Final project - Big Data For Official Statistics, Msc Data Science - La sapienza

Alessandro Taglieri, 1890945
February 2, 2021

Abstract

This project studies data shared by Government of San Francisco to predict the risk category of restaurants. The dataset used is LIVE and is updated daily by the government. This paper explores steps involved in pre-processing to improve training and test set. Post cleaning of data, important features are selected and data is then divided into training and testing split. Using training data, machine learning algorithm from various family are considered and trained. The performance of the algorithms is measured using several techniques and they are compared among them. All data anlysis on the dataset is available on a simple dashboard deployed on Heroku.

1 Introduction

Restaurant inspections are very important for the prevention of food borne diseases. To predict if any restaurant is continually behaving in common pattern or is a periodic defaulter, machine learning can be used. This will in turn help in predicting future risk category for the restaurants and those with high risk category can be inspected on regular basis. In this paper we'll understand the data recorded for each restaurant in San Francisco, California, recorded during inspection. This data will then be preprocessed and cleaned to remove any noise if present. Post cleaning the data it will be made to fit in different machine learning algorithms and models will be measured for their accuracy on predicting risk category. At the end various comparison between different models are done to select the best model for this dataset.

2 Dataset

2.1 Source and description

Dataset used in this paper is named as Restaurant Scores - LIVES Standard. The source of the dataset is DataSF[0]. When browsing web portal, different categories are available to select data from. This dataset falls under the category of Health and Social Services[1]. Dataset comprises of 17 columns and 53.7K rows. However, for this experiment data recorded from year 2016 – February 22, 2019 is used to train model and test various algorithms in machine learning. Below is the summary of dataset, Table 1, with column name and their data type.

(1)

Column	Data type
$business\_id$	Integer/Number
$business\_name$	String/Object
$business\_address$	String/Object
$business\_city$	String/Object
$business\_state$	String/Object
$business\_postal\_code$	Integer/Number
$business\_latitude$	Float/Decimal point
$business\_longitude$	Float/Decimal point
$business\_location$	String/Object
$business\_phone_number$	String/Object
$inspection\_id$	String/Object
$inspection\_date$	DateTime
$inspection\_score$	Float/Decimal point
$inspection\_type$	String/Object
$violation\_id$	String/Object
$violation\_description$	String/Object
$risk\_category$	String/Object

2.2 Preprocessing

To preprocess data, each column was carefully analyzed. The goal of preprocessing is to convert the data type of all columns in a particular symmetry. Our approach here, tries to convert all the columns to numerical data type. During the phase of preprocessing a lot of noisy data was removed. Also, during feature engineering a lot of redundant data from specific columns were removed and unique elements were kept, to help model train better. Below is detailed process of data preprocessing for each column with feature selection.

- $business\_name$: This column lists restaurant name. However, restaurants can be uniquely identified by $business\_id$, so it was dropped.
- $business\_address$: This feature is not important because we have postal codes, latitude and longitude data to observe business location. Hence, this was dropped..
- $business\_city$: It was observed that business city in whole dataset was San Francisco. As this column cannot contribute much towards training of data, it was dropped..
- $business\_state$: It was observed that business state in whole dataset was California. As this column cannot contribute towards training of data, it was dropped.
- $business\_location$: This column is combination of longitude and latitude. As it introduces redundancy in data, it was dropped.
- $business\_city$: It was observed that business city in whole dataset was San Francisco. As this column cannot contribute much towards training of data, it was dropped..
- $business\_phone\_number$: This doesn't seem an important feature when predicting risk category for restaurant. This column was dropped.This data was converted into categorical data. But after removing null values it was realized that only one category was consistent in all the rows. So, this column was dropped.
- $business\_type$: This data was converted into categorical data. But after removing null values it was realized that only one category was consistent in all the rows. So, this column was dropped.

