Large Scale Analytics Project Report

Predicting Restaurant Setup Using Yelp Dataset

Monica Dommaraju
Sri Sruthi Chilukuri
Vijayalaxmi Nagurkur

Index

1. Project description

Ideology and analysis

Restaurants have become the economy's best friends. The rise and fall of this industry has shown clear impact on the overall economy. It caters the businesses and socio-economic needs of the society and hence has played a significant role in shaping any country's economy standards. According to the latest statistics report from the National Restaurant Association[NRA], in just the United States itself, the restaurant industry has contributed nearly 4.5% to the Gross Domestic Product.This has grown immensionaly in the last two decades from 2% to 4% which clearly shows how strong the dependency has been. This growth is said to further continue if factors like a healthy labour market or a good wage market are sustained.

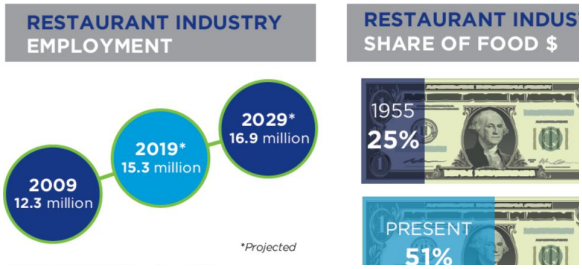


Image courtesy : NRA

A statistical survey carried out by NRA states that more than 9 in 10 restaurants have employees less than 50 and more than 7 in every 10 restaurants are independent units. These two statistics give us a clear implication that there are a wide variety of restaurants out there meeting every customer's requirement. With proper analysis, we will be able to retrieve information about those profitable factors that are responsible for a restaurants growth and with such proper analysis, we will be able to understand which of the features among them can give us an idea of the profitable factors that are affecting their boom.

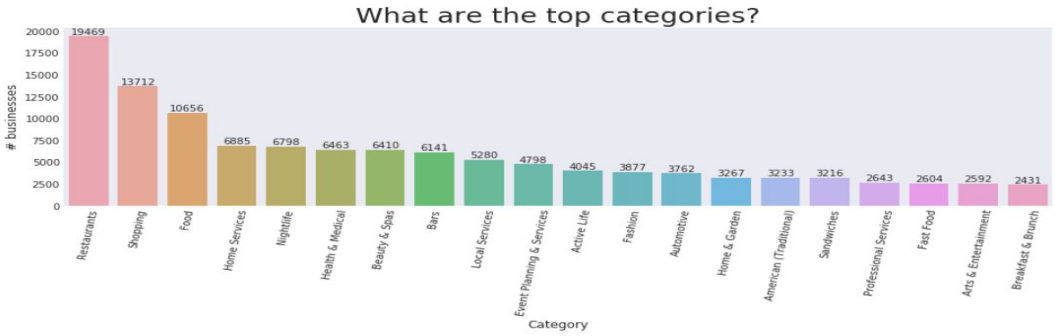
One of the most important features is the public opinion.What customers say about a business weighs a lot and becomes responsible for determining the life of that business. There are various review sites over the web like Yelp, Zomato, Trip Advisor etc where customers share their experiences.Websites have been a source of information for all business holders to know which areas of the business need improvisation. For example, if a customer is unhappy with a restaurant not having a WiFi or polite staff, the owners can look into those issues and fix them.

About the dataset

We are using the records from Yelp dataset challenge 2019 to analyze these trends. The dataset includes reviews of all kinds of businesses within two countries: the US and Canada. The dataset contains around 192609 records with business attributes like name, location, review count, stars, price range, etc. Hence, through this project, we would like to aid the process of decision-making of how good the idea is of setting up a new restaurant in a given place.

We tend to identify a certain set of attributes like GoodForKids, Restaurant Delivery, parking, Wi-Fi, Alcohol, Vegan, Chinese, Indian, etc from the dataset containing restaurants data and determine which variables affect the goodness of a restaurant. A recent study carried out on the popular review forum Yelp shows us that “Restaurants” is one of the most reviewed businesses. This gives us an advantage as we can take it up to study in detail and skeptical manner

Graph 1



Graph 1 shows us how the distribution statistics have been for each of the businesses. Around 19469 records have been found solely for the restaurants business.We have used this entire chunk to draw our insights.

2. Project Requirements

We have used a number of tools and technologies in every stage right from the data preprocessing to visualization, model building and deployment.

Tools and technologies used:

- Pyspark
- Amazon RDS
- Elastic search
- Logstash

- Kibana
- Google AI
- Google Cloud Storage
- Tableau
- TransmogrifAI AutoML
- MySqlWorkBench
- Flask (For RESTful APIs)
- React (For Front End)

Core libraries:

Core libraries that we have used for Data wrangling, visualizations and building machine learning models are

- Numpy
- Pandas
- Scikit-Learn
- XGBoost
- Matplotlib
- Seaborn
- GraphViz
- Findspark
- Google Python Client
- SqlAlchemy
- mysqlclient
- requests
- google_auth

3. The KDD process implementation on Yelp

Knowledge Discovery of the Databases is a data mining strategy that is aimed to extract non-trivial , insightful information from raw and unstructured data. The process typically consists of five major steps that are followed sequentially and iteratively until all the useful insights are extracted.

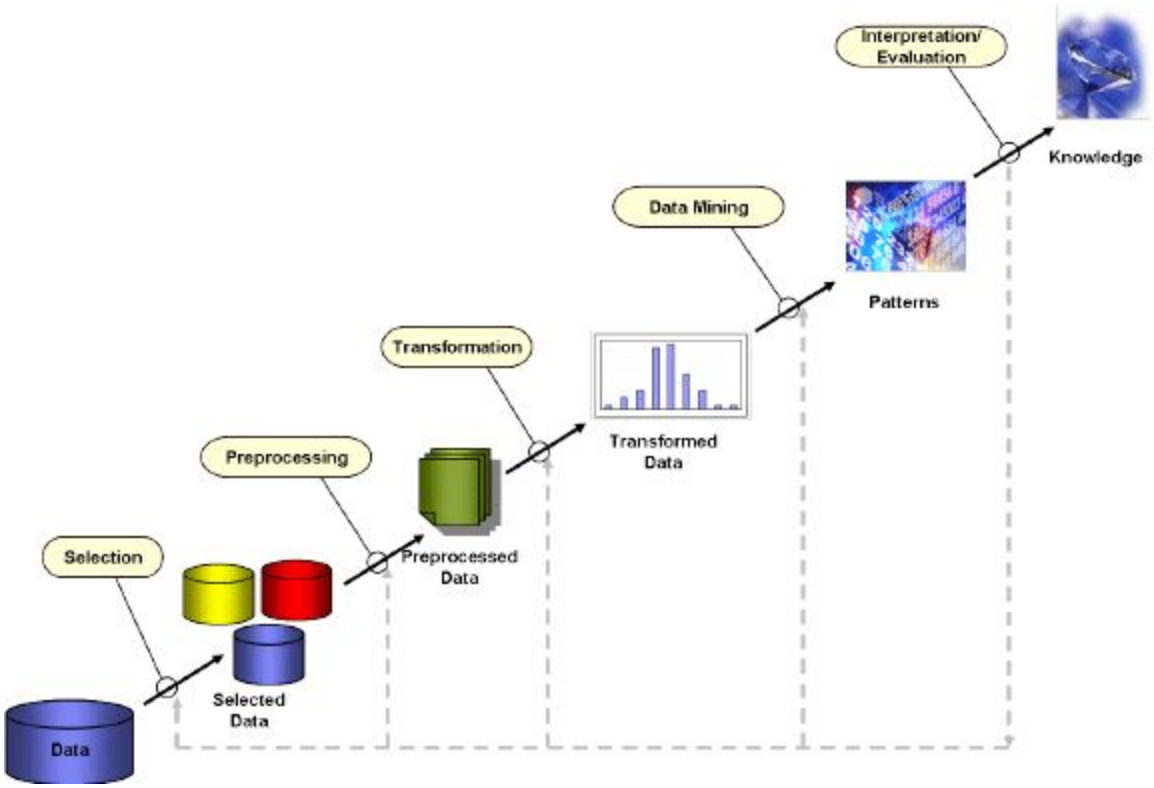


Image courtesy: ResearchGate

For our project, we have applied the five step methodology on the Yelp json file to understand various scenarios.Here are the steps followed for each signature step of the KDD process.

Data selection

- This step includes developing an understanding of the knowledge domain. We have researched on the restaurant industry to acquire knowledge about various types of restaurants, types of cuisines, food types, amenities provided by different restaurants,etc.
- The relevant and sufficient knowledge has been gained by our team about how the restaurant industry works, its contribution to our economy and what factors critically affect the rise and fall of the restaurants.
- Data selection also involves analysing the goals of the end user. We aim to build a model that provides the end user with a privilege to be able to predict how well his/her restaurant will boom with a given set of amenities even before it is setup.

Data preprocessing

- Preprocessing involves data cleaning, removing unwanted values and filling null values.After we have extracted the json file from Yelp, we have fed it to the Pyspark running locally on our machine. We obtained a processed flat file in comma separated value format.
- The sample space so obtained had a categories attribute with categorical data in the form of strings. These strings had to be converted into individual attributes and applied one hot encoding technique to make them more usable. One-hot encoding is a popular technique that is used to typically convert categorical data into a form that becomes easily understandable by the Machine Learning algorithms.
- After this process, some duplicate columns have been generated that had to combined using aggregation.
- All the restaurant related attributes have been filtered amongst which there were many categories like cafes, chinese, vietnamese, indian, american(old), american(new), etc. Through this selection process, we were able to reduce the file from 192609 samples to 59387 restaurant-related samples.
- When it comes to data reduction, columns like index, business ID, Name , that don't contribute to our model can be omitted which reduces the dimensionality of the dataset.
- Further, the attributes that contained more than 65% of the missing values can be eliminated.
- From this dataset, the samples that contained “Las Vegas” as the city have been extracted.
- We have selected restaurants in “Las Vegas” city for our analysis because it had maximum number of reviewers which has enabled us to have maximum number of insights.

Data Transformation

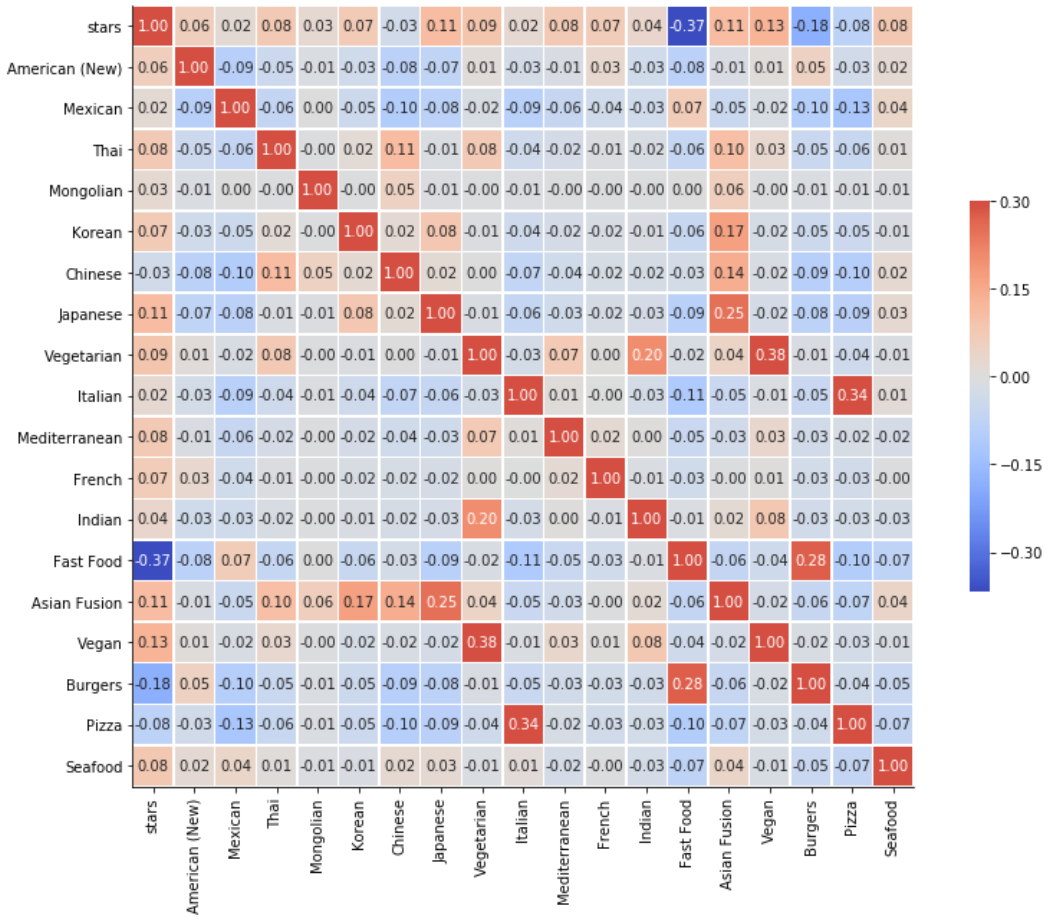
- The transformation of data refers to imputing missing values, performing feature engineering and dimensionality reduction.
- We have come a long way from processing the raw json file containing all the businesses data to creating a separate flat file to study the behaviour of region specific restaurants from the city “Las Vegas”.

Data mining Task

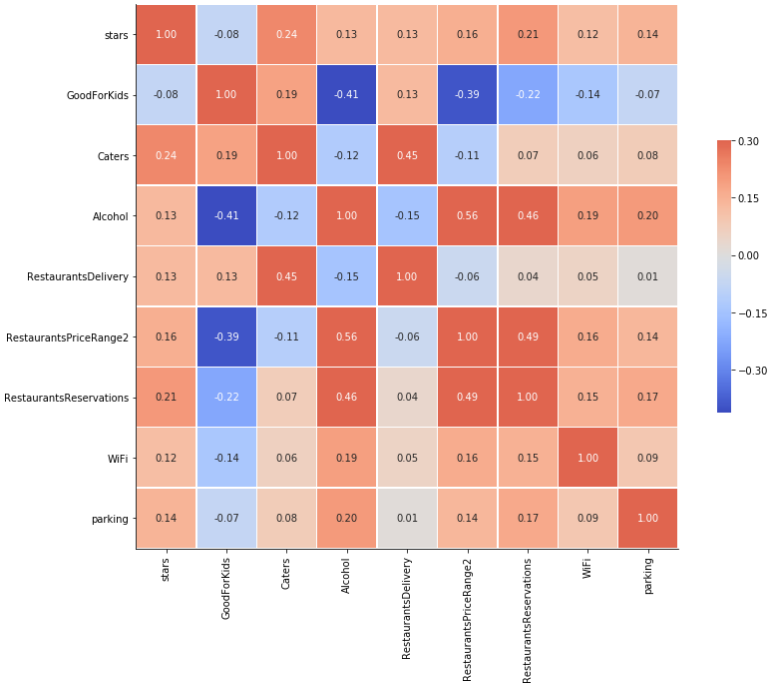
- The goal of the entire KDD process is to perform classification. After the data has been completely cleaned, structured and made model-friendly, we had to choose a range of classification algorithms to be applied on the dataset and evaluate performance.
- Various Machine Learning algorithms have been applied with parameter tuning.Some of these algorithms are:
    - Decision Tree
    - Naive Bayes'
    - K- nearest neighbour
    - Support vector Machines
    - Logistic Regression
    - Random forest
    - Gradient Boost Classifier
    - XGBoost Classifier

Interpretation and Evaluation

- Also, the performance of each of these models has been evaluated using various metrics like K-fold, Accuracy scores for test data and train data, classification report and confusion matrix. This draws us to conclude on various milestones as of how a specific model performed better over the other and why.
- Apart from the model based interpretations, there are few interpretations that could be derived by just visualizing the dataset itself. To be precise, after generating the correlation matrix between different cuisines and star ratings, we were able to draw certain inferences.We have also run the correlation matrix to identify the relation between the business star ratings and amenities/services provided by restaurants. We found that there is a high probability of a restaurant getting successful if it provides catering and restaurant reservations. Interestingly, we found that the restaurants that are good for kids have less star rating. Other interesting fact from our data analysis is that most of the restaurants serving alcohol also offer restaurant reservations. These facts can be taken into consideration while setting up a new restaurant business.
- Below is the correlation matrix that was developed to understand the feature correlations better.



Also, the relation between the amenities and the stars is depicted as following :



Knowledge

- We have developed an understanding of which model performed the best. This helped us understand if there was any need to perform feature engineering again to extract even more useful insights if we don't observe optimal accuracies.

