

# **User Review Based New Business Affinity Prediction System**

---

## **Intro to Data Science(CAP5571) Project Report**

### **Project Group 25**

**Professor : Daisy Zhe Wang**  
**TA : Xiaofeng Zhou**

**Team Members:**  
**Harshit Vijayvargia**  
**Harika Bukkapattanam**

## [Table of Contents](#)


<b>1.</b>	<b>Introduction .....</b>	
<b>1.1</b>	<b>Problem Synopsis .....</b>	
<b>1.2</b>	<b>Dataset .....</b>	
<b>1.3</b>	<b>Approach .....</b>	
<b>1.4</b>	<b>Results highlights.....</b>	
<b>2.</b>	<b>Dataset Description .....</b>	
<b>3.</b>	<b>Approach, algorithms, and tools.....</b>	
<b>3.1</b>	<b>Data cleaning.....</b>	
<b>3.2</b>	<b>Data preprocessing.....</b>	
<b>3.3</b>	<b>Algorithms implemented.....</b>	
<b>3.4</b>	<b>Tools and Technologies used.....</b>	
<b>3.5</b>	<b>Feature Generation using Sentiment Analysis .....</b>	
<b>3.6</b>	<b>End to End system diagram.....</b>	
<b>4.</b>	<b>Evaluation metrics .....</b>	
<b>4.1.1</b>	<b>Giving new business .....</b>	
<b>4.1.2</b>	<b>Prediction .....</b>	
<b>5.</b>	<b>Application and Challenges faced .....</b>	
<b>6.</b>	<b>Conclusion &amp; Future scope .....</b>	
<b>7.</b>	<b>References .....</b>	

## INTRODUCTION

### 1.1 Problem Statement

- Public online business directory systems like Yelp which publish crowd-sourced reviews about local businesses have huge amounts of data related to user-reviews.
- We describe our efforts to process such user reviews and business related raw information and determine the amount of attention a new business idea would gather based on its features using past user preferences in the given location.
- It will help registered businesses to customize its features according to what customers consider important and increase the probability of its success in a given location by looking at the predicted average star rating from our built model.

### 1.2 Dataset

-  dataset in JSON format containing 4,700,000 reviews from users for 156,000 businesses across 12 metropolitan areas.
- The dataset consists of various json files namely: business.json, review.json, user.json, checkin.json, tip.json and photos. We focus primarily on the business and review files.

### 1.3 Approach & Results

- Generated a new review text feature i.e the **sentiment score** processing all the user-reviews group by location using **NLP lexicon** and **PySpark Hadoop MapReduce** which will correctly represent the user's opinion towards a business.
- Used supervised **SVM classification** for building location based good model fits, performed accuracy analysis by selecting appropriate **kernel** , **gamma** and **c** parameters.
- Generated **new businesses** for various locations from existing dataset and **predicted average star rating** using the trained SVM RBF model fit.
- Trained the **SVM RBF** model using the entire dataset available and obtained a highest **accuracy** of 85% for training and testing datasets.
- Since Support Vectors provide very **fast prediction** performance, this held up on running numerous predictions even for the **entire dataset** thus **scaling well**.

### 1.4 Improvements

- Constructed a set of new businesses for each location using the data of businesses located in other cities.

- Predicted the average review stars for each of the newly created businesses in every location using the corresponding trained SVM RBF model fit.
- Visualized and analysed the prediction accuracies and runtime complexities.

## DETAILS

### 2.1 Data Description

#### ➤ Business.json

Contains business data including location data, attributes, and categories.

