CMPE 255- 02: DATA MINING

Prof. Kaikai Liu

PROJECT REPORT

on

“FOOD POISONING & RESTAURANT BUSINESS SUCCESS PREDICTION”

Rohan Patel (010745904)

1. Introduction

1.1. Motivation

Being one of the largest and fastest-growing industries, Restaurant businesses have pressure to succeed and maintain their brand name. Today's internet-savvy people not only assess an establishment by its food quality but also on aspects like customer service, ambiance, location, etc. which simply adds to the increasing pressure on these businesses. Moreover, people are quick to spread their opinions about a restaurant in their network and social media. A minute slip can cost a business a fortune and would take them months, if not years, to recover their position in the market.

These opinions and judgments are most widely shared on Yelp and similar platforms in form of reviews and ratings. This data can be used to identify and extract interesting information regarding the restaurant business from various customer interactions.

1.2. Objective

The main objective of this project is twofold - 1) to identify businesses that are likely to cause food poisoning to their customers and 2) to predict if a business will be successful based on its current circumstances.

- **Identifying food poisoning:** To identify restaurants that would lead to incidences of food poisoning, analyzed all the customer reviews for mention of any symptoms of or episodes of nausea and stomach infections.
- **Restaurant success:** To identify the success of the restaurant, analyze the location (address, longitude, and latitude), cuisine type (categories), number of stars, and count of reviews.

Applied appropriate data mining and machine learning techniques to achieve the objective of this project.

2. System Design and Implementation details

2.1. Algorithms considered

2.1.1. Food poisoning prediction from reviews

Each prediction required different attributes from the dataset. In the case of Predicting food poisoning, extract the business id, review text, and their corresponding star ratings for each record which is either a restaurant or a food vendor. Assigned a class to all the review records based on the occurrence of words related to any food poisoning or its symptoms.

Using the Naive Bayes classification, trained the model to assign appropriate classes to the test review. To identify if the business would cause food poisoning, aggregate the classifications for unique businesses and a score for food poisoning in terms of percentage is computed for each such business. The threshold for marking a business as one that may cause a food poisoning incident is 30%.

2.1.2. Restaurant Success

Classifiers considered for this part of the project are as follows: SVC, Decision Trees, Gradient Boosted Trees, Lasso Regression, and Random Forest Classifiers. Used all of the algorithms to predict stars among these 9 categories(1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0).

Also used DBScan to cluster close-by restaurants based on their longitude and latitude features of the dataset.

2.2. Technologies and Tools used

Python packages: Numpy, pandas, sklearn, string, NLTK
IDE and notebook used: Jupyter notebook and Google Colab notebooks

2.3. Implementation Details

2.3.1. Food Poisoning prediction

2.3.1.1. Data processing

- Since the dataset is huge and to save some time and effort every time the code is run, I uploaded the dataset in the form of CSV files on Google Drive. Accessing through Google Colab and their API to access the files on the drive.
- The data from yelp_business.csv and yelp_review.csv are loaded into Pandas Data frames.
- Considering business.csv, only the businesses for which the business category was either ‘restaurant’ or ‘food’ are retained for further analysis.
- Similarly, from review.csv only attributes like business_id, text, and stars are retained for the model training.
- These two data frames are then merged to make sure business_id is present in business.csv and review.csv and to retrieve the names of the restaurants.
- The text data of the reviews are cleaned to remove unwanted characters, stopwords, and HTML tags and later are lemmatized for the next step. WordLemmatizer, Whitespace tokenizer, pos_tag, and word_tokenize are used from the NLTK package to perform text cleaning.

2.3.1.2. Data Labeling and Classification

- Since the dataset is unlabelled, that is it does not have records that are already marked, assigned a class to all the review records based on the occurrence of words related to any food poisoning or their symptoms.