4. Feature Engineering

- In this step, we aimed to find the useful set of features that can contribute to the performance of our model. It involved filling missing values, dimensionality reduction, feature extraction and feature engineering. We had to omit the unnecessary features, modify the existing ones and also engineer new ones that could help us understand the correlations better.
- To do this, we have framed certain strategies:
    - The noise level attribute that had categorical columns has been label encoded.
    - To fill missing values in "Alcohol" ; we have framed a condition which says if the restaurant is good for kids, then alcohol is not available.
    - To fill the missing values in "catering" ; we have framed a condition which says if the restaurant has "food delivery"; then it provides catering services too.
    - To fill "Has TV" attribute we have imputed the value to be True if the "noise level" was loud.
    - To fill "WiFi" column, we have used the "expensive" attribute. If the restaurant was expensive, it was assumed to be having "WiFi.
    - Other attributes like "noise level", "outdoor seating", "Restaurant reservations" have been filled in with whatever their "mode" occurrence was in the entire dataset.

- To perform dimensionality reduction, we have followed a series of sequential steps that could reduce the sparseness of the data.
    - The attributes "garage" and "lot" have been combined to "has_parking".
    - If the "**price_range**" was greater than or equal to 3, then we have set the "valet parking" attribute to "true"
    - If the "**price_range**" was less than or equal to 1, then we have set the "street parking" attribute to "true".

- To detect any outliers and remove them, we have followed a series of steps:
    - For the "review_count" attribute, values that were greater than or less than 1.5 times the "InterQuartile Range" have been removed.

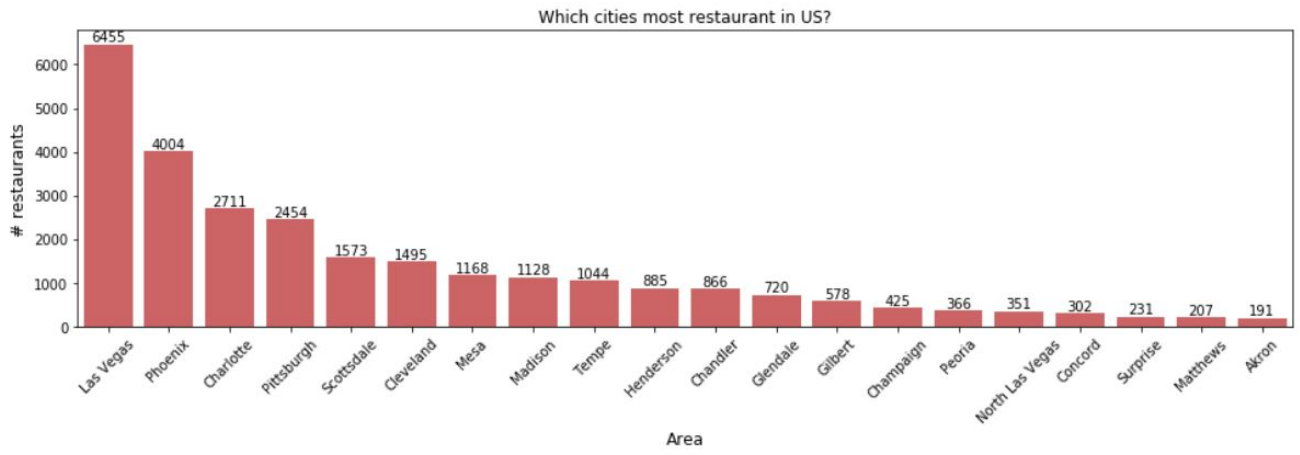
- As part of feature engineering, we have also created a new attribute named “score”. Its values are also built using an objective equation which weighs two most critical attributes i.e. the star_rating and reviews_count. The star_rating has been given a weight of 60% and the review_count a weight of 40%.
  Equation is :

$$0.6(star\ rating) + 0.4(review\ count) + \varepsilon_0 = \Delta(model)$$

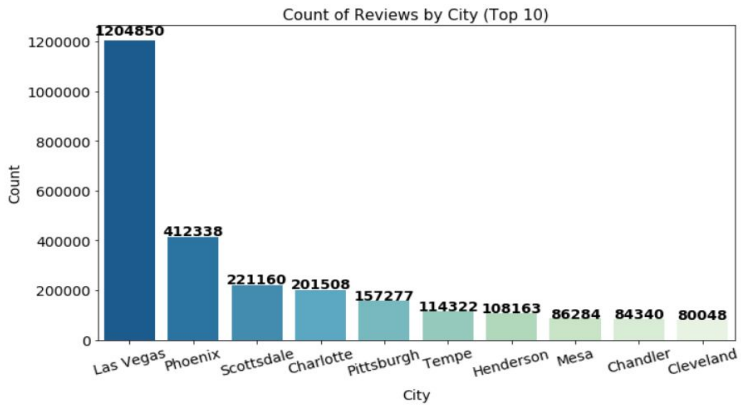
  Where, star_rating is the overall rating assigned the restaurant by the customer. It ranges between 1 to 5; 1 being the least rating and 5 being the highest. review_count is the total number of reviews available to that particular restaurant in that city. It takes any numerical value above 0. However, to avoid bias we have normalized these star_rating and review_count using MinMaxScaler to fall in the range of 0 and 1, This enabled us to calculate the final score based on the above formula.

5. Data visualization and Exploratory Data Analysis:

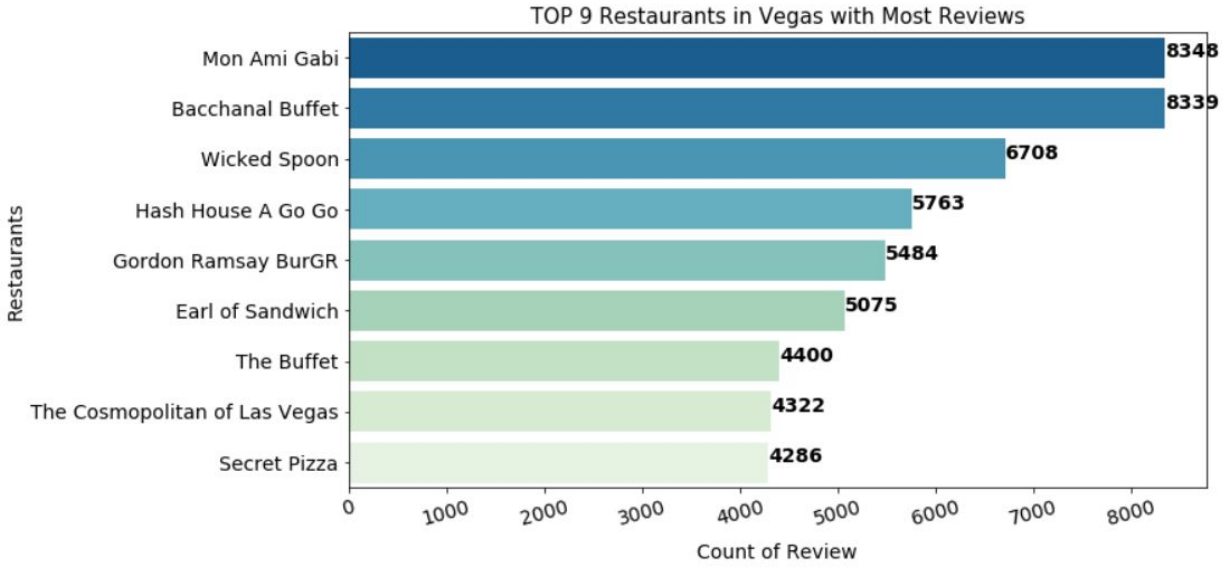
To visualize how many restaurants are located in each of the cities:



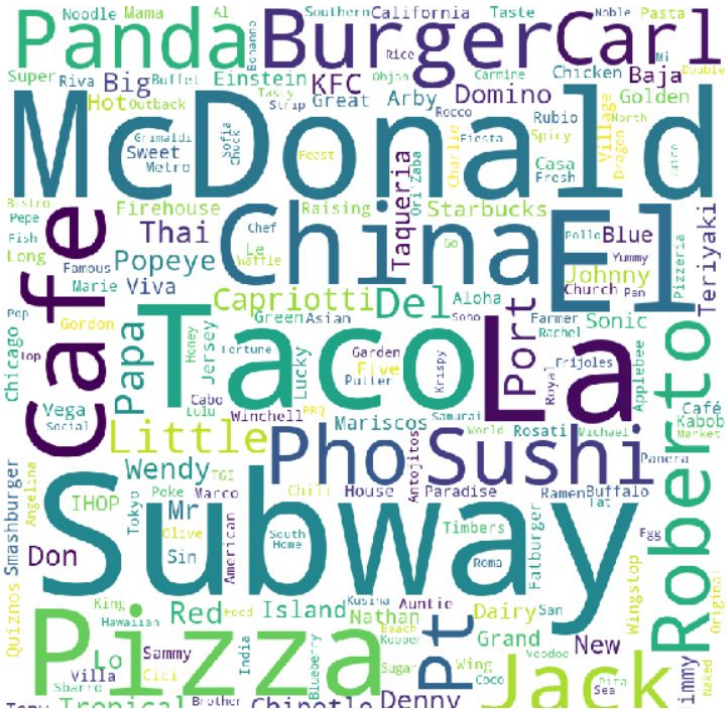
To visualize the city wise count of reviews:



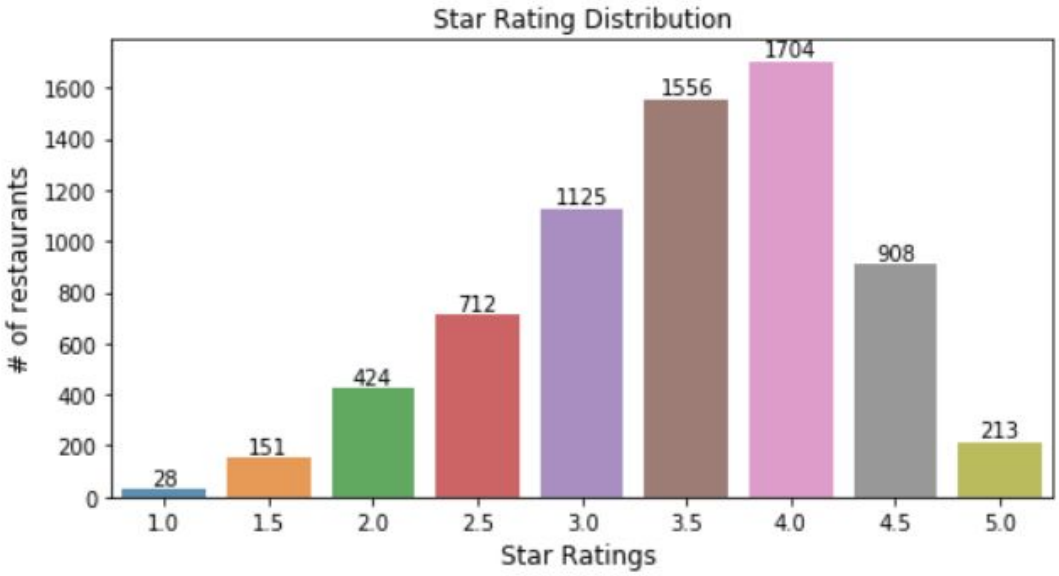
To visualize the top 9 restaurants in Vegas with more number of reviews



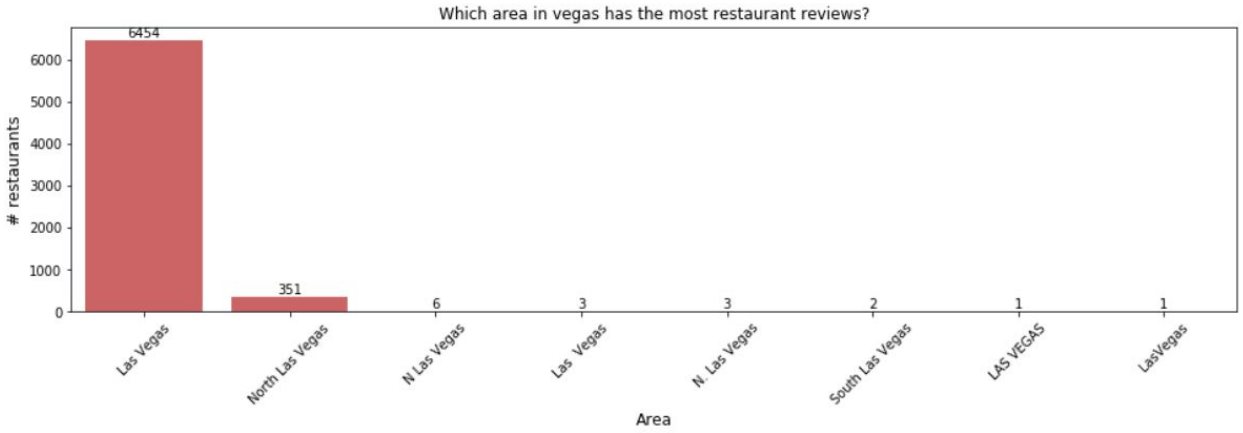
Generating word cloud for the restaurants in the city “Las Vegas”



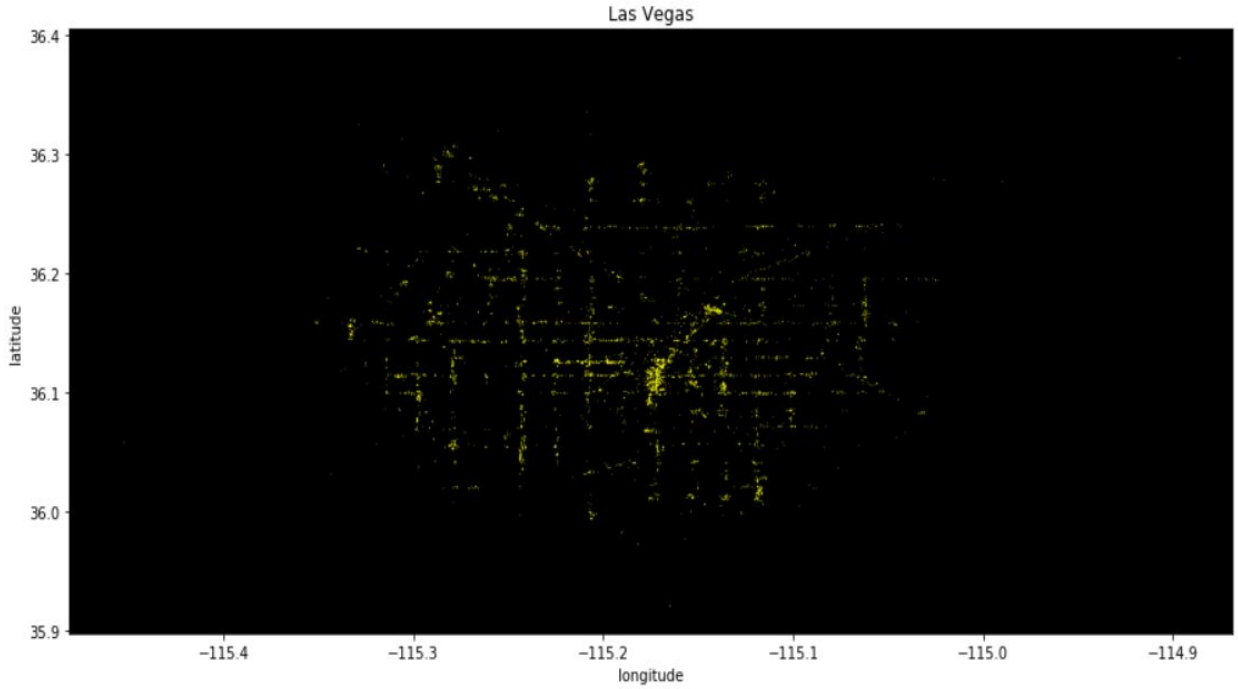
To visualize the number of restaurants having each of the star rating:



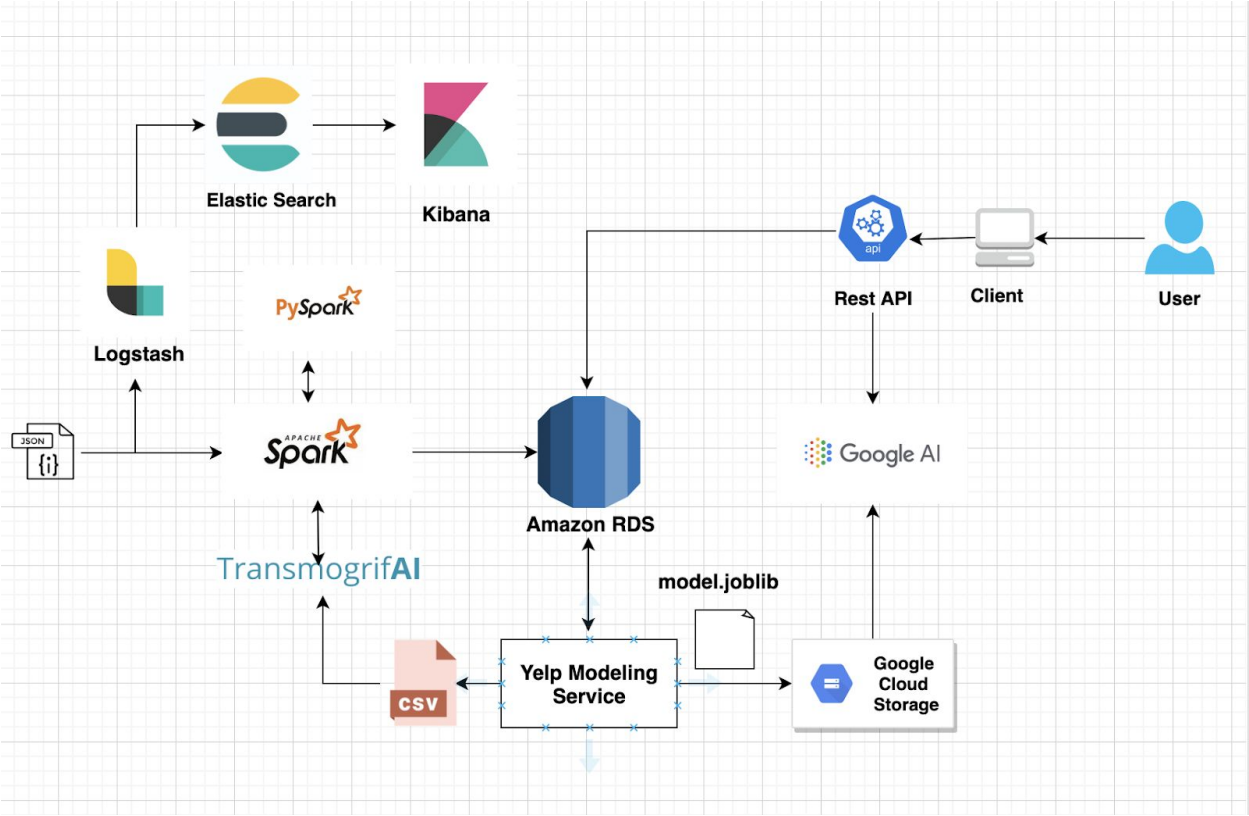
To visualize the number of restaurants in Las vegas, located area-wise.



