Group-20 – 'Moody Foodies'

ABSTRACT

Project Introduction and Description

Moody Foodies *is a food recommendation system which recommends food based on the mood of the customer. It aims to overcome the customer's paradox of choice by filtering down the options of food based on their current mood and demographics, additionally it also provides common food allergens info for the recommended dishes. The system generates recommendations using the data gathered through surveys and various other sources, as a next step the engine would find out the best restaurant options considering dish chosen, distance, affordability and rating as factors.*

BUSINESS UNDERSTANDING

Business Objective:

- *Overcome consumer's paradox of choice*
- *Enhance user food ordering experience*
- *Minimize the time to order food*

Business Constraint:

- *Handling Large amount of Data*
- *Dealing with ambiguous feedback data*

Business Success Criteria:

- *Develop an application using machine learning techniques which takes user demographics and mood as inputs to recommend food with good accuracy.*

ML Success Criteria:

- *We are building an end-to-end food recommendation system based on customer moods.*
- *We have flexible database.*
- *We can fine tune our model and improve it continuously through user feedback.*

DATA UNDERSTANDING

Data Description:

- *Primary Data Sources – New Survey Data or Pre-existing survey data*
- *Secondary Data Sources – Swiggy, Recipes and common food allergens*

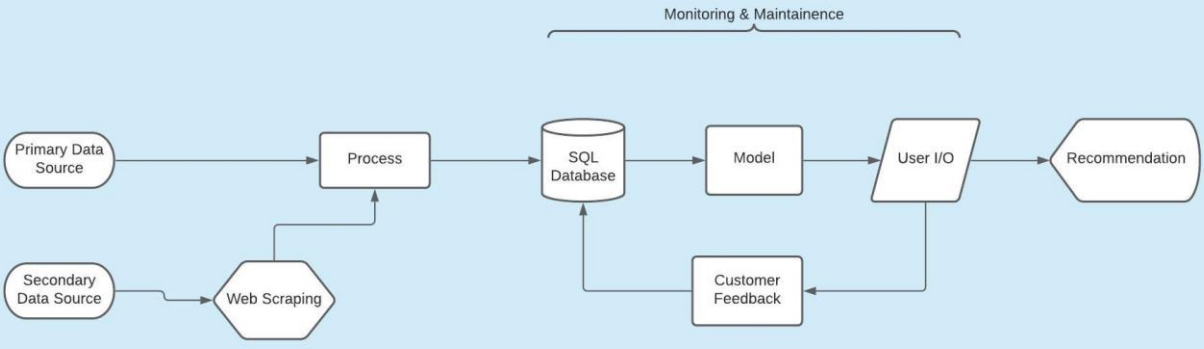
Data Collection:

- *Scraping the restaurants information like ratings, top food, cost for two, location from Swiggy using selenium/scrapy/beautiful soup.*
- *Collecting data through surveys and other existing online sources for mood-based food choice*

DATA PREPARATION

- *Normalizing Survey data.*
- *Cleaning scraped data from Swiggy*
- *EDA on survey data*
- *Removing outliers*

PROCESS ARCHITECTURE



MODELLING

- *K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.*
- *Depending on the Euclidean distance between the new data point and existing data points we find K nearest users who are similar, fetch the dishes chosen by these neighbors, create a frequency table and filter top N dishes as recommendations to the new user.*

EVALUATION

Average Precision (AP@N):

- *If we are asked to recommend N items, the number of relevant items in the full space of items is m, then:*

$$\text{AP@N} = \frac{1}{m}\sum_{k=1}^{N}(P(k) \text{ if } k^{th} \text{ item was relevant}) = \frac{1}{m}\sum_{k=1}^{N} P(k)\cdot rel(k),$$

- *where rel(k) is just an indicator that says whether that kth item was relevant (rel(k)=1) or not (rel(k)=0). I'd like to point out that instead of recommending N items we could have recommended, say, 2N, but the AP@N metric says we only care about the average precision up to the Nth item.*

The MEAN in MAP@N:

- *Average Precision, applies to a single data point (like a single user), whereas MAP averages the AP@N metric over all the |U| users. It's an average of an average.*

$$\text{MAP@N} = \frac{1}{|U|}\sum_{u=1}^{|}U|(\text{AP@N})_u = \frac{1}{|U|}\sum_{u=1}^{|}U|\frac{1}{m}\sum_{k=1}^{N} P_u(k)\cdot rel_u(k).$$

Mean average precision (MAP@N) of our model is currently 10%, due to the limitation in survey data collection versus the number of classes we have in our data. As a future improvement we could improve the precision by collecting more survey data.

DEPLOYMENT

Deployment Strategy:

- *Pre-process the data > splitting it into train and test > generating a machine learning model > Prediction > Scoring > Saving the Model*
- *Create Flask application > Run > Add templates > Create base layout > Create pages for user inputs and prediction > Load trained model, pass input data to model and predict > Modify prediction page and print prediction*
- *Create account in Heroku > GIT installation > Heroku_CLI_installation > Login to Heroku account > Install gunicorn > Declare app dependencies > Create Profile > Initialize git inside the project > Create_heroku_app > Add files to the GIT repository and deploy > Browse deployed URL*

MONITORING & MAINTENANCE

- *Identifying the root cause of deviation in result and tweaking the model accordingly*
- *Validating the accuracy of the model*
- *Ensuring new user inputs are collected with similar features in dataset.*
- *Check for regular updates.*

GENERIC REQUIRMENTS

- *Creating Heroku account*
- *Hardware Configuration*

  – 1 GB RAM

  – i3 Processor or greater

  – 10 GB SSD/HDD

  – 8 Cores
- *Software Configuration*

  – Windows 10/ Linux

  – Python 3.7 or greater

  – Heroku CLI
- *Libraries used*

  Pandas, Numpy, Plotly, matplotlib, collections, warnings, sklearn, pickle, ipywidgets, selenium, time.