- Considering the fact that people give lower ratings to those businesses which have a negative impression on them, for the labeling as food poisoning-related review, only considered those records for whom the value for star rating is less than 3.
- A predefined word list is created in such a way that it contains most of the words indicating or related to food poisoning or its symptoms.
- Later, the review dataset is split into a training dataset and a test dataset using sklearn.model_selection. The training dataset will be used to train the classifier model.
- Trained the Naive Bayes classification of the text model to assign appropriate classes to the test dataset and got an accuracy of 97.4% along with other performance metrics.
- Alternatively, also performed the K-Nearest Neighbors classification of the text and selected k = 150 as it provided a good accuracy of 98.1%, however, it took much more time than the Naive Bayes classifier.

2.3.1.3. Prediction

- Further, to identify if the business would cause food poisoning, aggregated the classifications for unique businesses and a score for food poisoning in terms of percentage is computed for each such business based on the reviews complaining about food poisoning and the total number of reviews for that particular restaurant.
- When a particular business_id is given, it can make out what is the percentage of reviews that complained of food poisoning.
- Can set a threshold value to say a restaurant will definitely give food poisoning, set the threshold to be 30%. This means if 30% of the total reviews mention food poisoning then that restaurant will definitely be labeled as unsafe.

2.3.2. Restaurant Success

2.3.2.1. Data Exploration

- Explored the stars, is_open, state, city, review count, name, postal code, categories, address, longitude, and latitude of the yelp_business dataset.
- Examined cuisine to stars by displaying it in order to better grasp the data.
- Discover the relationship between location, stars, and cuisine.

2.3.2.2. Data Cleaning

- Data cleaning was performed on the following features of the yelp_business.csv dataset:
- States: Checked and found that the number of states was greater than 50 (since restaurants in the USA only were the scope of this project). Found out that states from Canada were also included in the dataset, removing those states. Random states such as “ABE” were also encountered and were dropped from the data.

- Wrong State and City combinations: Some cities were found to be with wrong state combinations, for example, “Austin” was stated along with “CA”, such wrong combinations were rectified using the SImple Maps dataset for cities and states (https://simplemaps.com/data/us-cities).
- Address: addresses that were not available were marked as “NOT_AVAILABLE”.
- Performed one-hot encoding on the categories feature of the yelp_business.csv dataset.
- Made the name and location feature of the dataset lowercase.
- The longitude and Latitude feature of the dataset was narrowed down to only the longitudes and latitudes inside the boundary of the United States. Got the bounding longitudes and latitudes of the United States (Wikipedia).

2.3.2.3. Data Analysis

- The cleaned-up data was then used for analysis. The analysis was performed to cluster restaurants that were located near each other, using the longitude and latitude features of the dataset.

2.3.2.4. Data Modelling

- The data was then split into training, cross-validation, and testing data using the 60 - 36 - 4 split.
- Classifiers used for this were as follows: Decision Tree, SVC, Gradient Boosted Trees, Random Forest Classifier, and Lasso Regression,

3. Proof of concept evaluation

3.1. Datasets used

Name: Yelp Dataset
Source: https://www.kaggle.com/yelp-dataset/yelp-dataset/version/6
Type: .csv files
No. of files: 4
Files names: yelp_business.csv (31 MB), yelp_review.csv (4 GB), yelp_users.csv (1.36 GB), yelp_tip.csv (148 MB).

3.2. Experiments

3.2.1. Using JSON data

- The Yelp dataset available on the official site took a considerable amount of time to get loaded since it consisted of both business data as well as academic data in the local form of JSON.

- Further, it was more time-consuming when the data had to be converted from JSON to data frames.
- That is why I changed the dataset source from Yelp's official site to Kaggle's yelp data set which was already present in the form of a CSV file. Alternatively, had the option to select only the file that was necessary.

3.2.2. Using XGBoost

- While experimenting with the Yelp review dataset, first decided to use a business's categories and attributes as features to predict whether a restaurant has a high chance of food poisoning.
- Created a new column for every attribute and every category in the dataset, then fitted this data in a few ensemble methods like Random Forest and XGBoost. Also, the Decision Tree model was fitted on the same dataset.
- But the accuracy was really low at around 20%, the highest achievable accuracy was with the XGBoost model which was 50% accurate.
- The problem with this model was most of the attributes were not at all related to food poisoning.
- The attributes consisted of valet parking, parking, staff, wait for time, etc. which did not impact the food quality of a restaurant, hence this model was discarded, and came up with the idea of using review text and ratings of the restaurant.