To generate a heat map for all the restaurants in the Las Vegas city, based on the latitude and longitude values.



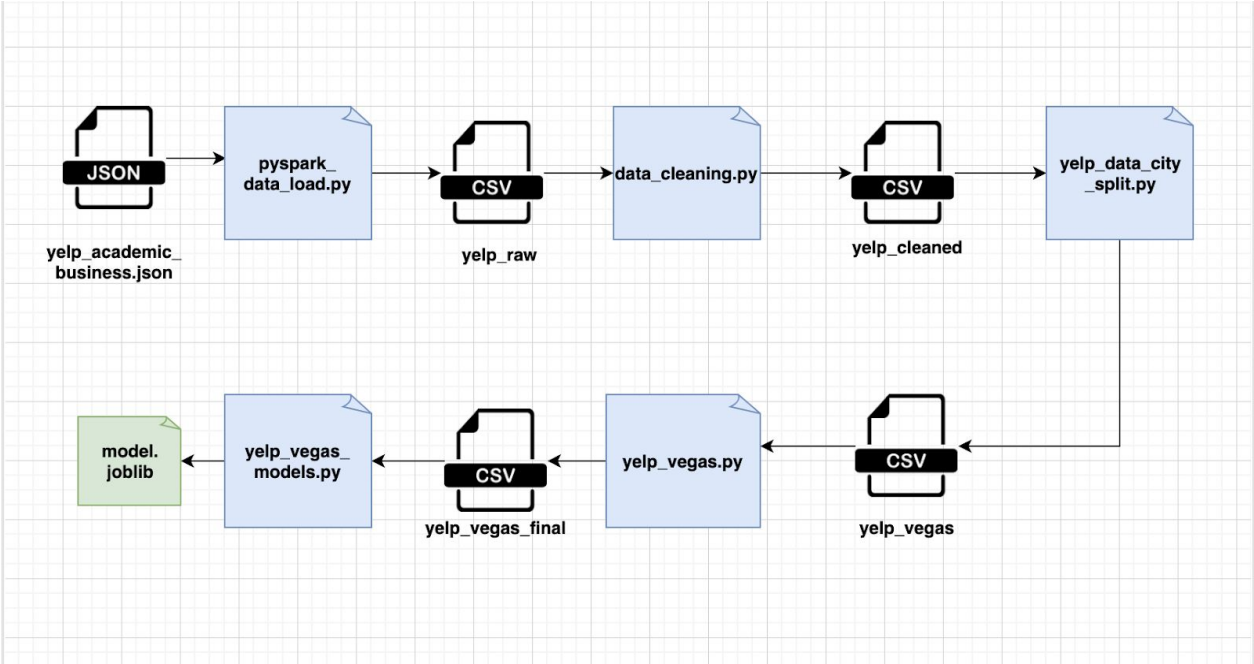
6. High level Architectural Design



1. **JsonFile:** The input we have got from Yelp was in the format of list of json strings.
2. **Logstash:** Logstash is a component of ELK stack by Elastic. This allows us to create pipelines with minimum effort. Logstash supports of hundreds of plugin to ingest data, transform data and export data. For our pipeline, we have used file plugin to ingest json data from yelp dataset file and exported it to elasticsearch using elastic plugin.
3. **Elasticsearch:** Elasticsearch is the heart of ELK stack and is the document repository. It stores json documents and indexes them in an efficient way using Lucene search engine. This makes the text searches very fast in an efficient manner. We are using elasticsearch to store all yelp json documents for our initial exploratory analysis.

4. **Kibana:** Kibana is the visualization tool that queries elasticsearch and display the results to the user. Kibana 7 also has new Machine learning module which can be used to visualize the characteristics of each and every feature.
5. **Spark:** Spark is the in-memory distributed computing framework that runs 100 times faster than hadoop. We used spark framework to perform initial data processing and cleaning. This helped us do some complex operation in very less time. Spark is natively developed in Scala. We have used PySpark version of spark for our data cleaning purposes.
6. **Amazon RDS:** We have used Amazon MySql RDS as our warehouse for storing datasets extracted after our analysis at various stages.
7. **Yelp Modelling Service:** This service is the core of our project. We have all our data cleaning, transformation code along with modelling code in this service.
8. **Model.joblib:** We have used joblib scikit library to export the trained models.
9. **Google Cloud Storage:** We have uploaded our model dumps into Google cloud storage bucket, so they can be used later for model deployment.
10. **Google AI:** Google AI service supports multiple ML operations like, creating pipelines, AutoML, Model deployments, Versioning etc. We have used Google AI for deploying our trained models.
11. **RESTful API:** We have developed a backend python Flask application to listen to requests from Client and forwards that request to Google AI model that we deployed in the previous step.
12. **Client:** We have also developed a React Front End application to enable users interact with our service.
13. **TransmorgrifAI:** TransmorgrifAI is an open source AutoML framework developed by salesforce which runs on top of Apache Spark. We have used this framework to validate our model selection and also to choose the best estimator parameters.

7. Data Flow Diagram & Component Level Design



The above figure shows the data flow of how we performed data analysis at each step until we exported our trained models.

Pyspark_data_load.py:

This module consumes the huge json dataset file that we acquired from Yelp. This module runs on top of PySpark and performs many data transformations on multiple unreadable features and converts them into readable .csv file. We have stored the cleaned dataframe at this stage into Database as Yelp_raw table.

Data_cleaning.py:

This module reads the data from yelp_raw table and perform expensive operations such as splitting the Categories features into multiple features, performing one hot encoding on these categorical features etc. This module finally saves this intermediate version of dataframe into DB as yelp_cleaned table for later use.

Yelp_data_city_split.py

This module reads the sql table generated from above step and perform basic data engineering operations like removing columns with more than 90% missing values, removing the columns with singular values etc. Finally this module splits the samples into subsets of data samples based on the city. It then save all these subsets into DB. One such subset we will be focusing later is yelp_vegas dataset.

Yelp_vegas.py

This is one of the important modules in this project, as it contains all the feature engineering tasks. We have done a lot of analysis on the features to impute missing values, tried to find correlation among features, reduced features by using data selection techniques, grouped columns to reduce dimensions and many such techniques. The final output dataset from this module is the final dataset that is ready to be fed to models. We saved this final dataset in the DB as yelp_vegas_final.

Yelp_vegas_models.py:

This is our final module which reads yelp_vegas_final dataset from DB. We have run multiple classification algorithms on this module. We have performed Kfold, cross validation on each model to make the scores more reliable. We have also integrated evaluation metrics like training and test scores, cross validation scores, classification report and confusion matrix to choose our final models. We have exported the final models using joblib library of scikit.

8. Data science algorithms and Features used:

We have used numerous machine learning algorithms to understand the behaviour of each of them when applied on our dataset. Our analysis, results and performance metrics are as follows:

1. Logistic Regression

- The basic classification algorithm that is used to estimate a fixed discrete value for a given set of independent variables. It defines the probability with which an event can occur by fitting the data to a special function called the logit function.
- The output of logistic regression lies with 0 and 1 as it outputs probability. Since our's is a multi-class classification model, we have 5 classes which denote stars ranging from 1 to 5.
- We have tried fitting our training data into our logistic regression model and tested the performance using various performance metrics.

$$\alpha + \beta x1 + \gamma x2 = (model)$$

- Where x1, x2 represent the attributes of the dataset and denotes a constant.
- Here are our performance metrics when we applied logistic regression

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.51
(+/- 0.01):
cross validation mean Accuracy score 0.51 (+/- 0.04)
Train accuracy score: 0.516093229744728
Test accuracy score: 0.5133037694013304
Confusion Matrix:
[[  0   9   7   0   0]
 [  0  65  59  21   0]
 [  0  35 200 141   0]
 [  0   5 148 198   0]
 [  0   1   4   9   0]]
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        16
           2       0.57      0.45      0.50       145
           3       0.48      0.53      0.50       376
           4       0.54      0.56      0.55       351
           5       0.00      0.00      0.00        14

    accuracy                           0.51       902
   macro avg       0.32      0.31      0.31       902
weighted avg       0.50      0.51      0.50       902
```

Note: (+/- denotes a standard deviation value)

2. Decision Tree

- The supervised learning algorithm that is used for most of the classification problems splits the data into two or more sets. The criterion used to separate these is that the independent attributes should be partitioned into distinct groups as possible.
- We have implemented decision tree using the gini index and entropy techniques.
- Entropy is a way to measure impurity whereas the Gini index is a criterion to minimise misclassification.Here are our observations:

If Impurity measure is Entropy:

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.46
(+/- 0.01):
cross validation mean Accuracy score 0.45 (+/- 0.03)
Train accuracy score: 0.905937846836848
Test accuracy score: 0.44789356984478934
Confusion Matrix:
[[  4   7   3   2   0]
 [ 11  58  49  27   0]
 [  6  73 160 128   9]
 [  1  31 129 181   9]
 [  0   1   6   6   1]]
Classification Report:
              precision    recall  f1-score   support

           1       0.18      0.25      0.21        16
           2       0.34      0.40      0.37       145
           3       0.46      0.43      0.44       376
           4       0.53      0.52      0.52       351
           5       0.05      0.07      0.06        14

   micro avg       0.45      0.45      0.45       902
   macro avg       0.31      0.33      0.32       902
weighted avg       0.46      0.45      0.45       902
```

Note: (+/- denotes a standard deviation value

If impurity measure is Gini- Index:

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.47
(+/- 0.01):
cross validation mean Accuracy score 0.48 (+/- 0.03)
Train accuracy score: 0.9142619311875694
Test accuracy score: 0.48558758314855877
Confusion Matrix:
[[  4   9   2   1   0]
 [  8  75  42  18   2]
```

```
 [  4  68 188 115   1]
 [  2  27 146 168   8]
 [  1   0   4   6   3]]
Classification Report:
              precision    recall  f1-score   support

           1       0.21      0.25      0.23        16
           2       0.42      0.52      0.46       145
           3       0.49      0.50      0.50       376
           4       0.55      0.48      0.51       351
           5       0.21      0.21      0.21        14

   micro avg       0.49      0.49      0.49       902
   macro avg       0.38      0.39      0.38       902
weighted avg       0.49      0.49      0.49       902
```

Note: (+/- denotes a standard deviation value

3. K-Nearest Neighbour

- It is a special regression technique that uses k closest training samples in the feature space.. However, selecting the value of K as the number of initial seeds becomes very important as it affects the performance.We have trained the model with varying K values ranging from 3 to 11; to test and decide on an optimal value.

K value in KNN	**Stratified K-Fold accuracy score**	**K-Fold cross validation mean accuracy score**	**Train Accuracy Score**	**Test Accuracy Score**
K=3	0.44 (+/- 0.01)	0.44 (+/- 0.03)	0.69	0.46
K=4	0.46 (+/- 0.01)	0.45 (+/- 0.04)	0.65	0.47
K=5	0.47 (+/- 0.01)	0.46 (+/- 0.03)	0.63	0.49
K=6	0.47 (+/- 0.01)	0.46 (+/- 0.03)	0.61	0.51
K=7	0.48 (+/- 0.00)	0.47 (+/- 0.03)	0.60	0.51
K=8	0.48 (+/- 0.01)	0.47 (+/- 0.01)	0.61	0.50

K=9	0.49 (+/- 0.01)	0.47 (+/- 0.03)	0.60	0.51
K=10	0.49 (+/- 0.01)	0.48 (+/- 0.03)	0.59	0.51

Note: (+/- denotes a standard deviation value)

Performance metrics for k = 3 to 12

- **k=3**

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.42
(+/- 0.00):
cross validation mean Accuracy score 0.41 (+/- 0.04)
Train accuracy score: 0.6426193118756937
Test accuracy score: 0.4212860310421286
Confusion Matrix:
[[  2  12   1   1   0]
 [  2  72  52  19   0]
 [ 10  94 158 113   1]
 [  6  59 139 147   0]
 [  1   2   5   5   1]]
Classification Report:
              precision    recall  f1-score   support

           1       0.10      0.12      0.11        16
           2       0.30      0.50      0.38       145
           3       0.45      0.42      0.43       376
           4       0.52      0.42      0.46       351
           5       0.50      0.07      0.12        14

   micro avg       0.42      0.42      0.42       902
   macro avg       0.37      0.31      0.30       902
weighted avg       0.44      0.42      0.42       902
```

- **For k=4**

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.44
(+/- 0.01):
cross validation mean Accuracy score 0.44 (+/- 0.02)
Train accuracy score: 0.6018312985571587
Test accuracy score: 0.43237250554323725
```

```
Confusion Matrix:
[[  2  12   1   1   0]
 [  3  65  58  19   0]
 [  5  73 205  92   1]
 [  3  35 195 118   0]
 [  0   2   5   7   0]]
Classification Report:
              precision    recall  f1-score   support

           1       0.15      0.12      0.14        16
           2       0.35      0.45      0.39       145
           3       0.44      0.55      0.49       376
           4       0.50      0.34      0.40       351
           5       0.00      0.00      0.00        14

   micro avg       0.43      0.43      0.43       902
   macro avg       0.29      0.29      0.28       902
weighted avg       0.44      0.43      0.43       902
```

- **For k=5**

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.44
(+/- 0.01):
cross validation mean Accuracy score 0.45 (+/- 0.02)
Train accuracy score: 0.6012763596004439
Test accuracy score: 0.42904656319290463
Confusion Matrix:
[[  1  12   2   1   0]
 [  2  60  64  19   0]
 [  3  76 191 106   0]
 [  2  42 172 135   0]
 [  0   0   5   9   0]]
Classification Report:
              precision    recall  f1-score   support

           1       0.12      0.06      0.08        16
           2       0.32      0.41      0.36       145
           3       0.44      0.51      0.47       376
           4       0.50      0.38      0.43       351
           5       0.00      0.00      0.00        14
```

```
   micro avg       0.43      0.43      0.43       902
   macro avg       0.28      0.27      0.27       902
weighted avg       0.43      0.43      0.42       902
```

- **For k=6**

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.44
(+/- 0.01):
cross validation mean Accuracy score 0.44 (+/- 0.02)
Train accuracy score: 0.5771365149833518
Test accuracy score: 0.4079822616407982
Confusion Matrix:
[[  0  13   3   0   0]
 [  3  61  62  19   0]
 [  2  78 190 106   0]
 [  1  41 192 117   0]
 [  0   1   5   8   0]]
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        16
           2       0.31      0.42      0.36       145
           3       0.42      0.51      0.46       376
           4       0.47      0.33      0.39       351
           5       0.00      0.00      0.00        14

   micro avg       0.41      0.41      0.41       902
   macro avg       0.24      0.25      0.24       902
weighted avg       0.41      0.41      0.40       902
```

- **For k=7**

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.45
(+/- 0.01):
cross validation mean Accuracy score 0.46 (+/- 0.01)
Train accuracy score: 0.5751942286348501
```

```
Test accuracy score: 0.4490022172949002
Confusion Matrix:
[[  0  13   2   1   0]
 [  1  59  65  20   0]
 [  2  56 208 110   0]
 [  1  32 180 138   0]
 [  0   2   4   8   0]]
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        16
           2       0.36      0.41      0.38       145
           3       0.45      0.55      0.50       376
           4       0.50      0.39      0.44       351
           5       0.00      0.00      0.00        14

   micro avg       0.45      0.45      0.45       902
   macro avg       0.26      0.27      0.26       902
weighted avg       0.44      0.45      0.44       902
```

