6/18/2021

Foundation Project 1

Final Submission - AMPBA Batch 15

SUBMISSION BY

-	Deep Kamal Singh	(12020053)
-	Kshitij Sharma	(12020062)
-	Mohua Sinha	(12020015)
-	Nidhi Srivastava	(12020078)
-	Soumaya Ranjan	(12020060)

Table of Contents

1 Abstract

Recommendation Systems are simple statistical based algorithms that helps consumers with the most relevant and accurate recommendations. Food recommendation system typically identify and recommend preferred restaurants with the suggested food items, to the users (consumers/app users) based on consumers preferences, demographics, locations etc by filtering useful features from a set of restaurants database. During this project, we have tried to study the evolution of food recommendation engines over a last few years and the underlying algorithms leveraged to build and recommend the restaurants to the users. While we have tried to analyse the evolution of these algorithms and their shortcomings, we are trying to build a recommendation engine based on ALS technique by considering additional features in terms of gastronomical and medical compatibility data (to address user specific allergies). Our suggested recommendation engine leverages the multimodal recommender system which over a period time will improve its prediction accuracy based on recursive model approach.

2 Business & Data Understanding

2.1 Define the Scope of the ML Application

In the current scenario, consumers have multiple options for food both in terms of Dining out outlets as well as delivery options. To select an outlet for dining out /delivery options, consumers get recommendation based on different analytical models like Content based filtering, Collaborative filtering, Latent Factor Collaborative and so on. The underlying features on which recommendation are made are around user history and experiences in terms of Ratings, reviews hygiene, offers, distance etc. However, very limited analytics is being used in recommendations specific to consumers preferences and taste of food based on seasons/ moods.

2.1.1 Business Problem

- Present food recommendation engines are not effective
- Significant churn on the account of limited/sub-optimal choices to customers, impacting future sales and retention

2.1.2 Business Objectives

Maximize revenue by providing the most preferred and likeable food options

2.1.3 Business Constraints

Minimize consumer fatigue by providing most optimal choices
Minimize the searching time for the preferred choices

2.2 Success Criteria

2.2.1 Business Success Criteria

- Revenue enhancement for food outlets in following 3 ways
  - Increase in footfalls/order volume
  - Increase in order volume per order by cross-selling other popular food items
  - Increase the likeability factor/positive reviews helping further revenues

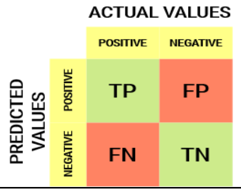
2.2.3 Economic Success Criteria

The economic success criteria could be determined broadly determined through variety of metrics from both app as well as food outlet perspective, and can be monetized

- Click through rate (CTR)
- Conversion rate overall & Conversion rate per user
  Proportion of orders with recommendations & Recommend items per order

2.2.2 ML Success Criteria

- Evaluation Metrics that will be used for assessing the ML will be the following:
  - **Accuracy**, which will be determined by minimizing low prediction errors by using split -validation of data for comparison
    Accuracy = No. of successful recommendations/ Total No. of recommendations
  - **Precision , Recall and F1 Score** - **Precision** indicates the fraction of relevant items among all the recommended items to a user and is calculated as (TP/ (TP+FP))**Recall** represents the number of relevant recommended items to the total number of items (Calculated as TP/(TP+FN) that should be recommended. Precision and recall metrics are calculated by computing a confusion matrix. **F1 Score** is the weighted score of Precision and Recall



		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP	FP
	NEGATIVE	FN	TN

2.3 Feasibility

2.3.1 Applicability of ML technology

ML based technology goes beyond the traditional ranking mechanism of ratings and location allows to rank the prefiltered dish and restaurant candidates based on additional contextual information such as day, mood, season, user information like allergies etc. ML based technology leverages supervised classification methods for ranking like Logistic regressions, Support Vector mechanism, Decision tree-based method such as Gradient Boosting and combination of couple of techniques for recommendations

2.3.2 Legal constraints

The legal constraints are more around "User privacy" for recommendation systems. These can be classified into 3 broad categories architectures, algorithmic, and policy approaches

- Non-adherence to policy approaches-GDPR legislation, i.e. lack of explicit guidelines and sanctions to clearly provide information and regulate data collection, use, and storage
- Inadequate Privacy-enhancing architectures- privacy risks of storing user data in separate and decentralized databases,
- Improper algorithmic solution (encryption) with the risk of losing the data to external agents for unwarranted usage

