CSE 5334 Course Project

Yelp Dataset Challenge – Project Report

Puneeth Umesh Bharadwaj (1001106478)

Shreekanth Kuppur Shreedhar (1001117162)

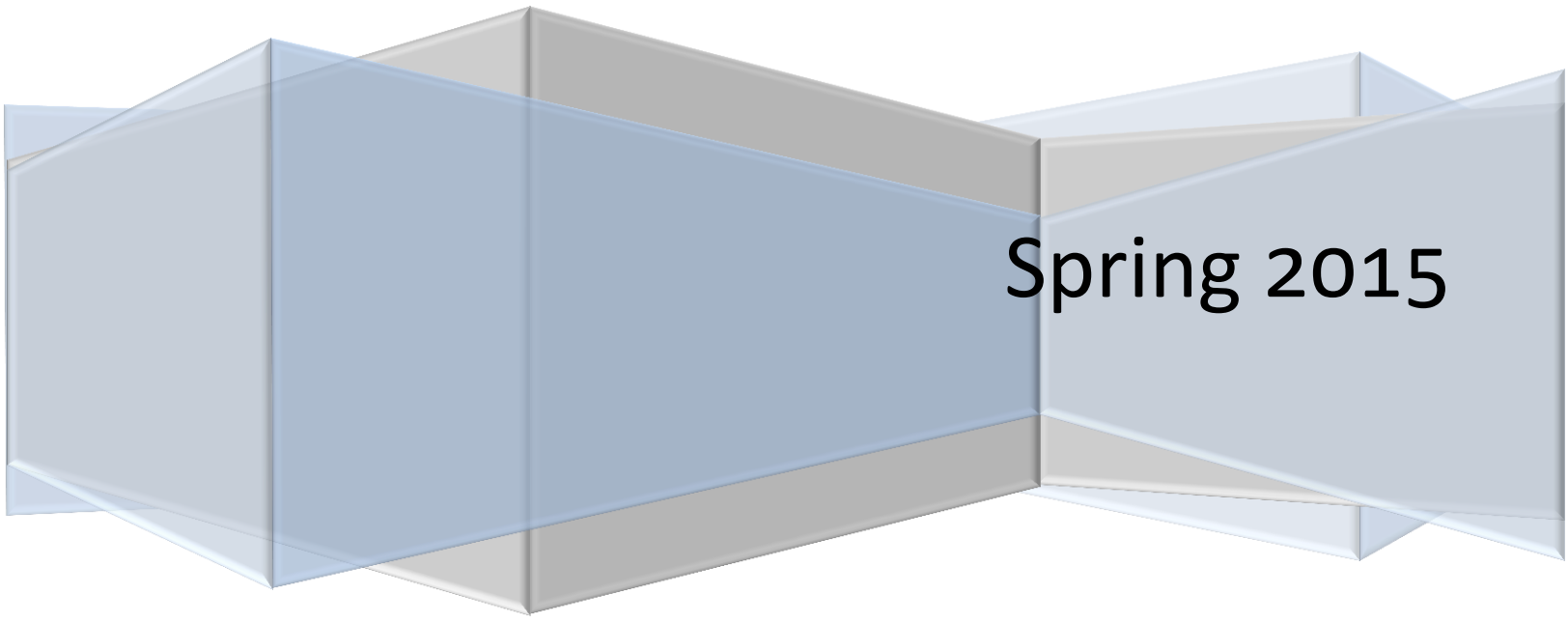


Table of Contents

Yelp Data Set Challenge:

The requirement for this project was to work with Yelp Data Set Challenge. You are required to analyze and mine the dataset and present the design, implementation, and results of your study. You have the full freedom to define the topic of your study. It can be classic data mining related tasks discussed in our course (classification, clustering, association rule, link analysis, graph mining, …) or those not discussed much in our course (collaborative filtering, prediction and regression analysis, sequential and time-series patterns, spatio-temporal data mining, sentiment analysis, named entity recognition and disambiguation, record linkage, data cleaning, …). It can even be new types of analysis that gains knowledge and insights from data. [1]

Project Proposal:

We have dataset provided by yelp, which has user, business and reviews data's of various restaurants. We use the review data and see the user reviews present, based on this data we predict and present to the user closest top 10 reviewed restaurants, display it on a map. This would help a user to know more restaurants closest to his liking and would help him to explore more. The project title is '**Restaurant Predictor**'.