3.2.3. Using K-Nearest Neighbour Classification

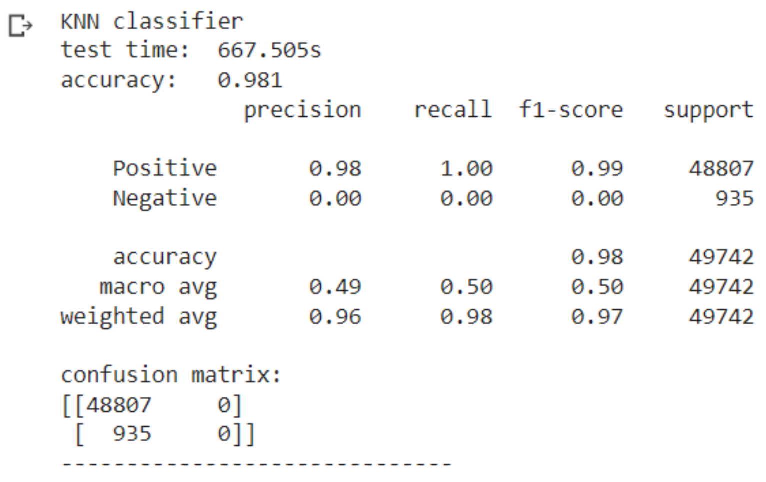
- While selecting a classification model, experimented by performing KNN Classification to classify the test data.
- It produced an accuracy of 98.45% with k = 150, however, the major hurdle was the prediction time. Since KNN classifiers are slower when handling large amounts of data.

3.3. Analysis of Results

3.3.1. Analysis of Classifier models for Food Poisoning

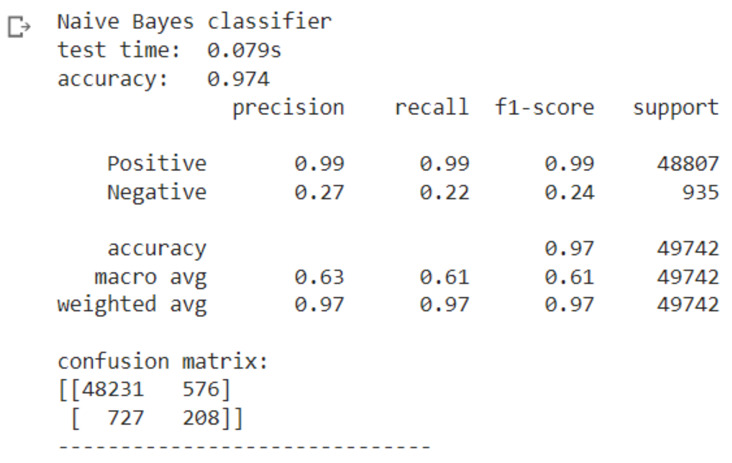
- When performing text classification, I noticed that the KNN classifier model performed better than the Naive Bayes model in terms of accuracy, however, it took longer to run.
- Since accuracy from the Naive Bayes classifier was not that far behind and produced quicker results, so opted to go with it.
- Apart from accuracy and time, both the classifiers performed comparably on other performance metrics.

