PARENTAL ADVISORY
EXPLICIT CONTENT

Explicit songs prediction

Ben Lahav
Ben Bukai

Step by Step..

Research Question

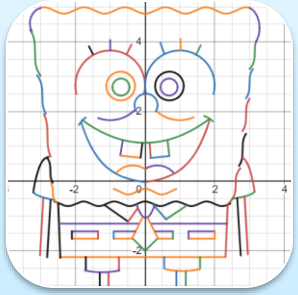
- Can we predict if a song has explicit content?

Data Acquisition

- API - Spotify, Genius & Deezer
- Origin Scraping
  - Britannica
  - Famous-Birthdays
  - Ranker
- Content Scraping
  - AZLyrics

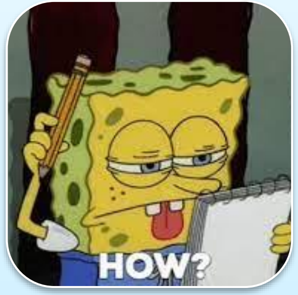
Data Preparation

- Clean the empty, not used in model, description data etc.
- Feature Engineering - parse values, convert data types etc.



EDA

- Visualize the data and make analysis based on it.
- With visualizing, fins the data to correct before the ML phase.



Machine Learning

- Make final corrections on the dataset.
- Try different ML algorithms, hyper-values and scores to find the best pipeline
for this data.

Research Question

- Can we predict if a given song has inappropriate words in it?
- This subject is very hot nowadays and talked about a lot.
- The content that kids and teenagers are exposing to in "legit" media channels has a lot of inappropriate parts.
- Our model can predict explicit content in songs and can be feature or a start of one in a big application that has high explicit-awareness for kids, teenagers and parents.

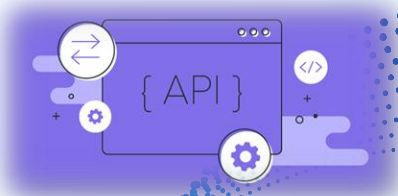
Data Acquisition

API

- We are requesting data via API with pythonic open source packages that makes wrappers over the API and makes the syntax a bit cleaner and readable.
- For getting music tracks - Spotify API
- For getting Lyrics data - Genius API
- For getting explicit content - Deezer API

Scraping

- To make our data richer and predict better, we wrote crawlers that uses Beautiful Soup and Selenium for make the HTTP requests.
- For artists origins - scrape Britannica.com, Famousbirthdays.com, Ranker.com
- For word analysis - scrape AZLyrics.com




```
def manage() -> None:
    """
    Manager of the Data acquisition module.

    :return: new Dataframe
    """
    df = spotify_manager()
    for func in [genius_manager, deezer_manager,
                 lyrics_crawler_manager, artist_origin_manager]:
        df = func(df)
```

Data Preparation

- To prepare the data for EDA & ML phases we need to clean & sharpened it a bit.
- This phase is crucial to predictions success because raw data is mostly not sufficient enough for ML.
- We are cleaning NaN values, converting Ordinal string features to numeric representation, Parse columns and extract relevant data, implementing feature engineering , ensemble data from scrapers to one algorithm etc.

```
def manage(df: pd.DataFrame) -> pd.DataFrame:
    """
    Manager function that runs this preparation module
    with a pipeline of functions.

    :param df: Dataframe from raw data
    :return: Dataframe prepared for visualizations and ML model
    """
    df_copy = encode_column(df.copy(), "artist_name")
    for func in [remove_nan_values, remove_noise_cols, origin_to_country_codes,
                 save_df, popular_genre_dummies, save_df, drop_correlated_features]:
        df_copy = func(df_copy)
        print(f"finished {func.__name__}")

    return df_copy
```

