Data 301: Genre Predictions

JOSH ABEL, KEVIN LABEL

Introduction

- Brainstorming (we couldn't find an old photo of Kevin, so we borrowed one)

Billboard Top 100 Data

- Towardsdatascience.com, credit: Rosebud Anwuri
- Spotify API

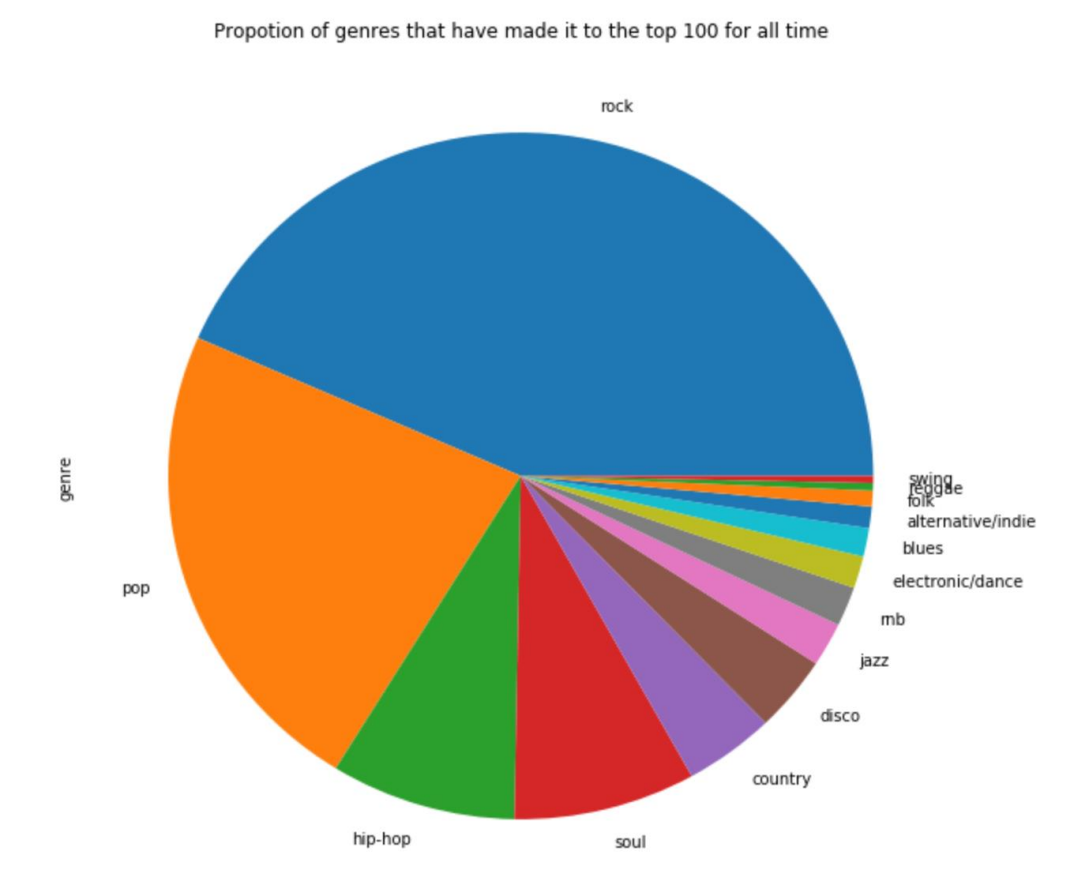
[1]:
```
import pandas as pd
Rosebud_data = pd.read_csv("https://raw.githubusercontent.com/RosebudAnwuri/TheArtandScienceofData/master/The%20Making%
Rosebud_data.head()
```

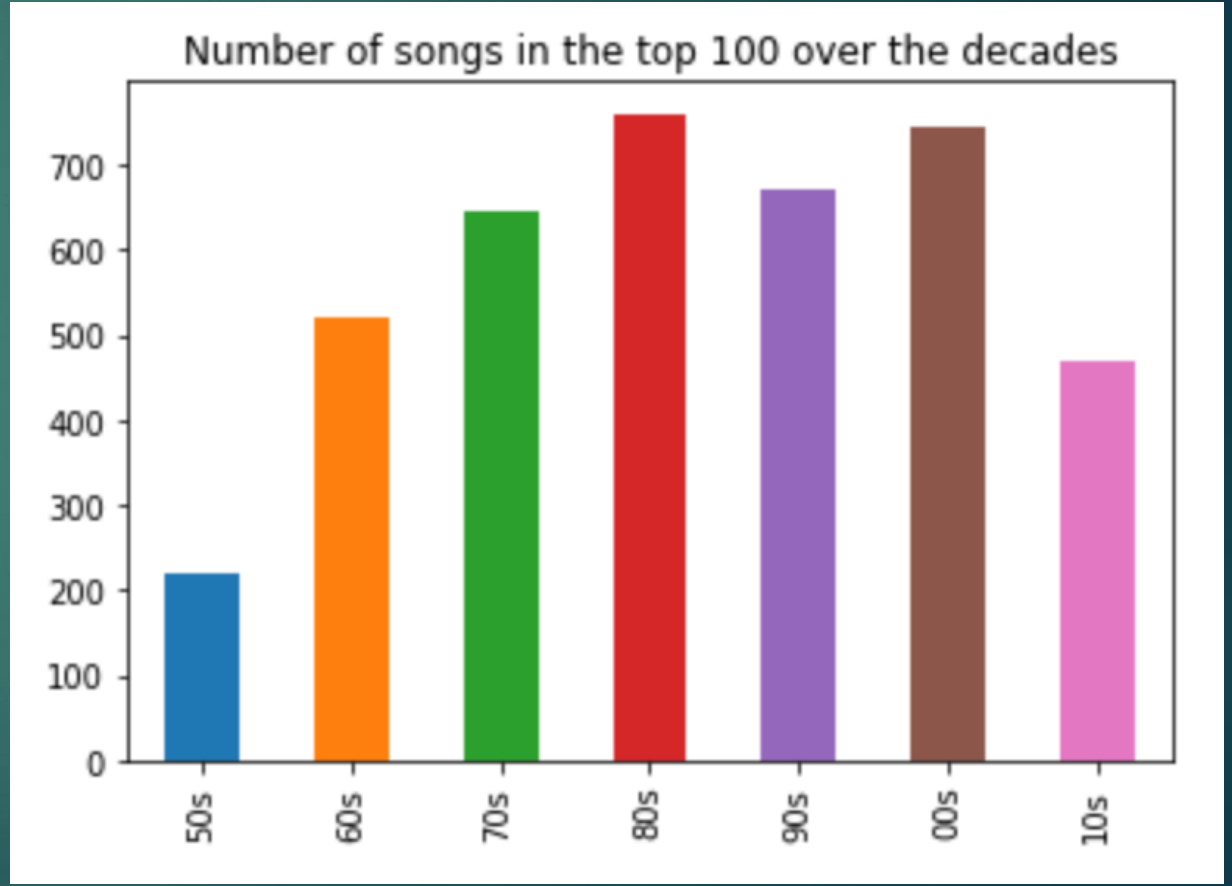
[1]:

	lyrics	num_syllables	pos	year	fog_index	flesch_index	num_words	num_lines	title	f_k_grade	...	tempo	duration
0	Mona Lisa, Mona Lisa, men have named you\nYou'...	189.0	0.199	1950	5.2	88.74	145	17	Mona Lisa	2.9	...	86.198	2075
1	I wanna be Loved\nBy Andrews Sisters\n\nOooo-o...	270.9	0.224	1950	4.4	82.31	189	31	I Wanna Be Loved	3.3	...	170.869	1980
2	I was dancing with my darling to the Tennessee...	174.6	0.351	1950	5.2	88.74	138	16	Tennessee Waltz	2.9	...	86.335	1827
3	Each time I hold someone new\nMy arms grow col...	135.9	0.231	1950	4.4	99.23	117	18	I'll Never Be Free	0.9	...	82.184	1580
4	Unfortunately, we are not licensed to display ...	46.8	0.079	1950	6.0	69.79	32	3	All My Love	6.0	...	123.314	1909

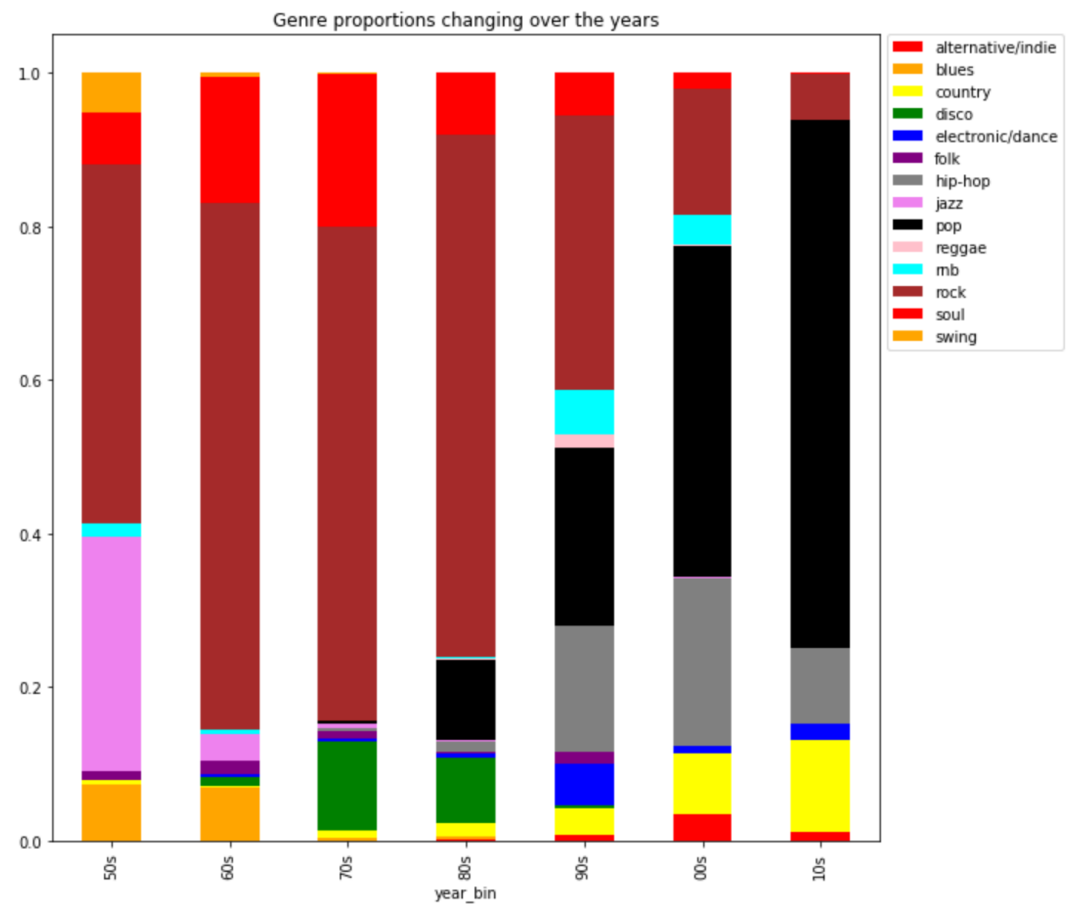
5 rows × 37 columns

Data Observation

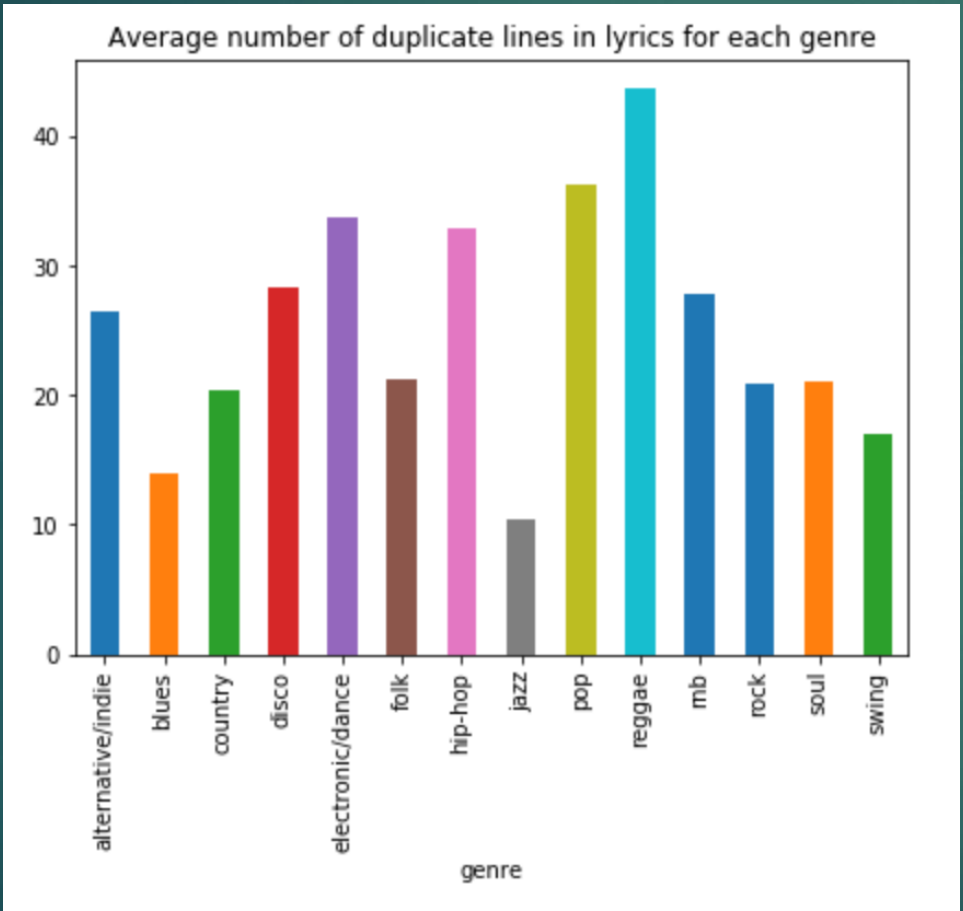




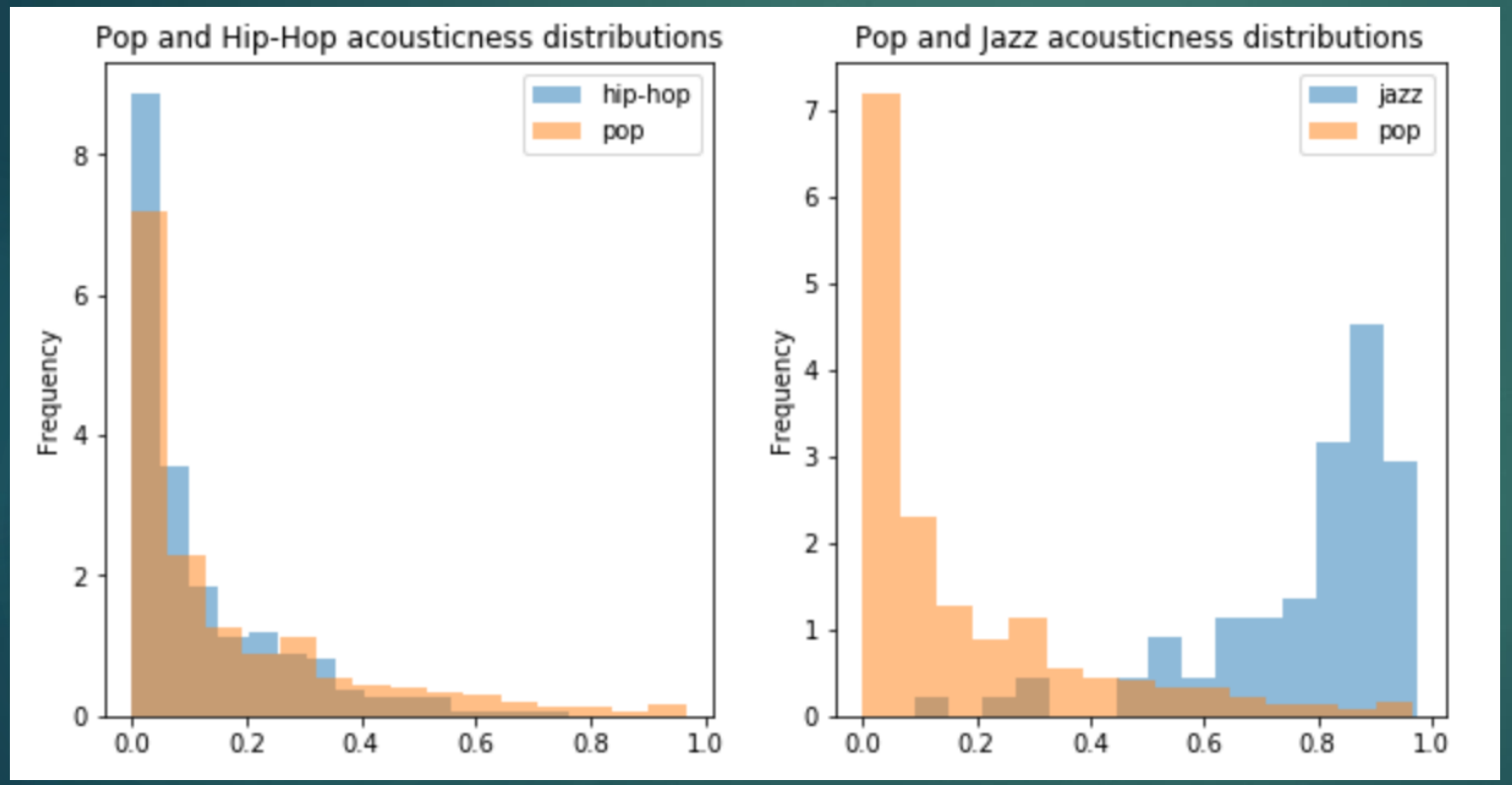
Data Observation (Genre by Decade)