- **For k=8**

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.45
(+/- 0.01):
cross validation mean Accuracy score 0.46 (+/- 0.01)
Train accuracy score: 0.5668701442841287
Test accuracy score: 0.44789356984478934
Confusion Matrix:
[[  0  14   2   0   0]
 [  1  61  67  16   0]
 [  1  57 213 105   0]
 [  1  31 189 130   0]
 [  0   1   4   9   0]]
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        16
           2       0.37      0.42      0.39       145
           3       0.45      0.57      0.50       376
```

```
           4       0.50      0.37      0.43       351
           5       0.00      0.00      0.00        14

   micro avg       0.45      0.45      0.45       902
   macro avg       0.26      0.27      0.26       902
weighted avg       0.44      0.45      0.44       902
```

- **For k=9**

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.45
(+/- 0.01):
cross validation mean Accuracy score 0.46 (+/- 0.02)
Train accuracy score: 0.5613207547169812
Test accuracy score: 0.4467849223946785
Confusion Matrix:
[[  0  13   3   0   0]
 [  0  57  69  19   0]
 [  0  57 201 117   1]
 [  1  36 169 145   0]
 [  0   2   3   9   0]]
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        16
           2       0.35      0.39      0.37       145
           3       0.45      0.53      0.49       376
           4       0.50      0.41      0.45       351
           5       0.00      0.00      0.00        14

   micro avg       0.45      0.45      0.45       902
   macro avg       0.26      0.27      0.26       902
weighted avg       0.44      0.45      0.44       902
```

- **For k=10**

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.46
(+/- 0.01):
```

```
cross validation mean Accuracy score 0.46 (+/- 0.02)
Train accuracy score: 0.5543840177580466
Test accuracy score: 0.4379157427937916
Confusion Matrix:
[[  0  14   1   1   0]
 [  0  58  65  22   0]
 [  0  54 208 113   1]
 [  1  29 192 129   0]
 [  0   1   5   8   0]]
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        16
           2       0.37      0.40      0.39       145
           3       0.44      0.55      0.49       376
           4       0.47      0.37      0.41       351
           5       0.00      0.00      0.00        14

   micro avg       0.44      0.44      0.44       902
   macro avg       0.26      0.26      0.26       902
weighted avg       0.43      0.44      0.43       902
```

- **For k=11**

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.46
(+/- 0.01):
cross validation mean Accuracy score 0.46 (+/- 0.01)
Train accuracy score: 0.553274139844617
Test accuracy score: 0.4501108647450111
Confusion Matrix:
[[  0  13   3   0   0]
 [  1  59  63  22   0]
 [  0  51 201 123   1]
 [  1  25 179 146   0]
 [  0   1   6   7   0]]
Classification Report:
              precision    recall  f1-score   support
```

```
           1       0.00      0.00      0.00        16
           2       0.40      0.41      0.40       145
           3       0.44      0.53      0.49       376
           4       0.49      0.42      0.45       351
           5       0.00      0.00      0.00        14

   micro avg       0.45      0.45      0.45       902
   macro avg       0.27      0.27      0.27       902
weighted avg       0.44      0.45      0.44       902
```

4. Naive Bayes

- It is a classifier that is based on Bayes' Theorem with an assumption that all the attributes are independent of each other.
- It assumes that a particular feature present in the dataset is completely unrelated to any other feature. We have tried and tested our model using Gaussian Naive Bayes' and Bernoulli Naive Bayes. Here are our observations:

4.1 Guassian naive bayes

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.15
(+/- 0.00):
cross validation mean Accuracy score 0.15 (+/- 0.04)
Train accuracy score: 0.17619311875693675
Test accuracy score: 0.15631929046563192
Confusion Matrix:
[[ 16   0   0   0   0]
 [115  21   1   6   2]
 [186 100  26  34  30]
 [113  65  23  75  75]
 [  9   0   0   2   3]]
```

```
Classification Report:
              precision    recall  f1-score   support

           1       0.04      1.00      0.07        16
           2       0.11      0.14      0.13       145
           3       0.52      0.07      0.12       376
           4       0.64      0.21      0.32       351
           5       0.03      0.21      0.05        14

   micro avg       0.16      0.16      0.16       902
   macro avg       0.27      0.33      0.14       902
weighted avg       0.49      0.16      0.20       902
```

Note: (+/- denotes a standard deviation value)

4.2 Bernoulli naive bayes

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.51
(+/- 0.00):
cross validation mean Accuracy score 0.51 (+/- 0.03)
Train accuracy score: 0.5124861265260822
Test accuracy score: 0.532150776053215
Confusion Matrix:
[[  3  11   2   0   0]
 [ 13  90  30  12   0]
 [  6  85 184  99   2]
 [  0  39 104 203   5]
 [  0   3   3   8   0]]
Classification Report:
              precision    recall  f1-score   support

           1       0.14      0.19      0.16        16
           2       0.39      0.62      0.48       145
           3       0.57      0.49      0.53       376
           4       0.63      0.58      0.60       351
           5       0.00      0.00      0.00        14
```

```
   micro avg       0.53      0.53      0.53       902
   macro avg       0.35      0.38      0.35       902
weighted avg       0.55      0.53      0.53       902
```

5. Support Vector Machines

This is a classifier that is defined by a separating hyperplane.In simple words, it can be said that for a given training data, the classifier puts a hyperplane in such a way that it best separates the classes.

SVM Linear

```
K-fold with StratifiedKFold(n_splits=5,random_state=100, shuffle=True
) splits along with cross_validate accuracy scores 0.51 (+/- 0.01):
cross validation mean Accuracy score 0.50 (+/- 0.05)
Train accuracy score: 0.5177580466148723
Test accuracy score: 0.5144124168514412
Confusion Matrix:
[[  0   9   7   0   0]
 [  0  64  69  12   0]
 [  0  30 214 132   0]
 [  0  11 154 186   0]
 [  0   2   5   7   0]]
Classification Report:
               precision    recall  f1-score   support

           1       0.00      0.00      0.00        16
           2       0.55      0.44      0.49       145
           3       0.48      0.57      0.52       376
           4       0.55      0.53      0.54       351
           5       0.00      0.00      0.00        14

    accuracy                           0.51       902
   macro avg       0.32      0.31      0.31       902
weighted avg       0.50      0.51      0.51       902
```

Note: (+/- denotes a standard deviation value)

SVM poly

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.47
(+/- 0.00):

cross validation mean Accuracy score 0.46 (+/- 0.03)
Train accuracy score: 0.494173140954495
Test accuracy score: 0.48337028824833705
Confusion Matrix:
[[  0   6  10   0   0]
 [  0  30 111   4   0]
 [  0  12 301  63   0]
 [  0   1 245 105   0]
 [  0   1   9   4   0]]
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        16
           2       0.60      0.21      0.31       145
           3       0.45      0.80      0.57       376
           4       0.60      0.30      0.40       351
           5       0.00      0.00      0.00        14

    accuracy                           0.48       902
   macro avg       0.33      0.26      0.26       902
weighted avg       0.51      0.48      0.44       902
```

Note: (+/- denotes a standard deviation value)

SVM rbf

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.53
(+/- 0.01):

cross validation mean Accuracy score 0.53 (+/- 0.04)
Train accuracy score: 0.5563263041065483
Test accuracy score: 0.5399113082039911
Confusion Matrix:
[[  0   8   8   0   0]
 [  0  64  66  15   0]
 [  0  28 223 125   0]
```

```
 [  0   9 142 200   0]
 [  0   1   2  11   0]]
 Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        16
           2       0.58      0.44      0.50       145
           3       0.51      0.59      0.55       376
           4       0.57      0.57      0.57       351
           5       0.00      0.00      0.00        14

    accuracy                           0.54       902
   macro avg       0.33      0.32      0.32       902
weighted avg       0.53      0.54      0.53       902
```

Note: (+/- denotes a standard deviation value)

6. Random Forest

- This classifier generates a number of sequential decision trees on various sub-samples and uses mean value to improve the predictive accuracy. This helps us control over-fitting. Here are our performance results:

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.53
(+/- 0.01):
cross validation mean Accuracy score 0.52 (+/- 0.06)
Train accuracy score: 0.9062153163152054
Test accuracy score: 0.5343680709534369
Confusion Matrix:
[[  2   9   3   2   0]
[  1  70  54  20   0]
[  3  47 218 108   0]
[  0  16 143 192   0]
[  0   1   3  10   0]]
Classification Report:
              precision    recall  f1-score   support
           1       0.33      0.12      0.18        16
           2       0.49      0.48      0.49       145
           3       0.52      0.58      0.55       376
```

```
          4       0.58      0.55      0.56       351
          5       0.00      0.00      0.00        14
  micro avg       0.53      0.53      0.53       902
  macro avg       0.38      0.35      0.36       902
weighted avg       0.53      0.53      0.53       902
```

Note: (+/- denotes a standard deviation value)

7. Gradient Boost Classifier

- Decision tree classifiers use Gradient boosting technique. Boosting refers to transforming a weak classifier to a strong classifier.It combines multiple weak models to generate a strong predictive one.

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.57
(+/- 0.01):
cross validation mean Accuracy score 0.56 (+/- 0.03)
Train accuracy score: 0.6531631520532741
Test accuracy score: 0.5920177383592018
Confusion Matrix:
[[  1  12   2   1   0]
 [  1  76  55  13   0]
 [  2  32 235 105   2]
 [  0   8 120 221   2]
 [  0   0   7   6   1]]
Classification Report:
              precision    recall  f1-score   support

           1       0.25      0.06      0.10        16
           2       0.59      0.52      0.56       145
           3       0.56      0.62      0.59       376
           4       0.64      0.63      0.63       351
           5       0.20      0.07      0.11        14

   micro avg       0.59      0.59      0.59       902
   macro avg       0.45      0.38      0.40       902
weighted avg       0.59      0.59      0.59       902
```

Note: (+/- denotes a standard deviation value)

8. XGBoost

- XGBoost builds a sequential decision tree model using sequential ensemble technique.
- This technique assigns weights to each of the dataset item and passes it to the model. After regression , a weak classifier model is generated. The average of such weak classifier models results gives the final result.
- This technique is iterative and improved our performance scores along with computation speed.

```
K-fold     with     StratifiedKFold(n_splits=5,     random_state=100,
shuffle=True) splits along with cross_validate accuracy scores 0.57
(+/- 0.01):
cross validation mean Accuracy score 0.56 (+/- 0.04)
Train accuracy score: 0.6226415094339622
Test accuracy score: 0.5842572062084257
Confusion Matrix:
[[  0  12   3   1   0]
 [  0  74  60  11   0]
 [  0  34 225 117   0]
 [  0   7 115 228   1]
 [  0   0   6   8   0]]
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        16
           2       0.58      0.51      0.54       145
           3       0.55      0.60      0.57       376
           4       0.62      0.65      0.64       351
           5       0.00      0.00      0.00        14

   micro avg       0.58      0.58      0.58       902
   macro avg       0.35      0.35      0.35       902
weighted avg       0.57      0.58      0.57       902
```

Note: (+/- denotes a standard deviation value)

9. Big Data Tools usage

We have used the following big data tools to visualize and transform the raw and unstructured data:

1. Apache Spark
    - A popular data analytics engine that is used in the field of Big Data and machine learning.
    - It has many easy to use API's that can be used to manipulate large datasets and also includes a collection of more than 100 operators for transforming data.
    - It also contains many high-level libraries that support SQL queries.
    - Apache spark is said to be 100x more efficient and fast than hadoop when it comes to large scale data processing.
    - Following is the command to install pyspark:

        brew install apache-spark

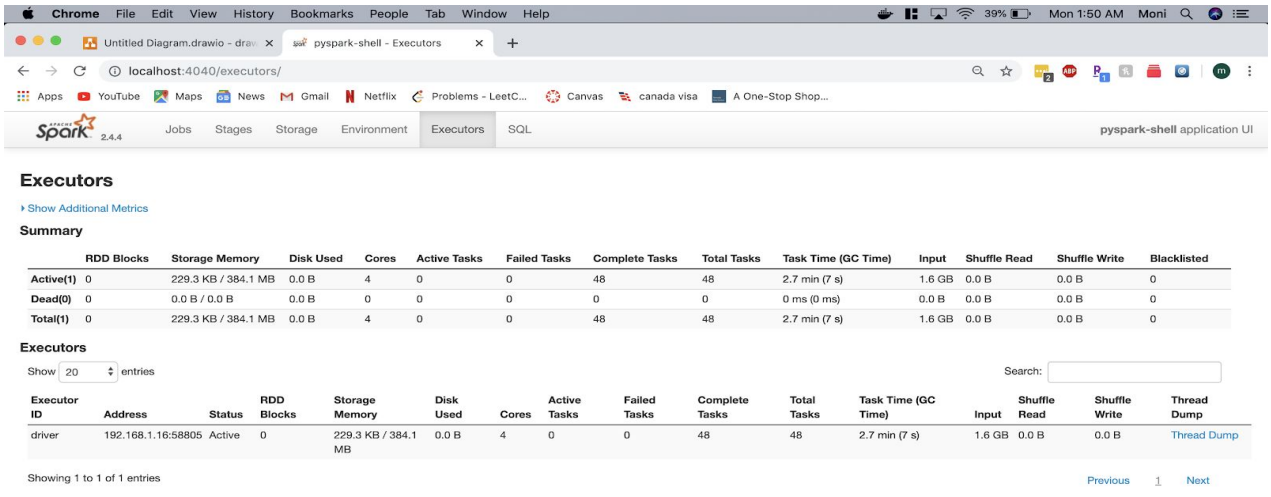
```
/Users/moni/projects/yelp/venv/bin/python /Users/moni/projects/yelp/pre_processing/pyspark_data_load.py
19/12/02 02:04:55 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.4
      /_/

Using Python version 3.5.7 (default, Nov 30 2019 17:31:22)
SparkSession available as 'spark'.
```