Motivation:

We have a huge data in our hand which allows us to work on various data mining operations. We have various angles at looking at the data and trying to analyze the data. The motivating factor for our project proposal was reviewers or any user is on a constant lookout for new restaurants to try out in their locality. It helps when they have some source to suggest them restaurants in their locality or the places where they have visited earlier and provided feedback. Given a user this application would look at all their reviews may be in different locality or the same and predict them top 10 restaurants closest to their review locality and also closest to liking. This would help the users narrowing down on which new restaurant to visit without having to search for them and wasting time.

Changes from Initial Proposal:

In the initial proposal we planned to focus more on the check-in dataset along with the user review dataset. Also our major plan was to focus more on just the location aspect while predictions. After discussions with Dr.Li few changes in plan were made. We planned to use just the review and business datasets. We were able to provide top 10 restaurants using these datasets with good accuracy.

Data Mining Tasks:

We would talk about the file preprocessing, data mining tasks performed, libraries used in this section.

Data Used:

The data files considered in creating this web application were User data, Business data and the Review datasets. These were considered as we need the user file to get the details of user and match with the review, as in which user has how many reviews. Business data to get the details of restaurants used in prediction also getting the information of user reviewed restaurants. These data were included currently for our study; this could be further extended using the check-in data as well.

Data Pre-Processing:

The data provided by Yelp is in JSON format. We had to process this data and convert it to comma separated values for it to be easier to be loaded into database. We used already implemented code for this task. This code can be seen in Covert_Json_to_CSV.py [2] script in the provided scripts.

After the file conversions from json to csv, we loaded the data into tables. The design to use database was made the data would be more structured and it help us to perform joins and look at data with two or more datasets joined. Database used in this project was MySQL Version 5.6.19. New database was created for this as shown below,

```
create database dmfinal2;
use dmfinal2;
```

The tables were created as an example shown below. Initially we tried creating the table with review_id as primary key but due to vast amount of data the loading of data took more time than expected and the tables were created with no primary key.

```
create table review(
funny int,
useful int,
cool int,
user_id varchar(25),
review_id varchar(25),
text text,
business_id varchar(25),
stars int,
date date,
type varchar(50));
```

All table creations can be seen in SQL_Operations.sql script. MySQL has option to do a data dump from csv file to Tables using Load Data Local Infile. The data loaded to tables using this as shown below,

```
load data local infile ‘File_Path’
into table review
fields terminated by '\t'
lines terminated by '\n';
```

Further queries were performed on the database to get a better hang and understanding of data. Some interesting queries as suggested by [3] were as shown below.

Top 25 Business with most reviews.

```
SELECT name, review_count
FROM business
ORDER BY review_count DESC
LIMIT 25
```

Top 25 Coolest Restaurants.

```
SELECT r.business_id, name, SUM(cool) AS coolness
FROM review r JOIN business b
ON (r.business_id = b.business_id)
WHERE categories LIKE '%Restaurants%'
GROUP BY r.business_id, name
ORDER BY coolness DESC
LIMIT 25
```

Some more queries performed by us.

To view the top 10 User based on number of reviews,

```
SELECT name,review_count
FROM user
ORDER BY review_count DESC
LIMIT 10;
```

Most Useful Review found by users based on review data.

```
SELECT *
FROM review
ORDER BY useful DESC
LIMIT 1;
```

These helped in better understanding the data. The required review data was pulled using the below query.

```
SELECT DISTINCT CONCAT((REPLACE(REPLACE(categories, '\r', ''), '\n', '')),' ') category,
        (REPLACE(REPLACE(attributes, '\r', ''), '\n', '')) attributes,b.business_id,
         user_id,concat(b.name,' ') name,b.review_count,b.stars,b.longitude,b.latitude
FROM review r, business b
WHERE r.business_id = b.business_id
AND categories LIKE '%Restaurants%'
AND user_id LIKE '%USED_ID_REQUIRED%';
```

Using this Query we got all the Reviews and corresponding Business data of a given user. This can be used to get the data for any given user. Also if data needs to extracted using user name instead of User ID we can do so by joining the User table as well and getting the User_ID of that user and querying. The query result was exported to CSV file.