Another important step about preprocessing is to delete all rows that contains NaN values in main feature (like $risk\_category$, $violation\_description$, $inspection\_score$). While checking null and unique values it was observed that postal codes and latitude were having zero values. These rows were dropped from the dataset, as it can impact training of the data. Further, $inspection\_date$ was broken down into three columns namely year, month and day to help model train better.

2.3 Exploratory Data Analysis

I did dome analysis to explore the original dataset. At first I show some distributions about $risk\_category$ and $violation\_description$. For this last attribute I've grouped all description in four main category.

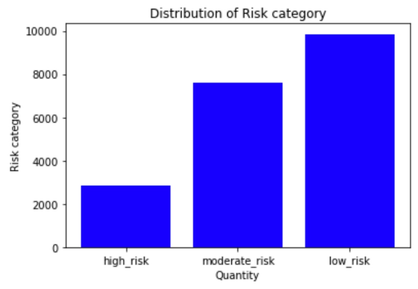

Figure 1: Distribution of Risk Category attribute

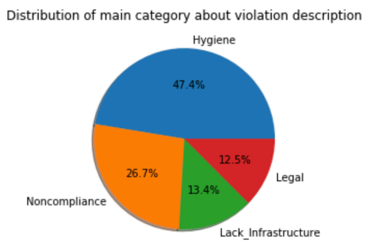

Figure 2: Distribution of violation's description

Then I did an analysis to see the distribution of all Restaurants in a map. Figure 3 represents all restaurants with different colors based on their inspection score while Figure 4 shows a sample of restaurants with marked points that are colored with three different colors (red: high risk, orange: moderate risk, green: low risk).

Then I explored data over the time. In Figure 5 show the amount of inspections during the years while the

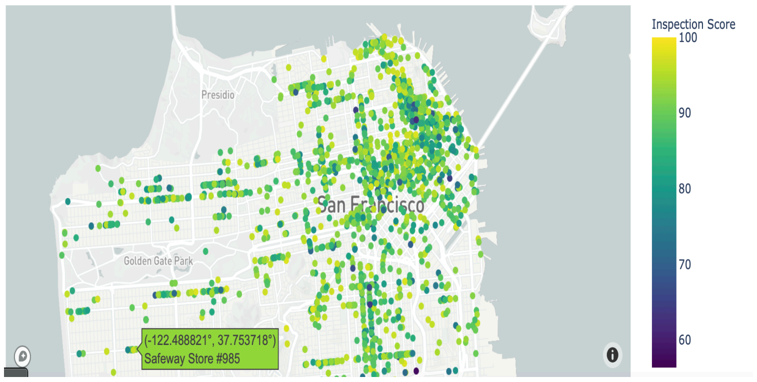

Figure 3: Restaurant over the maps with their inspection score

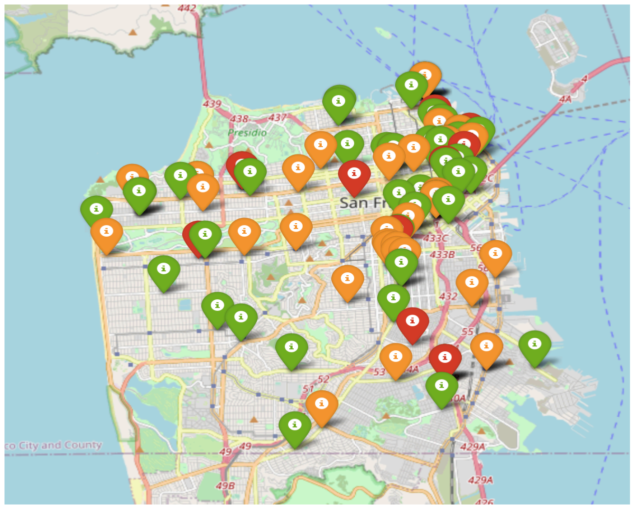

Figure 4: Restaurant over the maps with their risk category

Figure 6 shows the amount of inspections divided by three different risk category during years.