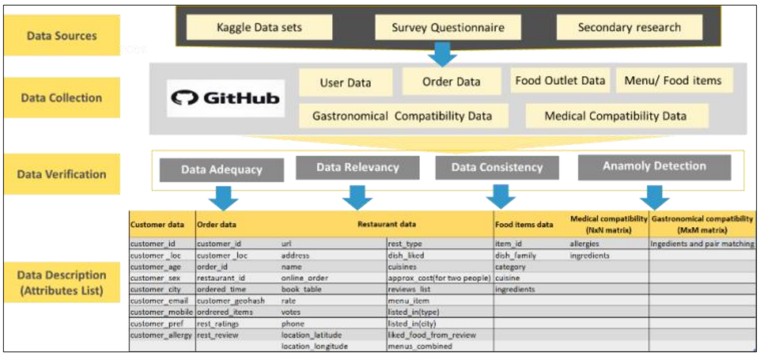
2.3.3 Requirements on the application

The requirements from the application would be from both consumer as well as food outlet perspective and is explained as follows:

- Consumer perspective- Consumer need to provide basic information in terms of demographic information, reviews, food preferences in terms of likes, allergies and other personal info based on the "User acceptance guidelines" and "Terms of Reference"
- Food outlet perspective- Food outlet needs to provide the menu, ingredients, services provided by them and this information would be used as input for recommendations to users

2.4 Data Collection & Data Quality Verification

The data collection process would be done through multiple mediums and data control was done through GitHub, which was used to organize our data, EDA, models preparation and model validation. Our recommendation engine is based on multiple data set around User, Order, food outlet, Medical Compatibility & Gastronomical compatibility



Customer data	Order data	Restaurant data		Food items data	Medical compatibility (NxN matrix)	Gastronomical compatibility (MxM matrix)
customer_id	customer_id	url	rest_type	item_id	allergies	Ingedients and pair matching
customer_loc	customer_loc	address	dish_liked	dish_family	ingredients	
customer_age	order_id	name	cuisines	category		
customer_sex	restaurant_id	online_order	approx_cost(for two people)	cuisine		
customer_city	ordered_time	book_table	reviews_list	ingredients		
customer_email	customer_geohash	rate	menu_item			
customer_mobile	ordrered_items	votes	listed_in(type)			
customer_pref	rest_ratings	phone	listed_in(city)			
customer_allergy	rest_review	location_latitude	liked_food_from_review			
		location_longitude	menus_combined			

3 Data Preparation

3.1 Select Data

3.1.1 Feature selection

It was desirable to reduce the number of input variables to both reduce the computational cost of modeling and improve the performance of the model. We have combined "average_cost_for_two","price_range","aggregate_rating","votes" into one feature column and applied StandardScaler() to scalerize the newly created "feature" column.

3.1.2 Data selection

The food industry generates a huge volume of data and most of the times, the available historical data can be geographically biased. We could have used hypothesis testing to evaluate the right selection of a sample.

3.1.3 Unbalanced Classes

On splitting the data , we saw that only 11.1% were negatives (low rating)) and 88.9% were positives values and the Logistic loss objective function treated the positive class with slightly higher weight. For this purpose, we have calculated the BalancingRatio as follows:

#BalancingRatio= numNegatives/dataset_size

3.2 Data wrangling

As a standard way of data cleaning we will apply approaches for Noise reduction and Data imputation on incoming data, further We have used the Imputer function of MLlib library to Compensate for Missing Values in our dataset.

Feature engineering

We have selected independent features using Correlation Matrix with Heatmap and Univariate Selection.

Standardize Data

File format – incoming data will depend upon the source format, however after first level of processing we will store it in columnar format internally

Normalization

We have applied Normalization of our feature through Standard Scaler function in Python.