Business.json

```
{
  "business_id": (unique string business id),
  "name": (the business's name),
  "neighborhood": (the neighborhood's name),
  "address": (the full address of the business),
  "city": (the city),
  "state": (2 character state code),
  "postal code": (the postal code),
  "latitude": (latitude),
  "longitude": (longitude),
  "stars": (star rating, rounded to half-stars),
  "review_count": (number of reviews),
  "is_open": (0 or 1 for closed or open),
  "attributes": {
    Key business_attribute: value/object attribute
  },
  "categories": [array of strings of business categories],
  "hours": {
    key day : value hours(hours are using a 24hr clock)
  }
}
```

#### ➤ Review.json

Contains full review text data including the user\_id that wrote the review and the business\_id the review is written for.

```
{
  "review_id": (22 character unique review id),
  "user_id": (22 character unique user id),
  "business_id": (22 character business id),
```

```
"stars": (star rating),
"date": (date formatted YYYY-MM-DD),
"text": (the review itself),
"useful": (number of useful votes received),
"funny": (number of funny votes received),
"cool": (number of cool votes received)
}
```

## 2.2 References

- [https://www.yelp.com/html/pdf/YelpDatasetChallengeWinner\\_InferredFuture.pdf](https://www.yelp.com/html/pdf/YelpDatasetChallengeWinner_InferredFuture.pdf)
- <http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>
- <http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

## APPROACH

### 3.1 Data Preparation

- Converted the business.json and review.json files to business.csv & review.csv files using the csv and json libraries in python.

### 3.2 Feature Engineering: Sentiment Analysis Using Hadoop MapReduce

- We wrote the code in python but since the review file is of size 3.9 Gb we implemented it using PySpark MapReduce and ran it on an AWS EC2 instance.

Our MapReduce algorithm is as follows:

1. The input to our Mapper is the reviews file in JSON format using the `spark.read.json` to read the file and obtain an RDD.
2. In the Mapper, we calculate the sentiment score of all the reviews and output the `business_id` as key and a combination of `review_id` and `sentiment score` as value.
3. In the Reducer, we take the average value of sentiment scores for each business id i.e aggregating by business id and then reducing by key.

### 3.2 Data Cleaning

The obtained csv files were processed as follows:

- Firstly, the businesses related to eateries were filtered out to enable us to get a more concise version of our problem statement.
- Secondly, obtained the business eateries and their associated sentiment score feature obtained from the MapReduce step.

- Thirdly, **extracted** around 29 **relevant business attributes** out of the available 95 attributes based on the **correlation** to the average review stars of the businesses and by removing the sparsely populated attributes across the businesses.
- Finally, obtained the **model input** by combining the business eateries with the relevant attributes, business associated sentiment score and the average user review stars per location resulting in below structure:

City	Business_id	Relevant Features	Sentiment Score
------	-------------	-------------------	-----------------

### 3.3 Supervised Learning

#### 3.3.1 Data Preprocessing

- Converted all the **symmetric** business attribute's values to **binary**.
- **SVM RBF** kernel assumes that all features are centered around 0 and have variance in the same order. Preprocessed model input data attributes by **standard scaling** using the sklearn preprocessing module.
- Given k feature vectors of n features in the training set, the mean and standard deviation of each feature is computed.

$$\mu^i = \frac{1}{k} \sum_{j=0}^k x_j^i$$

$$\sigma^i = \sqrt{\frac{1}{k} \sum_{j=0}^k (\mu^i - x_j^i)^2}$$

- Both the training and test data sets are then scaled using the scaling factors generated for the training set.

$$x_{scaled}^i = \frac{x^i - \mu^i}{\sigma x^i}$$

#### 3.3.2 Data Splitting

- Divided the model input data into **training and testing** sets using the train\_test\_split method of sklearn cross validation package in python.

#### 3.3.3 Model Fitting: Algorithms

##### 3.3.3.1 Random Forest

- We used the sklearn ensemble **RandomForestClassifier** package and ran it our model input with parameters: n\_estimators as 10(number of trees), min\_sample\_split as 2(minimum number of samples required to split an internal node) and with remaining default values.
- Even though the classifier produced **good individual** training accuracies for various city models, it **overfitted** the data thus resulting in large difference between testing and training accuracies.