Data Preparation -

Feature Engineering

artist_genres
('gauze pop', 'indietronica', 'shiver pop')
()
('australian hip hop',)
('glam rock', 'mellow gold', 'piano rock')
('british soul', 'pop', 'pop soul', 'uk pop')
('canadian contemporary r&b', 'canadian pop', 'pop')
('pop', 'uk pop')
('afro r&b',)
('lgbtq+ hip hop', 'pop')
('modern rock', 'pop')
('modern rock', 'rock')
('canadian pop', 'pop')
('canadian contemporary r&b', 'canadian pop', 'pop')
()
('dance pop', 'pop')
('alt z', 'gen z singer-songwriter', 'pop')

other_genre	disco	folk	funk	hip hop	electro	r&b	soul	metal	house	edm	country	rap	rock	pop
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

artist_name
Glass Animals
GAYLE
The Kid LAROI
Elton John
Adele
The Weeknd
Ed Sheeran
CKay
Lil Nas X
Jaymes Young
Imagine Dragons
Justin Bieber
The Weeknd
ACRAZE
Doja Cat
Lauren Spencer-Smith

artist_name
736
1851
604
51
1893
582
325
1117
879
808
970
1893
31
536
1093
422

S
origin
Oxford, England
Australia
London, England
London, England
Toronto, Canada
Hebden Bridge, England
Nigeria
Lithia Springs, GA
Washington
Las Vegas, NV
London, Canada
Toronto, Canada
Staten Island, NY
Tarzana, CA
Portsmouth, England
London, England
Bayamon, PR

	N
	country_code
0	826
0	36
0	826
1	826
1	124
1	826
0	566
1	840
1	840
0	840
1	124
1	124
0	840
1	840
1	826
1	826

Data Preparation -

Data Cleaning

```python
def remove_noise_cols(df: pd.DataFrame) -> pd.DataFrame:
    not_valid_cols = [col for col in df.columns if "Unnamed" in col.split(":")] + ["track_uri", "track_name", "album"]
    return df.drop(columns=not_valid_cols)
```

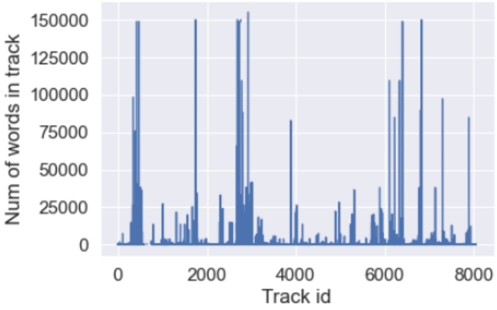
```python
def parse_column(df: pd.DataFrame, col: str) -> pd.DataFrame:
    """
    Function that parses string saves Series column to Tuple.
    This is for the `artist_genres` column.

    :param df: the dataframe to change.
    :param col: the column to parse.
    :return updated Dataframe
    """
    df_copy = df.copy()
    df_copy[col] = pd.Series([eval(instance[col]) for idx, instance in df_copy.iterrows()])

    return df_copy
```

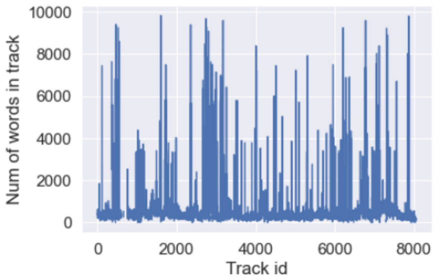
```python
def remove_nan_values(df: pd.DataFrame) -> pd.DataFrame:
    return df.dropna()
```

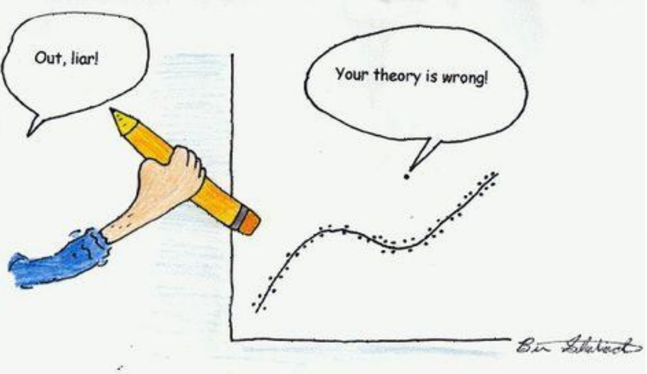
Data Preparation -
Detect and remove Outliers