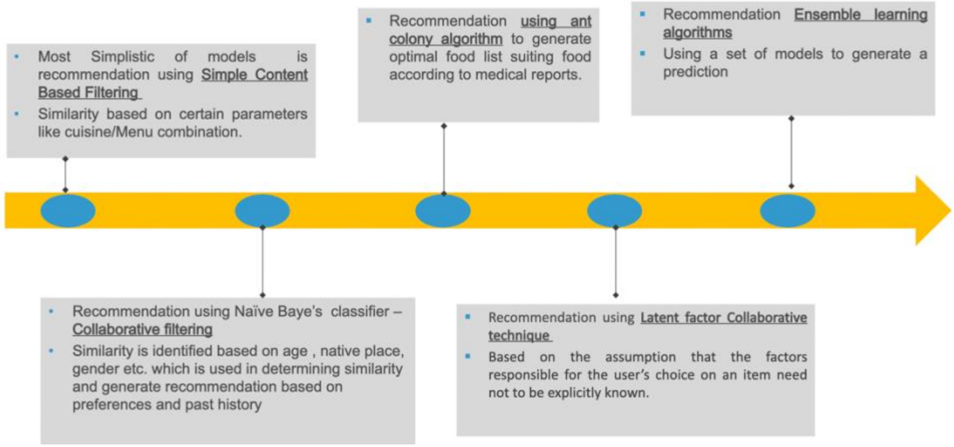
4 Modelling

Base Idea: The seasonal, taste-based, and nutritional data will be used as reference data to map with the customer personalized data to create an appropriate training dataset for our recommendation model, further we will make use of labelled data from user's given ratings on food items and restaurants,

1) To build a Recommendation system we are using ensemble methods
    a. We will follow MCDA – Multi criteria decision analysis matrix match, and apply ALS-WR on user's ratings for food item and restaurants
    b. We are using Frequent pattern mining techniques to obtain associations based on items ordered together, and food items that have common ingredients list
    c. Capture the preferences from food reviews from social platforms data.
2) Further learning in this direction led us to form the data points or features, which we will feed as input to our model
    a. Our target is to achieve prediction score (rmse <= 0.80) from the ALS-WR,
3) Based on the prediction score, will increase our training dataset, if required, to further tune our model

4.1 Literature research on similar problems

The research on multiple food recommender system helped us build base for the existing methods used, we studied in depth about the different algorithms and its working and concluded to build on our recommender system. The list includes as follows



- Most Simplistic of models is recommendation using Simple Content Based Filtering
- Similarity based on certain parameters like cuisine/Menu combination.

- Recommendation using Naïve Baye's classifier – Collaborative filtering
- Similarity is identified based on age , native place, gender etc. which is used in determining similarity and generate recommendation based on preferences and past history

- Recommendation using ant colony algorithm to generate optimal food list suiting food according to medical reports.

- Recommendation using Latent factor Collaborative technique
- Based on the assumption that the factors responsible for the user's choice on an item need not to be explicitly known.

- Recommendation Ensemble learning algorithms
- Using a set of models to generate a prediction

4.2 Define quality measures of the model

4.2.1 Performance

- F1 Score – to know how precise the classifier is (how many instances it classifies correctly) and how robust it is (it does not miss a significant number of instances)
- Confusion Matrix – to know a detailed breakdown of correct and incorrect classifications for each class
- RMSE (Root mean Square Error) - to have the model focus on large errors

4.2.2 Robustness

- Data Sanity Checks like Checking for datatype mismatches, variations in how values are entered, and missing values.
- To detect and prevent against shilling attacks, we have proposed to build a probabilistic generative model.

4.2.3 Scalability

We have used Distributed computing model (using Apache Spark) to handle the huge Volume and Velocity of data. Algorithm is also designed to be scalable. We have also applied the principle of 'Separation of Concerns' by keeping the training system different from the production system.

4.2.4 Explain-ability

We have focused on Feature Importance, Counterfactual Explanations and Adversarial Perturbations to illuminate the decisions our model is making.

4.2.5 Model Complexity	4.2.6 Resource Demand
We have designed a multi-model recommendation system where-in we have a Real-time Consolidator (Conditioner) and a Reducer layer. There is also a Hit/miss ratio feedback mechanism in place.	1. High compute systems for distributed processing – Apache Spark (N+1 horizontal and vertical scalability), minimum 32 GB, 24 Cores 2. Publicly hosted system for Web App and prediction engine – EC2 on AWS , High network and storage with good compute power – 16GB RAM, 500GB SSD , 5GB/s Network.