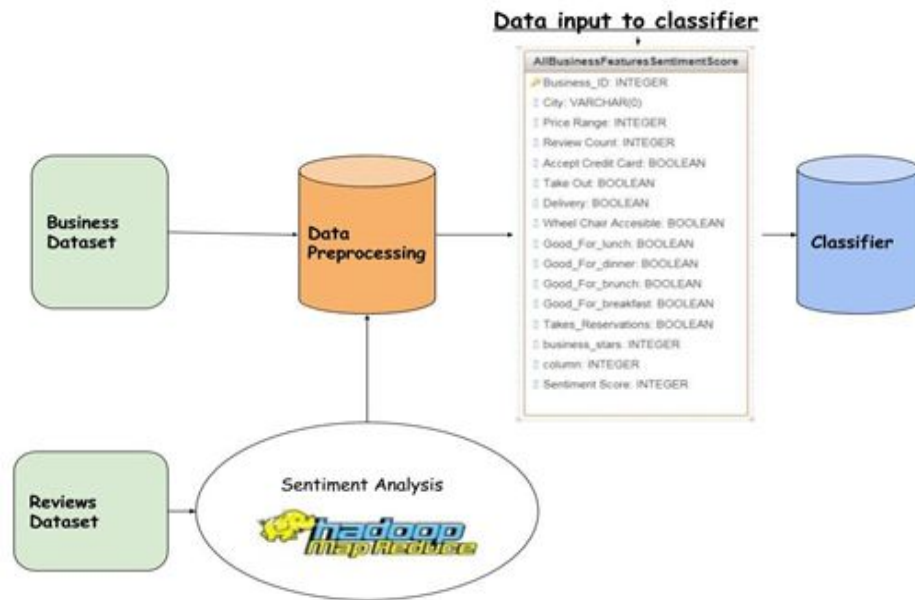
### 3.3.3.2 Support Vector Machine

- The dataset has **numerous sparse** business attributes and hence we chose SVM classifier since it works well with high dimensional data.
- SVM aims to find a function **margin function**  $f(x)$  that does not deviate more than some away from the training data.
- By having the margin, the training process can determine a subset of training points as having the most influence on the model parameters.
- The points with the most influence end up defining the margin and are called **support vectors**.
- Thus, the final model that is produced depends only on a subset of the training data. **Predictions** then run only with these smaller number of support vector making the process **fast**.
- We used the sklearn **svm.SVC** package and ran it on our model input by testing with **different gamma** values as well as **various kernels**. **RBF** kernel resulted in the best accuracy and hence we baselined using it.

## 3.4 Tools & Technologies



## 3.5 System Design



## METRICS & EVALUATION

### 4.1 New Business Generation

- Constructed a set of new businesses for each city using the data of businesses located in other cities.

### 4.2 Prediction

- In order to gauge our model, we predicted the review stars for each of the newly created businesses in every location using the trained SVM RBF model fits and calculated the average rating for each location. Following results were observed for 5

City	predicted_avg_rating	actual_avg_rating
Toronto	2.9	3
Phoenix	4.02	4
Charlotte	4.78	5
Las Vegas	3.91	4
Richmond	4.13	4

locations:

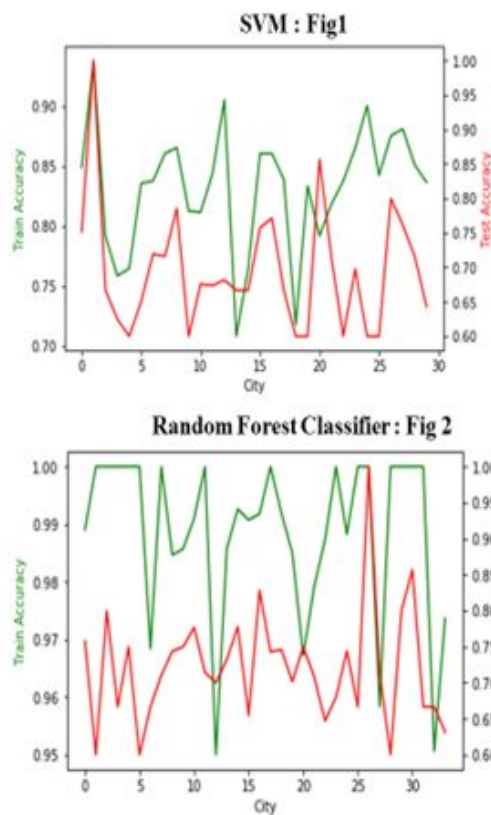
Ex: The average predicted rating for new businesses in toronto location is 2.9 and the actual average rating for businesses already in toronto is 3.

## RESULTS



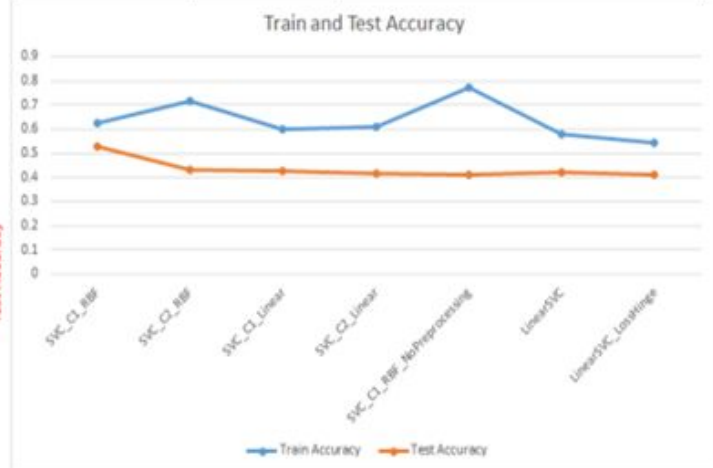
## 5.1 Classifier Selection

Classifier Selection on the basis of accuracy



**SVM model parameters selection : Fig 3**

Train Accuracy	Test Accuracy	Parameters
0.624123402	0.542374837	c = 1, kernel = "rbf"
0.714505147	0.433934534	c = 2, kernel = "rbf"
0.600952494	0.427557664	c = 1, kernel = "linear"
0.775047514	0.413936467	c = 1, kernel = "rbf", no preprocessing
0.617386536	0.426587355	c = 1, kernel = "rbf", randomState = 0
0.577037933	0.423913976	C = 1, gamma = 2, kernel = 'poly'



As we can see from the above graphs. SVM showed better accuracy on testing data whereas random classifier overfitted the model. Hence we selected SVM classifier with parameters ( c=1, Kernel = "rbf")

## 5.2 Analysis

- We obtained model fits for all the cities using the SVM RBF classifier which produced an average training accuracy of 86% and training accuracy of 75%.
- We used the trained model fits and predicted average review stars for 201 number of cities using AWS EC2 instance which took around 50 minutes 25 seconds.

## CONCLUSION

- **Feature Selection:** We analyzed and extracted relevant features out of the available 95 business features, also avoiding the null valued ones.
- **Data Cleaning:** To overcome the problem of loading very large reviews file in python during data cleaning, we used the *chunksize* option of *read\_csv* function followed by concatenation.

- **Sentiment Analysis Of Reviews:** Due to local memory limits, sentiment score evaluation of 3 Gb user reviews text data was performed using Hadoop MapReduce run on AWS EC2 instance.
- **Standardizing Data:** SVM RBF kernel assumes that all features are centered around 0 and have variance in the same order. We preprocessed model input data attributes by standard scaling using the sklearn preprocessing packages in python.
- **Supervised Learning:** We described a series of techniques used to infer future attention of businesses using numerous kinds of extracted relevant features. These features were generated from business statistics and raw review text. We find that of all features, the temporal features provided the best prediction results.