

▼ 1 Predicting NYC Restaurant Health Violations

Contents ⚙

2 Introduction

- ▼ 2 Introduction
 - ▼ 2.1 Objective
 - 2.1.1 Imports
- ▼ 3 Obtain
 - ▼ 3.1 Obtaining Restaurant Info
 - 3.1.1 Understanding NYC Data
 - 3.1.2 Target Variable -- NYC Health Violations
 - 3.1.3 Engineering Target Variable
 - ▼ 3.2 Obtaining Yelp Business Data
 - 3.2.1 Yelp Business Search
 - 3.2.2 Yelp Reviews
 - 3.2.3 Joining Yelp Reviews
 - 3.3 Joining NYC DOHMH Data
- ▼ 4 Scrubbing The Data
 - 4.1 Drop businesses missing data
- 5 Exploring The Dataset
- ▼ 6 Preprocessing For Further Modeling
 - 6.0.1 Lets find most frequent words
- 7 Preprocessing For Model Implementation
- ▼ 8 Models
 - ▼ 8.1 Bag-of-Words Model
 - 8.1.1 Model Evaluations
 - ▼ 8.2 Text Preprocessing Pipeline
 - 8.2.1 Multinomial Naive Bayes
 - 8.2.2 Tuned Multinomial Naive Bayes
 - 8.2.3 Logistic Regression
 - 8.2.4 Tuned Logistic Regression
 - 8.2.5 Decision Tree Model
 - 8.2.6 Tuned DecisionTree Model
 - 8.2.7 Support Vector Classification
 - 8.2.8 Tuned SupportVector Classification
- ▼ 9 Conclusions
 - 9.1 Best Model
 - According to the CDC, more than 48 million Americans per year become sick from food, and an estimated 75% of the outbreaks came from food prepared by caterers, delis, and restaurants. In fact, most foodborne illnesses occur in restaurants.
 - 9.2 Next Steps
 - 9.3 Appendix

2.1 Objective

The goal for this project is to leverage public citizen generated data from social media to narrow the search for critical health and safety violations in New York City. As the City of New York manages



an open data portal, everyone can access historical hygiene inspections and violation records. By combine these two data source this project aims to determine which words, phrases, ratings, and patterns among restaurants lead to critical health and safety violations. This model can assist city health inspectors do their job better by prioritizing the kitchens most likely to be in violation of code.

Contents

- 1 Predicting NYC Restaurant Inspections
- 2 Introduction
 - 2.1 Objective
 - 2.1.1 Imports
- 3 Obtain
 - 3.1 Obtaining Restaurant Information
 - 3.1.1 Understanding NYC Data
 - 3.1.2 Target Variable -- N
 - 3.1.3 Engineering Target
 - 3.2 Obtaining Yelp Business Data
 - 3.2.1 Yelp Business Search
 - 3.2.2 Yelp Reviews
 - 3.2.3 Joining Yelp Reviews
 - 3.3 Joining NYC DOHMH Data
- 4 Scrubbing The Data
 - 4.1 Drop businesses missing data
- 5 Exploring The Dataset
- 6 Preprocessing For Further Analysis
 - 6.0.1 Lets find most frequent words
- 7 Preprocessing For Modeling
- 8 Models
 - 8.1 Bag-of-Words Model
 - 8.1.1 Model Evaluations
 - 8.2 Text Preprocessing Pipelines
 - 8.2.1 Multinomial Naive Bayes
 - 8.2.2 Tuned Multinomial Naive Bayes
 - 8.2.3 Logistic Regression
 - 8.2.4 Tuned Logistic Regression
 - 8.2.5 Decision Tree Model
 - 8.2.6 Tuned DecisionTree Model
 - 8.2.7 Support Vector Classification
 - 8.2.8 Tuned SupportVectorClassification
- 9 Conclusions
 - 9.1 Best Model Results
 - 9.2 Next Steps and Future
- 9.3 Appendix - Deep NLP
 - 9.3.1 Word2Vec Embeddings
 - 9.3.2 Classifying With Embeddings
 - 9.3.3 Pretrained Vector Model
 - 9.3.4 Mean Word Embedding

```
In [2]: 1 from IPython import display
2 from bs4 import BeautifulSoup as bs
3 import requests
4 import json
5 import pandas as pd
6 import numpy as np
```

Contents

| | |
|---|---|
| 1 Predicting NYC Restaurant Violations | import matplotlib.pyplot as plt |
| 2 Introduction | import matplotlib inline |
| 2.1 Objective | import seaborn as sns |
| 2.1.1 Imports | import geopandas as gpd |
| 2.2 Data Sources | from shapely.geometry import Point, Polygon |
| 3 Obtain Data | import plotly.express as px |
| 3.1 Obtaining Restaurant Data | import warnings |
| 3.1.1 Understanding NYC Data | 3.1.1 Understanding NYC Data |
| 3.1.2 Target Variable | import time |
| 3.1.3 Engineering Target Variable | 3.1.2 Target Variable -- N |
| 3.1.4 Geocoding | 3.1.3 Engineering Target Variable |
| 3.2 Obtaining Yelp Business Data | import folium |
| 3.2.1 Yelp Business Search | import folium.plugins as plugins |
| 3.2.1.1 Yelp Reviews | 3.2.1.1 Yelp Reviews |
| 3.2.2 Yelp Business Details | 3.2.2 Yelp Business Details |
| 3.2.3 Joining Data | 3.2.3 Joining Data |
| 3.2.4 Joining NYC Data | import glob, os |
| 3.2.5 Joining NYC Data | import fastparquet |
| 3.3 Joining NYC Data | |
| 4 Scrubbing The Data | |
| 4.1 Drop businesses missing data | call_apis = False |
| 4.2 Exploring The Dataset | warnings.filterwarnings("ignore") |
| 5 Exploring The Dataset | pd.set_option('display.max_colwidth', 200) |
| 6 Preprocessing For Further Analysis | pd.set_option('display.max_columns', 50) |
| 6.0.1 Lets find most frequent words | 6.0.1 Lets find most frequent words |
| 7 Preprocessing For Modeling | pd.options.display.float_format = '{:.2f}'.format |
| 8 Models | from sklearn.preprocessing import OneHotEncoder |
| 8.1 Bag-of-Words Model | |
| 8.1.1 Model Evaluation | |
| 8.2 Text Processing | import spacy |
| 8.2.1 Multinomial Naive Bayes | from spacy.load("en_core_web_lg") |
| 8.2.2 Tuned Multinomial Naive Bayes | from spacy import displacy |
| 8.2.3 Logistic Regression | import nltk |
| 8.2.4 Tuned Logistic Regression | import string |
| 8.2.5 Decision Tree Model | from nltk.collocations import * |
| 8.2.6 Tuned Decision Tree Model | from nltk import word_tokenize, wordpunct_tokenize |
| 8.2.7 Support Vector Classification | from nltk.tokenize import RegexpTokenizer |
| 8.2.8 Tuned Support Vector Classification | from nltk.sentiment.vader import SentimentIntensityAnalyzer |
| 8.3 Word Embedding | from Wordcloud import WordCloud |
| 9 Conclusions | from nltk import FreqDist |
| 9.1 Best Model Results | from nltk.corpus import stopwords |
| 9.2 Next Steps | from nltk.stem import WordNetLemmatizer |
| 9.3 Appendix | from gensim.models import Word2Vec |
| 9.3.1 Word2Vec Embedding | from gensim.utils import simple_preprocess |
| 9.3.2 Classifying With Embeddings | from nltk import word_tokenize |
| 9.3.3 Pretrained Vector Embedding | # nltk.download('wordnet') |
| 9.3.4 Mean Word Embedding | # sk-learn |
| 9.3.4.1 Sk-learn Configuration | from sklearn import set_config |
| 9.3.4.2 Sk-learn Metrics | set_config(display='diagram') |
| 9.3.4.3 Sk-learn Confusion Matrix | from sklearn import metrics |
| 9.3.4.4 Sk-learn Confusion Matrix | from sklearn.metrics import confusion_matrix |

```
57 from sklearn.metrics import classification_report
58 from sklearn.model_selection import train_test_split
59 from sklearn.model_selection import GridSearchCV
60 from sklearn.pipeline import Pipeline
61 from sklearn.feature_extraction.text import TfidfVectorizer, TfidfTrans
62 from sklearn.tree import DecisionTreeClassifier
63 from sklearn.dummy import DummyClassifier
64 from sklearn.utils.class_weight import compute_class_weight
65 from sklearn.svm import SVC
66 from sklearn.svm import LinearSVC
67 from sklearn.linear_model import LogisticRegression
68 from sklearn.naive_bayes import MultinomialNB
69 from sklearn.model_selection import cross_val_score
70 import imblearn.pipeline
71 from imblearn.over_sampling import RandomOverSampler
```