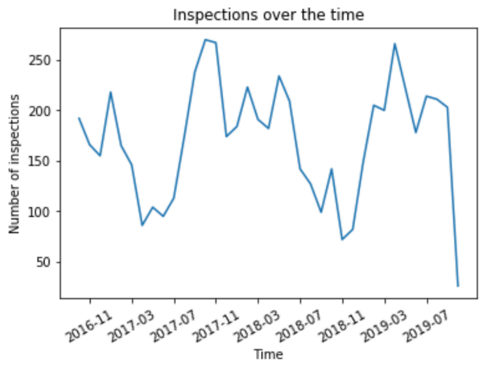

Figure 5: Amount of inspection during years

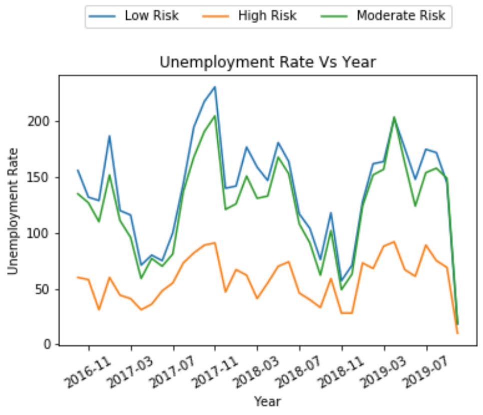

Figure 6: Amount of inspection during years

Finally I've focused to select the best five and the worst five restaurants. I've chosen them among all restaurant that has at least twenty inspections. Best restaurants are considered restaurant that got less inspection with worst inspection's result (High Risk). Worst restaurants are considered restaurant that got more inspection with worst inspection's result (High Risk). The following Figures represents distribution of different inspections (based on risk Category) for both best and worst five restaurants.

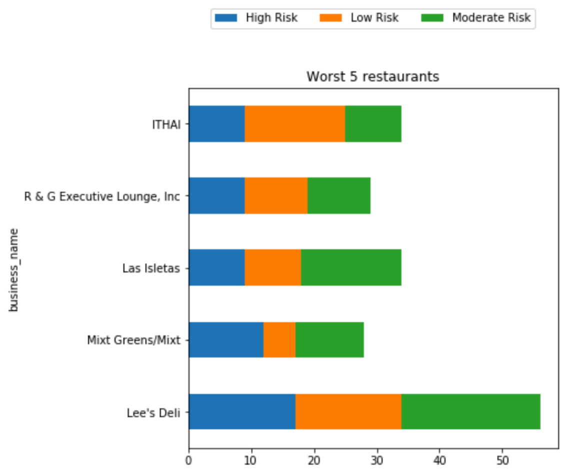

Figure 7: Worst five restaurants

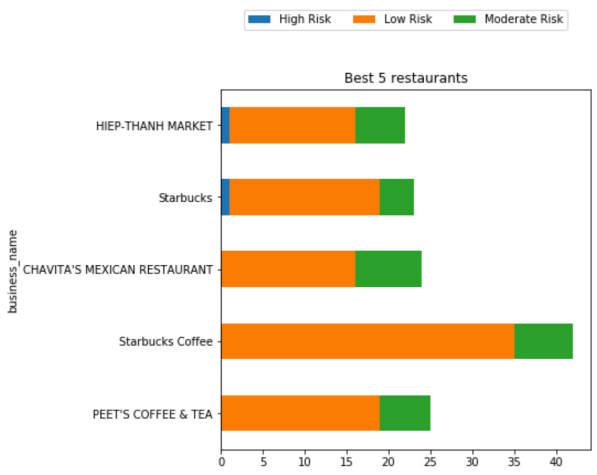

Figure 8: Best five restaurants

2.4 Feature selection

Feature selection for training data is very important. If proper features are not selected than it can either underfit or overfit the model. Thus, making our model highly unreliable. It is very important to select features which can convey some story or pattern in the data. Our model will try to learn pattern within the data and will make predictions based on it. Some of the below features were re-engineered and some were kept as is.

