Dine Wise Recommender System

(https://github.com/aaru9dua/Cloud_Framework_for_Restaurant_recommendation)
Aarushi Dua, Ingrid Arreola, Smit Borasaniya

1. Introduction

Recommendation systems are prevalent in today's digital age, and our group wanted to work on a project that would be in this space. Specifically we wanted to focus on a recommendation system that would personalize food recommendations. As cloud computing continues to revolutionize the way that we access, store and share information recommendation systems are becoming increasingly essential for businesses that want to offer their customers personalized and efficient services.

The goal of making a cloud-based restaurant recommendation system is to provide personalized recommendations to users based on their preferences, past dining experiences, and other relevant data. This system uses advanced algorithms and machine learning techniques to analyze vast amounts of data, including user reviews, restaurant ratings, menus, location data, and other relevant factors to generate a list of recommended restaurants that meet the user's criteria.

The main objective of the system is to enhance the user's dining experience by providing them with tailored and accurate recommendations in real-time. By doing so, the system can increase customer satisfaction and engagement, as well as help restaurants improve their operations, optimize their menus, and enhance their services. Additionally, cloud-based restaurant recommendation systems can benefit businesses by increasing their visibility and revenue. By our model of providing location based recommendation based on the category of restaurant we try to achieve the utmost accuracy.

2. Background and Related Work

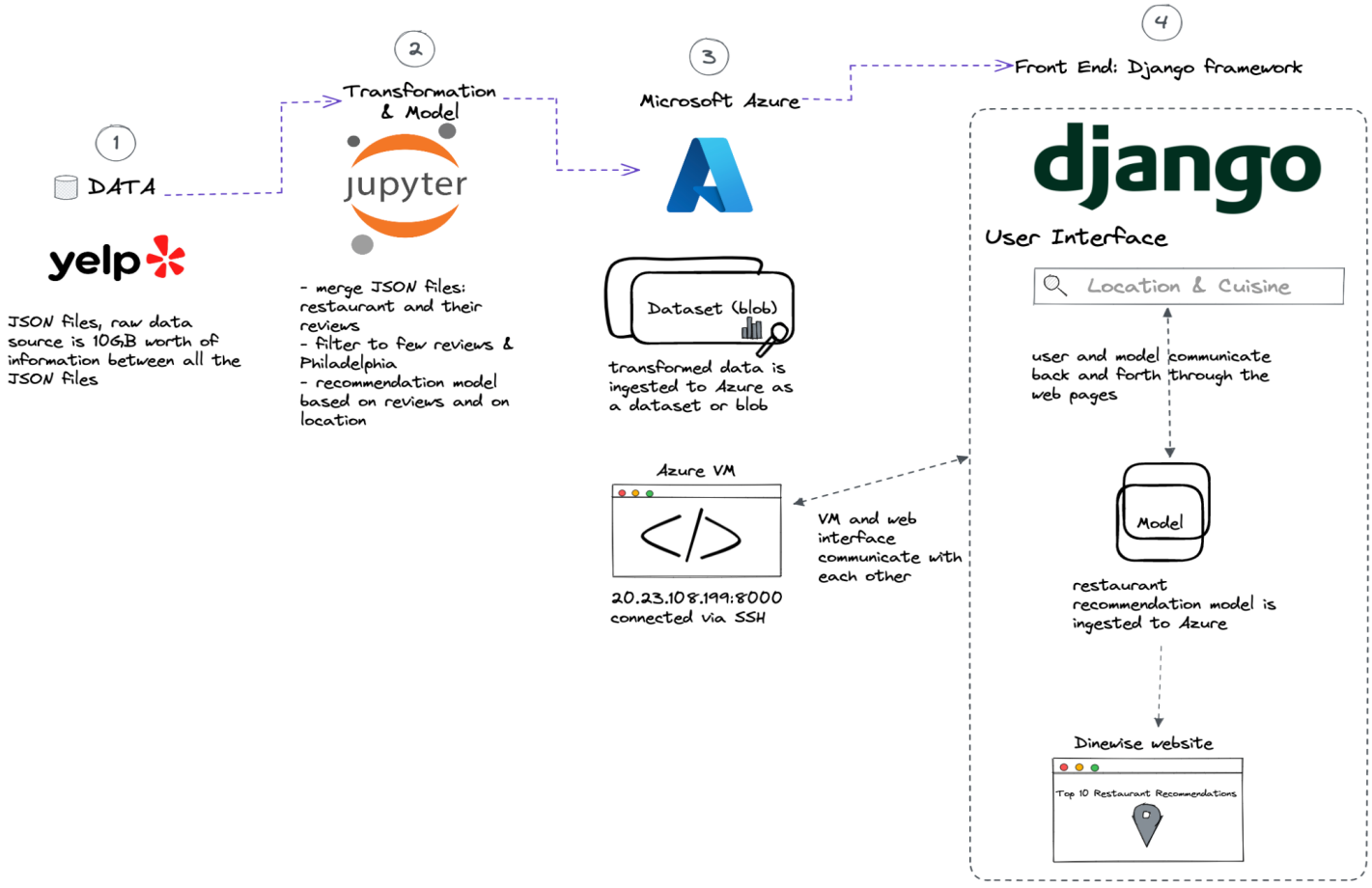
The global food service market is projected to grow from $2,540.05 billion in 2022 to $5,194.60 billion by 2029, at a CAGR of 10.76% in the forecast period.[1] The surge in dining out expenditure has aided the expansion of the global market. Furthermore, the rapid expansion of numerous restaurant chains has found popularity in both developing and developed nations. Customers' purchasing habits can be influenced by product recommendations. Uber Eats provides a user-friendly method of ordering food. While UberEats now recognizes consumers' intentions, there are still moments when users are unsure what they would like to eat. In these cases, the app offers a customized solution for each customer by recommending restaurants.