Contents ☰

```
 65 from sklearn.svm import SVC
 66 from sklearn.svm import LinearSVC
 67 from sklearn.linear_model import LogisticRegression
 68 from sklearn.naive_bayes import MultinomialNB
 69 from sklearn.model_selection import cross_val_score
 70 import imblearn.pipeline
 71 from imblearn.over_sampling import RandomOverSampler
 72
 73 # Deep Learning
 74 from tensorflow import keras
 75 from keras.models import load_model
 76 from tensorflow.keras.layers import Input, Dense, LSTM, Embedding
 77 from tensorflow.keras.layers import Dropout, Activation, Bidirectional,
 78 from tensorflow.keras.models import Sequential
 79 from tensorflow.keras import initializers, regularizers, constraints, optimizers
 80 from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
 81 from tensorflow.keras.preprocessing import text, sequence
 82 from tensorflow.keras.utils import to_categorical
 83 from keras.preprocessing import text, sequence
 84 from keras_preprocessing.sequence import pad_sequences
 85
 86 6.0.1 Lets find most freq
 87
 88 7 Preprocessing For Modelin
 89
 90 8 Models
```

To [8.1.1] Model Evaluations

executed in 8ms, finished 10:07:22 2022-08-16
8.2.1 Multinomial Naive Bayes

3 Obtain

This project utilises 3 data sources:
1. Market Share
2. Product Categories
3. Customer Demographics

[Historical](#) health and hygiene inspections recorded by New York City Department of Health and

▼ 9 Conclusions Mental Hygiene (DOHMH) public health inspectors

9.1 Best Model Results API business data and ratings

9.2 Next Steps: Help users generate reviews

9.3.1 Word2Vec Embedds

9.3.2 Classification With New York City has required restaurants to re-

9.3.3 Pretrained Vector Model

Received from sanitary inspections. Since 2010, New York City has required restaurants to post letter grade Word-Ember. www1.nyc.gov/html/doh/html/grades/grades.shtml

9.3.4 MCG grades that correspond to scores received from sanitary inspections.

- "A" grade: 0 to 13 points for sanitary violations
 - "B" grade: 14 to 27 points for sanitary violations

- "C" grade: 28 or more points for sanitary violations

3.1 Obtaining Restaurant Inspection Results from NYC Open Data Portal

Contents

| | CAMIS | DBA | BORO | BUILDING | STREET | ZIPCODE | PHONE | DESC |
|-------------------------------------|---|----------|-----------|----------|----------|----------|------------|------|
| 1 Predicting NYC Restaurant | | | | | | | | |
| 2 Introduction | The dataset can be obtained here | | | | | | | |
| 2.1 Objective | https://data.cityofnewyork.us/Health/DOHMH-New-York-City-Restaurant-Inspection-Results/43nn-pn8j | | | | | | | |
| 2.1.1 Imports | | | | | | | | |
| 3 Obtain | https://data.cityofnewyork.us/Health/DOHMH-New-York-City-Restaurant-Inspection-Results/43nn-pn8j | | | | | | | |
| 3.1 Obtaining Restaurant Data | | | | | | | | |
| 3.1.1 Understanding NYC | | | | | | | | |
| 3.1.2 Target Variable | The dataset was downloaded and saved to this repository. Let's load it in and explore its contents. | | | | | | | |
| 3.1.3 Engineering Target | | | | | | | | |
| 3.2 Obtaining Yelp Business Data | Detailed descriptions about each column can be found in the Restaurant Inspection Data Dictionary. | | | | | | | |
| 3.2.1 Yelp Business Seal | | | | | | | | |
| 3.2.2 Yelp Reviews | | | | | | | | |
| 3.2.3 Joining Yelp Reviews | | | | | | | | |
| 3.3 Joining NYC DOHMH Data | <code>doch_df = pd.read_csv('data/nyc_open_data/DOHMH_New_York_City_Restaurant_Inspection_Results.csv')</code> | | | | | | | |
| 4 Scrubbing The Data | | | | | | | | |
| 4.1 Drop businesses missing | executed in 2.83s, finished 15:02:00 2022-08-12 | | | | | | | |
| 5 Exploring The Dataset | | | | | | | | |
| Out[4]: | | | | | | | | |
| 6 Preprocessing For Further | | | | | | | | |
| 6.0.1 Lets find most freq | | | | | | | | |
| 7 Preprocessing For Modeling | | | | | | | | |
| 8 Models | | | | | | | | |
| 8.1 Bag-of-Words Model | | | | | | | | |
| 8.1.1 Model Evaluations | | | | | | | | |
| 8.2 Text Preprocessing Pipelines | | | | | | | | |
| 8.2.1 Multinomial Naive Bayes | 0 50008319 | POULETTE | Manhattan | 790 | 9 AVENUE | 10019.00 | 2129569488 | |
| 8.2.2 Tuned Multinomial | | | | | | | | |
| 8.2.3 Logistic Regression | | | | | | | | |
| 8.2.4 Tuned Logistic Regression | | | | | | | | |
| 8.2.5 Decision Tree Model | | | | | | | | |
| 8.2.6 Tuned DecisionTree Model | | | | | | | | |
| 8.2.7 Support Vector Classification | | | | | | | | |
| 8.2.8 Tuned SupportVector Model | | | | | | | | |
| 9 Conclusions | | | | | | | | |
| 9.1 Best Model Results | 14678734 TINY'S DINER Bronx 3603 RIVERDALE AVENUE 10463.00 7187087600 | | | | | | | |
| 9.2 Next Steps and Future | | | | | | | | |
| 9.3 Appendix - Deep NLP | The dataset contains 186,227 inspection results. However, when an inspection results in more than one violation, values for associated fields are repeated for each additional violation record. So let's check how many individual restaurants are in the dataset. | | | | | | | |
| 9.3.1 Word2Vec Embedding | | | | | | | | |
| 9.3.2 Classifying With Embedding | | | | | | | | |
| 9.3.3 Pretrained Vector Embedding | | | | | | | | |
| 9.3.4 Mean Word Embedding | | | | | | | | |

3.1.1 Understanding NYC DOHMH Data

```
In [5]: 1 # How many unique restaurants are in this dataset?
2 n_unique = doh_df['CAMIS'].nunique()
3 print(f'There are {n_unique} unique restaurants in the dataset. ')
```

executed in 28ms, finished 15:02:00 2022-08-12

There are 19792 unique restaurants in the dataset.