K Nearest Neighbors:

In pattern recognition, the k-Nearest Neighbors algorithm (or k-NN for short) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression: In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors. k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms. [4](Source: Wikipedia)

Application of KNN:

We used K Nearest Neighbor algorithm to predict the restaurants to the user. We used existing python library 'NearestNeighbors from sklearn.neighbors' [5][6][7], to get the k Nearest Neighbors. The library trains the data using a Nmpy Array which was got using the NumPY library. The array required should have all the features which we need to train the data on. For this we used all the categories of the user review provided business, also the latitude and longitude to get the predictions closer to the locality he has already reviewed. We used the review count and number of stars attributes to get better results. The training of data was done as shown below,

```
import numpy as np
X = np.array(final_user_arr)
nbrs = NearestNeighbors(n_neighbors=15, algorithm='ball_tree').fit(X)
```

We used the K count to be 15, i.e 15 nearest neighbors. After the training the user reviews data, we applied the classifier on business data. Same attributes were used i.e. the category of business, we restricted the data to Restaurants this can be extended to all data if need be. Also the Latitude, Longitude, Review Count and Number of Star attributes. The application of classifier on Business data to be classified was done as shown below,

```
numpy_array = np.array([return_arr])
distances, indices = nbrs.kneighbors(numpy_array)
```

This returns the distance and index of the current business from the user reviewed businesses. We save this in a data structure, and sort it in ascending order of distance as the smaller the distance the closer it is to the user reviewed businesses. Later this can be used to retrieve the closest Business or Restaurants and display it to the user. This can be seen in Restaurant_Predictor_using_KNN.py script.

Validation of Classifier:

We performed the validation of the classifier to make sure the tasks we performed by us is valid and the classification is working as expected. The logic behind the validation was, suppose we split the user review data obtained from the database to around 80% train and 20% validation data and we train the data and apply the trained classifier on the Business data we would get top restaurants, out of which at least a few should belong to the 20% test data of user.

Using the above same concept we randomly divided the user review file into 80% and 20%, we used the NumPy library for this as shown below,

```
import numpy as np
file_line_num_arr = sorted(np.random.choice(range(0,589),400,replace = False))
```

Use file_line_num_arr and check if the line number belongs to this array add it train file else add it to validation file. Then we perform training and classifying of the data as specified previously. After the restaurants and the distance values are obtained we compare this result with the validation file created from the user review file. We tried executing the validation script quite a few times to make sure the result we got was legit. We did find change the K value also restriction of top results like top 30 or top 40 and so on. We found few varying results for the same.

The maximum we found was 8 predicted restaurants matching with the user review out top 30 predicted restaurants. We did find zero results matching the top 30 results returned but that was 1 or 2 out of 10 runs. Similar runs could be performed with changing number of K and top results simultaneously. The validation helped us verify the classifier used and our approach. The validation operation can be found in Validation_KNN.py script in the submission.

These were the main data mining operation performed for this project.

After KNN:

The results obtained after the classification were stored in an array of tuples, the tuples being the Business ID and the distance score of that Business/Restaurant from the user review. The array had tuples sorted in ascending order of distance. Ascending order as lesser the distance more close it is to user data. The next part was to display the obtained result on browser. We created a dictionary with the Business_ID as key and the value being an array of Business/Restaurant name, Latitude and Longitude of the Business for future use. The resulting business ids obtained were then looked up in the dictionary to get the business name, latitude and longitude of the business. We write the obtained values to a comma separated value file, this is written by separating the values using comma as separator and newline character as field terminator. This file is used in the web application to display the results on map.

Visualization of the Outcome:

Visualization of the outcome was done using a web server application, generated results for random 10 users and gave the option to select the user on the web site. When a user is selected and 'Go' is pressed it shows top 10 restaurants closest to that user which the user has not visited on a Map. When observed on the map the locations predicted appear closer a particular location or a city showing that the results are based on location of user's previous review.

Web Application:

We have used the following in our projects web application

1. Flask, version 0.10.1
2. Google Maps Javascript API v3
3. Bootstrap CSS

Flask is a lightweight framework written in Python, based on Werkzeug and Jinja2. This allows us to integrate our Python application with the web framework, to display our results. [8]

The user selects his/her name in the dropdown list and submits it to the website. The "request" of Flask picks up the username and sends it to the python code written on the server side. The "views.py" handles the user request. Based on the name, the corresponding data is picked up and redirected to "suggestions.html". The "suggestion.html" renders the Google Map markers of the top 10 restaurants. These are the restaurants predicted by KNN to closest to Users previous reviews.