- $business\_id$: This column is very important as it uniquely identifies all the restaurants.
- $business\_postal\_code$: This column can uniquely identify region of the restaurant, which can be an important feature to consider.
- $business\_latitude$: This column lists latitude of restaurant, which can be used on a geographical map and estimate popular location.
- $business\_longitude$: Same as above, it can be used on a geographical map and estimate popular location.
- $inspection\_id$: This column is a combination of business id and $inspection\_date$. So it was dropped. E.g. 62010-20160415. Here 62010 is business id and 20160415 is date on which inspection was conducted.
- $inspection\_date$: This column lists date on which inspection was conducted. It also has a time unit attached to it which was trimmed during pre- processing as it was same in all rows.
- $inspection\_score$: This is very important feature and wasn't altered.
- $violation\_id$: This column uniquely identifies violation description. So, it was part of the training data.
- $violation\_description$: As this data had finite unique values, it was converted to categorical numerical data. i.e. Hygiene:1, Legal:2, Noncompliance:3 and Lack of Infrastructure:4
- $risk\_category$: This column is very important part of the dataset. We will be predicting risk category from the dataset. So, categories were converted to numerical data. i.e. Low Risk:1, Moderate Risk:2, High Risk:3

Converting data to numerical categorical values also favored in class balancing, and most of classes which were having least count got eliminated.

The following figure shows correlation between all attributes in the preprocessed dataset.