Contents ⚙️

| | |
|---|---|
| 1 In [6]: | # Get more information about the dataset contents |
| ▼ 2 Introduction | 2 doh_df.info() |
| ▼ 2.1 Objective | executed in 1.10s, finished 15:02:01 2022-08-12 |
| 2.1.1 Imports | |
| ▼ 3 Obtain | <class 'pandas.core.frame.DataFrame'> |
| 3.1 Obtaining Restaurant Data | RangeIndex: 186227 entries, 0 to 186226 Data columns (total 26 columns): |
| 3.1.1 Understanding NYC Column | Non-Null Count Dtype |
| 3.1.2 Target Variable | ----- |
| 3.1.3 Engineering Target | 186227 non-null int64 |
| 3.2 Obtaining Yelp Businesses | 186200 non-null object |
| 3.2.1 Yelp Business Search | 186227 non-null object |
| 3.2.2 Yelp Reviews | 185799 non-null object |
| 3.2.3 Joining Yelp and DOB | 186227 non-null object |
| 3.3 Joining NYC DOT and DOB | 183146 non-null float64 |
| ▼ 4 Scrubbing The Data | 186214 non-null object |
| 4.1 Drop businesses missing CUISINE DESCRIPTION | 186226 non-null object |
| 5 Exploring The Dataset | 8 INSPECTION DATE |
| 6 Preprocessing For Further Analysis | 9 ACTION |
| 6.0.1 Lets find most freq VIOLATION CODE | 10 VIOLATION DESCRIPTION |
| 7 Preprocessing For Modeling | 11 VIOLATION DESCRIPTION |
| ▼ 8 Models | 12 CRITICAL FLAG |
| 8.1 Bag-of-Words Model | 13 SCORE |
| 8.1.1 Model Evaluations | 186227 non-null int64 |

```
In [7]: 1 doh_df['ZIPCODE'].astype(str)
```

▼ 8.2 Text Preprocessing Pipelines

executed in 224ms, finished 15:02:02 2022-08-12

| | |
|----------------------------------|--|
| Ou8.2.1: 8.2.1 Tuned Multinomial | 10019.0 |
| 8.2.3 Logistic Regressi | 10463.0 |
| 8.2.4 Tuned Logistic Re | 10022.0 |
| 8.2.5 Decision Tree Mod | 10022.0 |
| 8.2.6 Tuned DecisionTre | 11368.0 |
| 8.2.7 Support Vector Clas | ... |
| 8.2.8 Tuned SupportVec | 186222 nan |
| 186223 | 10019.0 |
| ▼ 9 Conclusions | 186224 10036.0 |
| 9.1 Best Model Results | 186225 10472.0 |
| 9.2 Next Steps and Future | 186226 10461.0 |
| ▼ 9.3 Appendix | Name: ZIPCODE, Length: 186227, dtype: object |
| 9.3.1 Word2Vec Embedd | |
| 9.3.2 Classifying With Er | |
| 9.3.3 Pretrained Vector N | |
| 9.3.4 Mean Word Embed | |

```
In [8]: 1 doh_df['Community Board'].astype(str)
```

executed in 207ms, finished 15:02:02 2022-08-12

```
Out[8]: 0      104.0
        1      208.0
        2      105.0
        3      105.0
```

Contents ↗ 4.0

- 1 Predicting NYC Restaurants ..
- ▼ 2 Introduction 186222 nan
- ▼ 2.1 Objective 186223 105.0
 - 2.1.1 Imports 186224 105.0
- ▼ 3 Obtain 186225 209.0
- ▼ 3.1 Obtaining Restaurant Info 186226 210.0
 - Name: Community Board, Length: 186227, dtype: object
 - 3.1.1 Understanding NYC 186227
 - 3.1.2 Target Variable -- N 186228

```
In [8]: 1 doh_df['Council District'].astype(str)
```

- ▼ 3.2 Obtaining Yelp Businesses 186228
 - executed in 188ms, finished 15:02:02 2022-08-12
 - 3.2.1 Yelp Business Search 186229

```
Out[9]: 0      3.0
        1      11.0
        2      4.0
        3      4.0
```

- ▼ 4 Scrubbing The Data 186229 21.0
 - 4.1 Drop businesses missing 186230
- 5 Exploring The Dataset 186231 nan
- ▼ 6 Preprocessing 186232 Further 4.0
 - 6.0.1 Lets 186233 Post freq 4.0
- 7 Preprocessing 186234 Model 18.0
- ▼ 8 Models 186226 13.0
 - Name: Council District, Length: 186227, dtype: object
 - 8.1 Bag-of-Words Model 186228
 - 8.1.1 Model Evaluations 186229

```
In [8]: 1 doh_df['Census Tract'].astype(str)
```