Data Observation (Quantitative Labels)

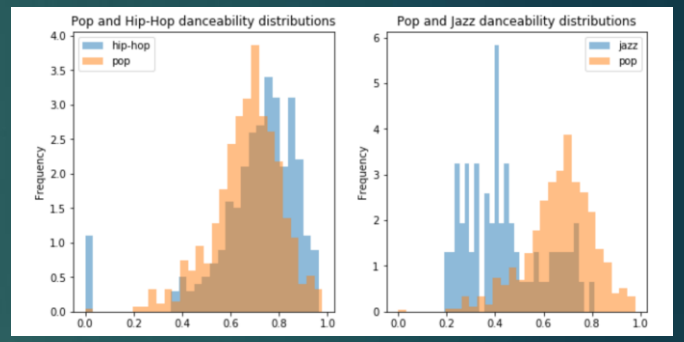


Data Observation (Quantitative Labels cont.)

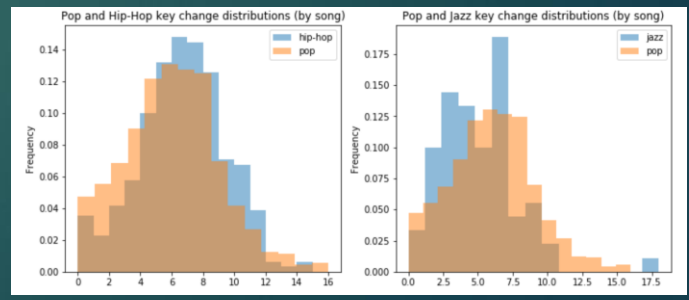


Other Examples

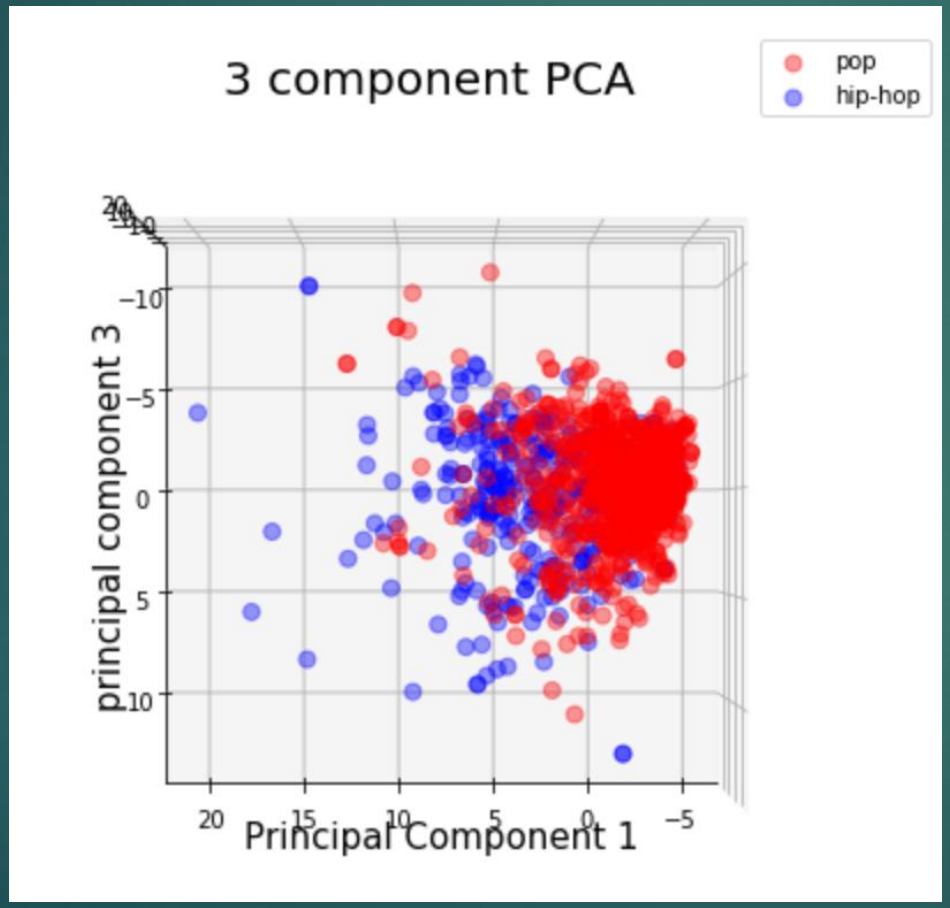
Obvious Feature Difference



Non-Obvious Feature Difference