3.3.1.1. K-Nearest Neighbor performance score



```
KNN classifier
test time:  667.505s
accuracy:   0.981
              precision    recall  f1-score   support

    Positive       0.98      1.00      0.99     48807
    Negative       0.00      0.00      0.00       935

    accuracy                           0.98     49742
   macro avg       0.49      0.50      0.50     49742
weighted avg       0.96      0.98      0.97     49742

confusion matrix:
[[48807     0]
 [  935     0]]
------------------------------
```

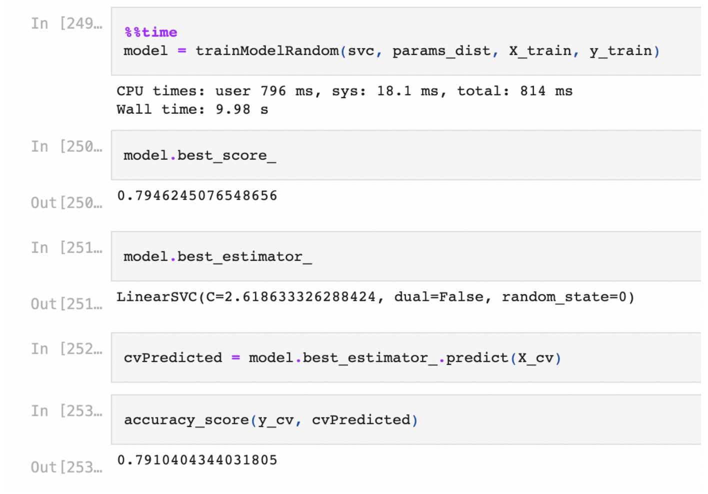
3.3.1.2. Naive Bayes performance score



```
Naive Bayes classifier
test time:  0.079s
accuracy:   0.974
              precision    recall  f1-score   support

    Positive       0.99      0.99      0.99     48807
    Negative       0.27      0.22      0.24       935

    accuracy                           0.97     49742
   macro avg       0.63      0.61      0.61     49742
weighted avg       0.97      0.97      0.97     49742

confusion matrix:
[[48231   576]
 [  727   208]]
------------------------------
```

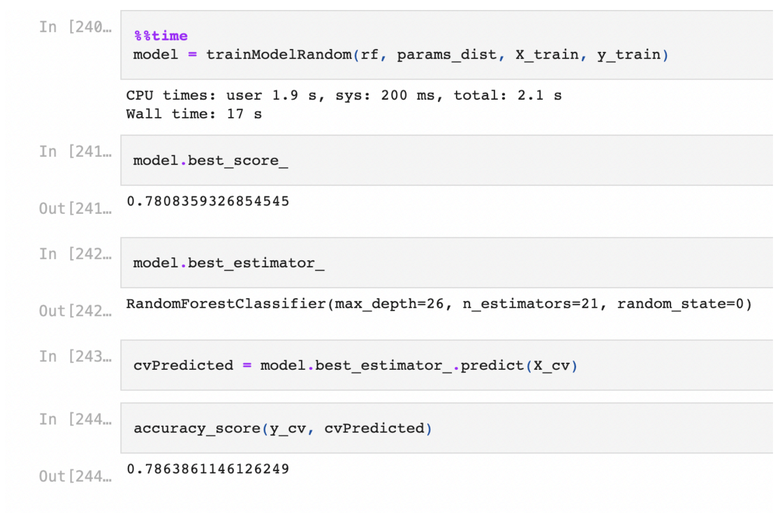
3.3.2. Analysis of Classifier models for Restaurant Success