Other websites such as Yelp offer a site and app for users to scroll through reviews for restaurants, add their own ratings, and search for restaurant options. This is helpful for users to locate restaurants and to see food options available, however it relies on a search query and doesn't yet extend into the realm of engaging with a user to provide more personalized options or additional engagement. Even search engines like google maps are great at searching and finding information on restaurants near a person but they do

not provide personalized engagement for a user. Our project would focus on extending this user experience so that a user has more engagement with restaurant recommendations.
This project aims to move beyond a search function and provide recommendations based on cuisine, allowing for an easier process for users to review their dining options. The project also strives to recommend cuisine options near a user's location.

3. Proposed Design / System

The design and system of this project is documented below with the visualization that shows the entire process used to create our cloud-based restaurant recommendation system. Each major step in the system is designated with a number and a header title. The rest of the report goes into the details of what specific steps were taken in the project.

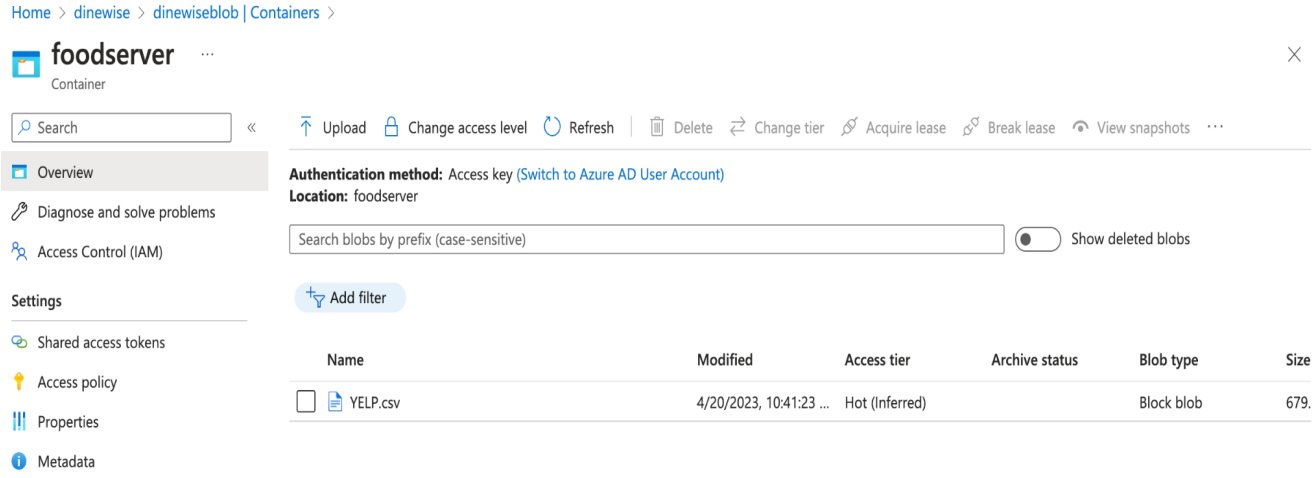


3. Proposed Architecture / Application

• Data Collection

Our project began with data collection and data exploration. We identified three initial APIs and signed up for access to these APIs. Only one of the APIs was accessible, the Yelp Fusion API, and we began an initial data exploration. We were limited by the number of requests we could send and found that we were getting back a smaller amount of data than we initially thought was available on the API. After this initial exploration we moved to using a dataset on Kaggle, the Yelp Dataset, which was created by the Yelp team and contains five JSON files and a user agreement.Out of 5 JSON files, we picked business.json and review.json. We loaded the json files using pandas. One of the early challenges was the size of the data, for example the review.json is a huge file (5 GB), and so we had to update our strategy to reduce the size of our dataset. We did this by filtering down to just the restaurant data, filter to 5 reviews per restaurant, and also filtering to just one city which had the majority of data. This reduced dataset was then uploaded as a dataset on Azure cloud.