4.3 Model Selection

4.4 Incorporate domain knowledge	4.5 Model training	4.6 Using unlabelled data and pre-trained models
• Online Food delivery – domain knowledge gathering • Fulfilment • Aggregation • Social media impact – harnessing it for good	• Available data • Train/Validate split • Trained model logistics to production CI	• Social media as source of data • Using existing models for cold-start , using existing gastronomical compatibility
4.7 Model Compression	**4.8 Ensemble methods**	**4.9 Assure reproducibility**
1. Matrix factorization for reduced dimensionality – ALS 2. Knowledge distillation – teaching small network step by step to behave exactly like bigger trained network, this process will be implemented down the line when data volume, sparsity, velocity is increased after production implementation.	Using FPM, ALS-WR and consolidating the final recommendations set, further we applied a reducer layer to strikeout recommendations which are not medically compatible or is in user's blocked list of restaurants	**Method reproducibility:** with multiple iterations we will ensure that changing data volume, data variety and velocity will not adversely impact recommendations **Result Reproducibility:** mathematical model of FPM and weighted regularization (ALS-WR) are applied with RMSE and F1 threshold – this is tested with alpha of 0.05.

5 Evaluation

5.1 Validate performance	5.2 Determine robustness
F1 Score Confusion Matrix Mean Square Error	Dataset sanctity Detection, Prevention against shilling attacks
5.3 Increase Explainability for ML practitioner & end user	**5.4 Compare results with defined success criteria**
Feature Importance Counterfactual Explanations Adversarial Perturbations	Accuracy (Successful recomm/ Total recomm) Precision (TP/ (TP+FP)) & Recall TP/(TP+FN) RMSE F1 Score = 2*(Recall * Precision) / (Recall + Precision) Click through rate (CTR) Conversion rate overall/Conversion rate per user Proportion of orders with recommendations Recommend items per order

6 Deployment

6.1 Define inference hardware

Have used Spark cluster with 32 cores on premises. Depending on data size increase in the future will scale the network horizontally.

6.2 Model evaluation under production condition

Model will be continuously monitored using tools like Prometheus or other available tools in market. Collecting and merging the ground truth and predictions to build a recursive model evolution will help achieve the desired metrics.

6.3 Assure user acceptance and usability

Machine learning predictions will be projected as suggestions and user feedbacks or buying pattern based on the recommendations, are to be feed in to the system for next layer of prediction.

6.4 Minimize the risks of unforeseen errors

To avoid unforeseen errors, have to implement governance, policies and controls on the overall system.

6.5 Deployment strategy

The final saved Model is deployed to the Serverless Amazon EC2 instance (Ubuntu server) using Flask framework.

7 Monitoring and Maintenance

7.1 Non-stationary data distribution

Under non stationary environment, the issue of covariate shift can be resolved using importance weighted loss functions. Support of the test input distribution should be contained in the training input distribution.

7.2 Degradation of hardware

Hardware degradation can happen over a period of time due to factors like hyperparameter training layers and optimization. To achieve high performance model, have to use hardware accelerators.

7.3 System updates

Periodic system checks and updates will be part of the CICD process.

7.4 Monitor

Continuously monitoring using tools like Prometheus, ML monitor to check the data drift and covariance shift will be employed. Comparison of deployed model on benchmark distribution and real-world distribution to use the recursive model evaluation.

7.5 Update

With this setup of data, model and feedback monitoring, it will be easy to do periodic updates and CICD using Git, Jenkins and serverless AWS EC2.

8 References

1. Data verification: https://blog.fastforwardlabs.com/2019/08/28/two-approaches-for-data-validation-in-ml-production.html
2. Github: Food-Recommendation-System/Deliverables at main · FP1-Group10/Food-Recommendation-System (github.com)
3. Google Colab Code: https://colab.research.google.com/drive/1Re6az6aSS6YjXHQHLgAWrDKU0mf4hl3V?usp=sharing
4. https://radanalytics.io/applications/project-jiminy
5. https://www.datasciencecentral.com/m/blogpost?id=6448529:BlogPost:512183
6. https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf
7. https://spark.apache.org/docs/2.2.0/ml-collaborative-filtering.html
8. https://dzone.com/articles/introduction-to-recommender-systems
9. https://www.slideshare.net/databricks/building-an-implicit-recommendation-engine-with-spark-with-sophie-watson
10. https://radanalytics.io/applications/project-jiminy
11. https://towardsdatascience.com/how-to-build-a-restaurant-recommendation-system-using-latent-factor-collaborative-filtering-ffe08dd57dca