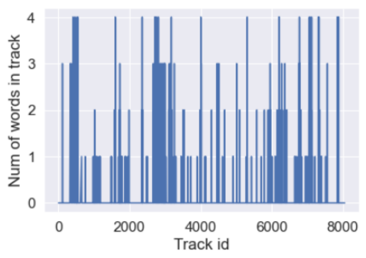
Outliers Detection

Removing Outlie





Scaling the data



EDA

Name

- Correlation matrix heatmaps.

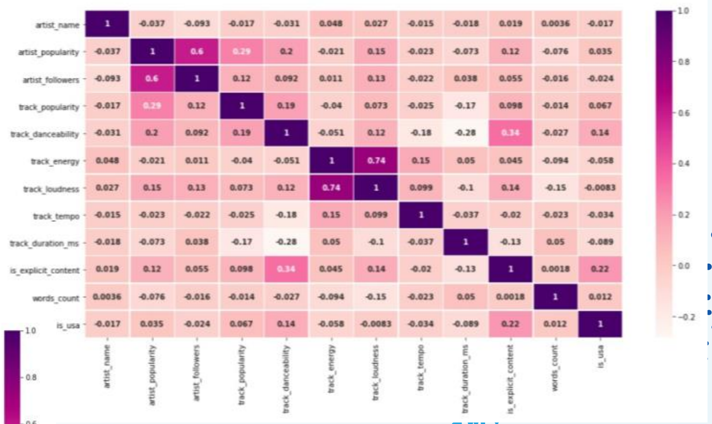
What can we see

- Strong or Weak (color) relationship between features

Conclusion

- Rap and hip hop are similar
- Track loudness and track energy are similar

Rest of dataset correlation matrix



re correlation matrix

EDA - scatter plots on features

Name

- Scatter plots to see insights from correlation matrix.

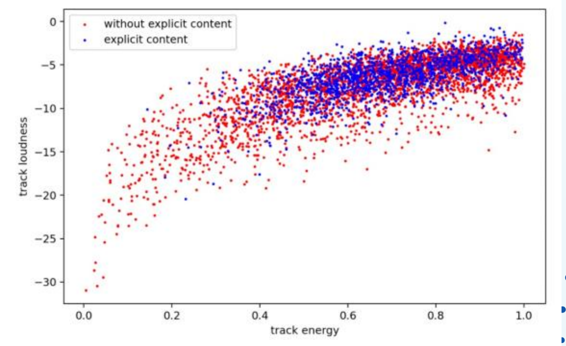
What can we see

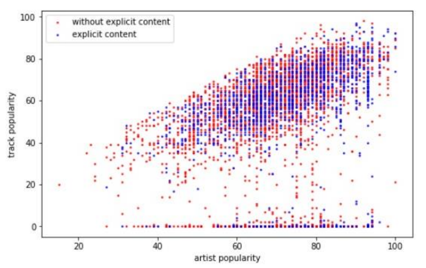
- Features that where strongly relation in corr matrix

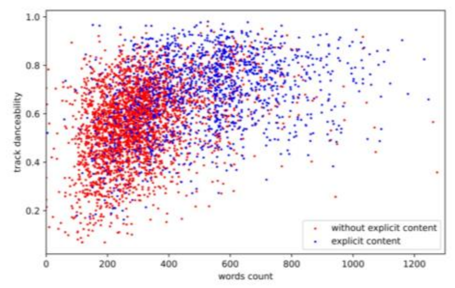
  and some that didn't.