The screenshot below shows the dataset which exists as an azure blob, this is leveraged in our project because we can call on the storage url, which is the blob, and use that very easily with Python to connect to the datasource.

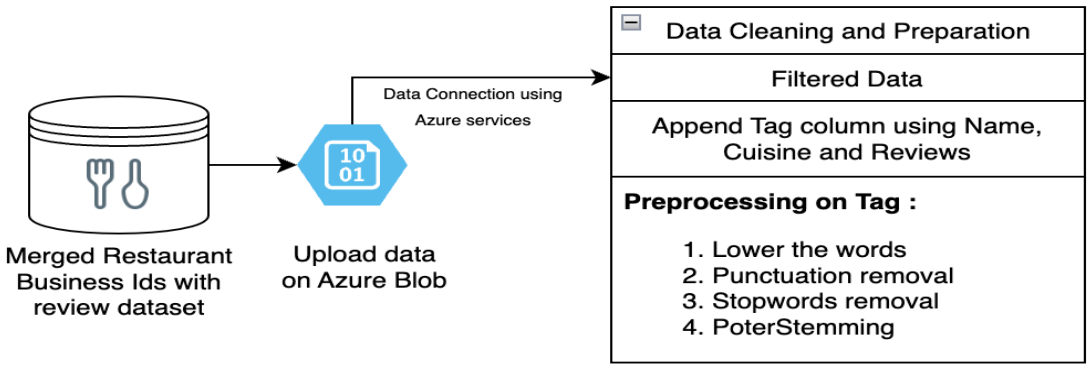


In our Python file we are able to connect to the Azure workspace, connect to the blob storage, and then kick off the recommendation system used for this project.

```
#CONNECT to data
account_name = 'dinewiseblob'
account_key = '6ePIp7JN5i49LZHTvmgUBSaYP5Q36uXdnjKV6I/I616LnZh/fcfXZTSSGqhXxxX6A30iTRwJ1bom+AStjlDKyg=='
container_name = 'foodserver'

connect_str = 'DefaultEndpointsProtocol=https;AccountName=' + account_name + ';AccountKey=' + account_key + ';EndpointSuffix=core.win

blob_service_client = BlobServiceClient.from_connection_string(connect_str)
container_client = blob_service_client.get_container_client(container_name)

sas_i = generate_blob_sas(account_name = account_name,
                          container_name = container_name,
                          blob_name = 'YELP.csv',
                          account_key=account_key,
                          permission=BlobSasPermissions(read=True),
                          expiry=datetime.utcnow() + timedelta(hours=1))
```

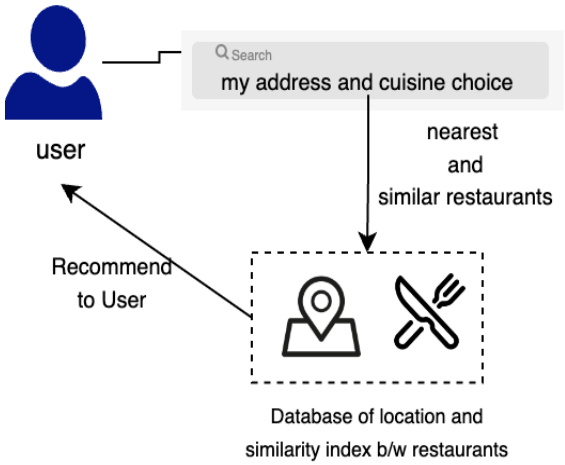
- **Data Cleaning and Preparation**

We picked up a part of the dataset from Yelp business and Review to start building a Content based filtering.



- **Content based filtering**

It is a type of recommendation based on restaurant's informations like reviews or cuisine style.



The steps involved in order to build this recommendation:

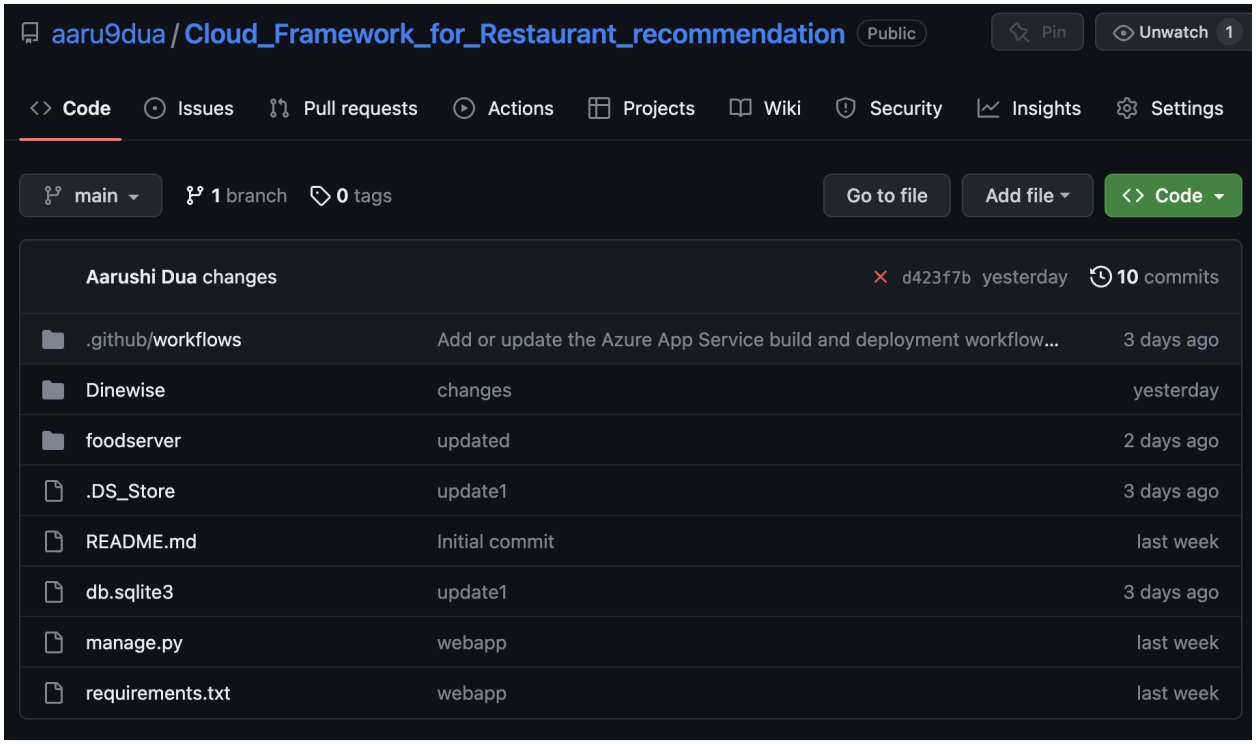
- **Data reduction:** Removed unnecessary columns from the merged Yelp dataset. Note: we are not taking location into consideration because that will be filtered once user's will enter his/her desired state/city.
- **Feature engineering:** Next, merge three columns : name, categories (cuisine) and reviews into a tag column to find the similarity between these textual informations.
- **Data Cleaning:** We performed removing punctuations, lowering each word and removed the commoner morphological and inflexional endings from words using PorterStemmer library from NLTK and at last removed stop-words.
- **Data transformation:** We Need To Convert Textual Data To A Numerical Value using CountVectorizer or TF-IDF. It focuses on the frequency of words and we set the maximum vocabulary size as 5000. We trained our pre-processes tag column in the feature extraction.
- **Similarity:** The next step is to calculate a similarity score. We used Cosine similarity to calculate similarities between each restaurant's description. That's the formula for cosine similarity.

$$cosine(x, y) = \frac{xy^\top}{|x||y|}$$

- **Get Recommendation:** We build a function which takes in a restaurant name. It finds its index in the similarity list and sorts all its scores and gives us the top N (Let's say 5) restaurant based on the user's choice.