- 8.2.1 Multinomial Naive Bayes 186230
 - executed in 192ms, finished 15:02:02 2022-08-12
 - 8.2.2 Tuned Multinomial

```
Out[10]: 0      13300.0
        1      29500.0
        2      11202.0
        3      11202.0
        4      43702.0
        5      ...
        6      ...
        7      ...
        8      ...
        9      ...
        10     ...
        11     ...
        12     ...
        13     ...
        14     ...
        15     ...
        16     ...
        17     ...
        18     ...
        19     ...
        20     ...
        21     ...
        22     ...
        23     ...
        24     ...
        25     ...
        26     ...
        27     ...
        28     ...
        29     ...
        30     ...
        31     ...
        32     ...
        33     ...
        34     ...
        35     ...
        36     ...
        37     ...
        38     ...
        39     ...
        40     ...
        41     ...
        42     ...
        43     ...
        44     ...
        45     ...
        46     ...
        47     ...
        48     ...
        49     ...
        50     ...
        51     ...
        52     ...
        53     ...
        54     ...
        55     ...
        56     ...
        57     ...
        58     ...
        59     ...
        60     ...
        61     ...
        62     ...
        63     ...
        64     ...
        65     ...
        66     ...
        67     ...
        68     ...
        69     ...
        70     ...
        71     ...
        72     ...
        73     ...
        74     ...
        75     ...
        76     ...
        77     ...
        78     ...
        79     ...
        80     ...
        81     ...
        82     ...
        83     ...
        84     ...
        85     ...
        86     ...
        87     ...
        88     ...
        89     ...
        90     ...
        91     ...
        92     ...
        93     ...
        94     ...
        95     ...
        96     ...
        97     ...
        98     ...
        99     ...
        100    ...
        101    ...
        102    ...
        103    ...
        104    ...
        105    ...
        106    ...
        107    ...
        108    ...
        109    ...
        110    ...
        111    ...
        112    ...
        113    ...
        114    ...
        115    ...
        116    ...
        117    ...
        118    ...
        119    ...
        120    ...
        121    ...
        122    ...
        123    ...
        124    ...
        125    ...
        126    ...
        127    ...
        128    ...
        129    ...
        130    ...
        131    ...
        132    ...
        133    ...
        134    ...
        135    ...
        136    ...
        137    ...
        138    ...
        139    ...
        140    ...
        141    ...
        142    ...
        143    ...
        144    ...
        145    ...
        146    ...
        147    ...
        148    ...
        149    ...
        150    ...
        151    ...
        152    ...
        153    ...
        154    ...
        155    ...
        156    ...
        157    ...
        158    ...
        159    ...
        160    ...
        161    ...
        162    ...
        163    ...
        164    ...
        165    ...
        166    ...
        167    ...
        168    ...
        169    ...
        170    ...
        171    ...
        172    ...
        173    ...
        174    ...
        175    ...
        176    ...
        177    ...
        178    ...
        179    ...
        180    ...
        181    ...
        182    ...
        183    ...
        184    ...
        185    ...
        186    ...
        187    ...
        188    ...
        189    ...
        190    ...
        191    ...
        192    ...
        193    ...
        194    ...
        195    ...
        196    ...
        197    ...
        198    ...
        199    ...
        200    ...
        201    ...
        202    ...
        203    ...
        204    ...
        205    ...
        206    ...
        207    ...
        208    ...
        209    ...
        210    ...
        211    ...
        212    ...
        213    ...
        214    ...
        215    ...
        216    ...
        217    ...
        218    ...
        219    ...
        220    ...
        221    ...
        222    ...
        223    ...
        224    ...
        225    ...
        226    ...
        227    ...
        228    ...
        229    ...
        230    ...
        231    ...
        232    ...
        233    ...
        234    ...
        235    ...
        236    ...
        237    ...
        238    ...
        239    ...
        240    ...
        241    ...
        242    ...
        243    ...
        244    ...
        245    ...
        246    ...
        247    ...
        248    ...
        249    ...
        250    ...
        251    ...
        252    ...
        253    ...
        254    ...
        255    ...
        256    ...
        257    ...
        258    ...
        259    ...
        260    ...
        261    ...
        262    ...
        263    ...
        264    ...
        265    ...
        266    ...
        267    ...
        268    ...
        269    ...
        270    ...
        271    ...
        272    ...
        273    ...
        274    ...
        275    ...
        276    ...
        277    ...
        278    ...
        279    ...
        280    ...
        281    ...
        282    ...
        283    ...
        284    ...
        285    ...
        286    ...
        287    ...
        288    ...
        289    ...
        290    ...
        291    ...
        292    ...
        293    ...
        294    ...
        295    ...
        296    ...
        297    ...
        298    ...
        299    ...
        300    ...
        301    ...
        302    ...
        303    ...
        304    ...
        305    ...
        306    ...
        307    ...
        308    ...
        309    ...
        310    ...
        311    ...
        312    ...
        313    ...
        314    ...
        315    ...
        316    ...
        317    ...
        318    ...
        319    ...
        320    ...
        321    ...
        322    ...
        323    ...
        324    ...
        325    ...
        326    ...
        327    ...
        328    ...
        329    ...
        330    ...
        331    ...
        332    ...
        333    ...
        334    ...
        335    ...
        336    ...
        337    ...
        338    ...
        339    ...
        340    ...
        341    ...
        342    ...
        343    ...
        344    ...
        345    ...
        346    ...
        347    ...
        348    ...
        349    ...
        350    ...
        351    ...
        352    ...
        353    ...
        354    ...
        355    ...
        356    ...
        357    ...
        358    ...
        359    ...
        360    ...
        361    ...
        362    ...
        363    ...
        364    ...
        365    ...
        366    ...
        367    ...
        368    ...
        369    ...
        370    ...
        371    ...
        372    ...
        373    ...
        374    ...
        375    ...
        376    ...
        377    ...
        378    ...
        379    ...
        380    ...
        381    ...
        382    ...
        383    ...
        384    ...
        385    ...
        386    ...
        387    ...
        388    ...
        389    ...
        390    ...
        391    ...
        392    ...
        393    ...
        394    ...
        395    ...
        396    ...
        397    ...
        398    ...
        399    ...
        400    ...
        401    ...
        402    ...
        403    ...
        404    ...
        405    ...
        406    ...
        407    ...
        408    ...
        409    ...
        410    ...
        411    ...
        412    ...
        413    ...
        414    ...
        415    ...
        416    ...
        417    ...
        418    ...
        419    ...
        420    ...
        421    ...
        422    ...
        423    ...
        424    ...
        425    ...
        426    ...
        427    ...
        428    ...
        429    ...
        430    ...
        431    ...
        432    ...
        433    ...
        434    ...
        435    ...
        436    ...
        437    ...
        438    ...
        439    ...
        440    ...
        441    ...
        442    ...
        443    ...
        444    ...
        445    ...
        446    ...
        447    ...
        448    ...
        449    ...
        450    ...
        451    ...
        452    ...
        453    ...
        454    ...
        455    ...
        456    ...
        457    ...
        458    ...
        459    ...
        460    ...
        461    ...
        462    ...
        463    ...
        464    ...
        465    ...
        466    ...
        467    ...