Conclusion

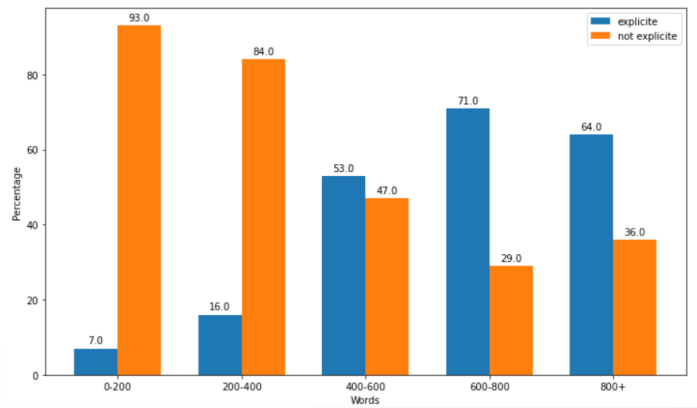
- We can see supported scatter plot for the relation between track energy and loudness.
- The others are less, we can see that they don't contruct "linear" form







EDA - bar plots

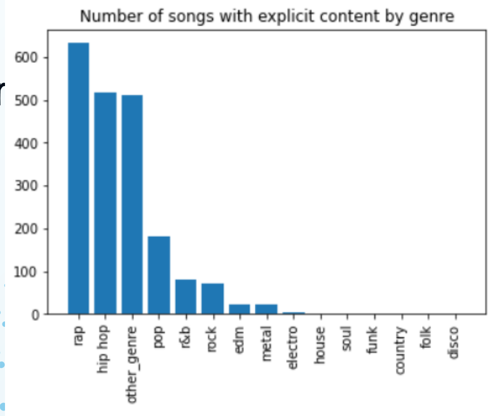


Name

- Bar plots that evaluates one-three features in a histogram bucket form.

What can we see

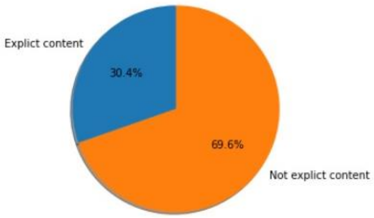
- Distribution with Bar plots with amount of songs in explicit content songs and not grouped by genr
- Distribution on number of words in song with ranges and explicit and not amount of songs.

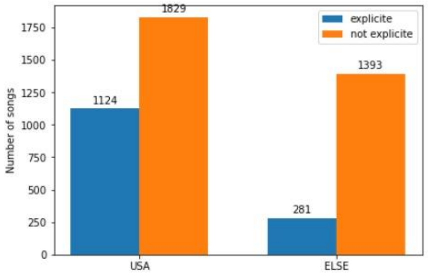


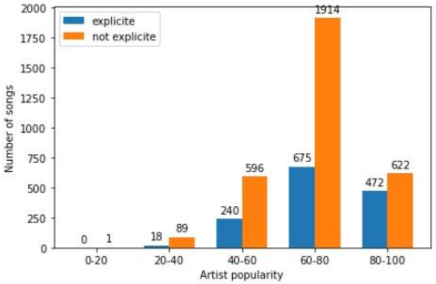
Conclusion

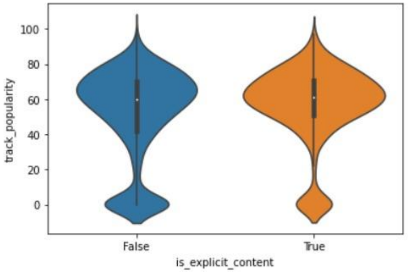
- Most of the explicit content comes from Rap, Hip Hop.
- As the number of words go high in a song, the chance to have explicit content is greater than that it wouldn't.

