Capstone Presentation

By: Adina Steinman

Business Problem

- Business goal is to build a recommendation system that can recommend new songs for brand new users to an app
- This is achieved through two stages: a classification analysis and an SVD-based recommendation system
- Classification analysis can be used prior to the rec system in order to eliminate the “cold start problem”; if ratings for a particular song are not present, we can run the song through our classification model to get a predicted rating to later be used in the recommendation system
- Recommendation system can then be used to generate ratings for a new user and to recommend songs based on a user’s existing ratings

Datasets

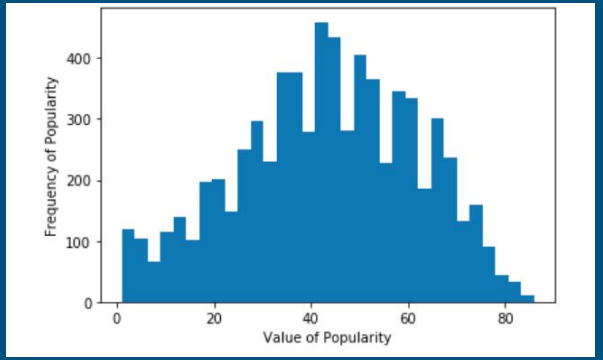
- Spotify API and the Spotipy wrapper were used
- Retrieved various forms of data from the API, including:
  - Artist data
  - Playlist track data
  - Genre data
  - Track audio features data
- Final dataset included 10,000 songs from a single playlist with 26 columns regarding features of the playlist's songs

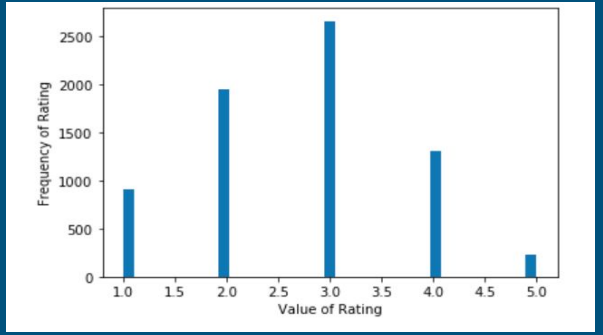
Feature Engineering

- Created a set of dummy variables that specified which decade the song was played in; it could be that newer songs have higher ratings than older songs due to their increased popularity
- Built a ‘ratings’ variable that compressed the original “popularity” measure into a 1-5 scale in order to more easily differentiate ratings from one another
    - Ratings were generated so that they matched the same overall distribution as the popularity scale
- Dataset contained over 500 genres; data was put into “buckets” that contained mainstream genre categories (pop, dance, etc.) and then each unique genre was transformed into a dummy variable

Exploratory Data Analysis

- Audio features: EDA looked to identify if each audio features distribution varied across different value of ratings
  - Higher rated songs had higher danceability, energy, valence, and duration
- Years: EDA was done to see if songs from recent years were rated more favourably than older songs
  - While the playlist mostly contained newer songs, older songs on average received higher ratings
- Genres: most popular genres across the playlist were dance, pop and rock
- Tracks: EDA was done to see top songs within each genre; will then do post-EDA on our recommendation system to see if these results differ





Vanilla Model Results: Classification Analysis

model	train_accuracy	train_precision	train_recall	train_f1	test_accuracy	test_precision	test_recall	test_f1
knn	0.555	0.555	0.555	0.555	0.348	0.348	0.348	0.348
bayes	0.375	0.375	0.375	0.375	0.382	0.382	0.382	0.382
decisiontree	0.996	0.996	0.996	0.996	0.299	0.299	0.299	0.299
bagging	0.995	0.995	0.995	0.995	0.370	0.370	0.370	0.370
randomforest	0.996	0.996	0.996	0.996	0.398	0.398	0.398	0.398
Adaboost	0.408	0.408	0.408	0.408	0.393	0.393	0.393	0.393
XGBoost	0.503	0.503	0.503	0.503	0.403	0.403	0.403	0.403
SVM	0.461	0.461	0.461	0.461	0.402	0.402	0.402	0.402

- Goal is to optimize precision: therefore XGBoost and SVM were turned further

XGBoost Model

1. XGBoost
    - Gridsearch was performed to include hyperparameters gamma, learning rate, subsample size, colsample by tree and number of estimators
    - Test Precision improved only slightly; could be due to other extraneous factors not included in the model that could impact ratings (not simply audio features)
    - Used the predict_proba function of the XGBoost model to generate probabilistic ratings for the model; this will provide more precise ratings for songs and not restrict ratings to the integer values in the 1-5 scale

