USER CHURN PREDICTION AND RECOMMENDATION ENGINE

Ruchita Rozario (301359835) | Slavvy Coelho (301359836)

1. Problem Definition

Churn rate, in its broadest sense, is a measure of the decline in the number of users using a certain service over a specific period. It is one of the major factors that determine the steady-state level of customers a business will support.

In this project, we aim to work on the KKBOX, Asia's leading music service dataset. The important questions that we wish to address through this project are the following:

a. What crucial factors should be kept in mind to determine whether or not the user will churn?
b. Can we devise a model that will predict whether a user will churn?
c. Can we create a recommendation system to retain the users that might churn based off the users' history?

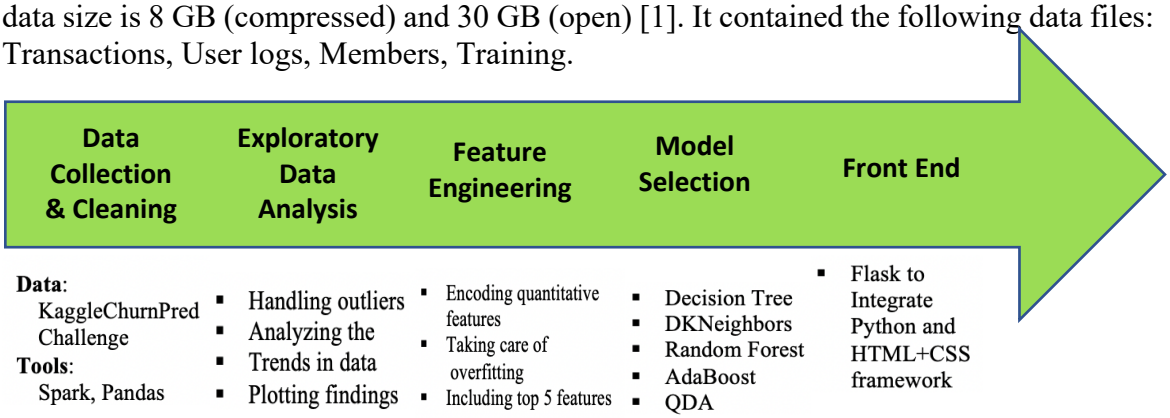
To address the above problems, we've divided our project into 2 major modules:

a. User Churn Predictor
b. Songs Recommender – User history based

2. Data and Pipeline

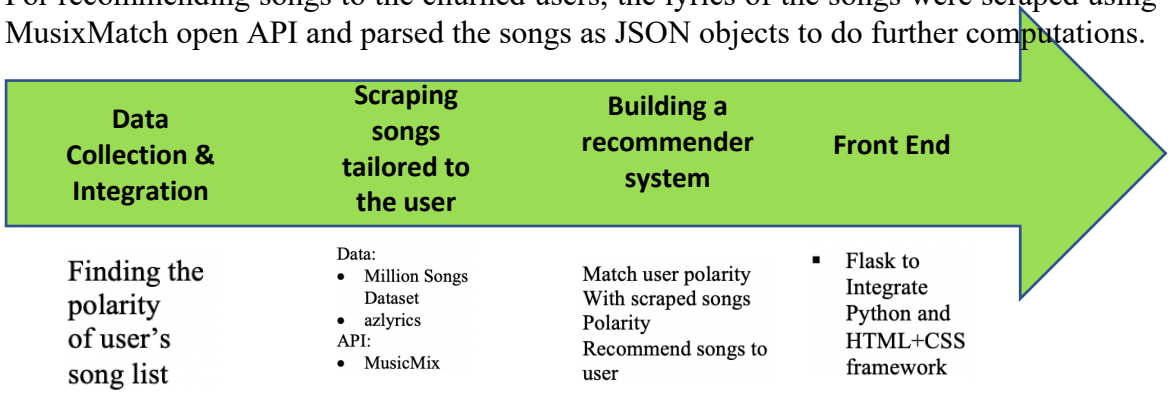
User Churn Prediction:

The data for prediction model was acquired from Kaggle's churn prediction challenge. The data size is 8 GB (compressed) and 30 GB (open) [1]. It contained the following data files: Transactions, User logs, Members, Training.