The Google Maps Javascript API v3, used in the "suggestions.html", will pick the latitude and longitude sent by "views.py". The API uses reverse geocoding. The term geocoding generally refers to translating a human-readable address into a location on a map. The process of doing the opposite, translating a location on the map into a human-readable address, is known as reverse geocoding. [9][10]

Bootstrap provides a set of stylesheets that provide basic style definitions for all key HTML components. These provide a uniform, modern appearance for formatting text, tables and form elements. In addition to the regular HTML elements, Bootstrap contains other commonly used interface elements. These include buttons with advanced features (e.g. grouping of buttons or buttons with drop-down option, make and navigation lists, horizontal and vertical tabs, navigation, breadcrumb navigation, pagination, etc.), labels, advanced typographic capabilities, and a progress bar. The components are implemented as CSS classes, which must be applied to certain HTML elements in a page. [11]

The source code can be found inside the SLP_Restaurant_Predictor folder. To run the application run the 'run.py' script this would start the service. To make is accessible without running the script each time manually, we have uploaded the scripts to Amazon Web Service Cloud EC2 instance and kept the script running at background. By this we can view the website from any browser using the URL provided below.

URL of Project Website:

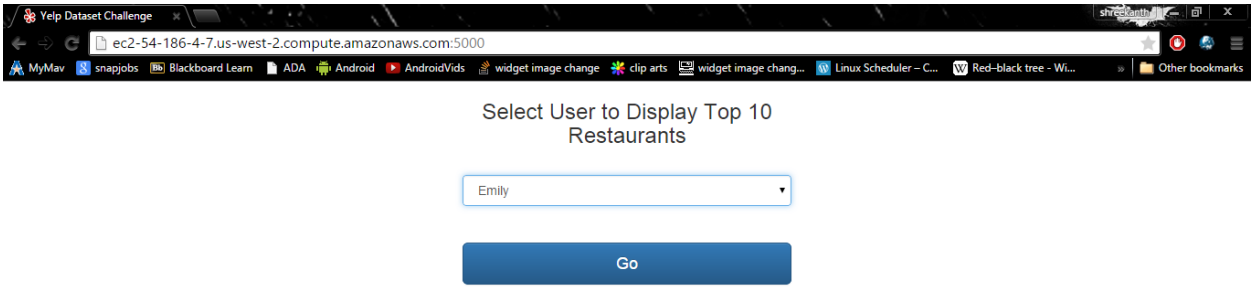
http://ec2-54-186-4-7.us-west-2.compute.amazonaws.com:5000/

Website Hosted on:

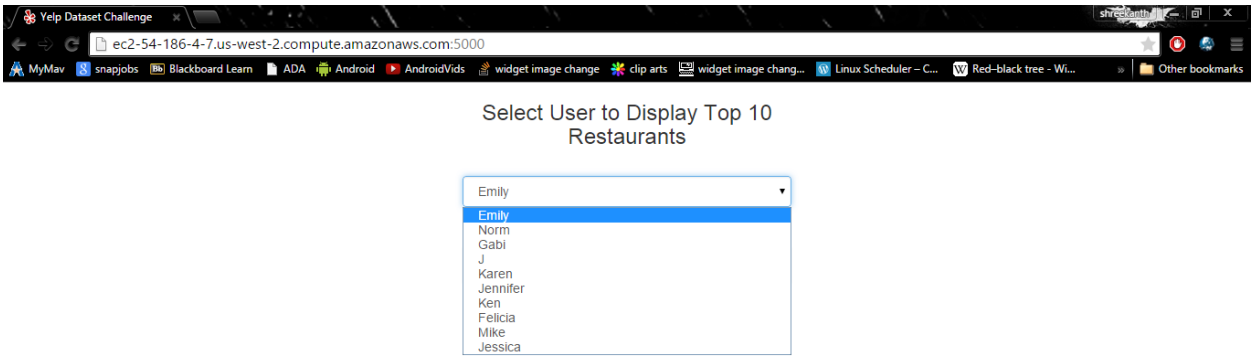
The website is hosted on Amazon Web Services Cloud Platform. We created a t2 Micro(free-tier) instance of Elastic Cloud Compute(EC2) and kept the flask script running in background as mentioned earlier.