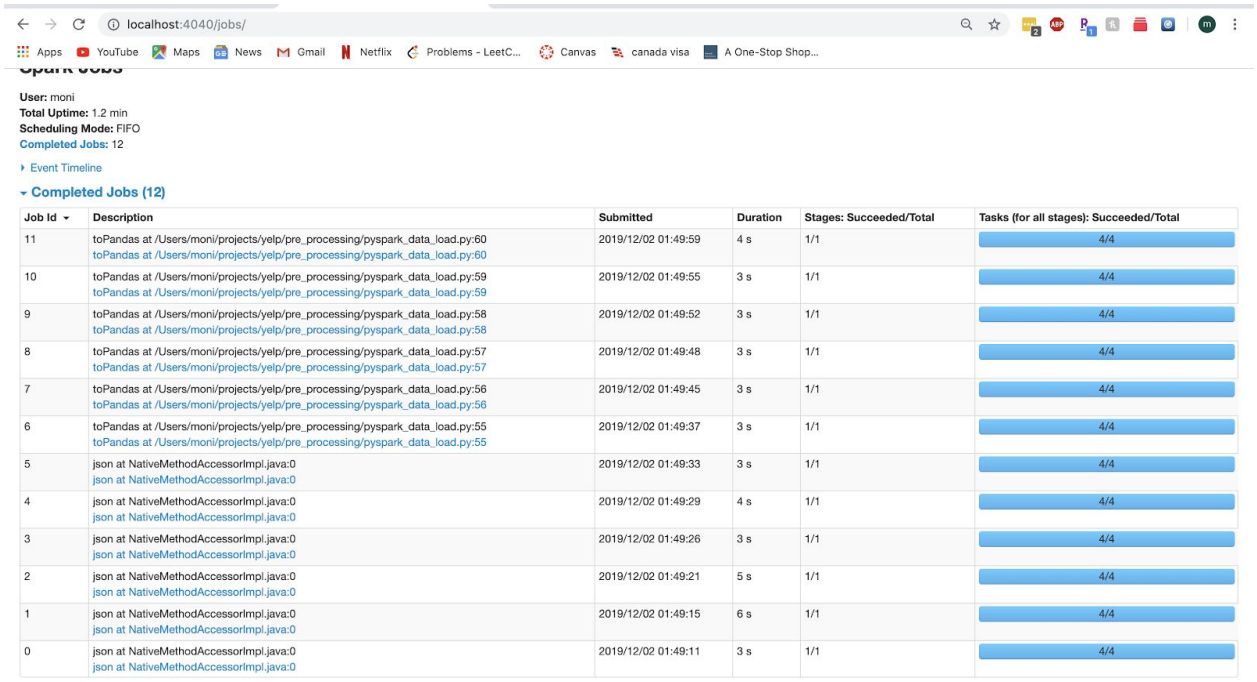
- Below are the screenshots of the attributes from the Yelp dataset getting parsed on the apache spark.

```
root
 |-- address: string (nullable = true)
 |-- attributes: struct (nullable = true)
 |    |-- AcceptsInsurance: string (nullable = true)
 |    |-- AgesAllowed: string (nullable = true)
 |    |-- Alcohol: string (nullable = true)
 |    |-- Ambience: string (nullable = true)
 |    |-- BYOB: string (nullable = true)
 |    |-- BYOBCorkage: string (nullable = true)
 |    |-- BestNights: string (nullable = true)
 |    |-- BikeParking: string (nullable = true)
 |    |-- BusinessAcceptsBitcoin: string (nullable = true)
 |    |-- BusinessAcceptsCreditCards: string (nullable = true)
 |    |-- BusinessParking: string (nullable = true)
 |    |-- ByAppointmentOnly: string (nullable = true)
 |    |-- Caters: string (nullable = true)
 |    |-- CoatCheck: string (nullable = true)
 |    |-- Corkage: string (nullable = true)
 |    |-- DietaryRestrictions: string (nullable = true)
 |    |-- DogsAllowed: string (nullable = true)
 |    |-- DriveThru: string (nullable = true)
 |    |-- GoodForDancing: string (nullable = true)
 |    |-- GoodForKids: string (nullable = true)
 |    |-- GoodForMeal: string (nullable = true)
 |    |-- HairSpecializesIn: string (nullable = true)
 |    |-- HappyHour: string (nullable = true)
 |    |-- HasTV: string (nullable = true)
 |    |-- Music: string (nullable = true)
 |    |-- NoiseLevel: string (nullable = true)
 |    |-- Open24Hours: string (nullable = true)
 |    |-- OutdoorSeating: string (nullable = true)
 |    |-- RestaurantsAttire: string (nullable = true)
 |    |-- RestaurantsCounterService: string (nullable = true)
 |    |-- RestaurantsDelivery: string (nullable = true)
```

```
|    |-- RestaurantsGoodForGroups: string (nullable = true)
|    |-- RestaurantsPriceRange2: string (nullable = true)
|    |-- RestaurantsReservations: string (nullable = true)
|    |-- RestaurantsTableService: string (nullable = true)
|    |-- RestaurantsTakeOut: string (nullable = true)
|    |-- Smoking: string (nullable = true)
|    |-- WheelchairAccessible: string (nullable = true)
|    |-- WiFi: string (nullable = true)
|-- business_id: string (nullable = true)
|-- categories: string (nullable = true)
|-- city: string (nullable = true)
|-- hours: struct (nullable = true)
|    |-- Friday: string (nullable = true)
|    |-- Monday: string (nullable = true)
|    |-- Saturday: string (nullable = true)
|    |-- Sunday: string (nullable = true)
|    |-- Thursday: string (nullable = true)
|    |-- Tuesday: string (nullable = true)
|    |-- Wednesday: string (nullable = true)
|-- is_open: long (nullable = true)
|-- latitude: double (nullable = true)
|-- longitude: double (nullable = true)
|-- name: string (nullable = true)
|-- postal_code: string (nullable = true)
|-- review_count: long (nullable = true)
|-- stars: double (nullable = true)
|-- state: string (nullable = true)
```

- Running apache stack without any workers and loading all the tasks:





localhost:4040/jobs/

Apps YouTube Maps News Gmail Netflix Problems - LeetC... Canvas canada visa A One-Stop Shop...

User: moni
Total Uptime: 1.2 min
Scheduling Mode: FIFO
Completed Jobs: 12

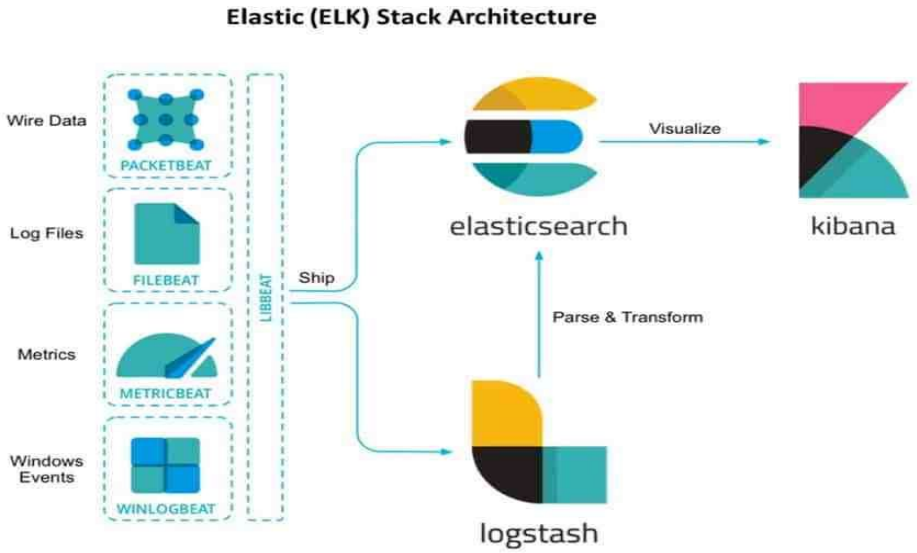
Event Timeline

Completed Jobs (12)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
11	toPandas at /Users/moni/projects/yelp/pre_processing/pyspark_data_load.py:60 toPandas at /Users/moni/projects/yelp/pre_processing/pyspark_data_load.py:60	2019/12/02 01:49:59	4 s	1/1	4/4
10	toPandas at /Users/moni/projects/yelp/pre_processing/pyspark_data_load.py:59 toPandas at /Users/moni/projects/yelp/pre_processing/pyspark_data_load.py:59	2019/12/02 01:49:55	3 s	1/1	4/4
9	toPandas at /Users/moni/projects/yelp/pre_processing/pyspark_data_load.py:58 toPandas at /Users/moni/projects/yelp/pre_processing/pyspark_data_load.py:58	2019/12/02 01:49:52	3 s	1/1	4/4
8	toPandas at /Users/moni/projects/yelp/pre_processing/pyspark_data_load.py:57 toPandas at /Users/moni/projects/yelp/pre_processing/pyspark_data_load.py:57	2019/12/02 01:49:48	3 s	1/1	4/4
7	toPandas at /Users/moni/projects/yelp/pre_processing/pyspark_data_load.py:56 toPandas at /Users/moni/projects/yelp/pre_processing/pyspark_data_load.py:56	2019/12/02 01:49:45	3 s	1/1	4/4
6	toPandas at /Users/moni/projects/yelp/pre_processing/pyspark_data_load.py:55 toPandas at /Users/moni/projects/yelp/pre_processing/pyspark_data_load.py:55	2019/12/02 01:49:37	3 s	1/1	4/4
5	json at NativeMethodAccessorImpl.java:0 json at NativeMethodAccessorImpl.java:0	2019/12/02 01:49:33	3 s	1/1	4/4
4	json at NativeMethodAccessorImpl.java:0 json at NativeMethodAccessorImpl.java:0	2019/12/02 01:49:29	4 s	1/1	4/4
3	json at NativeMethodAccessorImpl.java:0 json at NativeMethodAccessorImpl.java:0	2019/12/02 01:49:26	3 s	1/1	4/4
2	json at NativeMethodAccessorImpl.java:0 json at NativeMethodAccessorImpl.java:0	2019/12/02 01:49:21	5 s	1/1	4/4
1	json at NativeMethodAccessorImpl.java:0 json at NativeMethodAccessorImpl.java:0	2019/12/02 01:49:15	6 s	1/1	4/4
0	json at NativeMethodAccessorImpl.java:0 json at NativeMethodAccessorImpl.java:0	2019/12/02 01:49:11	3 s	1/1	4/4

2. ELK stack

- It is a combination of three open source objects: Elasticsearch, Logstash and Kibana.
- Elasticsearch is a NoSQL database whereas Logstash is a pipeline tool that takes in inputs from various sources, performs various transformations.Kibana is the GUI; a visualization layer that works on the top of elasticsearch which helps in searching of logs from elasticsearch.



Elasticsearch

```
~  elasticsearch                                                                                                    1 ↵
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a future release.
[2019-12-02T02:20:05,982][INFO ][o.e.e.NodeEnvironment    ] [as-MacBook-Pro.local] using [1] data paths, mounts [[/ (/dev/disk1s1)]], net usable_space [63.9gb], net total_space [
233.4gb], types [apfs]
[2019-12-02T02:20:06,001][INFO ][o.e.e.NodeEnvironment    ] [as-MacBook-Pro.local] heap size [990.7mb], compressed ordinary object pointers [true]
[2019-12-02T02:20:06,067][INFO ][o.e.n.Node               ] [as-MacBook-Pro.local] node name [as-MacBook-Pro.local], node ID [rG5Y73b5TCOGYWGLBxPGEw], cluster name [elasticsearch
_moni]
[2019-12-02T02:20:06,067][INFO ][o.e.n.Node               ] [as-MacBook-Pro.local] version[7.4.2], pid[50043], build[default/tar/2f90bbf7b93631e52bafb59b3b049cb44ec25e96/2019-10-
28T20:40:44.881551Z], OS[Mac OS X/10.14.5/x86_64], JVM[AdoptOpenJDK/OpenJDK 64-Bit Server VM/13.0.1/13.0.1+9]
[2019-12-02T02:20:06,068][INFO ][o.e.n.Node               ] [as-MacBook-Pro.local] JVM home [/usr/local/Cellar/elasticsearch-full/7.4.2/libexec/jdk/Contents/Home]
[2019-12-02T02:20:06,068][INFO ][o.e.n.Node               ] [as-MacBook-Pro.local] JVM arguments [-Xms1g, -Xmx1g, -XX:+UseConcMarkSweepGC, -XX:CMSInitiatingOccupancyFraction=75,
-XX:+UseCMSInitiatingOccupancyOnly, -Des.networkaddress.cache.ttl=60, -Des.networkaddress.cache.negative.ttl=10, -XX:+AlwaysPreTouch, -Xss1m, -Djava.awt.headless=true, -Dfile.enc
oding=UTF-8, -Djna.nosys=true, -XX:-OmitStackTraceInFastThrow, -Dio.netty.noUnsafe=true, -Dio.netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -Dio.ne
tty.allocator.numDirectArenas=0, -Dlog4j.shutdownHookEnabled=false, -Dlog4j2.disable.jmx=true, -Djava.io.tmpdir=/var/folders/3r/rqkdfxtx2jd8nsvf3ncllnnw0000gp/T/elasticsearch-145
43985878376737720, -XX:+HeapDumpOnOutOfMemoryError, -XX:HeapDumpPath=data, -XX:ErrorFile=logs/hs_err_pid%p.log, -Xlog:gc*,gc+age=trace,safepoint:file=/usr/local/var/log/elasticse
arch/gc.log:utctime,pid,tags:filecount=32,filesize=64m, -Djava.locale.providers=COMPAT, -Dio.netty.allocator.type=unpooled, -XX:MaxDirectMemorySize=536870912, -Des.path.home=/usr
/local/Cellar/elasticsearch-full/7.4.2/libexec, -Des.path.conf=/usr/local/etc/elasticsearch, -Des.distribution.flavor=default, -Des.distribution.type=tar, -Des.bundled_jdk=true]
[2019-12-02T02:20:08,058][INFO ][o.e.p.PluginsService     ] [as-MacBook-Pro.local] loaded module [aggs-matrix-stats]
[2019-12-02T02:20:08,059][INFO ][o.e.p.PluginsService     ] [as-MacBook-Pro.local] loaded module [analysis-common]
```

```
[2019-12-02T02:20:08,072][INFO ][o.e.p.PluginsService     ] [as-MacBook-Pro.local] no plugins loaded
[2019-12-02T02:20:11,647][INFO ][o.e.x.s.a.s.FileRolesStore] [as-MacBook-Pro.local] parsed [0] roles from file [/usr/local/etc/elasticsearch/roles.yml]
[2019-12-02T02:20:12,512][INFO ][o.e.x.m.p.l.CppLogMessageHandler] [as-MacBook-Pro.local] [controller/50063] [Main.cc@110] controller (64 bit): Version 7.4.2 (Build 473f61b8a5238
b) Copyright (c) 2019 Elasticsearch BV
[2019-12-02T02:20:13,269][DEBUG][o.e.a.ActionModule       ] [as-MacBook-Pro.local] Using REST wrapper from plugin org.elasticsearch.xpack.security.Security
[2019-12-02T02:20:14,002][INFO ][o.e.d.DiscoveryModule    ] [as-MacBook-Pro.local] using discovery type [zen] and seed hosts providers [settings]
[2019-12-02T02:20:14,919][INFO ][o.e.n.Node               ] [as-MacBook-Pro.local] initialized
[2019-12-02T02:20:14,920][INFO ][o.e.n.Node               ] [as-MacBook-Pro.local] starting ...
[2019-12-02T02:20:15,164][INFO ][o.e.t.TransportService   ] [as-MacBook-Pro.local] publish_address {127.0.0.1:9300}, bound_addresses {[::1]:9300}, {127.0.0.1:9300}
[2019-12-02T02:20:15,200][WARN ][o.e.b.BootstrapChecks    ] [as-MacBook-Pro.local] the default discovery settings are unsuitable for production use; at least one of [discovery.se
ed_hosts, discovery.seed_providers, cluster.initial_master_nodes] must be configured
[2019-12-02T02:20:15,205][INFO ][o.e.c.c.Coordinator      ] [as-MacBook-Pro.local] cluster UUID [7hLKyWT_Rg-cv3mfaNqPFg]
[2019-12-02T02:20:15,264][INFO ][o.e.c.c.ClusterBootstrapService] [as-MacBook-Pro.local] no discovery configuration found, will perform best-effort cluster bootstrapping after [3
s] unless existing master is discovered
[2019-12-02T02:20:15,539][INFO ][o.e.c.s.MasterService    ] [as-MacBook-Pro.local] elected-as-master ([1] nodes joined)[{as-MacBook-Pro.local}{rG5Y73b5TCOGYWGLBxPGEw}{-TxrXS9CSFO
HyYTryBSP-Q}{127.0.0.1}{127.0.0.1:9300}{dilm}{ml.machine_memory=8589934592, xpack.installed=true, ml.max_open_jobs=20} elect leader, _BECOME_MASTER_TASK_, _FINISH_ELECTION_], ter
m: 5, version: 113, reason: master node changed {previous [], current [{as-MacBook-Pro.local}{rG5Y73b5TCOGYWGLBxPGEw}{-TxrXS9CSFOHyYTryBSP-Q}{127.0.0.1}{127.0.0.1:9300}{dilm}{ml.
machine_memory=8589934592, xpack.installed=true, ml.max_open_jobs=20}]}
[2019-12-02T02:20:15,704][INFO ][o.e.c.s.ClusterApplierService] [as-MacBook-Pro.local] master node changed {previous [], current [{as-MacBook-Pro.local}{rG5Y73b5TCOGYWGLBxPGEw}{-
TxrXS9CSFOHyYTryBSP-Q}{127.0.0.1}{127.0.0.1:9300}{dilm}{ml.machine_memory=8589934592, xpack.installed=true, ml.max_open_jobs=20}]}, term: 5, version: 113, reason: Publication{ter
m=5, version=113}
[2019-12-02T02:20:15,980][INFO ][o.e.h.AbstractHttpServerTransport] [as-MacBook-Pro.local] publish_address {127.0.0.1:9200}, bound_addresses {[::1]:9200}, {127.0.0.1:9200}
[2019-12-02T02:20:15,981][INFO ][o.e.n.Node               ] [as-MacBook-Pro.local] started
[2019-12-02T02:20:16,196][INFO ][o.e.l.LicenseService     ] [as-MacBook-Pro.local] license [23cc32c3-aee7-47d8-9d92-f30905a3a1a7] mode [basic] - valid
[2019-12-02T02:20:16,197][INFO ][o.e.x.s.s.SecurityStatusChangeListener] [as-MacBook-Pro.local] Active license is now [BASIC]; Security is disabled
[2019-12-02T02:20:16,208][INFO ][o.e.g.GatewayService     ] [as-MacBook-Pro.local] recovered [7] indices into cluster_state
[2019-12-02T02:20:17,048][INFO ][o.e.c.r.a.AllocationService] [as-MacBook-Pro.local] Cluster health status changed from [RED] to [YELLOW] (reason: [shards started [[yelp-business
][0]]]).
```