        468    ...
        469    ...
        470    ...
        471    ...
        472    ...
        473    ...
        474    ...
        475    ...
        476    ...
        477    ...
        478    ...
        479    ...
        480    ...
        481    ...
        482    ...
        483    ...
        484    ...
        485    ...
        486    ...
        487    ...
        488    ...
        489    ...
        490    ...
        491    ...
        492    ...
        493    ...
        494    ...
        495    ...
        496    ...
        497    ...
        498    ...
        499    ...
        500    ...
        501    ...
        502    ...
        503    ...
        504    ...
        505    ...
        506    ...
        507    ...
        508    ...
        509    ...
        510    ...
        511    ...
        512    ...
        513    ...
        514    ...
        515    ...
        516    ...
        517    ...
        518    ...
        519    ...
        520    ...
        521    ...
        522    ...
        523    ...
        524    ...
        525    ...
        526    ...
        527    ...
        528    ...
        529    ...
        530    ...
        531    ...
        532    ...
        533    ...
        534    ...
        535    ...
        536    ...
        537    ...
        538    ...
        539    ...
        540    ...
        541    ...
        542    ...
        543    ...
        544    ...
        545    ...
        546    ...
        547    ...
        548    ...
        549    ...
        550    ...
        551    ...
        552    ...
        553    ...
        554    ...
        555    ...
        556    ...
        557    ...
        558    ...
        559    ...
        560    ...
        561    ...
        562    ...
        563    ...
        564    ...
        565    ...
        566    ...
        567    ...
        568    ...
        569    ...
        570    ...
        571    ...
        572    ...
        573    ...
        574    ...
        575    ...
        576    ...
        577    ...
        578    ...
        579    ...
        580    ...
        581    ...
        582    ...
        583    ...
        584    ...
        585    ...
        586    ...
        587    ...
        588    ...
        589    ...
        590    ...
        591    ...
        592    ...
        593    ...
        594    ...
        595    ...
        596    ...
        597    ...
        598    ...
        599    ...
        600    ...
        601    ...
        602    ...
        603    ...
        604    ...
        605    ...
        606    ...
        607    ...
        608    ...
        609    ...
        610    ...
        611    ...
        612    ...
        613    ...
        614    ...
        615    ...
        616    ...
        617    ...
        618    ...
        619    ...
        620    ...
        621    ...
        622    ...
        623    ...
        624    ...
        625    ...
        626    ...
        627    ...
        628    ...
        629    ...
        630    ...
        631    ...
        632    ...
        633    ...
        634    ...
        635    ...
        636    ...
        637    ...
        638    ...
        639    ...
        640    ...
        641    ...
        642    ...
        643    ...
        644    ...
        645    ...
        646    ...
        647    ...
        648    ...
        649    ...
        650    ...
        651    ...
        652    ...
        653    ...
        654    ...
        655    ...
        656    ...
        657    ...
        658    ...
        659    ...
        660    ...
        661    ...
        662    ...
        663    ...
        664    ...
        665    ...
        666    ...
        667    ...
        668    ...
        669    ...
        670    ...
        671    ...
        672    ...
        673    ...
        674    ...
        675    ...
        676    ...
        677    ...
        678    ...
        679    ...
        680    ...
        681    ...
        682    ...
        683    ...
        684    ...
        685    ...
        686    ...
        687    ...
        688    ...
        689    ...
        690    ...
        691    ...
        692    ...
        693    ...
        694    ...
        695    ...
        696    ...
        697    ...
        698    ...
        699    ...
        700    ...
        701    ...
        702    ...
        703    ...
        704    ...
        705    ...
        706    ...
        707    ...
        708    ...
        709    ...
        710    ...
        711    ...
        712    ...
        713    ...
        714    ...
        715    ...
        716    ...
        717    ...
        718    ...
        719    ...
        720    ...
        721    ...
        722    ...
        723    ...
        724    ...
        725    ...
        726    ...
        727    ...
        728    ...
        729    ...
        730    ...
        731    ...
        732    ...
        733    ...
        734    ...
        735    ...
        736    ...
        737    ...
        738    ...
        739    ...
        740    ...
        741    ...
        742    ...
        743    ...
        744    ...
        745    ...
        746    ...
        747    ...
        748    ...
        749    ...
        750    ...
        751    ...
        752    ...
        753    ...
        754    ...
        755    ...
        756    ...
        757    ...
        758    ...
        759    ...
        760    ...
        761    ...
        762    ...
        763    ...
        764    ...
        765    ...
        766    ...
        767    ...
        768    ...
        769    ...
        770    ...
        771    ...
        772    ...
        773    ...
        774    ...
        775    ...
        776    ...
        777    ...
        778    ...
        779    ...
        780    ...
        781    ...
        782    ...
        783    ...
        784    ...
        785    ...
        786    ...
        787    ...
        788    ...
        789    ...
        790    ...
        791    ...
        792    ...
        793    ...
        794    ...
        795    ...
        796    ...
        797    ...
        798    ...
        799    ...
        800    ...
        801    ...
        802    ...
        803    ...
        804    ...
        805    ...
        806    ...
        807    ...
        808    ...
        809    ...
        810    ...
        811    ...
        812    ...
        813    ...
        814    ...
        815    ...
        816    ...
        817    ...
        818    ...
        819    ...
        820    ...
        821    ...
        822    ...
        823    ...
        824    ...
        825    ...
        826    ...
        827    ...
        828    ...
        829    ...
        830    ...
        831    ...
        832    ...
        833    ...
        834    ...
        835    ...
        836    ...
        837    ...
        838    ...
        839    ...
        840    ...
        841    ...
        842    ...
        843    ...
        844    ...
        845    ...
        846    ...
        847    ...
        848    ...
        849    ...
        850    ...
        851    ...
        852    ...
        853    ...
        854    ...
        855    ...
        856    ...
        857    ...
        858    ...
        859    ...
        860    ...
        861    ...
        862    ...
        863    ...
        864    ...
        865    ...
        866    ...
        867    ...
        868    ...
        869    ...
        870    ...
        871    ...
        872    ...
        873    ...
        874    ...
        875    ...
        876    ...
        877    ...
        878    ...
        879    ...
        880    ...
        881    ...
        882    ...
        883    ...
        884    ...
        885    ...
        886    ...
        887    ...
        888    ...
        889    ...
        890    ...
        891    ...
        892    ...
        893    ...
        894    ...
        895    ...
        896    ...
        897    ...
        898    ...
        899    ...
        900    ...
        901    ...
        902    ...
        903    ...
        904    ...
        905    ...
        906    ...
        907    ...
        908    ...
        909    ...
        910    ...
        911    ...
        912    ...
        913    ...
        914    ...
        915    ...
        916    ...
        917    ...
        918    ...
        919    ...
        920    ...
        921    ...
        922    ...
        923    ...
        924    ...
        925    ...
        926    ...
        927    ...
        928    ...
        929    ...
        930    ...
        931    ...
        932    ...
        933    ...
        934    ...
        935    ...
        936    ...
        937    ...
        938    ...
        939    ...
        940    ...
        941    ...
        942    ...
        943    ...