MORE RELEVANT EDA EXAMPLES

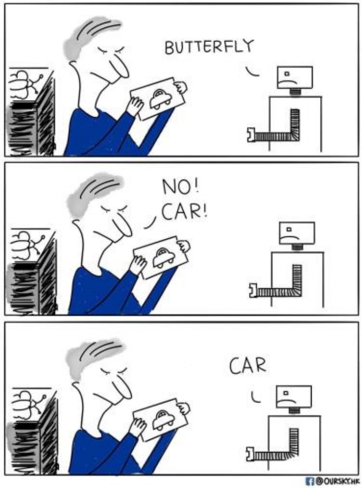


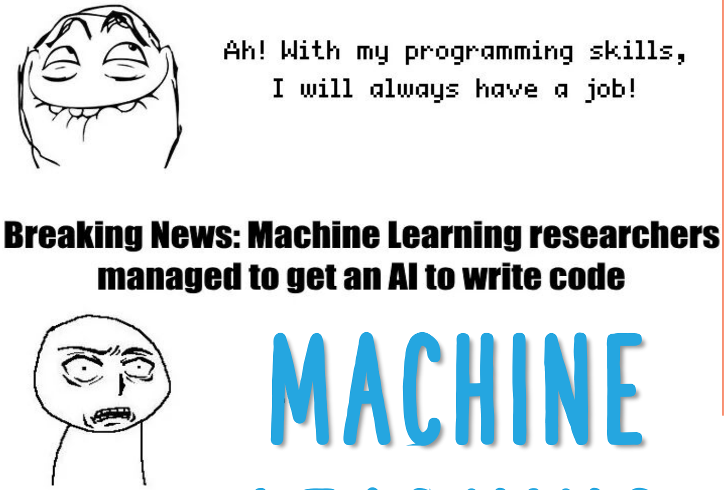


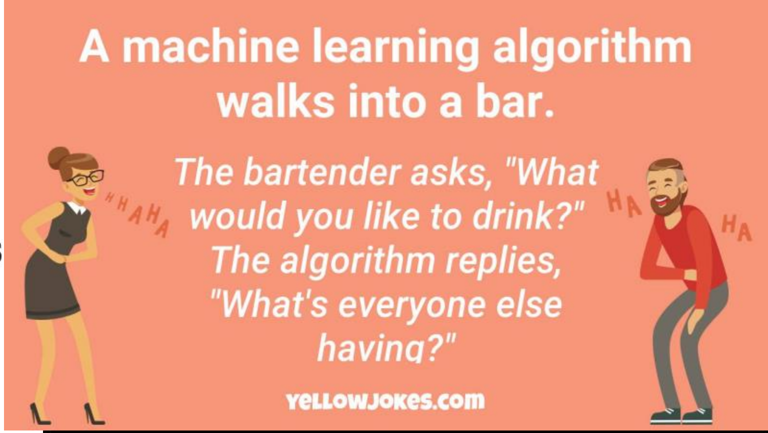


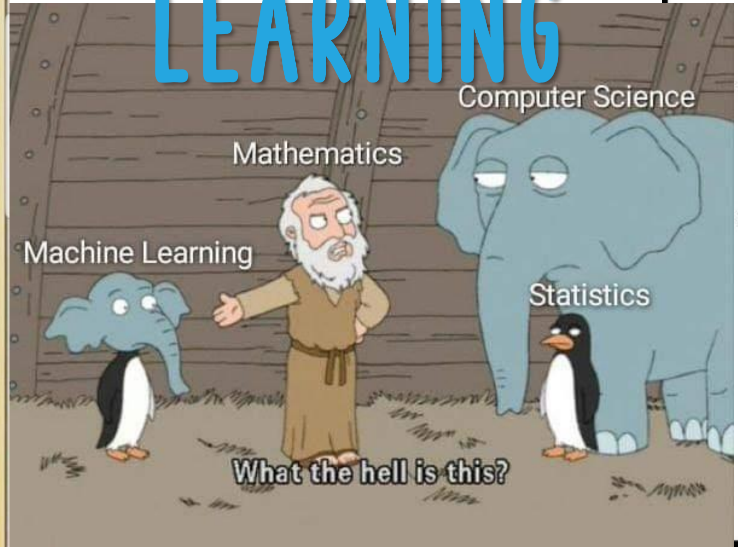


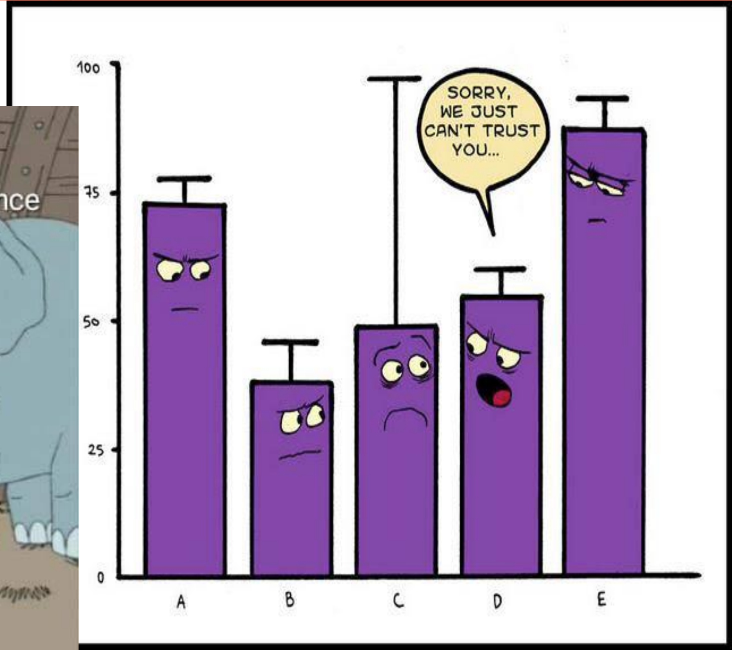
MACHINE LEARNING











Machine Learning

- With the help of GridCV function of sklearn.model_selection we could find the best algorithms that would suit our need to answer this research question.
- We chose 3 algorithms and their hyper-parameters values as input (after filtering some).
  - As we see Random Forest gives us the best score!
  - As we can see in the classification report that we did for Random Forest with the best parameters, it's easier to find not explicit content then to predict true positive one.

```python
params_decision_tree = {"max_depth": [7,9,15], "min_samples_split": [3,5,7,10]}
params_random_forest = {"n_estimators": [150,200, 250],"max_depth": [7,11,15]}
params_knn = {"n_neighbors": range(3, 20)}

clf_params = [(DecisionTreeClassifier(), params_decision_tree),
              (RandomForestClassifier(), params_random_forest),
              (KNeighborsClassifier(), params_knn)]


for idx, (clf, params) in enumerate(clf_params):
    clf_cv = GridSearchCV(clf,params,cv=10)
    clf_cv.fit(X,y)
    print(alg_names[idx])
    print("==========================")
    print(f"best params are: {clf_cv.best_params_}")
    print(f"best score is: {clf_cv.best_score_}")
```

```
Decision Tree
==========================
best params are: {'max_depth': 7, 'min_samples_split': 3}
best score is: 0.8213048214454863
Random Forest
==========================
best params are: {'max_depth': 15, 'n_estimators': 250}
best score is: 0.8528587666950838
KNN