Screen Shots of Web application:

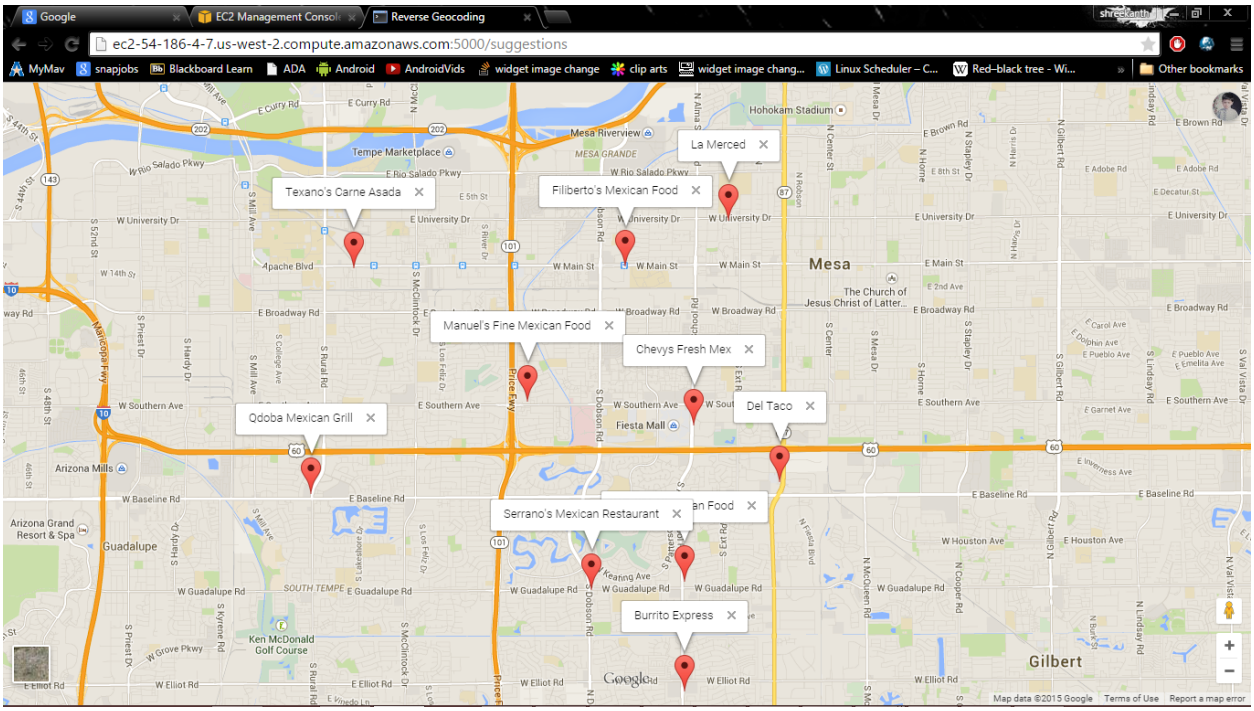
Below is the screen shot of the first screen when the URL is accessed. The default user selected is 'Emily'.



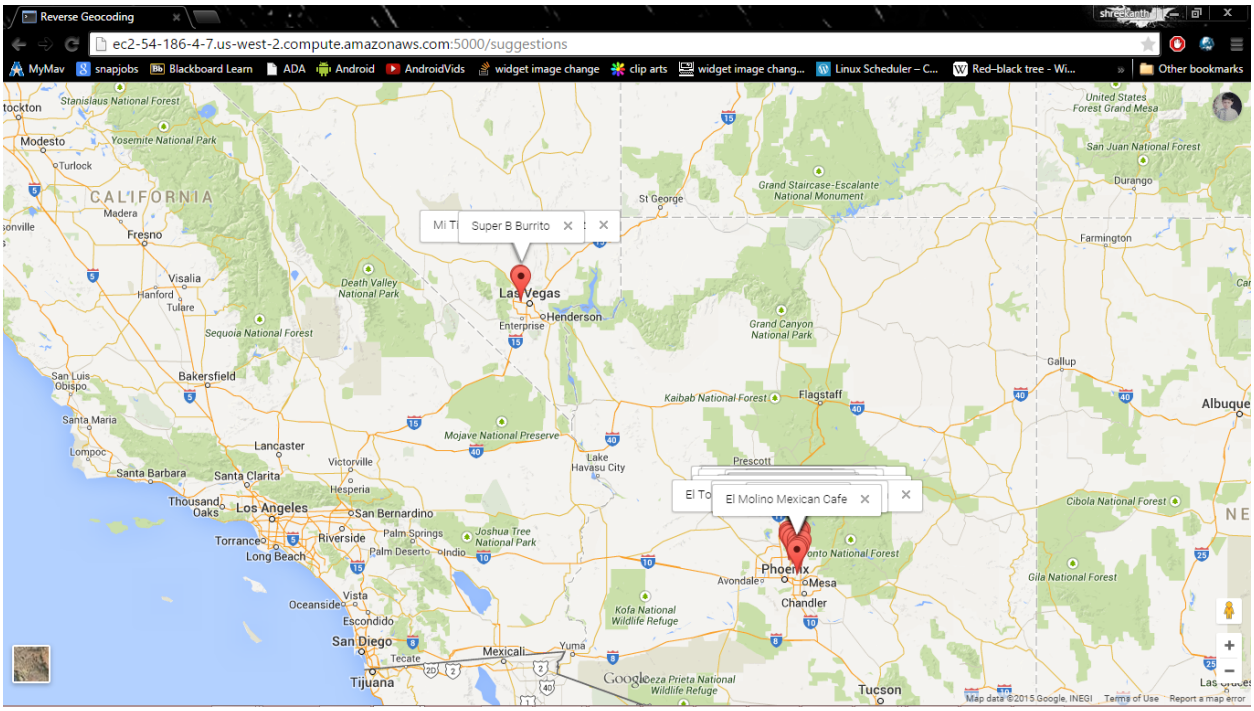
Click on the drop down to select a different user as shown in the screen shot below.