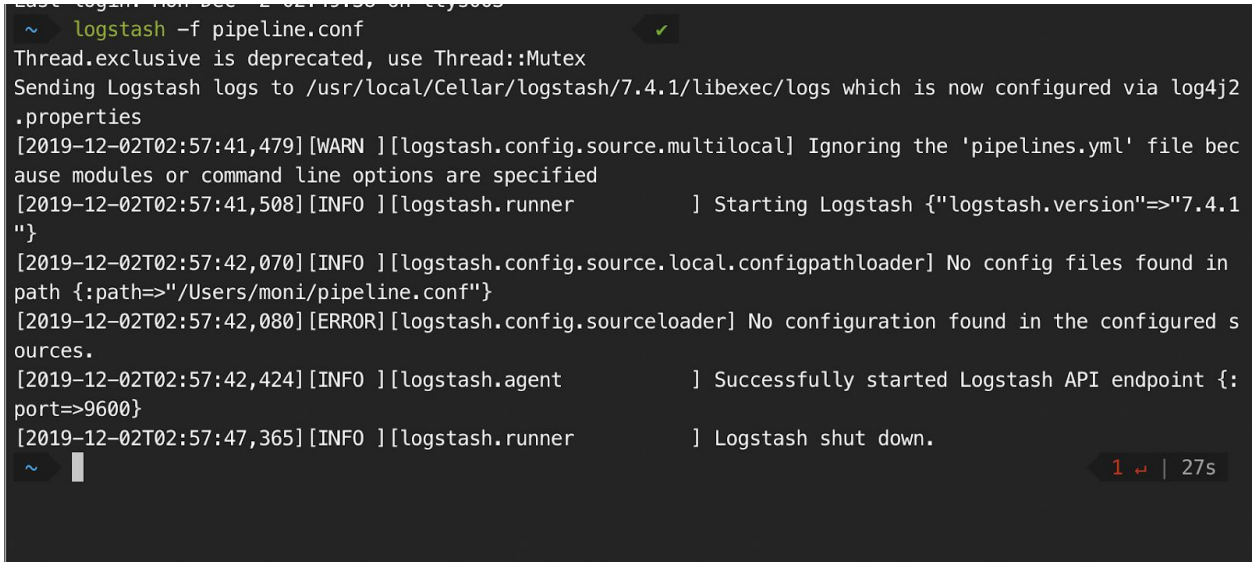
- On the local server

localhost:9200

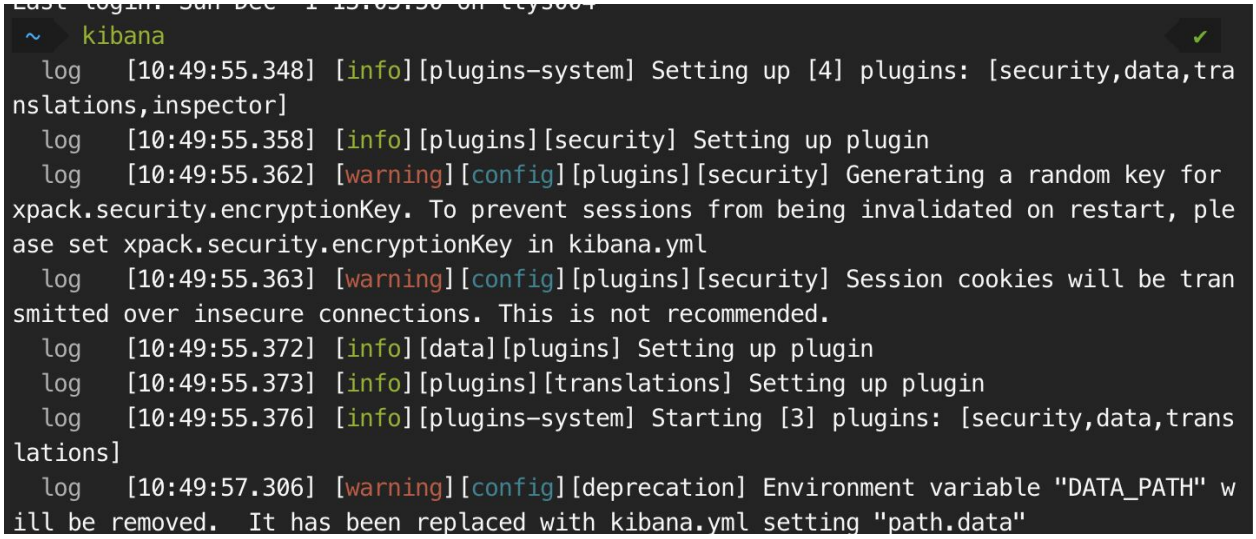
Apps YouTube Maps News Gmail Netflix Problems - LeetC...


```
{
  "name" : "as-MacBook-Pro.local",
  "cluster_name" : "elasticsearch_moni",
  "cluster_uuid" : "7hLKyWT_Rg-cv3mfaNqPFg",
  "version" : {
    "number" : "7.4.2",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "2f90bbf7b93631e52bafb59b3b049cb44ec25e96",
    "build_date" : "2019-10-28T20:40:44.881551Z",
    "build_snapshot" : false,
    "lucene_version" : "8.2.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

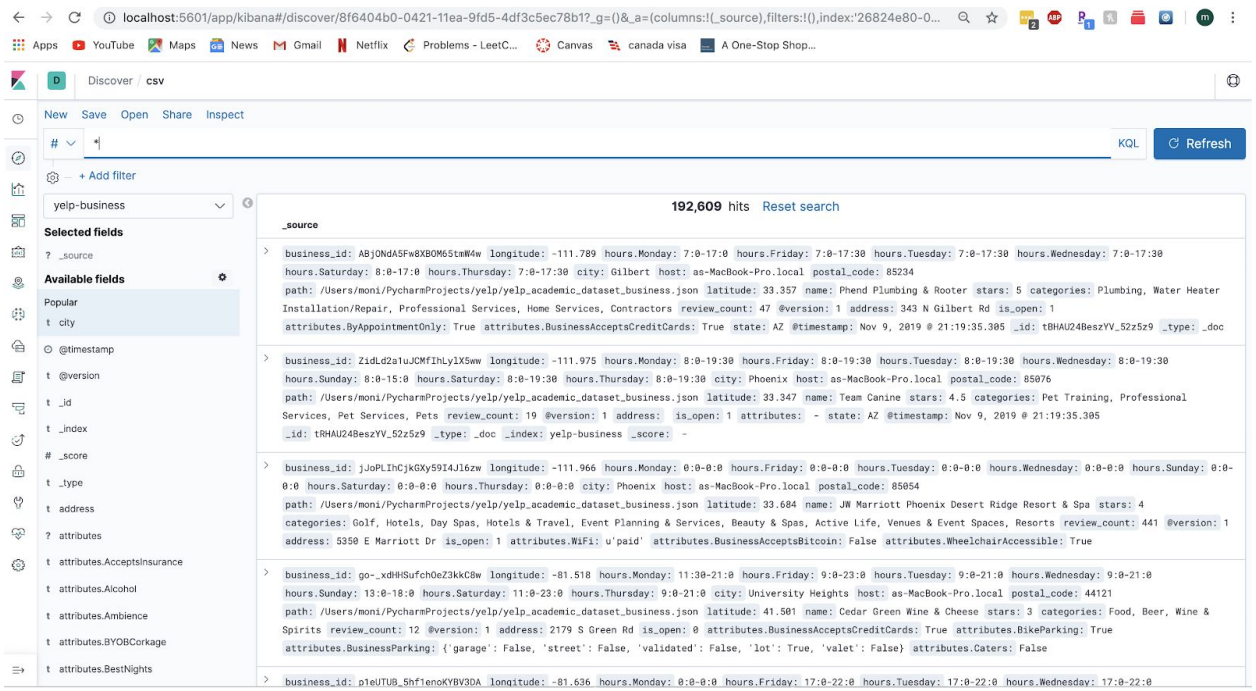
- Starting logstash

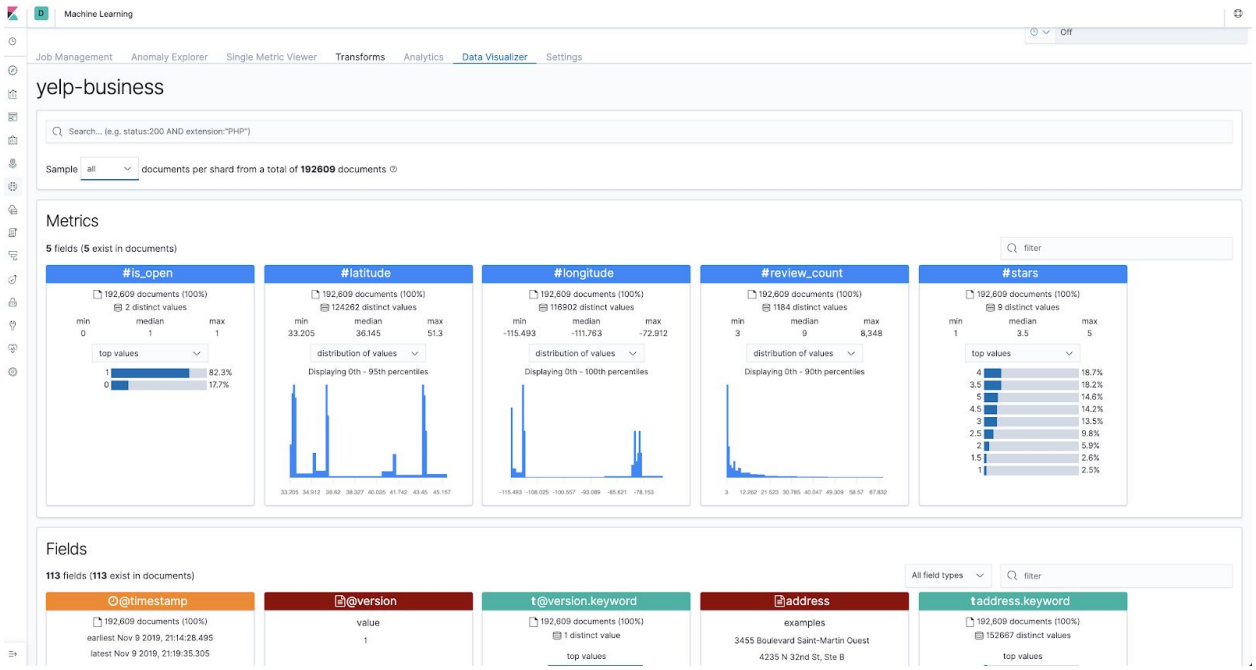


- Starting Kibana



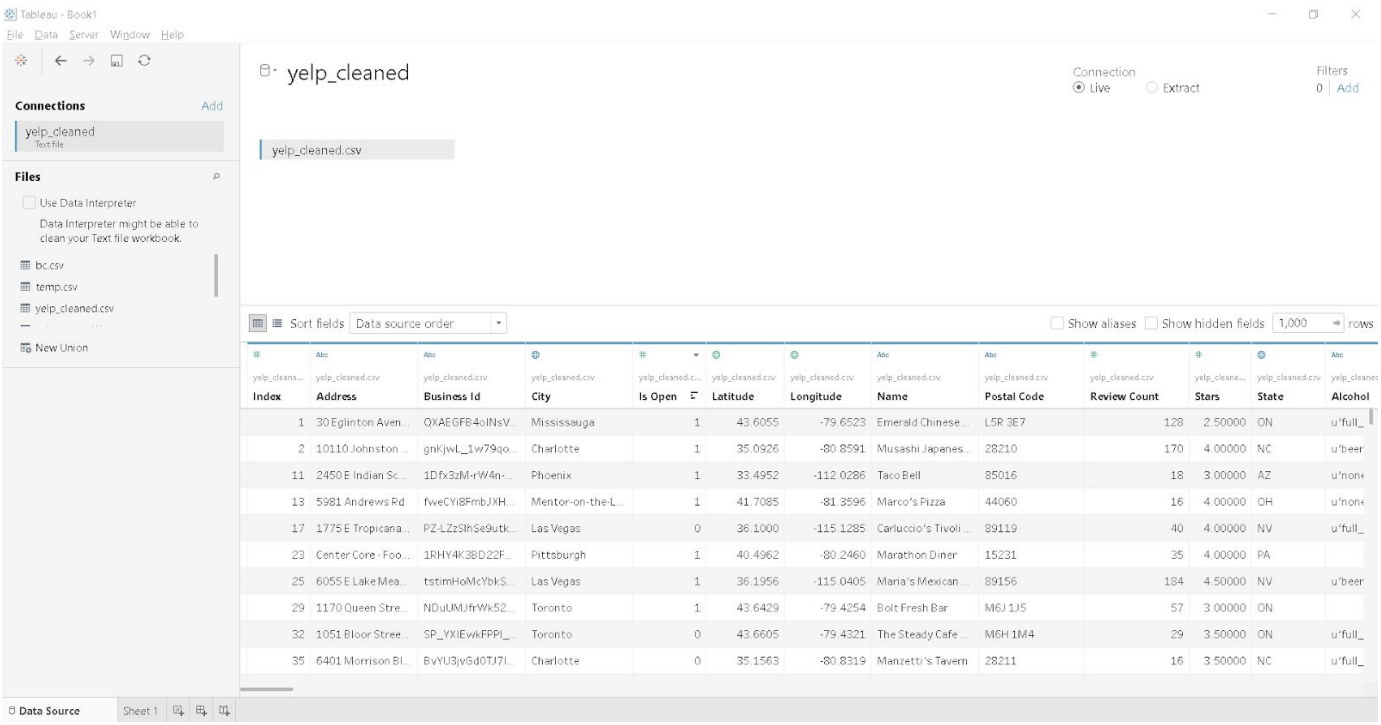
Kibana


localhost:5601/app/kibana#/discover/8f6404b0-0421-11ea-9fd5-4df3c5ec78b1?_g=()&_a=(columns:!(_source),filters:!(),index:'26824e80-0...
Apps YouTube Maps News Gmail Netflix Problems - LeetC... Canvas canada visa A One-Stop Shop...
Discover / csv
New Save Open Share Inspect
KQL Refresh
+ Add filter
yelp-business
192,609 hits Reset search
_source
Selected fields
_source
Available fields
Popular
city
@timestamp
@version
_id
_index
_score
_type
address
attributes
attributes.AcceptsInsurance
attributes.Alcohol
attributes.Ambience
attributes.BYOBCorkage
attributes.BestNights


Machine Learning
Job Management Anomaly Explorer Single Metric Viewer Transforms Analytics Data Visualizer Settings
yelp-business
Search... (e.g. status:200 AND extension:"PHP")
Sample all documents per shard from a total of 192609 documents
Metrics
5 fields (5 exist in documents)
#is_open
#latitude
#longitude
#review_count
#stars
Fields
113 fields (113 exist in documents)
@timestamp
@version
t @version.keyword
address
t address.keyword

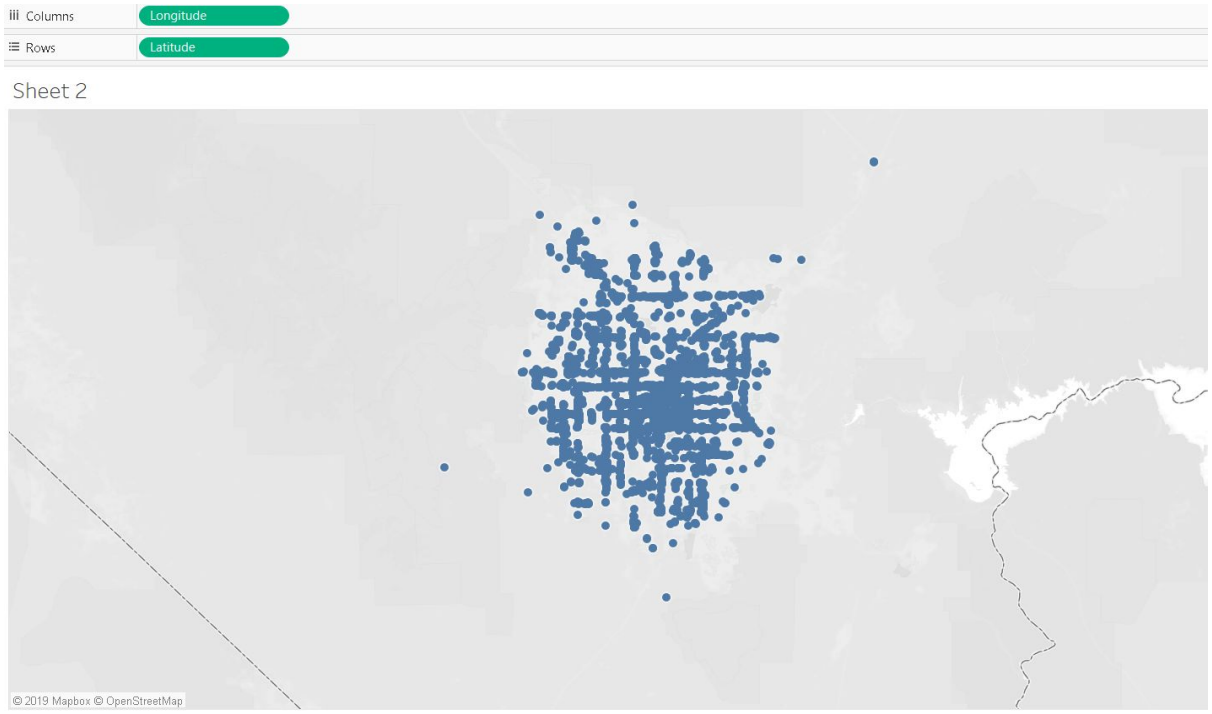
Tableau

- Tableau is yet another visualisation and analytics tool that is used to draw various insights about various elements of our dataset.
- We have analysed out Yelp dataset consisting of various features like postal_code, Is_open, Business_Id, etc.
- The worksheet of a Tableau desktop version looks like as below:



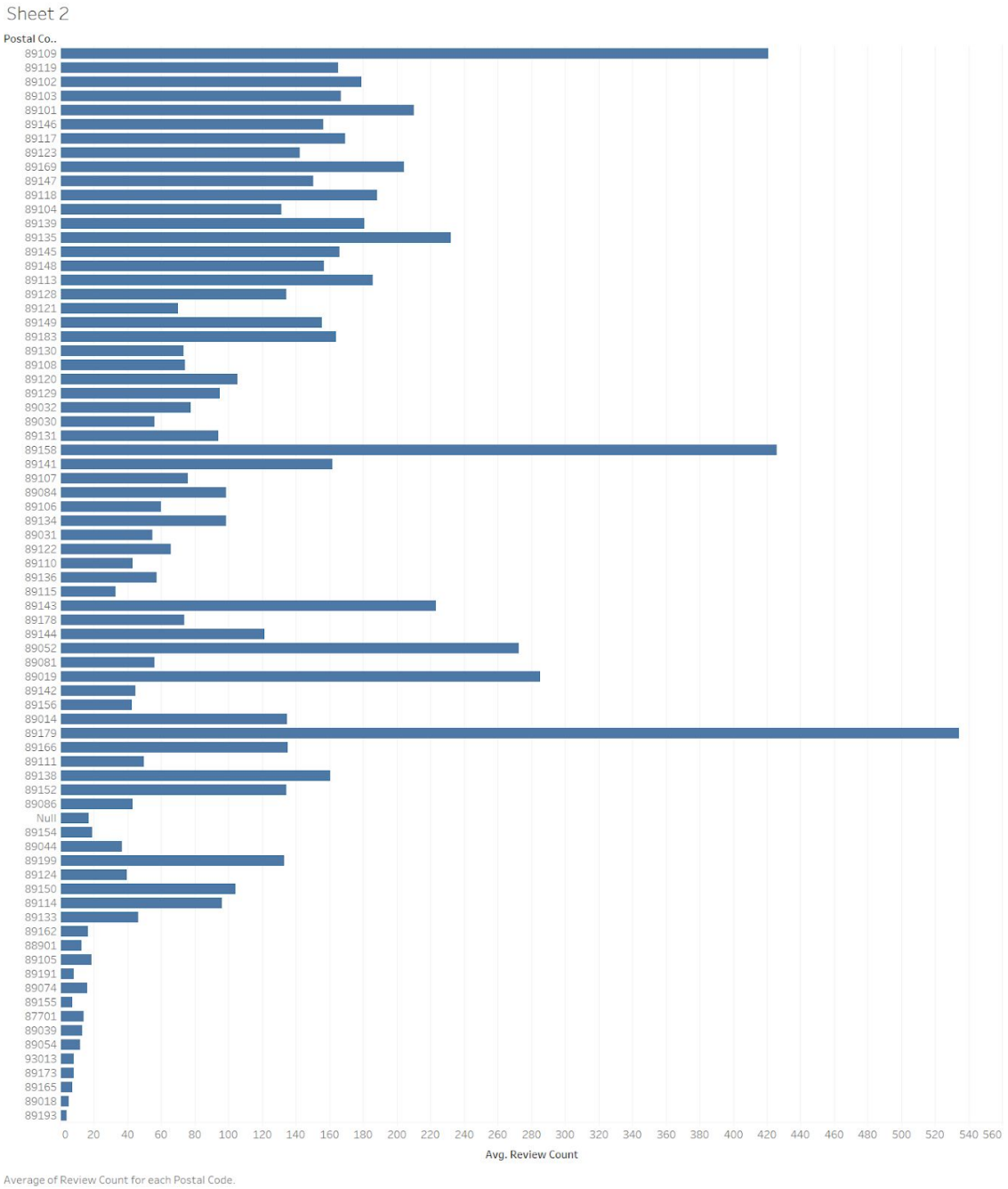
Visualization 1

We have visualised the area-wise distribution of the data using Maps. This has given us a clear insight of which are in Las Vegas has been most populated with restaurants and which one is done least. Each restaurant is represented as a dot and is described by the latitude and longitude points.


Columns
Longitude
Rows
Latitude
Sheet 2
© 2019 Mapbox © OpenStreetMap

Visualisation 2

This is a bar-graph representation plotted between the Average review count for each of the postal codes.



10 . RESTful Server Side Design

- We have developed a RESTful backend API service to accept json request that contains the features user needs to have to set up a restaurant business.
- We have used Google Python Client library and OAuth libraries to setup secure connection from our backend to the Google AI platform where our model was deployed.
- We have created server-to-server security credentials on Google cloud and used those credentials in order to enable client send prediction requests to model.
- Backend exposes following APIs to the client and listens on 5000 port.