==========================
best params are: {'n_neighbors': 4}
best score is: 0.7590532348204982
```

```python
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, f1_score

X = df.drop(columns=["is_explicit_content"])
y = df["is_explicit_content"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

clf = RandomForestClassifier(max_depth=15, n_estimators=250)
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)

targets=["without explicit content", "with explicit content"]

print("Test Results:")
print("==============")
print(classification_report(y_true=y_test, y_pred=y_pred, target_names=targets))
```

```
Test Results:
==============
                          precision    recall  f1-score   support

without explicit content       0.87      0.95      0.91       618
   with explicit content       0.87      0.70      0.77       296

                accuracy                           0.87       914
               macro avg       0.87      0.82      0.84       914
            weighted avg       0.87      0.87      0.86       914
```

Next steps to improve prediction

- The data is not even, there are 1/3 explicit songs and 2/3 that aren't, the classifier has a hard time due to misbalancing, more instances that have explicit content would improve prediction.
- The genre feature in Spotify is a Cowboys land, the artist tags his songs with tags that he can create, this made us engineering this feature a lot. With reducing/make better labels of genres, the prediction would be better.
- Converting origins to continents or regions, a lot of explicit content songs comes from Latin America also but making the feature binary (USA/not) without this ability was better than 2000 countries before data preparation.