Deployment on Django Framework

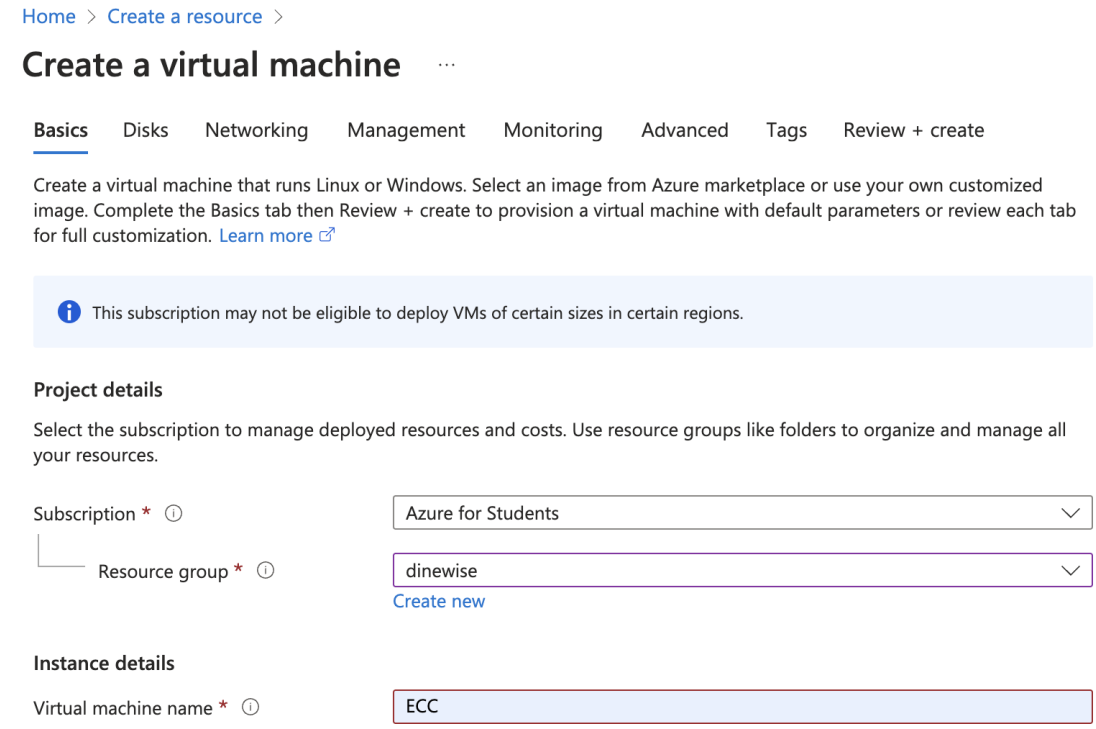
- Install Django: Django is installed running pip command in the terminal: pip install Django. Make sure to have the latest version of pip installed.
- Create a new Django project: To create a new Django project, run the following command in the terminal: django-admin startproject projectname.
- Create a new Django app: In Django, a project can have multiple apps. To create a new app, run the following command in the terminal: python manage.py startapp appname.
- Define models: Models are used to define the structure of the database. Define models in the models.py file in the app directory.
- Create database tables: Run the following command in the terminal to create database tables for the models: python manage.py makemigrations and then python manage.py migrate.
- Create a superuser: Run the following command in the terminal to create a superuser: python manage.py createsuperuser. Follow the prompts to create a username and password.
- Start the development server: Run the following command in the terminal to start the development server: python manage.py runserver. This will start the server at http://localhost:8000/.
- Once Django's setup is complete, we added our backend logic of filtering data for the recommendation in views.py and created recommendation.html and results.html to feed and show the output of the data being recommended.
- Once the app was running we integrated and hosted it on Azure and before that uploaded the final framework on github.

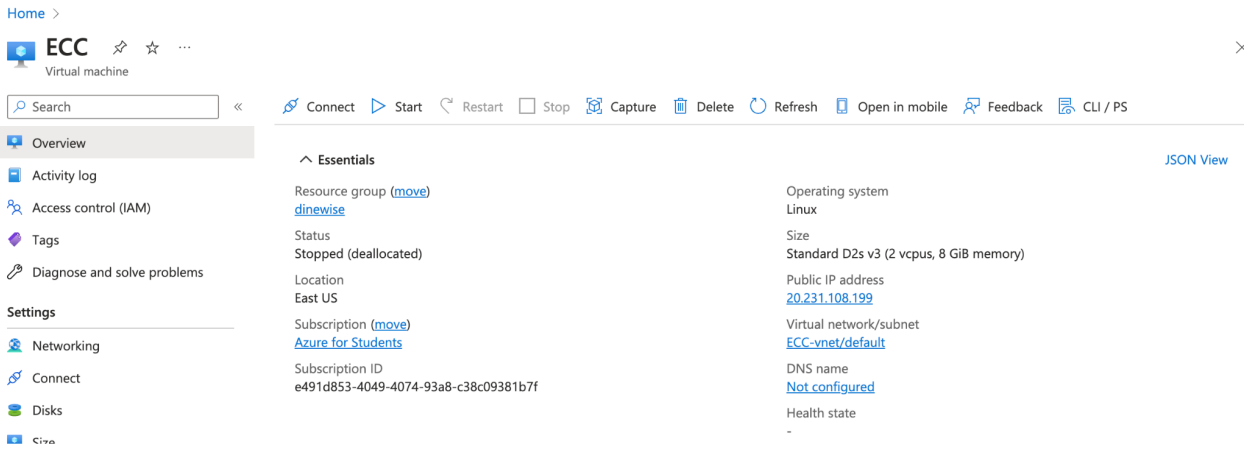


- **Deployment on Cloud**

We deployed our django framework using Azure Virtual Machine. The following steps lead to our final website-

1. Create an Azure Virtual Machine:
    - Log in to the Azure portal (portal.azure.com).
    - Click on "Create a resource" and search for "Virtual Machine."
    - Follow the prompts to configure the VM, including selecting the desired operating system, size, and networking options.
    - Set up SSH access during the VM creation process by providing a public SSH key.





