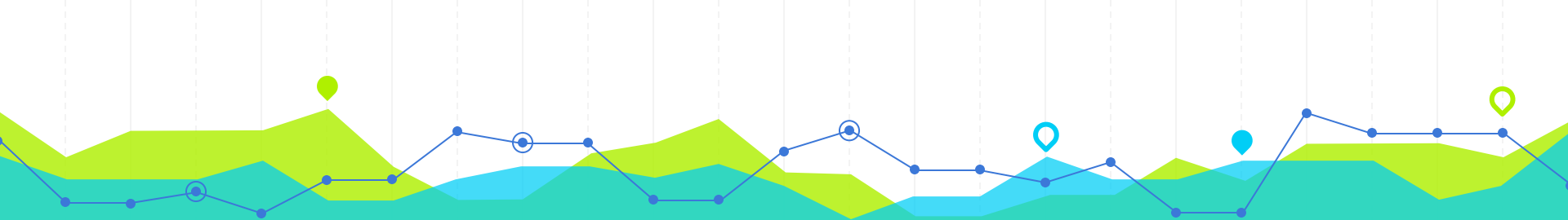
Accuracy using best param: 0.682

# MACHINE LEARNING WITHOUT TIME

```
model.getBestParams('best_param_no_time.txt')
```

Best clf is Logistic Regression with {'C': 0.03125}





# WHY TWO MODELS?

# MODEL WITHOUT TIME

```
LogisticRegression(C=0.03125, class_weight=None, dual=False,  
    fit_intercept=True, intercept_scaling=1, max_iter=100,  
    multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,  
    solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```



# MODEL WITHOUT TIME

72.7%

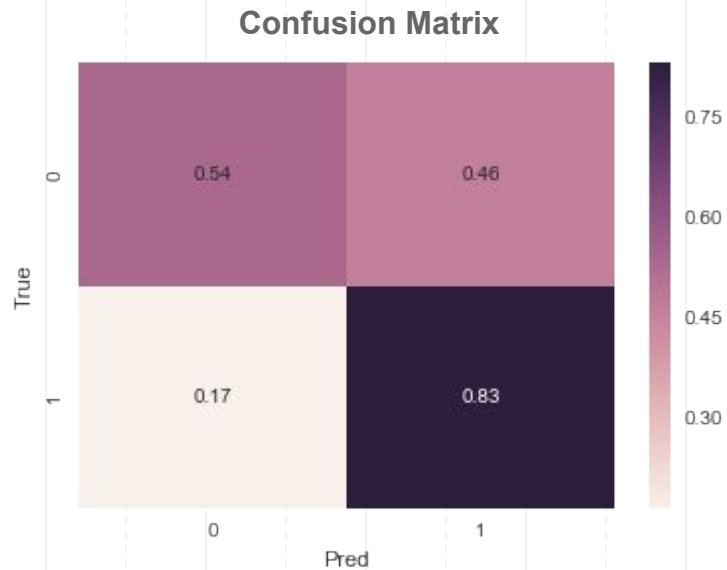
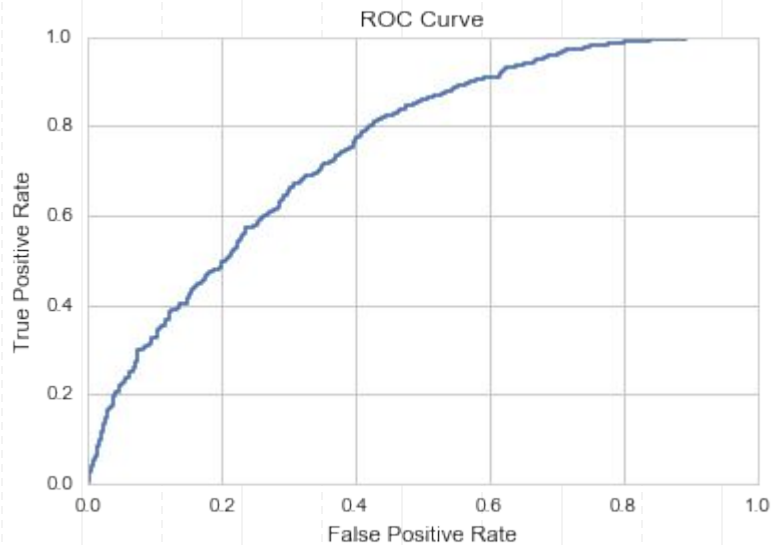
Training Set

71.0%

Test Set



# MODEL WITHOUT TIME



# MODEL WITH TIME

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',  
                        max_depth=8, max_features='auto', max_leaf_nodes=None,  
                        min_samples_leaf=2, min_samples_split=2,  
                        min_weight_fraction_leaf=0.0, n_estimators=256, n_jobs=1,  
                        oob_score=False, random_state=None, verbose=0,  
                        warm_start=False)
```