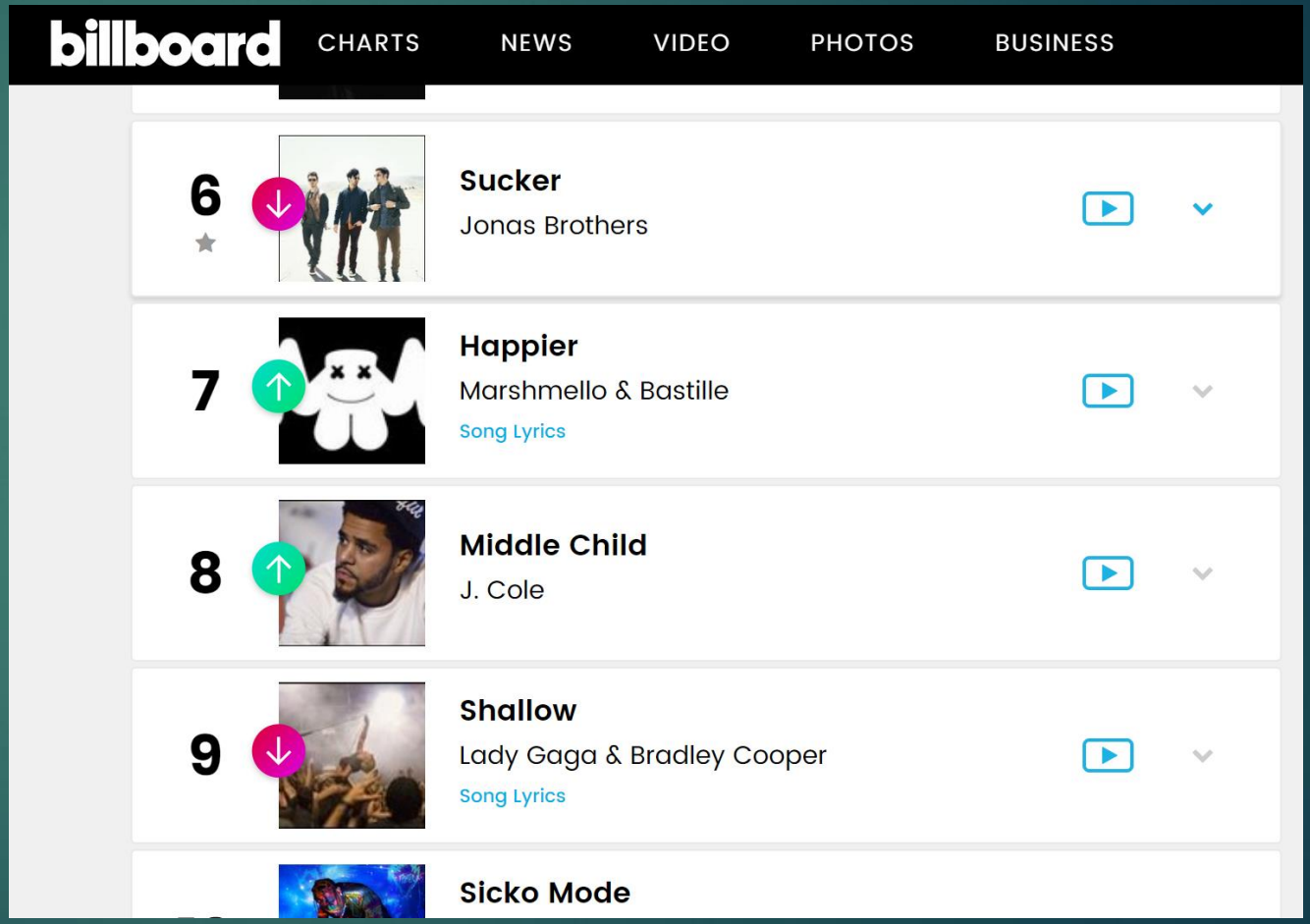


Data Observation (Principal Component Analysis)



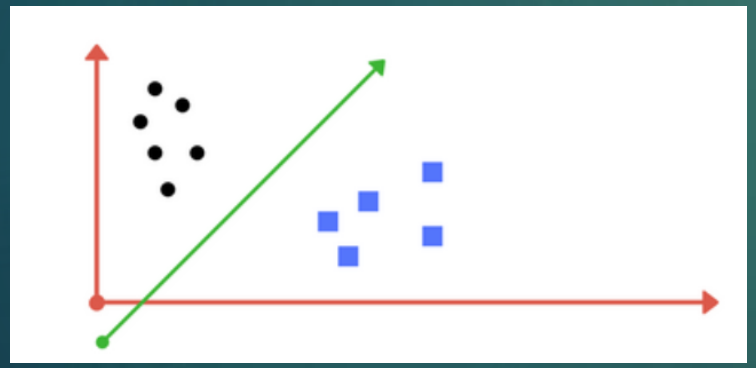
Machine Learning

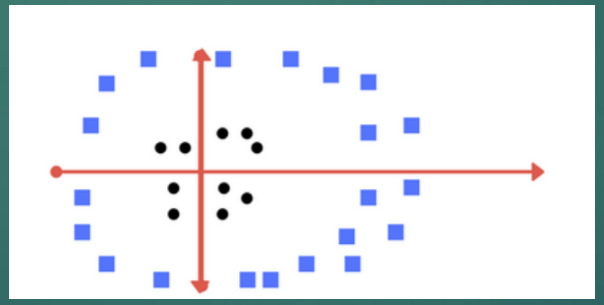
- Process:
  - SVM
  - KNN
  - Ensemble
  - Choose the best model
  - Regression Imputation
  - Run on Current Top 100