Songs Recommender:

For recommending songs to the churned users, the lyrics of the songs were scraped using MusixMatch open API and parsed the songs as JSON objects to do further computations.



3. Methodology

User Churn Prediction:

- *Data Collection and EDA*

  In order to ensure that our model is able to handle huge amount of user data, we deployed the decided model on Apache Spark which is an open-source cluster-computing framework. The idea of using distributed computing engines is to distribute the calculations to multiple low-end machines (commodity hardware) instead of a single high-end one. This definitely speeds up the learning phase and allows us to create better models and ensures scalability.

  We initially took care of outliers and missing data. All the qualitative variables were encoded (eg: male, female = 0, 1). We further performed analysis and generated the following plots:

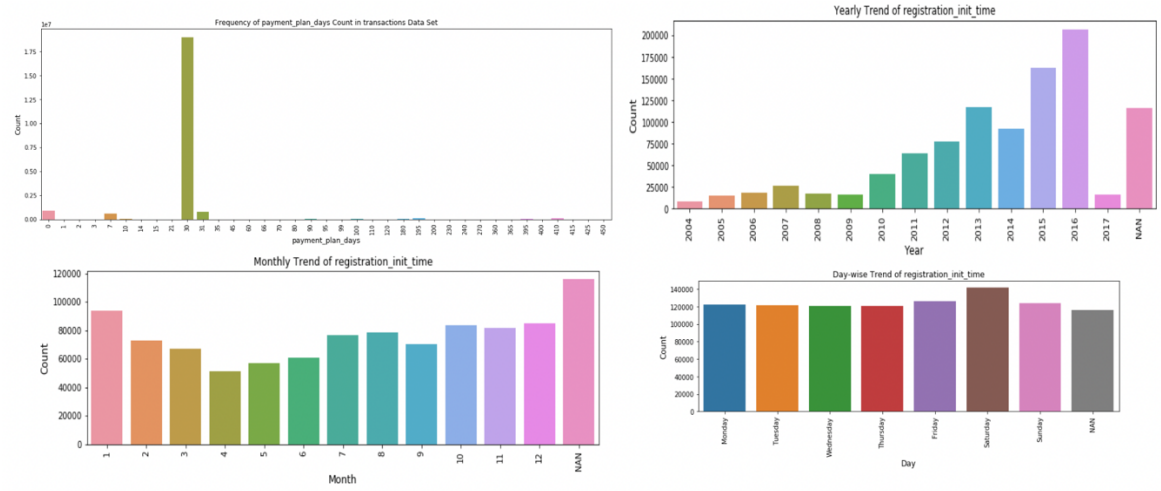


Fig: Analysis on transactions data

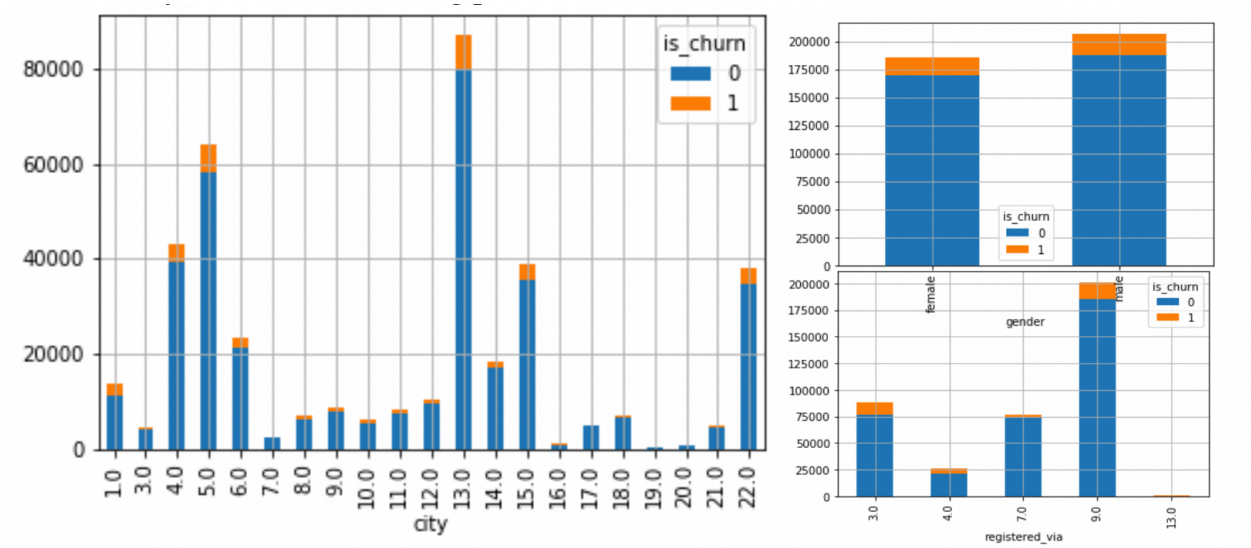
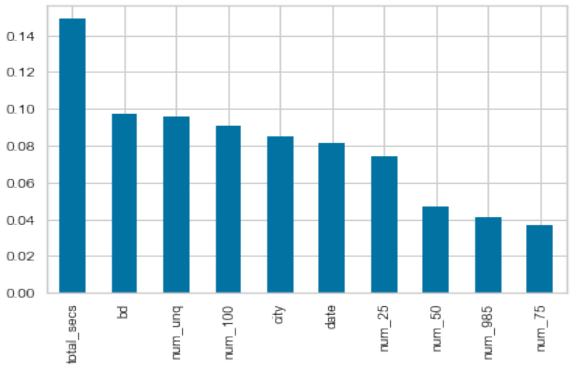


Fig: Analysis based on churning patterns

- *Feature Engineering*

  In order to include only relevant features in the model (i.e. features that influence the prediction) and the avoid overfitting, we computed the feature importances of all the features and included the top 5 most important features in the prediction model. Also, quantitative features were encoded as integer values.

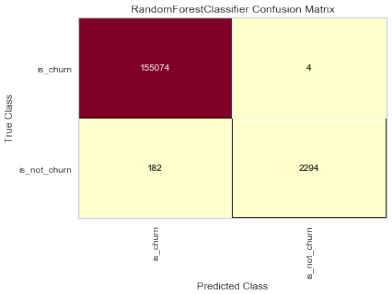


- *Model Selection*

  For our prediction, we used the following 5 models are following are the results:

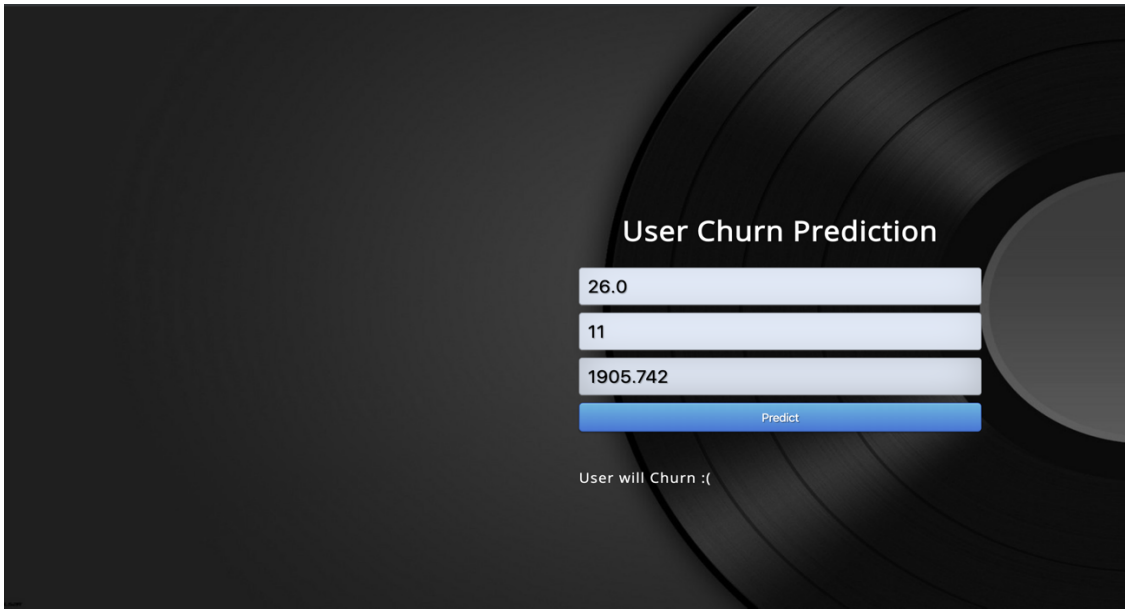
	DecisionTree	*KNeighbors*	*RandomForest*	*AdaBoost*	*QDA*
F1 score	0.97459	0.53767	0.98021	0.49804	0.55626
Precision	0.90322	0.04722	0.92603	0.01646	0.03252
Recall	0.98090	0.52188	0.96323	0.50098	0.58512

The highest accuracy in these reports was given by Random Forest model which we then chose as the best model with an F1 score of ~0.98. We decided to go ahead and implement this as the final model.



- *Front End*

  Once the model was trained and saved, we built a front end that takes top three most important features and predicts if based on these parameters, the user will churn or not. We used flask with HTML+CSS framework, to integrate built model with the front end.



Songs Recommender:

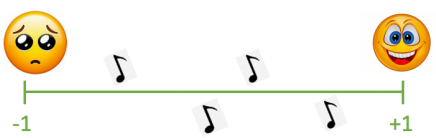
After we predict if a particular user is going to churn or not, we need to put directed efforts towards customers that show tendency to churn. As our application is a music streaming service, we performed sentiment analysis on the users most heard songs.

- *Sentiment Analysis for Polarity*

  Here, we essentially worked with two terms of the sentiment analysis domain- Polarity and Subjectivity.

  Polarity: Polarity is float which lies in the range of [-1,1] where 1 means positive statement and -1 means a negative statement.

  Subjectivity: Subjective sentences generally refer to personal opinion, emotion or judgment whereas objective refers to factual information. Subjectivity is also a float which lies in the range of [0,1].



Polarity

Subjectivity

Following was the algorithm used for recommendation:

a. *Lyrics of about 5000 songs were scraped using MusixMatch API and the values of Polarity and Subjectivity were recorded for each song using TextBlob.*
b. *These songs were stored in a database.*
c. *The average polarity for each user's playlist is calculated.*
d. *This value is mapped to the corresponding songs in the database and the top 5 songs are recommended to the user.*

The MusixMatch API was used with the help of following API call:

```
api_call = base_url + lyrics_matcher + format_url + artist_search_parameter + artist_name + track_search_parameter + track_name + api_key
```

```
MENU OPTIONS
1 - User-ID Based
2 - Polarity Based
0 - Exit

> 1

Enter User-ID
> 9797
0.26397890787848266
You generally like songs with polarity  0.26397890787848266
Here's a list recommended for you:
             Song Name                   Artist Name
0  Burn the house down                           AJR
1        Move to Miami  Enrique Iglesias ft. Pitbull
2               Colour     MNEK ft. HAILEE STEINFELD
3                 2002                   Anne Marie
4  Just wanna love you       Chris Cab ft. J. Balvin
```