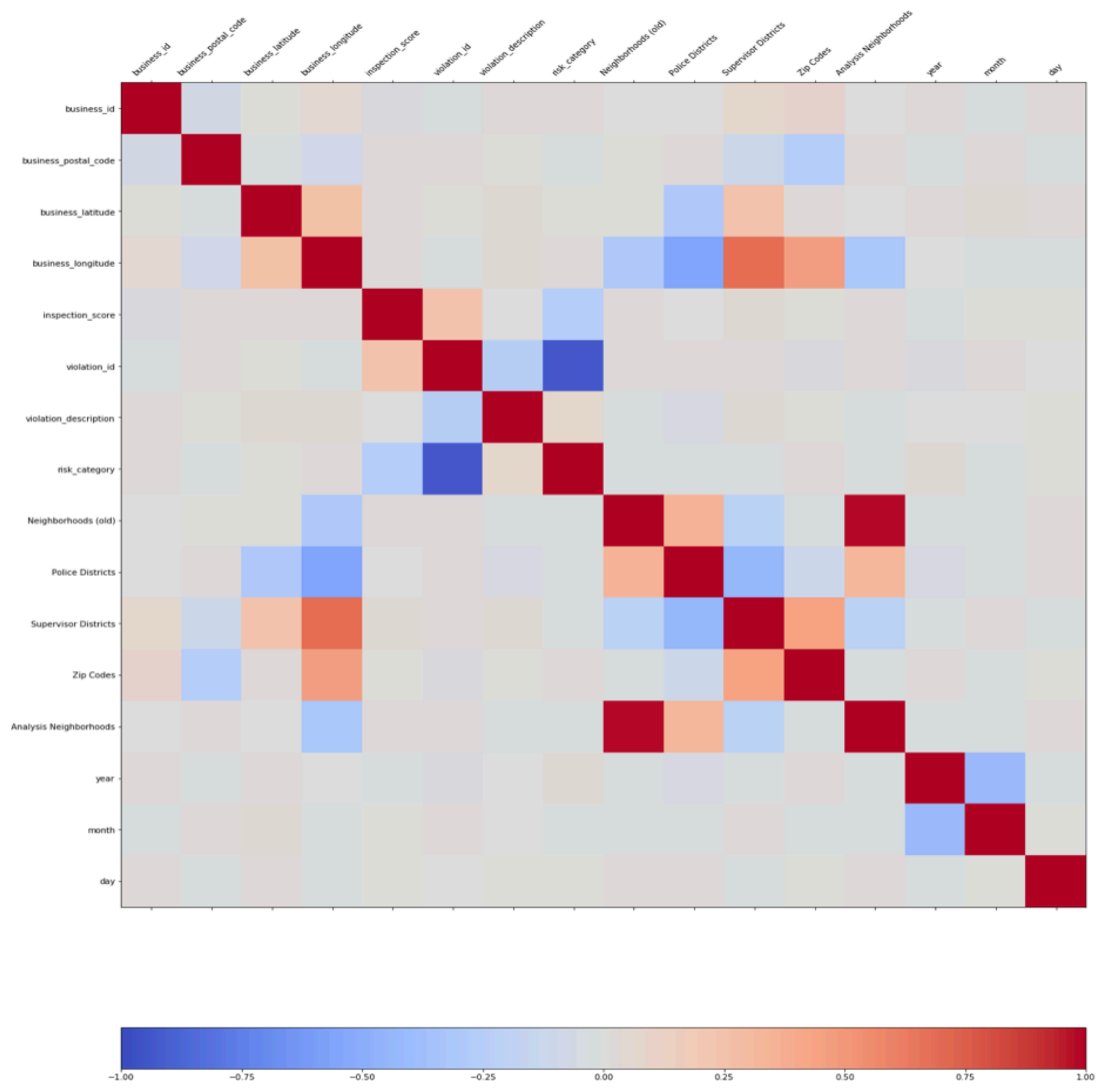

Figure 9: Restaurant over the maps with their inspection score

3 Data Setup and model evaluation

To train model, two separate datasets were used. One dataset was used for training and other for testing. After training the model, it was evaluated on few performance parameters like precision, recall, accuracy and box plot. Below sections explains the setup and evaluation in detail.

3.1 Data setup

Our model will be trained to classify data into one of three risk categories. So, risk category will become predict label. After selecting the predictor, we will split our data into train and test datasets. Usually, training data is bigger than the test set. So, in our case training data will account for 80% of the total data and test data will account for 20%. After split-data I proceeded to train different models with different techniques. For every model I did an hyperparameter optimization to get best scores for each model. After training, test data was used for determining prediction and score. Based on score, multiple performance evaluation metrics were computed, which helped in determining the best model for the dataset.

3.2 Model Evaluation

To select the best model, each model will be quantified to some metric value and graph plots. These metric values and graphs will help in the judgement of the best model for the dataset. I've used scikit-learn[2] library to implement each metrics (except cross validation) and each machine learning model. These parameters are explained in detail below:

- K-Fold Cross Validation: Cross validation is procedure to evaluate machine learning model on a limited data sample using resampling. It is often referred to as k-fold cross validation because k decides number of groups a given data sample is to be split into. Cross validation is used in machine learning model on unseen data to estimate the skill of the model which is less biased and less optimistic estimate, as compared to other methods.
- Accuracy: It is the ratio of number of correct predictions to total number of samples. Higher the accuracy better the model fits data. If accuracy is too high i.e. above 95% it means model is overfitting the data.
- Precision: It is the ratio of number of correctly predicted positives to total number of positives in the sample.
- Recall: It is the ratio of number of correctly predicted positives to total number of all relevant samples. Which means all samples which should have been predicted as positives.
- F1-score:F1 score is a harmonic mean between precision and recall. It signifies how robust and precise the model is.
- Algorithm comparison with box plot: Box plot is used to evaluate performance of multiple models by plotting box mean accuracy, generated from k-fold cross validation.

4 Model

4.1 Random Forest

It fits number of decision tree classifier on various subsample to form final prediction using averaging and controls overfitting of the model. In this project this algorithm is not useful beacause we have an overfitting on the data and also with Cross-Validation technique didn't avoid this overfitting. The following table shows main metrics, it's easy to understand that there is an overfitting.

Performance indicator	**Value**
Accuracy	0.9989
Precision	0.9989
Recall	0.9989
F1-score	0.9989

4.2 SGD Classifier

Stochastic gradient descent is very efficient approach as it uses convex loss functions to train model with linear classifier. As we can see in the following table, this method has poorly performance. We should avoid to use this model.