# MODEL WITH TIME

88.9%

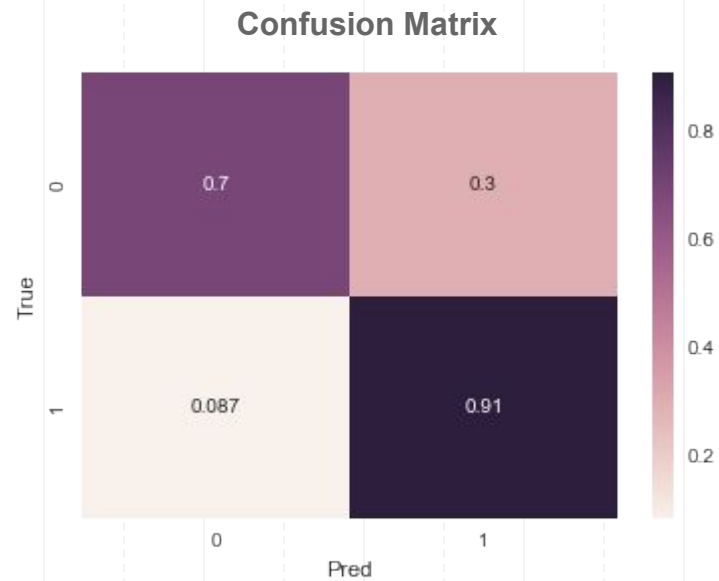
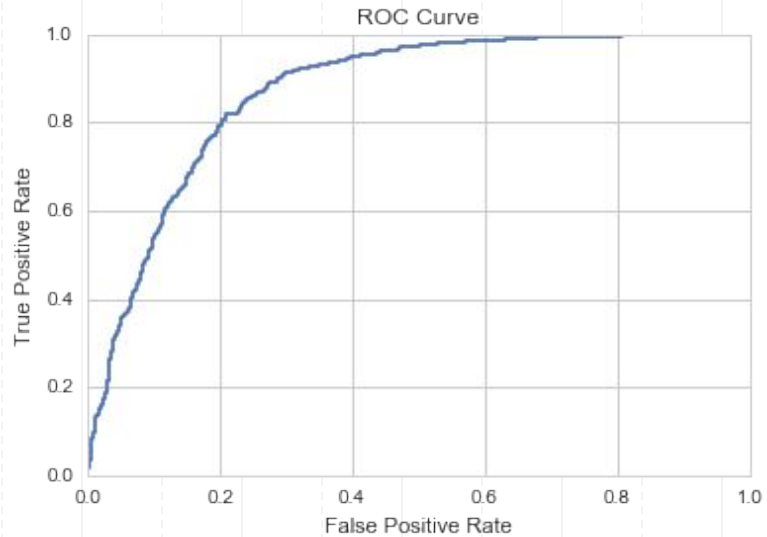
Training Set