        944    ...
        945    ...
        946    ...
        947    ...
        948    ...
        949    ...
        950    ...
        951    ...
        952    ...
        953    ...
        954    ...
        955    ...
        956    ...
        957    ...
        958    ...
        959    ...
        960    ...
        961    ...
        962    ...
        963    ...
        964    ...
        965    ...
        966    ...
        967    ...
        968    ...
        969    ...
        970    ...
        971    ...
        972    ...
        973    ...
        974    ...
        975    ...
        976    ...
        977    ...
        978    ...
        979    ...
        980    ...
        981    ...
        982    ...
        983    ...
        984    ...
        985    ...
        986    ...
        987    ...
        988    ...
        989    ...
        990    ...
        991    ...
        992    ...
        993    ...
        994    ...
        995    ...
        996    ...
        997    ...
        998    ...
        999    ...
        1000    ...
        1001    ...
        1002    ...
        1003    ...
        1004    ...
        1005    ...
        1006    ...
        1007    ...
        1008    ...
        1009    ...
        1010    ...
        1011    ...
        1012    ...
        1013    ...
        1014    ...
        1015    ...
        1016    ...
        1017    ...
        1018    ...
        1019    ...
        1020    ...
        1021    ...
        1022    ...
        1023    ...
        1024    ...
        1025    ...
        1026    ...
        1027    ...
        1028    ...
        1029    ...
        1030    ...
        1031    ...
        1032    ...
        1033    ...
        1034    ...
        1035    ...
        1036    ...
        1037    ...
        1038    ...
        1039    ...
        1040    ...
        1041    ...
        1042    ...
        1043    ...
        1044    ...
        1045    ...
        1046    ...
        1047    ...
        1048    ...
        1049    ...
        1050    ...
        1051    ...
        1052    ...
        1053    ...
        1054    ...
        1055    ...
        1056    ...
        1057    ...
        1058    ...
        1059    ...
        1060    ...
        1061    ...
        1062    ...
        1063    ...
        1064    ...
        1065    ...
        1066    ...
        1067    ...
        1068    ...
        1069    ...
        1070    ...
        1071    ...
        1072    ...
        1073    ...
        1074    ...
        1075    ...
        1076    ...
        1077    ...
        1078    ...
        1079    ...
        1080    ...
        1081    ...
        1082    ...
        1083    ...
        1084    ...
        1085    ...
        1086    ...
        1087    ...
        1088    ...
        1089    ...
        1090    ...
        1091    ...
        1092    ...
        1093    ...
        1094    ...
        1095    ...
        1096    ...
        1097    ...
        1098    ...
        1099    ...
        1100    ...
        1101    ...
        1102    ...
        1103    ...
        1104    ...
        1105    ...
        1106    ...
        1107    ...
        1108    ...
        1109    ...
        1110    ...
        1111    ...
        1112    ...
        1113    ...
        1114    ...
        1115    ...
        1116    ...
        1117    ...
        1118    ...
        1119    ...
        1120    ...
        1121    ...
        1122    ...
        1123    ...
        1124    ...
        1125    ...
        1126    ...
        1127    ...
        1128    ...
        1129    ...
        1130    ...
        1131    ...
        1132    ...
        1133    ...
        1134    ...
        1135    ...
        1136    ...
        1137    ...
        1138    ...
        1139    ...
        1140    ...
        1141    ...
        1142    ...
        1143    ...
        1144    ...
        1145    ...
        1146    ...
        1147    ...
        1148    ...
        1149    ...
        1150    ...
        1151    ...
        1152    ...
        1153    ...
        1154    ...
        1155    ...
        1156    ...
        1157    ...
        1158    ...
        1159    ...
        1160    ...
        1161    ...
        1162    ...
        1163    ...
        1164    ...
        1165    ...
        1166    ...
        1167    ...
        1168    ...
        1169    ...
        1170    ...
        1171    ...
        1172    ...
        1173    ...
        1174    ...
        1175    ...
        1176    ...
        1177    ...
        1178    ...
        1179    ...
        1180    ...
        1181    ...
        1182    ...
        1183    ...
        1184    ...
        1185    ...
        1186    ...
        1187    ...
        1188    ...
        1189    ...
        1190    ...
        1191    ...
        1192    ...
        1193    ...
        1194    ...
        1195    ...
        1196    ...
        1197    ...
        1198    ...
        1199    ...
        1200    ...
        1201    ...
        1202    ...
        1203    ...
        1204    ...
        1205    ...
        1206    ...
        1207    ...
        1208    ...
        1209    ...
        1210    ...
        1211    ...
        1212    ...
        1213    ...
        1214    ...
        1215    ...
        1216    ...
        1217    ...
        1218    ...
        1219    ...
        1220    ...
        1221    ...
        1222    ...
        1223    ...
        1224    ...
        1225    ...
        1226    ...
        1227    ...
        1228    ...
        1229    ...
        1230    ...
        1231    ...
        1232    ...
        1233    ...
        1234    ...
        1235    ...
        1236    ...
        1237    ...
        1238    ...
        1239    ...
        1240    ...
        1241    ...
        1242    ...
        1243    ...
        1244    ...
        1245    ...
        1246    ...
        1247    ...
        1248    ...
        1249    ...
        1250    ...
        1251    ...
        1252    ...
        1253    ...
        1254    ...
        1255    ...
        1256    ...
        1257    ...
        1258    ...
        1259    ...
        1260    ...
        1261    ...
        1262    ...
        1263    ...
        1264    ...
        1265    ...
        1266    ...
        1267    ...
        1268    ...
        1269    ...
        1270    ...
        1271    ...
        1272    ...
        1273    ...
        1274    ...
        1275    ...
        1276    ...
        1277    ...
        1278    ...
        1279    ...
        1280    ...
        1281    ...
        1282    ...
        1283    ...
        1284    ...
        1285    ...
        1286    ...
        1287    ...
        1288    ...
        1289    ...
        1290    ...
        1291    ...
        1292    ...
        1293    ...
        1294    ...
        1295    ...
        1296    ...
        1297    ...
        1298    ...
        1299    ...
        1300    ...
        1301    ...
        1302    ...
        1303    ...
        1304    ...
        1305    ...
        1306    ...
        1307    ...
        1308    ...
        1309    ...
        1310    ...
        1311    ...
        1312    ...
        1313    ...
        1314    ...
        1315    ...
        1316    ...
        1317    ...
        1318    ...
        1319    ...
        1320    ...
        1321    ...
        1322    ...
        1323    ...
        1324    ...
        1325    ...
        1326    ...
        1327    ...
        1328    ...
        1329    ...
        1330    ...
        1331    ...
        1332    ...
        1333    ...
        1334    ...
        1335    ...
        1336    ...
        1337    ...
        1338    ...
        1339    ...
        1340    ...
        1341    ...
        1342    ...
        1343    ...
        1344    ...
        1345    ...
        1346    ...
        1347    ...
        1348    ...
        1349    ...
        1350    ...
        1351    ...
        1352    ...
        1353    ...
        1354    ...
        1355    ...
        1356    ...
        1357    ...
        1358    ...
        1359    ...
        1360    ...
        1361    ...
        1362    ...
        1363    ...
        1364    ...
        1365    ...
        1366    ...
        1367    ...
        1368    ...
        1369    ...
        1370    ...
        1371    ...
        1372    ...
        1373    ...
        1374    ...
        1375    ...
        1376    ...
        1377    ...
        1378    ...
        1379    ...
        1380    ...
        1381    ...
        1382    ...
        1383    ...
        1384    ...
        1385    ...
        1386    ...
        1387    ...
        1388    ...
        1389    ...
        1390    ...
        13
```

```
In [11]: 1 print(f'There are {doh_df.duplicated(keep=False).sum()} duplicated rows')
executed in 812ms, finished 15:02:03 2022-08-12
```