Performance indicator	**Value**
Accuracy	0.3776
Precision	0.14260
Recall	0.3776
F1-score	0.207

4.3 AdaBoost Classifier

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. With cross-validation we noticed that it has an high variance on the training accuracy, but during the inference we obtain good results, as represented in the following table.

Performance indicator	**Value**
Accuracy	0.85560
Precision	0.75110
Recall	0.85560
F1-score	0.79500

4.4 K Nearest Neighbours

The main principle behind nearest neighbor is to find training samples closest in distance to new point and predict classes from these. With this model I obtained good result. As we can see in the following table, we got 83% about accuracy. It can be considered great model that I can use.

Performance indicator	**Value**
Accuracy	0.8308
Precision	0.8286
Recall	0.8308
F1-score	0.8289

4.5 Gaussian Naive Bayes Classifier

To select most probable hypothesis about the data, we can use prior knowledge. Bayes theorem provides a way by which we can calculate probability of hypothesis on the basis of prior knowledge. Also with this method we have Overfitting on the data, as you can see performances in the following table.

Performance indicator	**Value**
Accuracy	0.98195
Precision	0.98253
Recall	0.98195
F1-score	0.98168

4.6 Ridge classifier

As it implements l2 regularization so it is called as ridge regression. It adds squared magnitude of coefficient as a penalty to loss function. It converts the label data into [-1, 1] and solves the problem with regression method. The highest value in prediction is accepted as a target class and for multiclass data muilti-output regression is applied. As represented in the following table, we obtained great result, for example 79% of accuracy. Like KNN, this model can be considered a good model to use for our goal.

Performance indicator	Value
Accuracy	0.79435
Precision	0.80278
Recall	0.79435
F1-score	0.74860

5 Experimental Results

To compare algorithms a box plot was plotted for all the models. The data points for box plot were calculated using 10-fold cross validation with random seed value and their mean accuracy across each cross-validation fold, for each algorithm, were plotted to evaluate the range and percentiles.

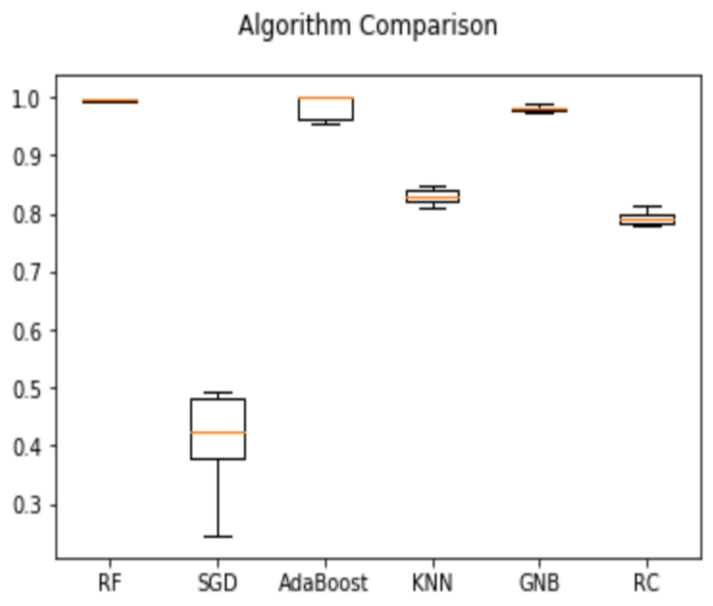

Figure 10: Model comparison by accuracy

6 Predicting violation description

I've decided to build another model that allow me to predict risk category, given a violation description. This task is done by using a Multinomial Naive Bayes model. I've considered only two columns from original dataset: $violation\_description$ that is the description of the violation referred to a single inspection and $risk\_category$ is the category assigned to this inspection that can be High, Moderate, Low Risk.