SVM Model

2. SVM Model

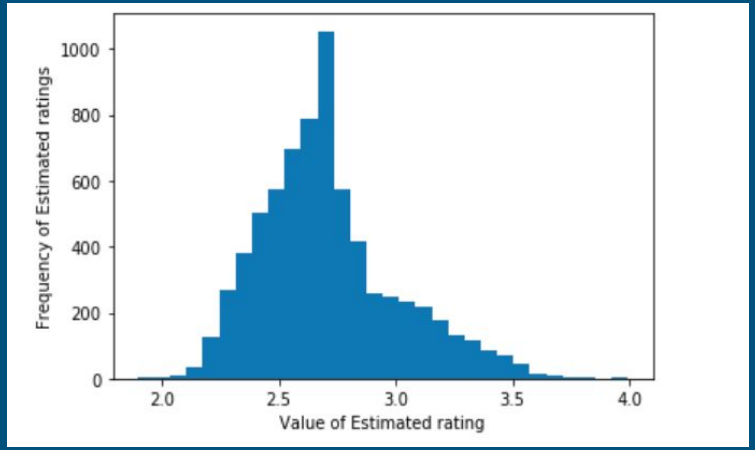
- Performed GridSearch to include hyperparameters C, gamma, decision function shape, and kernel
- Result improved test precision from 0.402 to 0.427; a moderate increase comparable with the improvement from the XGBoost model

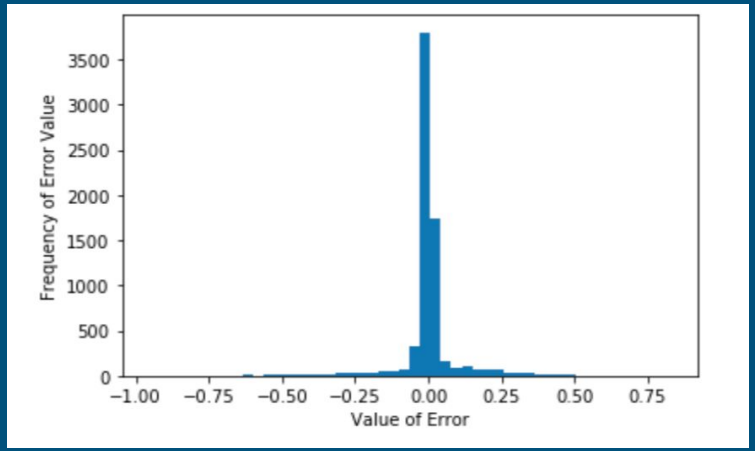
Recommendation System

- Used “artist_id” as a proxy for “user_id”, and “track_id” as a proxy for “item_id” in the surprise model
- Two different models were run; one with original integer ratings, another with the probabilistic ratings
  - For original ratings, several models were run: KNNBasic, KNNBaseline, KNNWtihMeans, and SVD. SVD performed the best with an RMSE of 0.875
  - For probabilistic ratings, SVD model was run and RMSE was 0.245, a significant increase from the integer ratings result.
  - The probabilistic model will be our final chosen model, as the errors are significantly smaller, which makes sense as the decimal value of ratings allowed for the differences between the estimated and original ratings to decrease

Post-Modeling EDA

- Post-model EDA was done on the estimation of the probabilistic ratings from the XGBoost model
- Distribution of the estimated ratings shows a peak at approx. 2.75
- Distribution of errors is centered around 0, with majority of errors in the -0.25 to 0.25 range





Conclusions and Future Work

- While the classification model addresses the cold start problem, the XGBoost model still has a high RMSE. Future work could look at including additional features that may impact a song's popularity
- Merging datasets with other APIs, such as the Genius API, or any other music-related API, to have more features could improve the results of classification models
- Supplement SVD model to include feature analysis with LightFM
- Improve the sparsity of the recommendation model matrix by finding additional data to use as artist IDs, either with songwriter or producer IDs (can either find ways to extract this from Spotify API or again use the Genius API to retrieve this information)

Thank You!

- GitHub Repo: https://github.com/adinas94/Spotify_Recommendation_System
- Spotify API Wrapper: https://spotipy.readthedocs.io/en/2.16.1/
- Linkedin: https://www.linkedin.com/in/adinasteinman/
- Email: adinasteinman@gmail.com