3.3.2.1. SVC



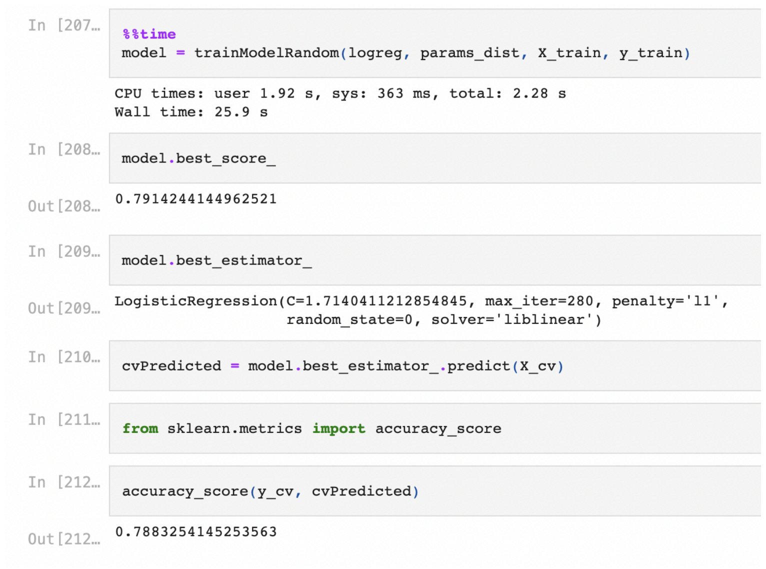
```
In [249…
%%time
model = trainModelRandom(svc, params_dist, X_train, y_train)

CPU times: user 796 ms, sys: 18.1 ms, total: 814 ms
Wall time: 9.98 s

In [250…
model.best_score_

Out[250…
0.7946245076548656

In [251…
model.best_estimator_

Out[251…
LinearSVC(C=2.618633326288424, dual=False, random_state=0)

In [252…
cvPredicted = model.best_estimator_.predict(X_cv)

In [253…
accuracy_score(y_cv, cvPredicted)

Out[253…
0.7910404344031805
```

3.3.2.2. Random Forest



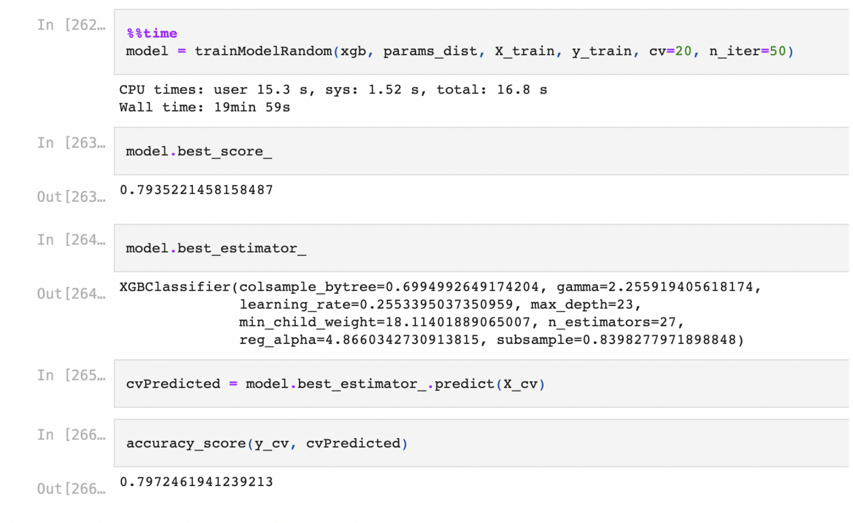
```
In [240…  %%time
          model = trainModelRandom(rf, params_dist, X_train, y_train)

          CPU times: user 1.9 s, sys: 200 ms, total: 2.1 s
          Wall time: 17 s

In [241…  model.best_score_

Out[241…  0.7808359326854545

In [242…  model.best_estimator_

Out[242…  RandomForestClassifier(max_depth=26, n_estimators=21, random_state=0)

In [243…  cvPredicted = model.best_estimator_.predict(X_cv)

In [244…  accuracy_score(y_cv, cvPredicted)

Out[244…  0.7863861146126249
```