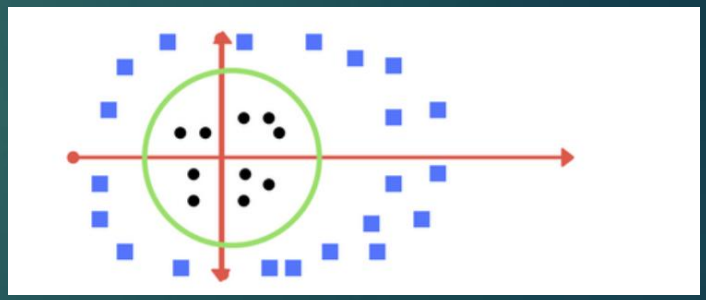


Machine Learning (SVM Model Explained)

- Separate Data by Linear Hyperplane
- Classify
- If not separable, made a new dimension and separate by hyperplane (ex: z = x^2 + y^2)





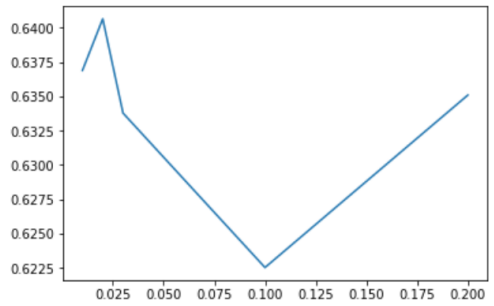


Machine Learning Models and Optimization

- Support Vector Machine (SVM) - best
- K-Nearest Neighbors
- Ensembler

```
%matplotlib inline
test.plot.line()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe8288fac88>
```

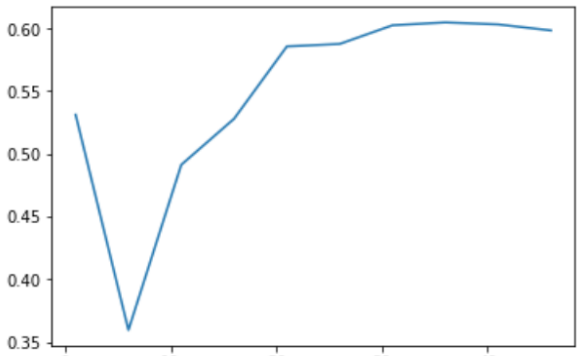



```
test.sort_values(ascending=False)
```

```
0.02    0.640656
0.01    0.636897
0.20    0.635099
0.03    0.633786
0.10    0.622529
dtype: float64
```

For the values that we have tested, the best value for m was 0.02.

```
6     0.359580
11    0.491347
16    0.528065
1     0.531255
21    0.585770
26    0.587730
46    0.598505
31    0.602621
41    0.603307
36    0.604901
dtype: float64
```




```
[8]: eclf1 = VotingClassifier(estimators=[('svc', model1), ('knn', model2)], voting='hard')

[9]: eclf1 = eclf1.fit(X_dict, y_train)

[10]: f1 = cross_val_score(eclf1,
                          X_dict,
                          y_train == "rock",
                          cv=10, scoring="f1").mean()

[11]: f1

[11]: 0.59278649135604111
```

Final Predictions on Top 100

- Regression Imputation
- Predicting for current 100

```
[15]: predictions = model1.predict(X.to_dict(orient="records"))

[16]: predictions[:20]

[16]: array(['rock', 'rock', 'rock', 'rock', 'rock', 'rock', 'rock', 'rock',
       'rock', 'rock', 'rock', 'rock', 'rock', 'jazz', 'jazz', 'rock',
       'rock', 'rock', 'rock', 'rock'], dtype=object)
```

```
[28]: top100_preds = model1.predict(X.to_dict(orient="records"))

[29]: top100_preds

[29]: array(['rock', 'rock', 'rock', 'rock', 'rock', 'rock', 'rock', 'rock',
       'hip-hop', 'rock', 'pop', 'rock', 'rock', 'rock', 'rock', 'rock',
       'hip-hop', 'pop', 'rock', 'rock', 'rock', 'country', 'hip-hop',
       'rock', 'rock', 'rock', 'rock', 'rock', 'rock', 'rock', 'hip-hop',
       'rock', 'rock', 'rock', 'rock', 'rock', 'pop', 'rock', 'rock',
       'rock', 'hip-hop', 'rock', 'rock', 'rock', 'rock', 'rock', 'rock',
       'rock', 'rock', 'pop', 'rock', 'rock', 'rock', 'rock', 'rock',
       'rock', 'rock', 'rock', 'country', 'rock', 'rock', 'rock', 'rock',
       'rock', 'rock', 'rock', 'rock', 'rock', 'rock', 'rock', 'rock',
       'rock', 'hip-hop', 'rock', 'rock', 'rock', 'rock', 'rock', 'rock',
       'rock', 'rock', 'pop', 'rock', 'hip-hop', 'country', 'rock', 'rock',
       'rock', 'rock', 'rock', 'rock', 'rock', 'rock', 'rock', 'rock',
       'rock', 'rock', 'rock', 'pop', 'rock'], dtype=object)

[30]: top100["genre"] = top100_preds
```