Click on go after selecting the user to display top 10 restaurants with the name populated from our application as shown below in the screen shot.



The below screen shot shows that the user has visited and provided reviews in two different places hence the prediction results are in two different places. Here to show the functionality we have increased the number of restaurant to being Top 20 closest restaurant.



Challenges Faced:

The major challenges faced while implementation of this project was the preprocessing of data as we wanted to load the data into a database to perform queries on the database. The data being in JSON format made it difficult to do the same. We had to search for solutions for the same we came across the conversion already provided in [12], in json_to_csv_converter.py. But this was of little help as the data got from this script was vague and not usable as per requirements. Hence we had to start all over again found the solution in [2].The code in this helped us cover the files to csv as required and load the data to database.

The second challenge faced was the implementation of K Nearest Neighbor algorithm. We were not sure initially regarding the implementation of KNN to be done manually or to use a library. We looked for any available library for the same and found NearestNeighbors in sklearn.neighbors library which helped us find the K nearest neighbors to the user review given. This was cost effective and provided the result within couple of seconds.

The final challenge we faced was the integration of all the process together. The query execution took about 30 to 40 seconds to complete and the process of prediction took another 30 to 40 seconds to complete, this was a problem when tried to run together when selected from the website. So

we decided to manually create the results for 10 random users and display the results. The users provided are as shown below

User Name	Review Count In User Table
Ken	11
Jessica	58
Felicia	430
Mike	223
Karen	1202
Emily	1353
Norm	1650
J	1971
Jennifer	1155
Gabi	1505

Link to Git Hub:

The link to our GIT hub repository is as below.

https://github.com/puneethshreekanth/5334Project.git

Conclusion:

To sum up, we worked on Yelp data set challenge which had all the business data and the users who use yelp, their checkins, the user reviews and so on. We proposed and worked on a restaurant predictor which would consider the previous reviews of any given user and provide top 10 restaurants closest to his liking and in his locality. K nearest neighbor was used in the implementation of the project. A validation task was also performed to validate the working of the classifier and overall project itself. The result obtained was displayed on a map via pins and showing the user the name of restaurant and the location of the restaurant. This application would help the users in finding restaurants closest to their liking and in his previous reviewed locality.

References:

[1] http://www.yelp.com/dataset_challenge

[2] https://raw.githubusercontent.com/romainr/yelp-data-analysis/master/convert.py

[3] http://blog.cloudera.com/blog/2013/04/demo-analyzing-data-with-hue-and-hive/

[4] http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[5] http://scikit-learn.org/stable/modules/neighbors.html

[6] http://stackoverflow.com/questions/14505716/implement-k-neighbors-classifier-in-scikit-learn-with-3-feature-per-object

[7] http://blog.yhathq.com/posts/classification-using-knn-and-python.html

[8] http://flask.pocoo.org/docs/0.10/

[9] https://developers.google.com/maps/documentation/javascript/

[10] http://en.wikipedia.org/wiki/Google_Maps

[11] http://en.wikipedia.org/wiki/Bootstrap_%28front-end_framework%29

[12] https://github.com/Yelp/dataset-examples