```
POST /predict HTTP/1.1
Host: localhost:5000
Content-Type: application/json
Accept: */*
Cache-Control: no-cache
Host: localhost:5000
Accept-Encoding: gzip, deflate
Content-Length: 73
Connection: keep-alive
cache-control: no-cache
{
      "Burgers":1,
      "parking":1,
      "RestaurantsReservations":1
}
```

- Backend transforms the above client request into a json format that GoogleAI can understand with all the features needed for model prediction.

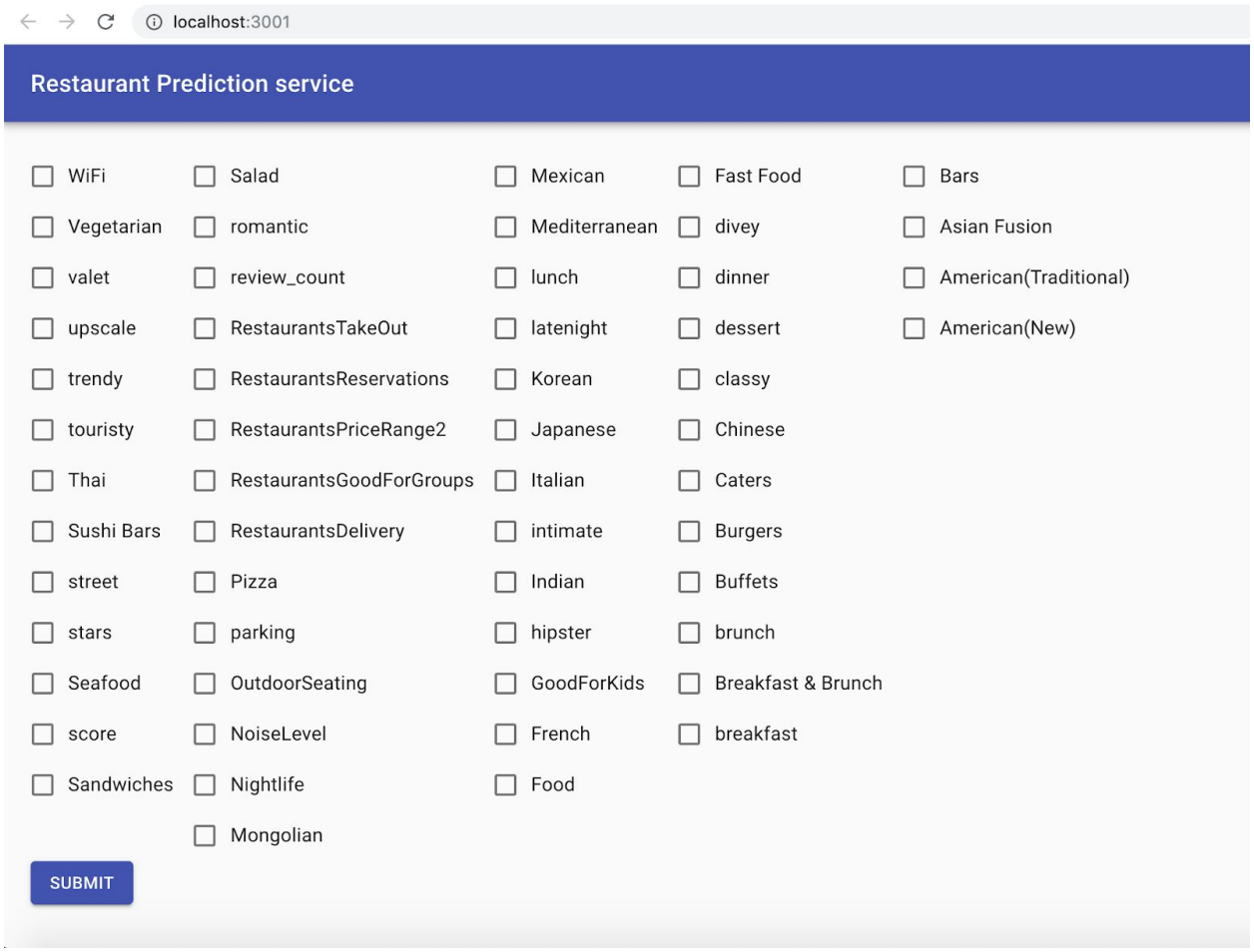
```
POST
https://ml.googleapis.com/v1/projects/yelp-final-260208/models/rfc
{
 "instances":[[1.0, 1.0, 0, 1, 1.0, 2.0, 0, 0.0, 1.0, 1.0, 0.0, 1.0,
0.0, 1.0, 0.0, 0.0, 1,0, 0, 0, 0, 0, 0, 0, 0.0, 0.0, 0.0, 0.0, 1.0,
0.0, 1.0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0]]
}
```

The response from Google AI is forwarded to client to display it to users.