Here are the predictions of the genres that were made by the model for the top 100 songs.

```
[31]: top100[["title", "artist","genre"]]
```

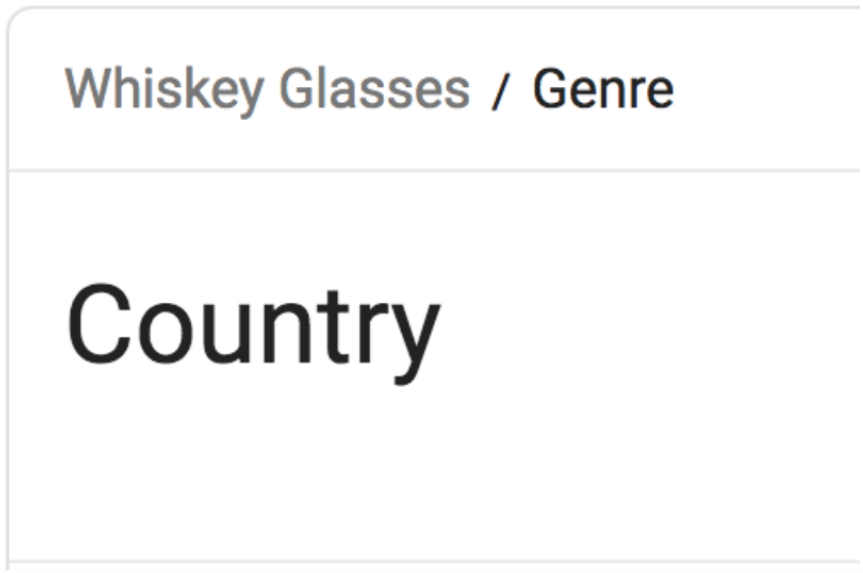
[31]:

	title	artist	genre
0	Sucker	Jonas Brothers	rock
1	7 Rings	Ariana Grande	rock
2	Please Me	Cardi B & Bruno Mars	rock
3	Sunflower (Spider-Man: Into The Spider-Verse)	Post Malone & Swae Lee	rock
4	Without Me	Halsey	rock
5	Shallow	Lady Gaga & Bradley Cooper	rock

Country Observation

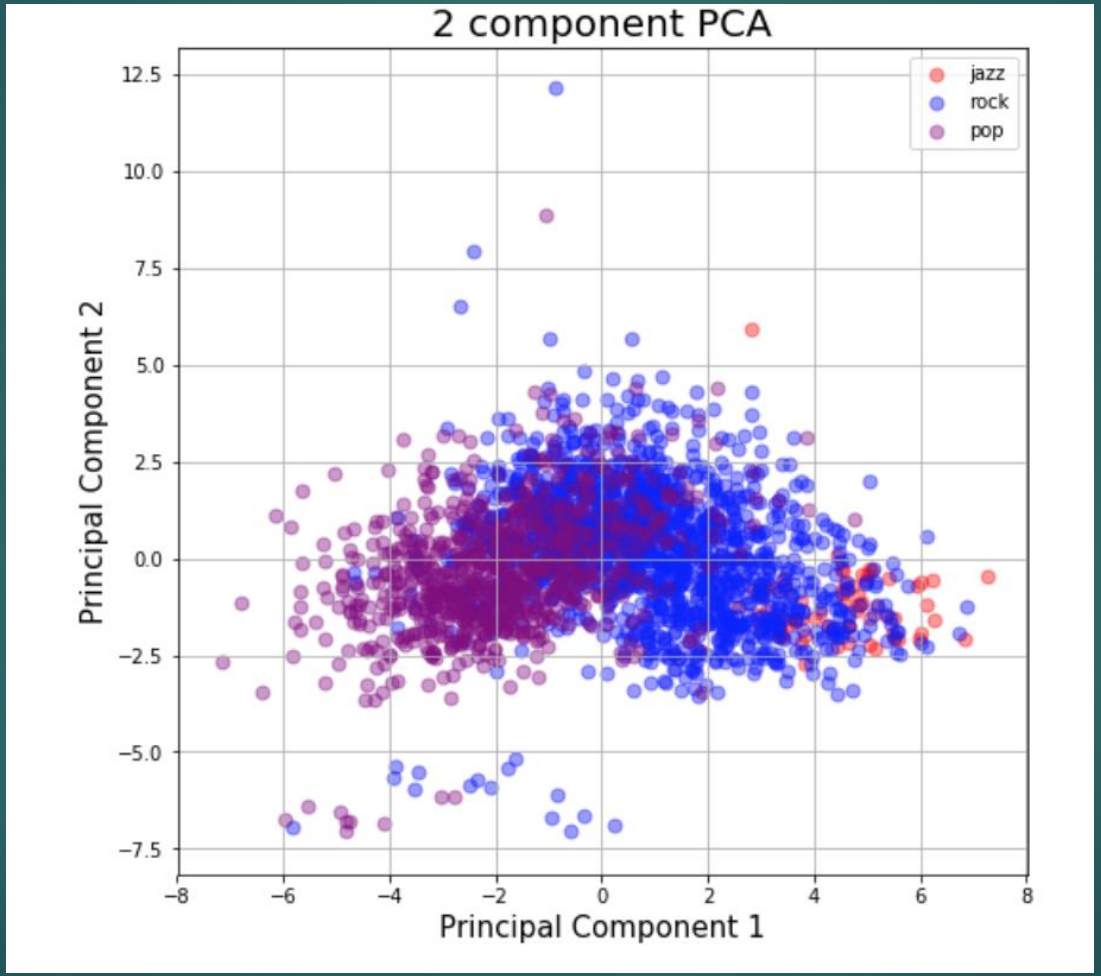
```
[32]: top100[["title", "artist","genre"]].loc[84]

[32]: title     Whiskey Glasses
      artist      Morgan Wallen
      genre             country
      Name: 84, dtype: object
```

We predicted this song to be country and it is!