2. Connect to the Azure VM via SSH:
   - Obtained the public IP address of the Azure VM from the Azure portal.
   - Opened a terminal or SSH client on our local machine.
   - Use the following command to connect to the VM:

ssh -i aarushidua aarushidua@20.231.108.199

3. Install necessary dependencies on the Azure VM like Python packages.

sudo apt install python3 python3-pip python3-venv git

4.Next, Cloned our Django project repository into the Vm:

Git clone https://github.com/aaru9dua/Cloud_Framework_for_Restaurant_recommendation.git

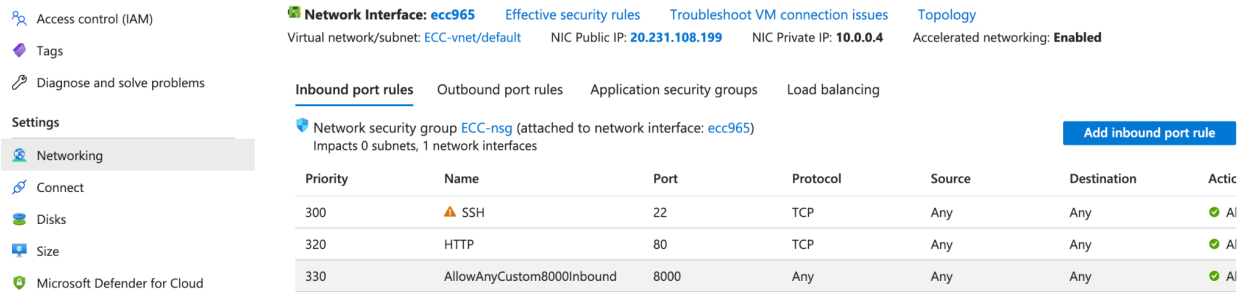
5.Installed project dependencies by installing packages from requirement.txt file

```
[aarushidua@ECC:~$ ls
 Cloud_Framework_for_Restaurant_recommendation  nltk_data
[aarushidua@ECC:~$ cd Cloud_Framework_for_Restaurant_recommendation/
 aarushidua@ECC:~/Cloud_Framework_for_Restaurant_recommendation$ pip install -r requirements.txt
```