82.3%

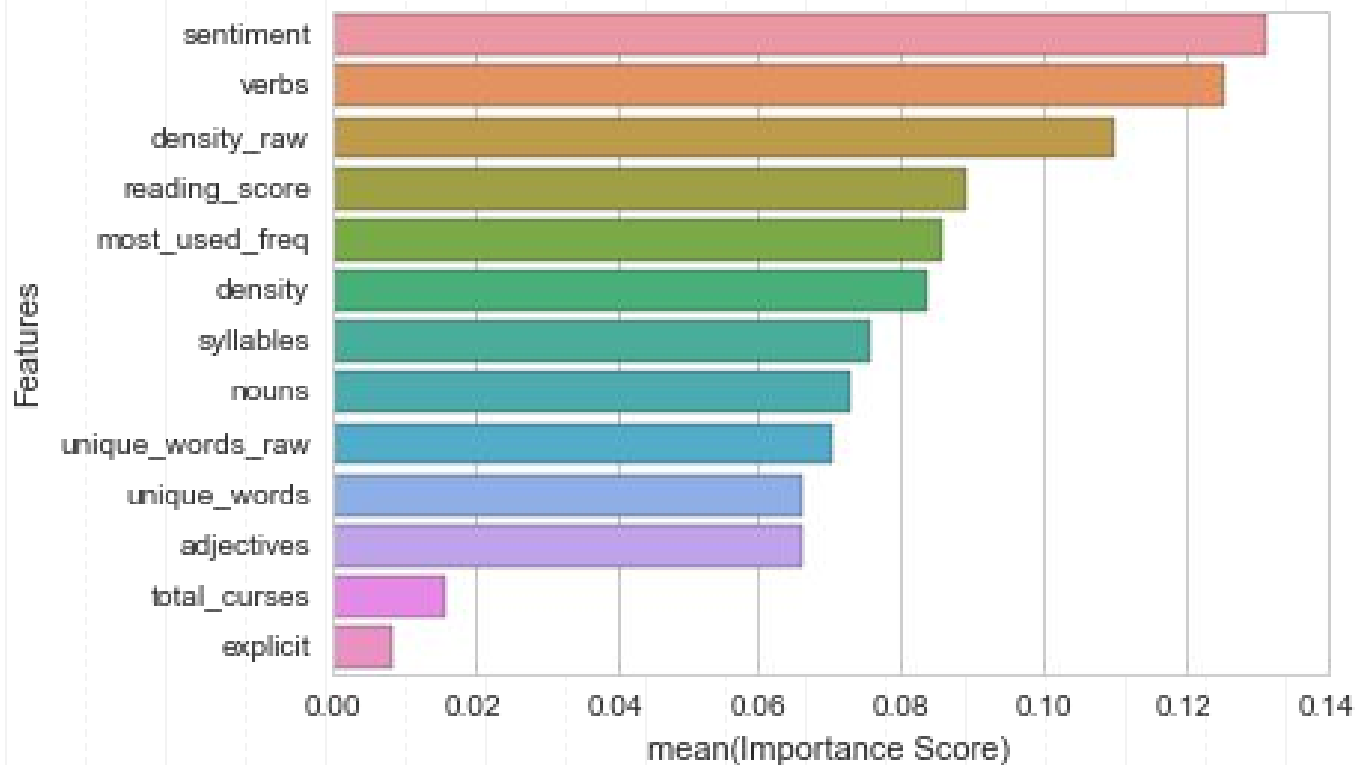
Test Set



# MODEL WITH TIME

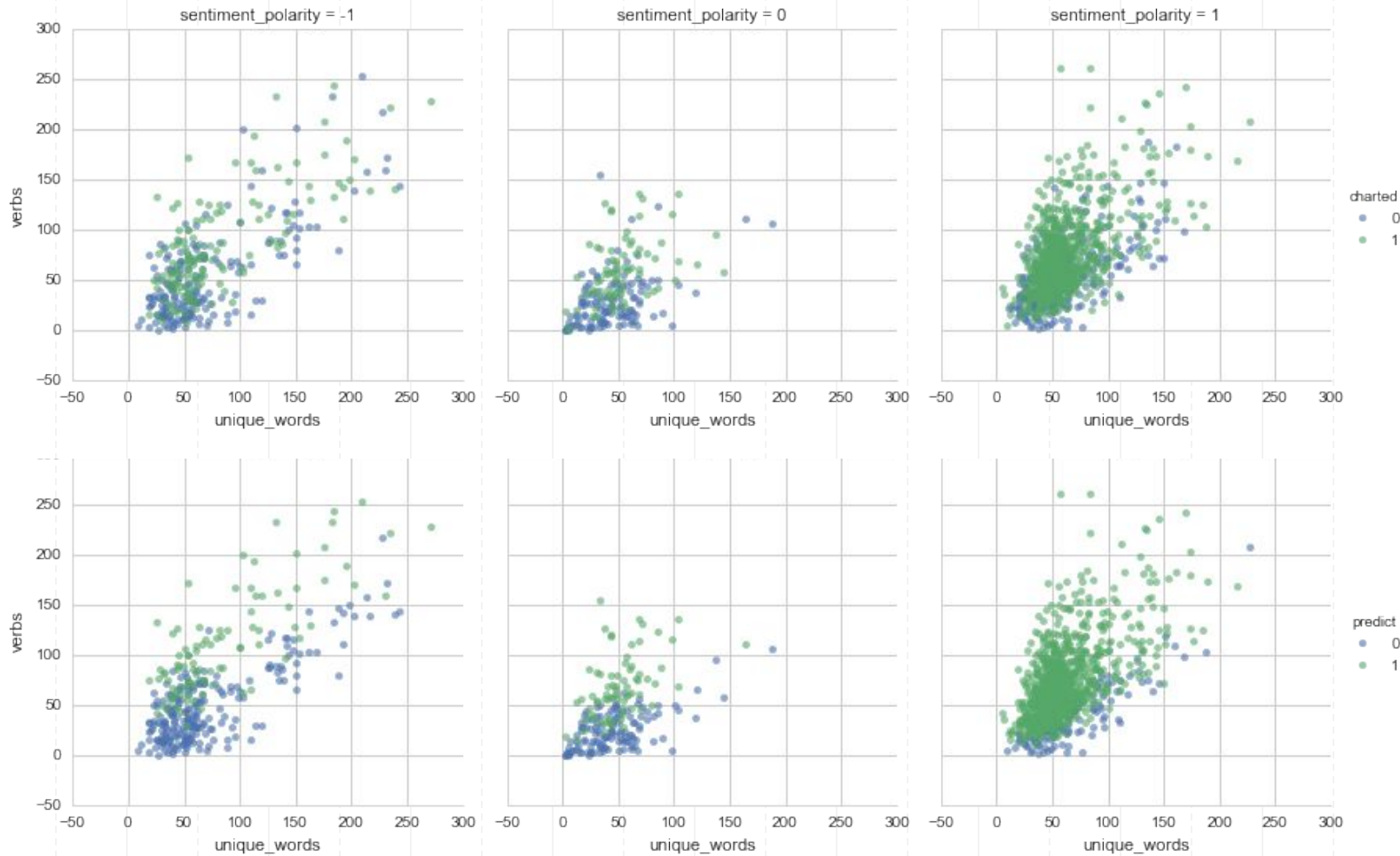


# MODEL FEATURE IMPORTANCE

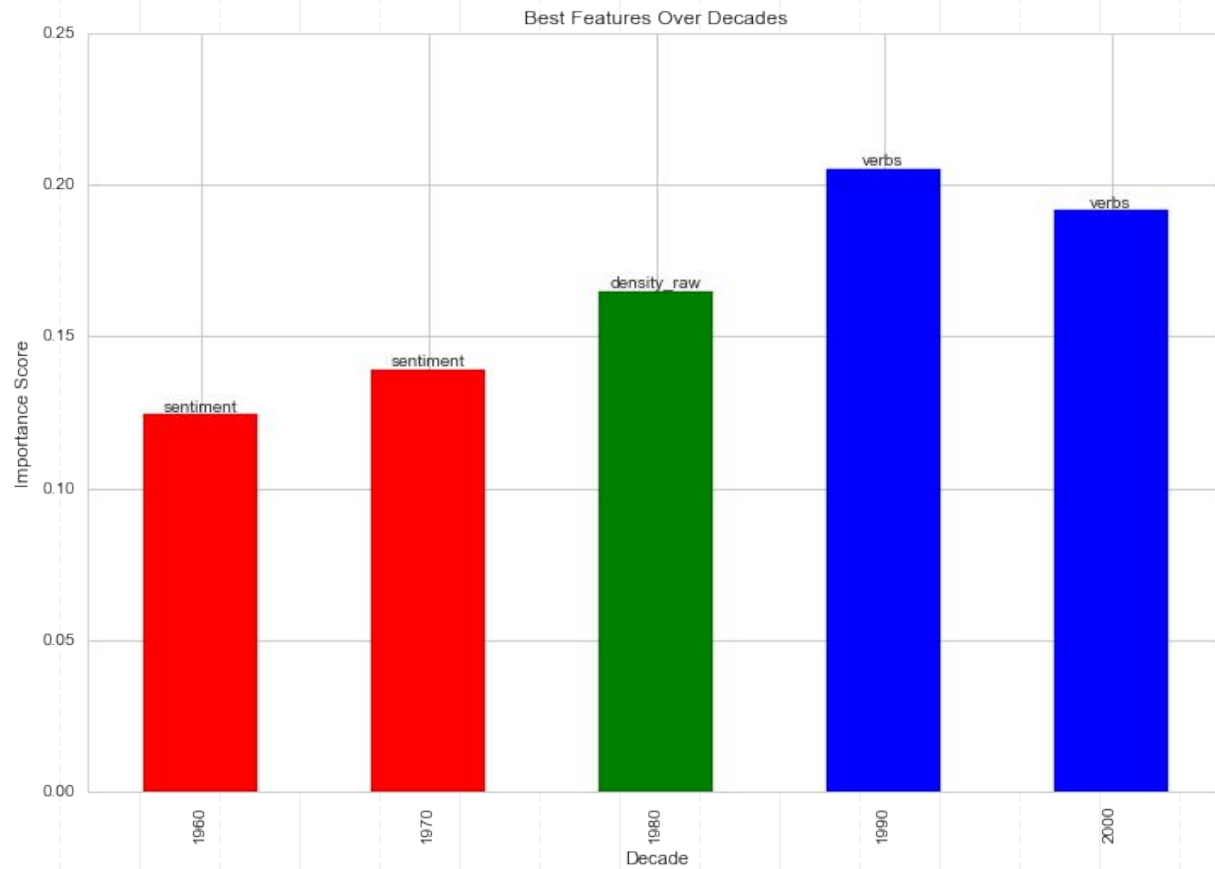