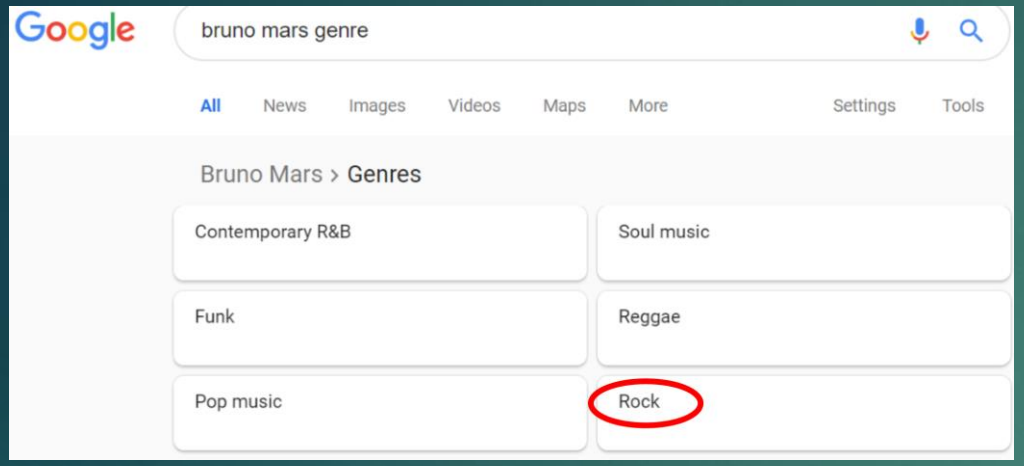


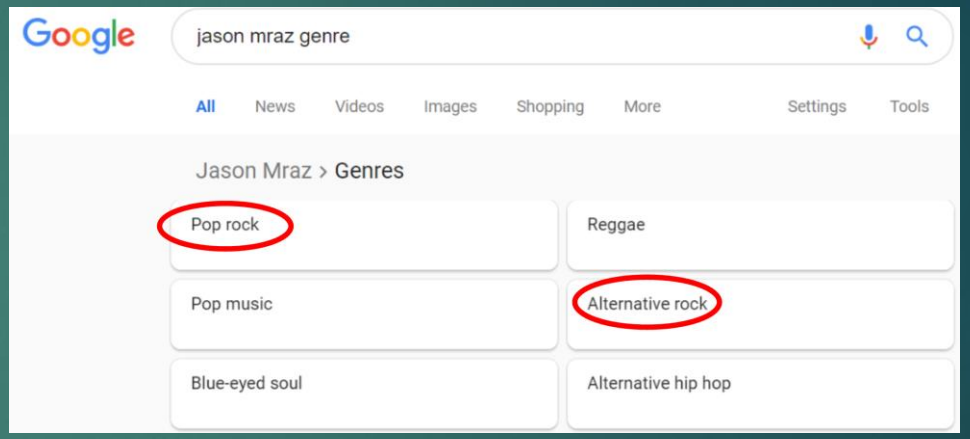

```
[34]: songs.genre.value_counts()

[34]: rock                1887
      pop                  885
      hip-hop              320
      soul                 305
      country              162
      disco                131
      jazz                  77
      rnb                   68
      electronic/dance      55
      blues                 49
      alternative/indie     37
      folk                  27
      reggae                13
      swing                 12
      Name: genre, dtype: int64

[33]: top100.genre.value_counts()

[33]: rock       84
      hip-hop     7
      pop         6
      country     3
      Name: genre, dtype: int64
```

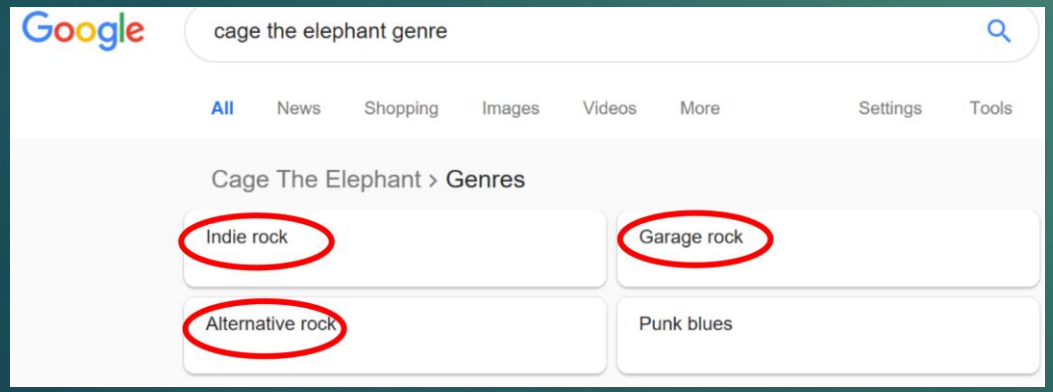
Final Conclusions



Final Conclusions







Thank you

Questions?