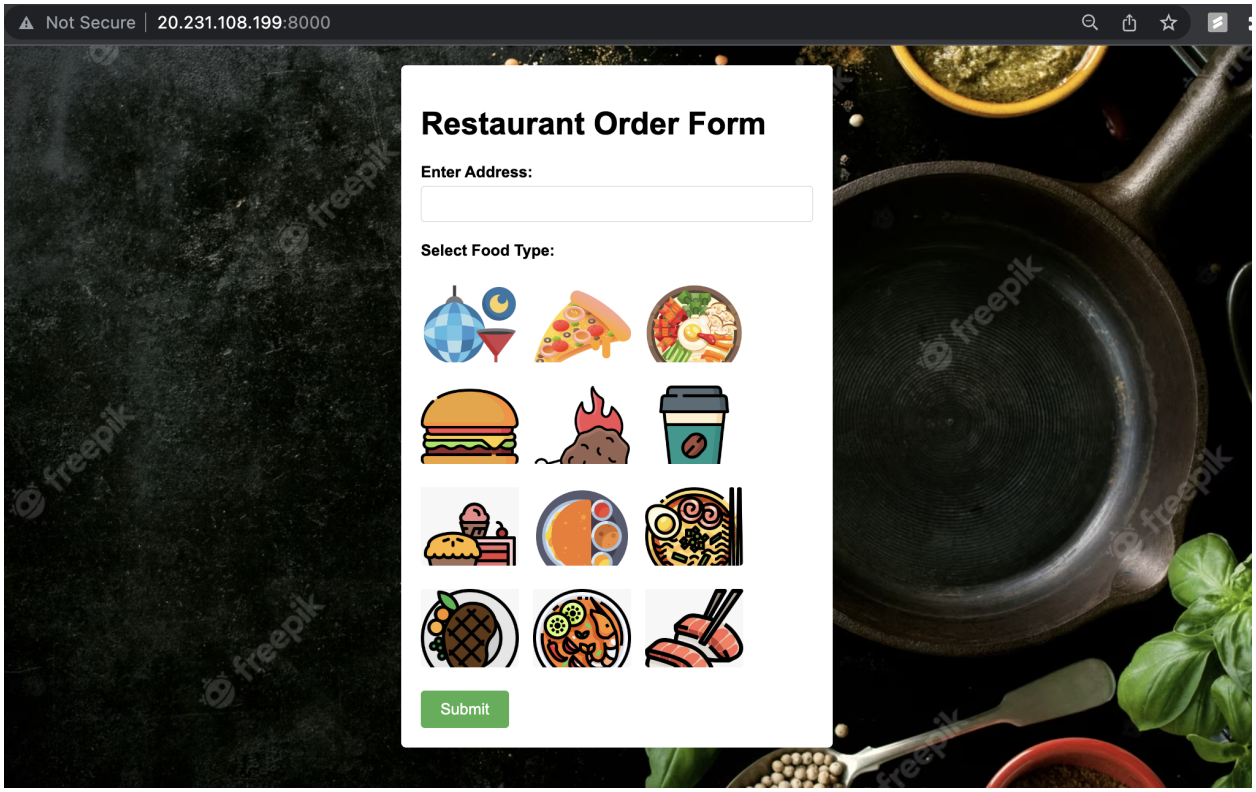
6.Started the Django development server on port 8000

```
[aarushidua@ECC:~/Cloud_Framework_for_Restaurant_recommendation$ python3 manage.py runserver 0.0.0.0:8000
Watching for file changes with StatReloader
Performing system checks...

[nltk_data] Downloading package stopwords to
[nltk_data]     /home/aarushidua/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /home/aarushidua/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
System check identified no issues (0 silenced).
April 30, 2023 - 02:27:01
Django version 3.2.4, using settings 'foodserver.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

7. Manually add the inbound port rule of 8000 in networking settings:



8.In our web browser, navigate to http://20.231.108.199:8000/, to see our website running successfully.

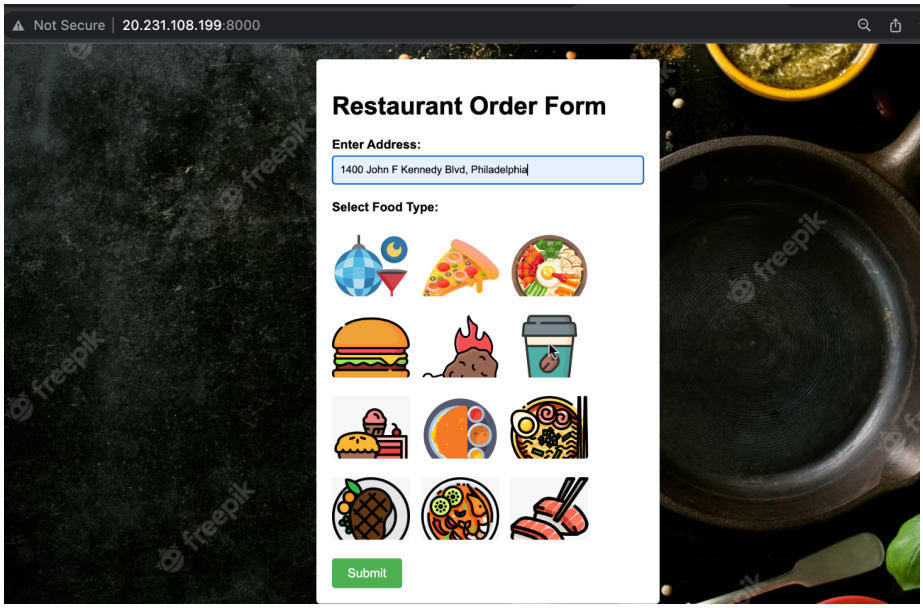


4. Evaluation / Result Analysis

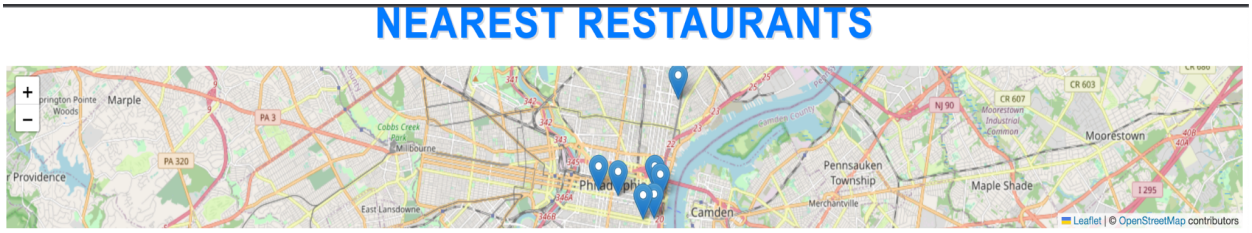
The final result of the project is the cloud deployment of our restaurant recommendation system, which has two main web pages for users to interact on. The virtual machine and the Django framework communicate with each other so that users can interact with the front end on a website. Users can enter in their address and select a food type. There are several food types included in the UI and the 12 categories represent a good selection from which users can pick from. After this the model runs in the background and provides recommendations which users can review on the next web page. The next web page displays a map along with a table of the recommendation results.

Our group tried out various food types and locations on the website to review the recommendations. We compared the results and were able to validate by visiting website menus as well as google maps to confirm that recommended restaurants were a reasonable distance from each.

Let’s try to evaluate our model whether it is effective in giving accurate results based on the desired choice of a User or not.



Here the User gave the address and the food type selected is Coffee and let's see the results



Name	Stars	Review Count	Categories	Address
Home Cuban Cafe	4.5	19	Restaurants, Vegan, Cuban, Cafes, Breakfast & Brunch, Latin American, Caribbean	17 N 3rd St, Philadelphia, PA
The Little Bird Bakery & Cafe	4.0	13	Restaurants, Cafes, Caterers, Food, Event Planning & Services, Bakeries	517 S 5th St, Philadelphia, PA
Buzz Cafe	4.5	53	Restaurants, Cafes, Coffee & Tea, Desserts, Breakfast & Brunch, Food	1800 N Howard St, Philadelphia, PA
Manhattan Bagel	3.0	133	Food, Sandwiches, Breakfast & Brunch, Coffee & Tea, Restaurants, Bagels	125 S 18th St, Philadelphia, PA
Bodega Bar	3.5	65	Restaurants, Cuban, Nightlife, Cocktail Bars, Latin American, Bars	1223 Walnut St, Philadelphia, PA
Main Street Cafe	4.0	11	Restaurants, American (Traditional), Cafes	215 E Main St, Norristown, PA
Dunkin'	2.0	16	Restaurants, Food, Coffee & Tea, Breakfast & Brunch, Donuts	101 Route 70 E, Marlton, NJ
Mtq Cafe	4.0	50	Restaurants, Vietnamese	113 Chestnut St, Philadelphia, PA
Bloomsday Cafe	4.5	73	Restaurants, Coffee & Tea, Food, Breakfast & Brunch, Bars, Wine Bars, Nightlife	414 S 2nd St, Philadelphia, PA

With our domain knowledge, we can observe these are mostly coffee places and after cross-referencing from Google map, these places are great cafes near the user's location.

5. Future Work