# FINDINGS



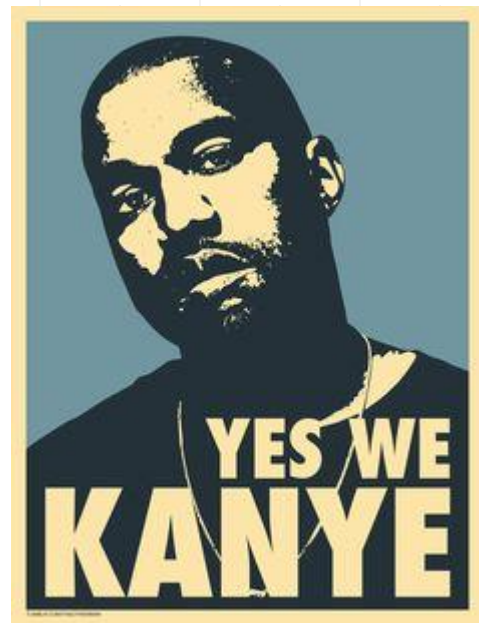
# FINDINGS



# NEXT STEPS

- **Compile my own Million Song Dataset**
- **Improve my reading score function**
- **Plea to gain access to MetroLyrics's API**

- **Sell to Kanye West \$\$\$**



# THANKS!

## Any questions?

You can find me at

[sabbirsphere.com](http://sabbirsphere.com) / [sabbir0ahmed0@gmail.com](mailto:sabbir0ahmed0@gmail.com)