Fig. Back-end of the recommendation

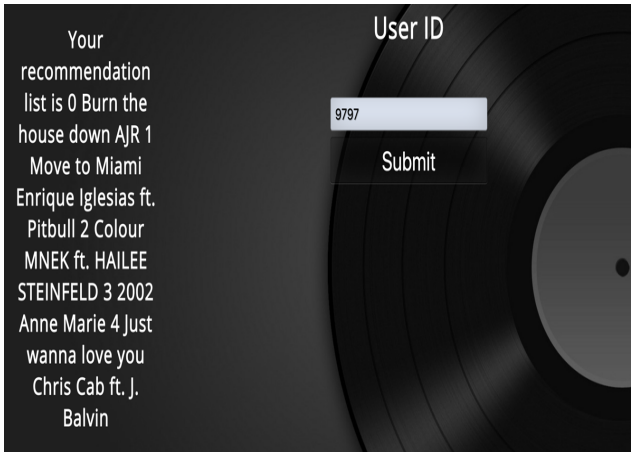


Fig. Front-end of the recommendation

4. Problems

- *Managing the size of data*
  The humongous size of data was difficult to deal with for data preprocessing and reading in. The python notebook kernels were proving to be inefficient for the computation of this scale.
  Solution: We implemented Apache Spark to parallelize the computations. It proved to be much faster and efficient.
- *Free-tier version of MusixMatch API*
  The free version of the API allowed only 50 requests per IP address per day. Hence, collecting data had to be a consistent job over several days.
- Usage of NLTK libraries for scraping the lyrics of the songs also took significant efforts since we were both not very well-versed with it. It improved our skills in that base.

5. Results

The findings from our analysis in this project will help any upcoming streaming service company to know which customer base to focus on. This will ensure customer retention and an overall security to the product.

1. Rightly predicting whether or not a user will churn requires the apt combination of the user's personal data and his data of usage of the service. Personal data includes his age, the city he comes from (a city customer is generally more likely to not churn out). Usage data includes average usage time, number of unique songs played, etc.
2. The most influential features to determine churning is a cusp of personal and usage data.
3. In targeted advertising, another key factor that is as important as knowing the customer base is the ability to retain that customer base. Hence, the second part of our project is building a recommender system to suggest songs that cater to the particular user's taste.
4. We successfully built a recommendation system by matching the sentiments of the user's songs to the ones in our library.

6. Project Summary

In conclusion, we intended to garner inferences from our data that would help streaming companies know their target audience to prevent churning. Firstly, using the predictions of the model, we separated the customer base that needed extra attention. The users' tastes were calculated on the basis of polarity of the music he/she listens to. Thereafter, we scraped songs using MusixMatch API and calculated the values of polarity for those songs. We also managed to come up with a way to retain the about-to-churn customers with targeted advertising. Finally, in the recommendation engine, user information was used to ensure that he is suggested songs according to his/her taste.

Getting the data	Scraping the lyrics using API	2
ETL	Encoding, missing values, outliers	1
Problem	Had good amount of eda, prediction and recommendation on sizeable data	4
Algorithmic Work	Using the vitality of Polarity as basis for recommendation algo.	3
Bigness/parallelization	Spark implementation	3
UI	Flask with HTML+CSS	2
Visualizations	Covered in EDA using Numpy and Pandas	1
Technologies	Spark, Flask, Scraper API, etc	3

REFERENCES

https://www.kaggle.com/c/kkbox-churn-prediction-challenge
https://www.kaggle.com/c/kkbox-music-recommendation-challenge
https://towardsdatascience.com/simple-way-to-deploy-machine-learning-models-to-cloud-fd58b771fdcf
https://github.com/krishnaik06/Deployment-flask
https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/
https://developer.musixmatch.com/
https://developer.musixmatch.com/documentation