6.1 Preprocessing Data

For this step I've cleaned only text in $violation\_description$. I proceeded with a text preprocessing in the following way:

- Tokenization: Every sentence is tokenized in different words.
- Remove short words (less than two characters).
- Removing stop words from the input sentences.

- Converting every items (list of words) in TfIdf vector. This transformation is made with tfidf (term frequency - inverted document frequency) principle that works on relevance of every word in different set (sentences).

Finally I obtained a tfidf vector for every violation's description. So our data is composed by them and risk category (target).

6.2 Model

I've chosen a Multinomial Naive Bayes model that allow me to get an optimal accuracy. The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work. This task is relatively easy because violation description are very similiar among them (I supposed that inspectors used frequently same terms). For this task I got only accuracy score that is showed in the following figure:

Performance indicator	**Value**
Accuracy	0.98

7 Dashboard

I've decided to represent data analysis and data visualization for this project with a web app. This web-app consists in a dashboard developed with Streamlit[3] framework and then I deployed it on Heroku[4]. It's available on the following link: https://bdos-project.herokuapp.com

This dashboard contains four distinct section:

- Introduction on the data: It contains an overview and general info on the original dataset.
- Inspections over the map: It shows distribution of the restaurants on the map with two distinct figure. It allow to user to insert max number of restaurant that user want see on the map.
- Data over the time: This panel allow to see some plots about distribution of some feature during time/years.
- Best Worst restaurants: It contains general info about all inspections about best and worst five restaurants.
- Predict risk category from violation description: This panel allows user (probably an inspector) to insert a possible violation description and then user can see a predicted risk category referred to violation. To perform this task it was needed to save trained model from the jupyter notebook and the dict related to text preprocessing. In this way when we want to do a prediction on the dashboard, it's needed only to upload dict and model and make inference with given input sentence.

8 Conclusion

This experiment tried to classify restaurants into three risk categories based on the data shared by Government of San Francisco. The dataset comprises of geographical location, inspection scores and violations by restaurants. However, the data is not clean and it contains lots of features which are not important, messy merely acts like noise for model during training. The data was pre-processed and was made suitable for training on various models. Models from all the families were considered and trained. It was observed that few models were overfitting the data, like Random Forest and Gaussian Naive Bayes classifiers. However, ridge classifier, AdaBoost classifier and KNN worked perfectly for this dataset. The mean accuracy for both of them was above 79%. For the second goal of this project (from text sentences, i.e violation description, predicting risk category), I've used Multinomial Naive Bayes model that got 0.98 of accuracy. It means that there can be overfitting because all sentences use words from a not-big dictionary (all sentences from the same category are very similiar). This means that it can be very easy to predict the properly category but there can be an high risk to overfit the data. Finally, this last model is usable in the dashboard[5]. There, you can find all interactive and interesting analysis on the original dataset. All material to perform all tasks of this project is available on the github repository[6], while code needed to develop dashboard is in other github repo[7].

9 Resources

- Github repository where you can find Jupyer notebbok that contains full code about analysis and ml model[1] implementation and presentation of this project(ppt): https://github.com/AlessandroTaglieri/BDOS
- Github repository that contains all code needed to develop dashboard on stramlit: https://github.com/AlessandroTaglieri/st heroku-BDOS

9.1 References

- -[0] SF source website: https://datasf.org/
- -[1] SF Restaurant Scores Dataset: https://data.sfgov.org/Health-and-Social-Services/Restaurant-Scores-LIVES-Standard/pyih-qa8i?rowindex=0=
- -[2] Scikit-learn: https://scikit-learn.org/stable/
- -[3] Streamlit: https://www.streamlit.io
- -[4] Heroku: https://dashboard.heroku.com
- -[5] Dashboard: https://bdos-project.herokuapp.com
- -[6] Project repository: https://github.com/AlessandroTaglieri/BDOS
- -[7] Dashboard repository: https://github.com/AlessandroTaglieri/streamlit-heroku-BDOS