There are 22010 duplicated rows.

```
In [12]: 1 # Let's drop these duplicated rows
2 doh_df.drop_duplicates(keep='first', inplace=True)
```

Contents

1 Predicting NYC Restaurant executed in 894ms, finished 15:02:04 2022-08-12

▼ 2 Introduction

```
In [13]: 1 # Confirming duplicates have been removed
2 doh_df.shape
```

▼ 3 Obtain executed in 18ms, finished 15:02:04 2022-08-12

Out[13]: (174725, 26)

3.1.1 Understanding NYC

3.1.2 Target Variable -- N

3.1.3 Engineering Target Since this project will be leveraging data publicly generated from social media a lookup value will be needed to call the API and join the tables. The Yelp API has an endpoint for Phone Search. This

▼ 3.2 Obtaining Yelp Business will allow us to pull Yelp business data for each restaurant by providing a telephone number. More information can be found in the [documentation here](#).

3.2.3 Joining (https://yelp3p.com/developers/documentation/v3/business_search_phone)

3.3 Joining NYC DOHMH {

▼ 4 Scrubbing The Data

```
In [14]: 1 # Checking that every restaurant has a phone number
2 missing_num = doh_df['PHONE'].isna().sum()
```

5 Exploring The Dataset print(f'There are {missing_num} restaurants missing a telephone number.')

▼ 6 Preprocessing For Further executed in 32ms, finished 15:02:04 2022-08-12

6.0.1 Lets find most freq

There are 13 restaurants missing a telephone number.

▼ 7 Preprocessing For Modelin

▼ 8 Models

▼ 8.1 Bag-of-Words # Since only 13 numbers are missing, these rows can be dropped

8.1.1 ModelEvaluation.dropna(subset=['PHONE'], inplace=True)

▼ 8.2 Text Preprocessing Pic

executed in 922ms, finished 15:02:04 2022-08-12

8.2.1 Multinomial Naive I

In [8.2.2] Tuned Multinomial

8.2.3 Logistic Regression PHONE].isna().sum()

8.2.4 Tuned Logistic Rec

executed in 27ms, finished 15:02:04 2022-08-12

8.2.5 Decision Tree Mod

Out[16]: 0

8.2.6 Tuned DecisionTre

8.2.7 Support Vector Cla

In [8.2.7.8] Tuned Support Vector

How many unique restaurants are remaining?

▼ 9 Conclusions n_unique = doh_df['CAMIS'].nunique()

9.1 Best Model Results print(f'There are {n_unique} unique restaurants remaining in the database')

9.2 Next Steps and Future executed in 1ms, finished 15:02:04 2022-08-12

▼ 9.3 Appendix - Deep NLP There are 19790 unique restaurants remaining in the dataset.

9.3.1 Word2Vec Embedds

9.3.2 Classifying With Er

Let's explore the date range for this dataset.

9.3.3 Pretrained Vector

9.3.4 Mean Word Embed

```
In [18]: 1 doh_df['INSPECTION DATE'] = pd.to_datetime(doh_df['INSPECTION DATE'])
2 begin_date = doh_df['INSPECTION DATE'].min()
3 end_date = doh_df['INSPECTION DATE'].max()
4 print(f'The data ranges from {begin_date} to {end_date}')
```

executed in 348ms, finished 15:02:05 2022-08-12

The data ranges from 2009-05-16 00:00:00 to 2022-03-23 00:00:00

Contents ⚙

- 1 Predicting NYC Restaurant Inspections in this dataset range from May 2009 up to March 2022.

- 2 Introduction

- 2.1 Objective

- 2.1.1 Imports

3.1.2 Target Variable -- NYCDOH Inspection Grades

- 3 Obtain

3.1 Obtaining Restaurant Info

Health Code Violations found during an inspection carries a point value, and a restaurant's score corresponds to a letter grade. A lower point score leads to a better letter grade:

- 3.1.1 Under Health Code Violations
- 3.1.2 Target Variable: Score
- 3.1.3 Engineering Target

3.2 Obtaining Yelp Grade

- 3.2.1 Yelp Business Grade: 0 to 13 points for sanitary violations
- 3.2.2 Yelp Reviews Grade: 14 to 27 points for sanitary violations
- 3.2.3 Joining Yelp Reviews
- 3.3 Joining NYC DOHMH Data

- 4 Cleaning The Data *Let see what the score distribution is*

```
4.1 Drop business_id: doh_df['SCORE'].hist(bins=113, figsize=(12,6));
```

- 5 Exploring The Dataset

executed in 1.00s, finished 15:02:06 2022-08-12

- 6 Preprocessing For Further

- 6.0.1 Lets find most freq

- 7 Preprocessing For Modeling

- 8 Models

- 8.1 Bag-of-Words Model

- 8.1.1 Model Evaluations

- 8.2 Text Preprocessing Pic

- 8.2.1 Multinomial Naive Bayes

- 8.2.2 Tuned Multinomial

- 8.2.3 Logistic Regression

- 8.2.4 Tuned Logistic Reg

- 8.2.5 Decision Tree Model

- 8.2.6 Tuned DecisionTree

- 8.2.7 Support Vector Classifier

- 8.2.8 Tuned SupportVec

- 9 Conclusions

- 9.1 Best Model Results

- 9.2 Next Steps and Future

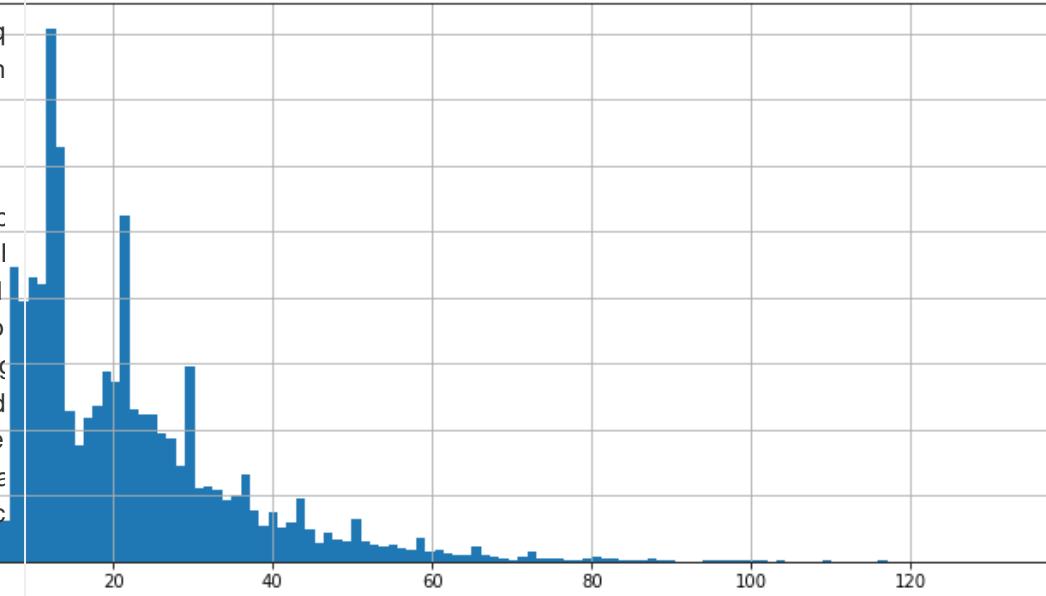
- 9.3 Appendix - Deep NLP

- 9.3.1 Word2Vec Embedding

- 9.3.2 Classifying With Embedding

- 9.3.3 Pretrained Vector Model

- 9.3.4 Mean Word Embedding



```
In [20]: 1 doh_df['SCORE'].describe()
```

executed in 42ms, finished 15:02:06 2022-08-12

```
Out[20]: count    174712.00
```

mean 21.36

std 14.36

min 0.00

25% 12.00

50% 18.00

75% 27.00

max 131.00

Name: SCORE, dtype: float64

2.1.1 Imports

Contents

1 Predicting NYC Restaurants

2 Introduction

2.1 Objective

2.1.1 Imports

3 Obtain

3.1 Obtaining Restaurant Inspections

3.1.1 Understanding NYC

3.1.2 Target Variable -- N

3.1.3 Engineering Target

Because the inspections dataset contains many duplicated businesses in its records, this project

3.2 Obtaining Yelp Businesses

3.2.1 Yelp Business Search

Will group records by businesses and get a count for each for their letter grades. The target variable

will be finding restaurants that have received 1 or more C Grades.