Our group utilized the Azure cloud computing platform to create a recommendation system that would personalize food recommendations for users. Our grouped learned about the ETL process required to create this system. We learned about how to utilize Azure to host the data, learned

about the machine learning service Azure has, as well as the virtual machine and Django framework used for deployment. There are several areas where future work could take place throughout the entire project system.

There is the possibility of future work around the area of data collection and transformation, this project could incorporate and add in the Yelp API to get the latest information of user reviews. The project utilized a Kaggle dataset however the future could add in another data source, and leverage the Yelp API to get more updated information. Additionally there could be future work in the area of the Azure platform. Our group utilized the student credit offered on Azure and in the future the project could focus on the virtual machine configuration such as testing the VM size, or looking at specific configurations to help improve the performance optimization and resource allocation. The future work could also focus on testing the scalability features to improve the user experience as the VM scales up and down depending on the demand on the system. Future work could further explore this by look at rightsizing the virtual machine, scheduling jobs and really understanding the management tools to monitor the resource usage. Finally, future work could focus on the user experience and experiment with additional pages for the recommendation system. Currently our team met our objectives in providing recommendations, however there are more features which could be explored and added to the project to further improve user experience.

6. Conclusion

Our group leveraged Azure as the cloud computing platform, such as the virtual machine, storage, and machine learning to build, deploy and manage our Dinewise restaurant recommendation project. We utilized a python / Jupyter notebook for data cleaning and machine learning, the Django framework for the front end, and Azure virtual machine as the platform for the front end, and Azure as the platform to support the testing and deployment.

Azure was a great platform to work on as they offer student credit and this helps reduce the economic barrier for us as students. We used the virtual machine which benefits the project due to its parallelism, allowing our project to have reduced processing time, allows for scalability, and permits us to execute multiple tasks simultaneously. The Azure virtual machine has the ability to scale up and down depending on the demand which exists at a given time on the system. Since this is a service on the cloud platform we can also monitor the performance, our utilization, and have real time knowledge on how the system is doing as users engage with the Dinewise project. Overall this was the platform that allowed us to store our dataset in Azure storage (the blob), allowing us to host the communication with the Django framework (virtual machine). In the future if the project was very popular then this platform would allow the group to rightsize the virtual machine, schedule jobs, and use the cost management tools to monitor and optimize the resources required for the project.

7. References

1. https://www.fortunebusinessinsights.com/food-service-market-106277
2. https://www.deploymachinelearning.com/start-django-project/
3. https://learn.microsoft.com/en-us/azure/virtual-machines/windows/quick-create-portal