```
{
    "predictions": [
        3
    ]
}
```

11. Client Side Design

- Our Restaurant Prediction Service asks the user to select a set of amenities wanted by the restaurant owner who wants to set up a restaurant in Las Vegas.
- For a given set of features say “Wi-Fi”, serves “Vegetarian “, “Mexican”, “breakfast”, the owner will be given a score as an output that will indicate how likely the restaurant is going to do well.
- We have created this frontend client using ReactJS. This application runs on port 3000 and proxies the requests to backend which is running on port 5000



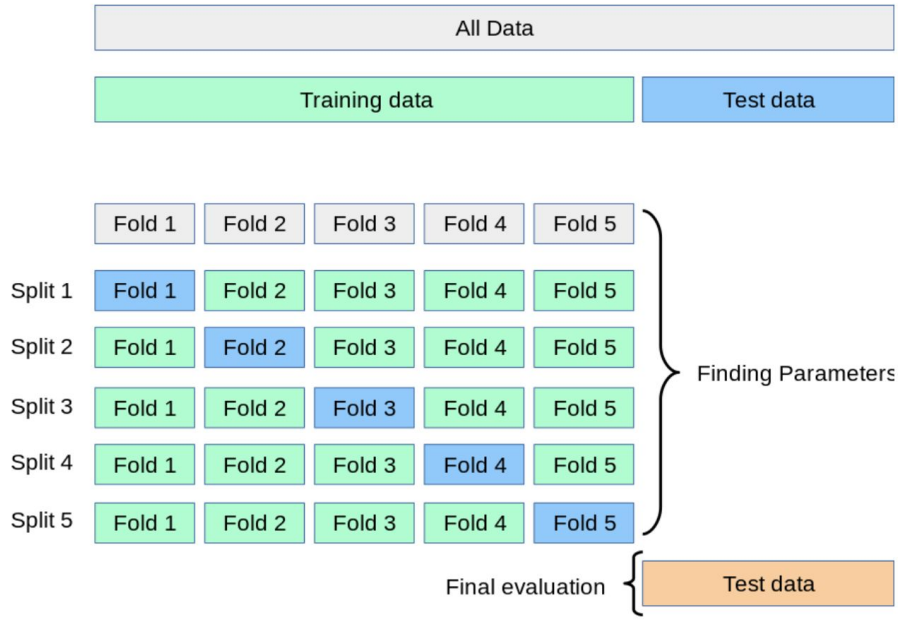
12. Testing (Data Validation/K-fold)

Before running our models, we basically split the data into different training and test datasets, once for each iteration. This strategy helps us to not show bias to any particular class label value. For example, while splitting the dataset if all the negative samples are in the trained data and positive samples are in the test data, the model will learn inappropriately. To avoid these types of errors we use k-fold stratified splitting and cross validation.

Cross Validation

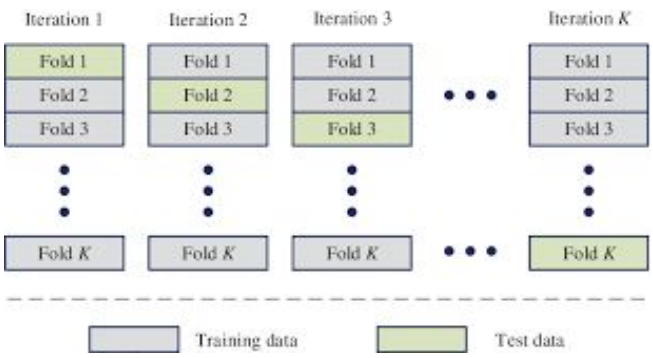
- Generally we split the whole dataset into training and test data and the training data is used for the model to learn and test data for predicting the scores. Cross validation to generally performed if the number of samples in the training dataset are low. This should help us identify how well the model fit with the data.

- Here, the dataset is first split into training and test dataset. Then the training dataset is again split into training and validation set.
- The model will learn from the training set and then perform prediction on the validation fold.
- The goal here is to know how well the model performs better when given a dataset.



K-fold Stratified splitting

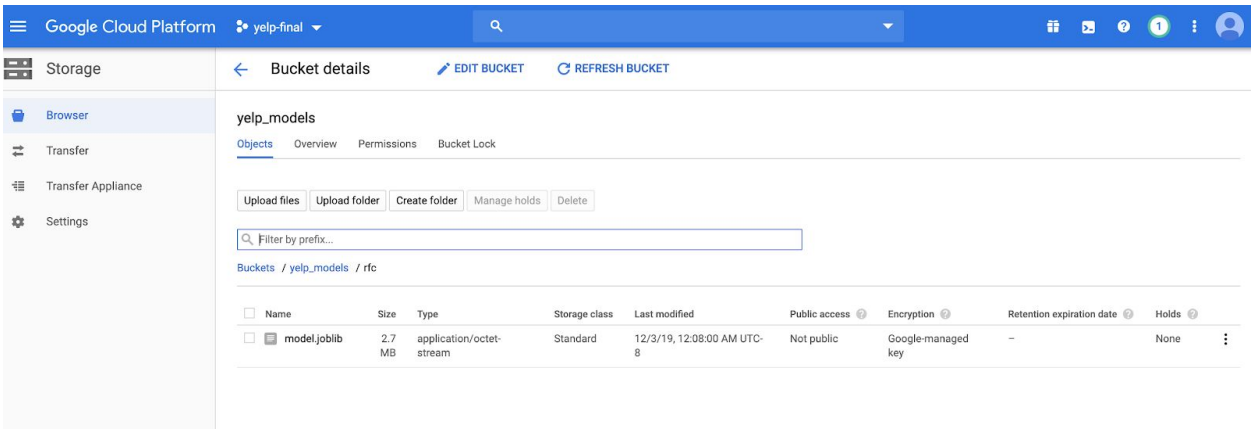
- Here, we first split our dataset based on K equal sized subsets. After splitting we consider one set as test data and k -1 subsets as train data.
- In each iteration, we consider one subset as test and other subsets as trained data and this is an iterative process. Repeat until all the K subsets are served as test data.
- By doing this K-fold, we get an accuracy each time and we consider the mean of all K fold accuracies and that is going to be out final score of our model.
- It will be used as one of our evaluation metrics for our models.
- In our project, we are using k-fold with 5 folds and trying to take the meaning of them for all the models.
- Stratified version of K-Fold helps us split the data in such a way that it doesn't show bias to any value/feature in the dataset.



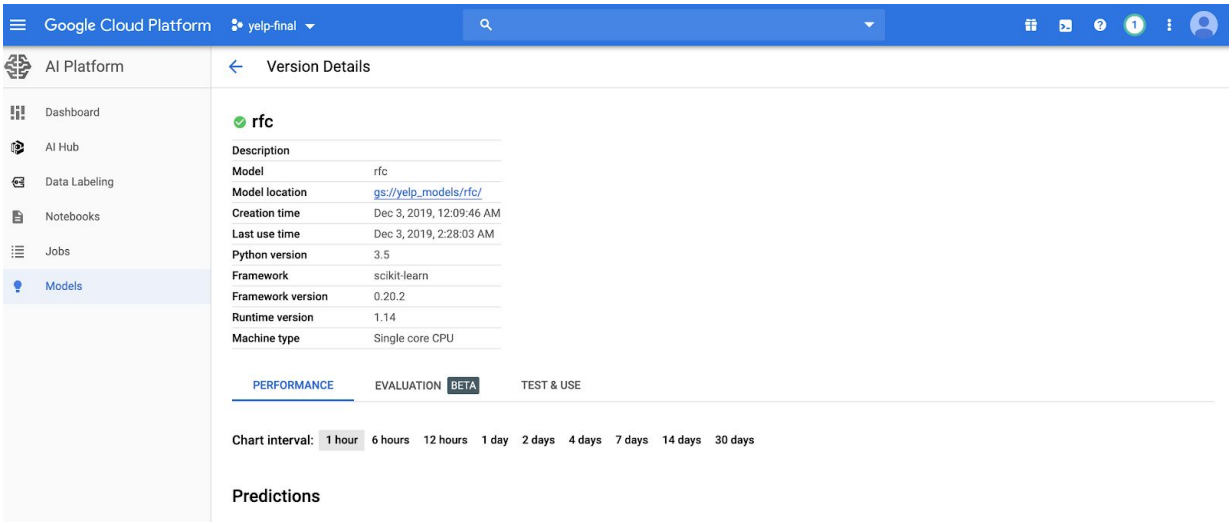
13. Model Deployment

Once we ran different classification models that supports multi class classification, we have evaluated the performance of these models and chose Random Forest Classifier, XG Boost Classifier and Logistic Regression Classifiers as winners. So we exported these models to local machine using sklearn joblib library. Once we have these joblib model files:

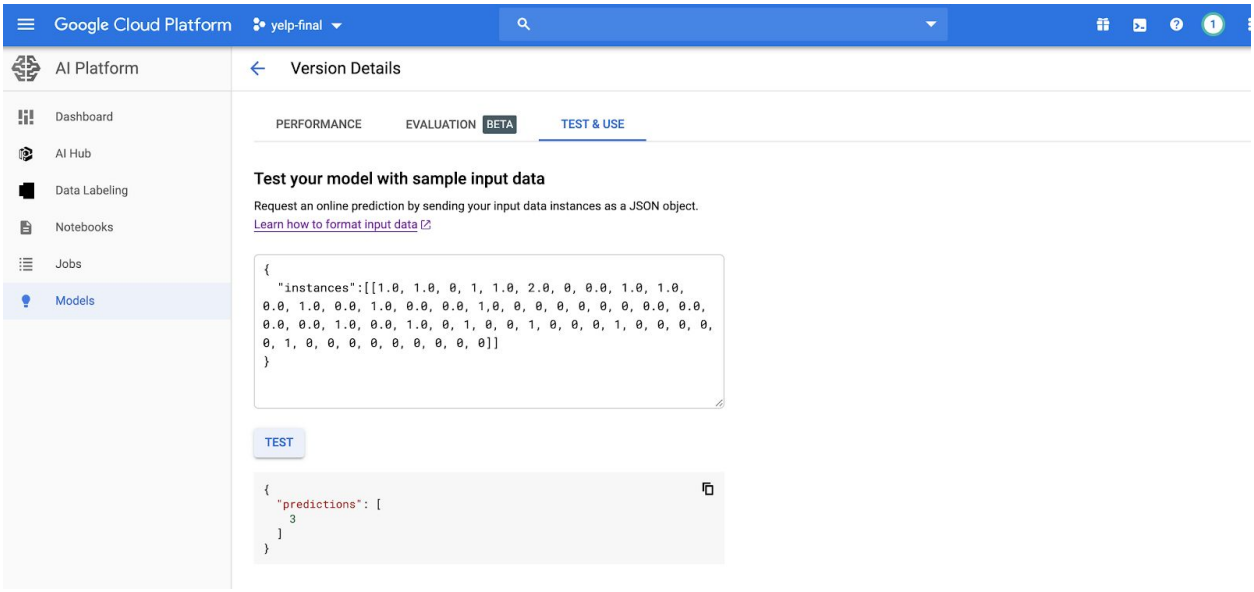
- We have uploaded these models into Google Cloud storage.

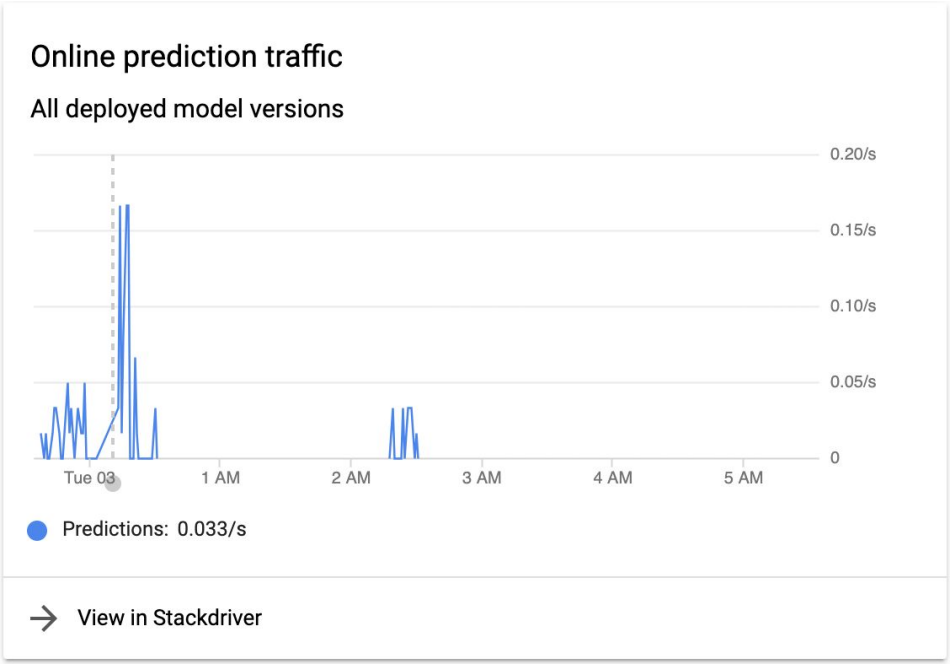


- Created a new model deployment cycle one for each model type.



- Provided the environment setup and the path of the model file to Google AI
- Google AI reads the model, deploys it and starts accepting API requests for prediction.





14. Design Patterns Used

- We have tried to make our code more modular with very less duplicate code.
- We have followed the checkpointing approach, to not lose the work and at the same time, this approach helped us to not run the entire project for every single change during development.

Singleton Pattern

- We have used Singleton pattern to create restrict the user create multiple objects. In our data to restrict create multiple db engines.
- Singleton class provides a global access to its object without the need for instantiating the object of the class.
- The main advantage of using this design pattern is to reduce memory usage whenever possible.

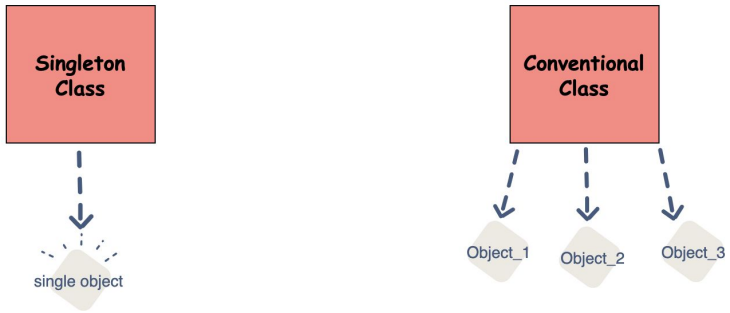


Image courtesy: www.educative.io

- Unlike Jupyter Notebooks style code, we have defined many python definitions in our code to reduce repetitions wherever possible.

15. AutoML

We have explored many AutoML tools and came across TransmorgrifAI framework developed by Salesforce. We have chosen this framework because of the following reasons.

1. Runs on top of spark.
2. One of the very few open source AutoML frameworks currently available.
3. Wanted to get hands on scala. TransmorgrifAI supports only Scala.
4. Well descriptive documentation and tutorials

AutoML is used to reduce the time of Machine Learning Engineer working on a project from months to hours. AutoML has the capabilities of applying standard data engineering techniques for pre-processing. However, given a noisy dataset like Yelp with hundreds of features, we decided to hand code the feature engineering steps and try the model prediction and parameter tuning given by AutoMl framework like TransmorgrifAI.

Here are the summary of results predicted by AutoML.

```
Model summary:
Evaluated OpLogisticRegression, OpRandomForestClassifier models using
Cross Validation and error metric.
Evaluated 8 OpLogisticRegression models with error metric between
[0.43766485387022713, 0.5795879488806348].
Evaluated 18 OpRandomForestClassifier models with error metric
between [0.4328438809755431, 0.5830757416060958].
+-------------------------------------------------------------+
|           Selected Model - OpRandomForestClassifier         |
+-------------------------------------------------------------+
| Model Param           | Value                               |
+-----------------------+-------------------------------------+
| cacheNodeIds          | false                               |
| checkpointInterval    | 10                                  |
| featureSubsetStrategy | auto                                |
| impurity              | gini                                |
```

```
| maxBins                | 32                                          |
| maxDepth               | 12                                          |
| maxMemoryInMB          | 256                                         |
| minInfoGain            | 0.001                                       |
| minInstancesPerNode    | 10                                          |
| modelType              | OpRandomForestClassifier                    |
| name                   | OpRandomForestClassifier_000000000040_2     |
| numTrees               | 50                                          |
| seed                   | 329511018                                   |
| subsamplingRate        | 1.0                                         |
| uid                    | OpRandomForestClassifier_000000000040       |
+------------------------+---------------------------------------------+
+-----------------------------------------------------------+
|                 Model Evaluation Metrics                  |
+-----------------------------------------------------------+
| Metric Name | Training Set Value | Hold Out Set Value     |
+-------------+--------------------+------------------------+
| error       | 0.3548148148148148 | 0.42324561403508776    |
| f1          | 0.6359772196544843 | 0.56753417116608       |
| precision   | 0.6270283846758737 | 0.5586041135202225     |
| recall      | 0.6451851851851852 | 0.5767543859649122     |
+-------------+--------------------+------------------------+
+-----------------------------------------------------------+
|                   Top Model Insights                      |
+-----------------------------------------------------------+
| Top Positive Correlations    |       Correlation Value    |
+------------------------------+----------------------------+
| latenight(latenight = null)  | -1.7976931348623157E308    |
| latenight                    | -1.7976931348623157E308    |
| breakfast(breakfast = null)  | -1.7976931348623157E308    |
| breakfast                    | -1.7976931348623157E308    |
| touristy(touristy = null)    | -1.7976931348623157E308    |
| touristy                     | -1.7976931348623157E308    |
| dinner(dinner = null)        | -1.7976931348623157E308    |
| dinner                       | -1.7976931348623157E308    |
| Japanese(Japanese = null)    | -1.7976931348623157E308    |
| Japanese                     | -1.7976931348623157E308    |
| Vegan(Vegan = null)          | -1.7976931348623157E308    |
19/12/02 22:24:23 INFO SparkContext: Invoking stop() from shutdown
hook
```

```
| Vegan                          | -1.7976931348623157E308 |
| French(French = null)          | -1.7976931348623157E308 |
| French                         | -1.7976931348623157E308 |
| American(American = null)      | -1.7976931348623157E308 |
+--------------------------------+-------------------------+
+-------------------------------------------------------------------------
--------------+
| Top Negative Correlations                                              |
Correlation Value |
+------------------------------------------------------------+----------
--------------+
| Bars                                                       |
1.7976931348623157E308 |
| Bars(Bars = null)                                          |
1.7976931348623157E308 |
| Chinese                                                    |
1.7976931348623157E308 |
| Chinese(Chinese = null)                                    |
1.7976931348623157E308 |
| RestaurantsDelivery                                        |
1.7976931348623157E308 |
| RestaurantsDelivery(RestaurantsDelivery = null)            |
1.7976931348623157E308 |
| Mongolian                                                  |
1.7976931348623157E308 |
| Mongolian(Mongolian = null)                                |
1.7976931348623157E308 |
| Italian                                                    |
1.7976931348623157E308 |
| Italian(Italian = null)                                    |
1.7976931348623157E308 |
| upscale                                                    |
1.7976931348623157E308 |
| upscale(upscale = null)                                    |
1.7976931348623157E308 |
| RestaurantsReservations                                    |
1.7976931348623157E308 |
| RestaurantsReservations(RestaurantsReservations = null)    |
1.7976931348623157E308 |
| NoiseLevel                                                 |
```

```
1.7976931348623157E308 |
+-------------------------------------------------------+----------
-------------+
+-----------------------------------------------+
| Top Contributions       |   Contribution Value |
+-------------------------+----------------------+
| review_count            |  0.12808948066565476 |
| FastFood                |  0.12741327022619447 |
| Caters                  |   0.0631842597131414 |
| Alcohol                 |  0.04675920193554789 |
| HasTV                   | 0.042905833591698486 |
| Burgers                 |  0.03996312015016243 |
| trendy                  | 0.038553615471395616 |
| RestaurantsPriceRange2  | 0.029462490474249717 |
| parking                 | 0.028335548592809828 |
| RestaurantsReservations | 0.026270283116490906 |
| casual                  | 0.026247578170924775 |
| NoiseLevel              |  0.02447247151706853 |
| WiFi                    |  0.02271822364671009 |
| American                |  0.02013292542769742 |
| OutdoorSeating          | 0.018578914884961877 |
+-------------------------+----------------------+
```

- As we can see above, AutoML has chosen **RandomForestClassifier** as the best model to predict along with the parameters.
- This framework ran on top on Spark and was able to come up with this model in a very short time.
- It has selected the best features in the dataset along with the top positively correlated and negatively correlated features

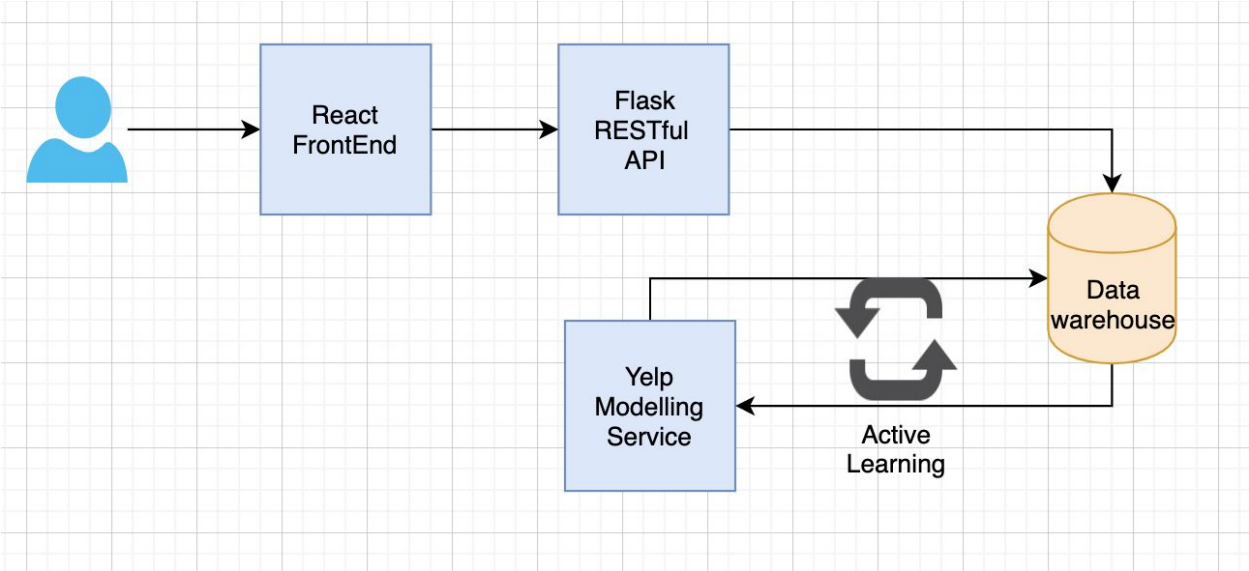
16. Data Engineering

- The data was fetched from Yelp and was used for cleaning, preprocessing and running the models. The data file was in json format and had multiple businesses. The restaurants related data had to be extracted in order to perform any EDA or run models on it.

17. Active Learning/Feedback Loop

We have also implemented the active learning approach on our model as follows.

- We already have the cleaned dataset with imputed values in our database.
- Whenever any user performs predictive operations on our model using the backend RESTful application, we are capturing the features as input attributes (X) and the prediction from our model deployed in the Google AI as the y label.
- We are storing this captured data in our datastore
- So for our next model training iteration, we have additional data that we already captured. We can use this data along with our dataset to improve the model performance.



18. Interpretability of the Model

After evaluating all the models we have run, we found out that RandomForest Classifier, XGBoost and Logistic Regression models performed better. Initially, Logistic regression looked to us like a right fit in terms of Mean cross validation, training and test accuracy scores as they were all close to each other. But random forest classifier looked to us like the best fit as we felt we were able to interpret the model in an optimistic way.

19. Links and references

- https://github.com/monicasjsu/yelp
- https://github.com/monicasjsu/flask_yelp
- https://github.com/vijaylaxmid/lsa-client
- https://github.com/monicasjsu/yelp-autoML