3.3.2.3. Lasso Regression



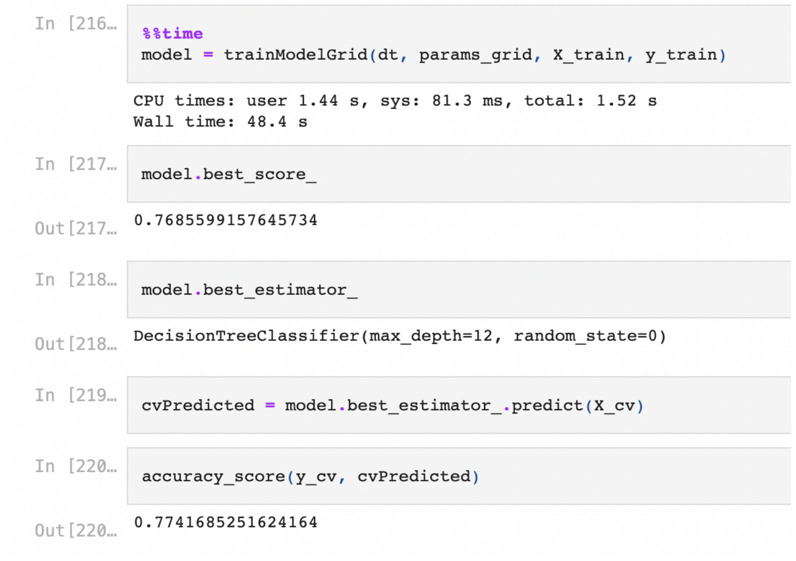
```
In [207…  %%time
          model = trainModelRandom(logreg, params_dist, X_train, y_train)

          CPU times: user 1.92 s, sys: 363 ms, total: 2.28 s
          Wall time: 25.9 s

In [208…  model.best_score_

Out[208…  0.7914244144962521

In [209…  model.best_estimator_

Out[209…  LogisticRegression(C=1.7140411212854845, max_iter=280, penalty='l1',
                             random_state=0, solver='liblinear')

In [210…  cvPredicted = model.best_estimator_.predict(X_cv)

In [211…  from sklearn.metrics import accuracy_score

In [212…  accuracy_score(y_cv, cvPredicted)

Out[212…  0.7883254145253563
```

3.3.2.4. Gradient Boosted Trees



```
In [262…  %%time
          model = trainModelRandom(xgb, params_dist, X_train, y_train, cv=20, n_iter=50)

          CPU times: user 15.3 s, sys: 1.52 s, total: 16.8 s
          Wall time: 19min 59s

In [263…  model.best_score_

Out[263…  0.7935221458158487

In [264…  model.best_estimator_

Out[264…  XGBClassifier(colsample_bytree=0.6994992649174204, gamma=2.255919405618174,
                        learning_rate=0.2553395037350959, max_depth=23,
                        min_child_weight=18.11401889065007, n_estimators=27,
                        reg_alpha=4.8660342730913815, subsample=0.8398277971898848)

In [265…  cvPredicted = model.best_estimator_.predict(X_cv)

In [266…  accuracy_score(y_cv, cvPredicted)

Out[266…  0.7972461941239213
```

3.3.2.5. Decision Trees



```
In [216…  %%time
          model = trainModelGrid(dt, params_grid, X_train, y_train)

          CPU times: user 1.44 s, sys: 81.3 ms, total: 1.52 s
          Wall time: 48.4 s

In [217…  model.best_score_

Out[217…  0.7685599157645734

In [218…  model.best_estimator_

Out[218…  DecisionTreeClassifier(max_depth=12, random_state=0)

In [219…  cvPredicted = model.best_estimator_.predict(X_cv)

In [220…  accuracy_score(y_cv, cvPredicted)

Out[220…  0.7741685251624164
```

3.4. Methodology Followed

3.4.1. Food Poisoning

The dataset was divided into train and test sets in an 80:20 ratio. Using the test set it can determine how accurate the model is.

3.4.2. Restaurant Success

60% of the data is utilized for training, 36% for cross-validation, and the remaining 4% is used as a test set.

4. Discussions and Conclusions

4.1. Difficulties faced

1. The Yelp dataset for reviews is really huge and it takes some time to load the whole data and convert it into a data frame. While using read_json for JSON files the system was crashing and hence had to use the CSV format available on Kaggle for the yelp dataset.

2. Also, the execution time was very high, and hence whenever I made little changes in the code, it took a lot of time to test it.

4.2. Conclusion

If someone wants to open a restaurant and look at the Yelp dataset, the Restaurant Business Success Predictor will assist them in making appropriate business decisions based on client preferences. If a customer wants to judge a restaurant based on the quality of food, or if someone wants to inspect a restaurant for food poisoning cases based on customer reviews, they can use the Food Poisoning Predictor section of the Project.