3.2.2 Yelp Reviews

3.2.3 Joining Yelp Reviews

```
In [21]: 1 # Copying to a new DataFrame
```

3.3 Joining NYC DOHMH Data

2 doh_graded = doh_df.copy()

4 Scrubbing The Data

4.1 Drop businesses

executed in 52ms, finished 15:02:06 2022-08-12

5 Exploring The Dataset

6 Preprocessing For Further

6.0.1 Lets find most freq

7 Preprocessing For Modeling

7.0.1 'VIOLATION DESCRIPTION', 'CRITICAL FLAG', 'GRADE',

7.0.2 'GRADE DATE', 'RECORD DATE', 'INSPECTION TYPE', inplace=True)

8 Models

8.1 Bag-of-Words Model

8.1.1 Model Evaluations

```
In [22]: # Creating new columns with NYCDOH Grades
```

8.2.1 Multinomial Naive

8.2.2 Tuned Multinomial

8.2.3 Logistic Regression

8.2.4 Tuned Logistic Reg

8.2.5 Decision Tree Model

```
In [23]: # Creating a new DF and grouping businesses and summing their Grade counts
```

8.2.7 Support Vector Clas

8.2.8 Tuned SupportVec

9 Conclusions

9.1 Best Model Results

9.2 Next Steps and Future

9.3 Appendix - Deep NLP

9.3.1 Word2Vec Embedds

```
In [24]: 1 # Checking the number of businesses that have received 1 or more B Grades
```

9.3.2 Classifying With Er

9.3.3 Pretrained Vector

9.3.4 Mean WORD EMOE

executed in 12ms, finished 15:02:09 2022-08-12

```
Out[25]: 9977
```

Of the 19,790 unique restaurants, 9,977 failed an initial cycle inspection at least once.

```
In [26]: 1 # Checking the number of businesses that have received 1 or more C Grade
          2 (doh_grouped['C'] > 0).sum()
executed in 29ms, finished 15:02:09 2022-08-12
```

Out[26]: 5648

Contents

The 19,790 unique restaurants, 5,648 severely failed an initial cycle inspection at least once and

1 Predicting NYC Risk of being closed by the DOHMH.

2 Introduction

2.1 Objective
2.1.1 Imports
In [27]: 1 # Creating the Target Variable 'Severe' for Restaurants that have score
 2 doh_grouped['Severe'] = (doh_grouped['C'] > 0).astype(int)

3 Obtain
3 nyc_df = doh_grouped.reset_index()

3.1 Obtaining Restaurant
4 nyc_df.drop(['A', 'B', 'C'], axis=1, inplace=True)
3.1.1 Understanding NYC
executed in 67ms, finished 15:02:09 2022-08-12

3.1.2 Target Variable -- N

In [28]: 1 # Checking out the new DF with the engineered target variable

3.2 Obtaining Yelp Business
2 nyc_df

3.2.1 Yelp Business Search

executed in 38ms, finished 15:02:09 2022-08-12

3.2.2 Yelp Reviews

Out[28]: 3.2.3 Joining Yelp Review

| | CAMIS | DBA | CUISINE DESCRIPTION | BORO | BUILDING | STREET | ZIPCODE |
|---------------------------------------|-------------------------|--------------------------------|--------------------------|-----------|----------|--------------------|--------------|
| 3.3 Joining NYC DOHMH { | | | | | | | |
| 4 Scrubbing The Data | 300751445 | MORRIS PARK BAKE SHOP | Bakery Products/Desserts | Bronx | 1007 | MORRIS PARK AVENUE | 10462.00 718 |
| 4.1 Drop businesses with 0 violations | 300751445 | | | | | | |
| 5 Exploring The Dataset | | | | | | | |
| 6 Preprocessing For Further Analysis | 1 30112340 | WENDY'S | Hamburgers | Brooklyn | 469 | FLATBUSH AVENUE | 11225.00 718 |
| 6.0.1 Lets find most freq | | | | | | | |
| 7 Preprocessing For Modelin | | DJ REYNOLDS PUB AND RESTAURANT | Irish | Manhattan | 351 | WEST 57 STREET | 10019.00 212 |
| 8 Models | 2 30191841 | RIVIERA CATERERS | American | Brooklyn | 2780 | STILLWELL AVENUE | 11224.00 718 |
| 8.1 Bag-of-Words Model | | | | | | | |
| 8.1.1 Model Evaluations | | | | | | | |
| 8.2 Text Preprocessing Pic | 3 40356018 | WILKEN'S FINE FOOD | Sandwiches | Brooklyn | 7114 | AVENUE U | 11234.00 718 |
| 8.2.1 Multinomial Naive | | | | | | | |
| 8.2.2 Tuned Multinomial | 4 40356483 | | | | | | |
| 8.2.3 Logistic Regressio | | | | | | | |
| 8.2.4 Tuned Logistic Re | | | | | | | |
| 8.2.5 Decision Tree Mod | | | | | | | |
| 8.2.6 Tuned DecisionTre | 19785 50115169 | Sobak | Korean | Manhattan | 51 | CANAL STREET | 10002.00 929 |
| 8.2.7 Support Vector Clas | | | | | | | |
| 8.2.8 Tuned SupportVec | 19786 50116155 | SEPTEMBER | Coffee/Tea | Brooklyn | 83 | SARATOGA AVENUE | 11233.00 508 |
| 9 Conclusions | | | | | | | |
| 9.1 Best Model Results | 19787 50117350 | SONG TEA | Coffee/Tea | Manhattan | 488 | 7 AVENUE | 10018.00 646 |
| 9.2 Next Steps and Future | | | | | | | |
| 9.3 Appendix | 19788 50117434 | LA POLLERA COLORADA | Spanish | Queens | 8213 | NORTHERN BLVD | 11372.00 718 |
| 9.3.1 Word2Vec Embedc | | | | | | | |
| 9.3.2 Classifying With T | 19789 50117959 | SOL MAYA RESTAURANT | American | Manhattan | 2061 | 2 AVENUE | 10029.00 212 |
| 9.3.3 Pretrained Vector I | | | | | | | |
| 9.3.4 Mean Word Embed | 19790 rows x 14 columns | | | | | | |

NYC DOH Data Exploration

In [29]: 1 nyc_df['BORO'].value_counts()

executed in 18ms, finished 15:02:09 2022-08-12

Out[29]:

| BORO | Count |
|---------------|-------|
| Manhattan | 7892 |
| Brooklyn | 4801 |
| Queens | 4468 |
| Bronx | 1920 |
| Staten Island | 704 |

Contents

1 Predicting NYC Restaurant Violations

2 Introduction

3 Obtaining Data

4 Scrubbing The Data

5 Exploring The Dataset

6 Preprocessing For Further Analysis

7 Preprocessing For Modeling

8 Models

9 Conclusions

9.1 Best Model Results

9.2 Next Steps and Future Work

9.3 Appendix - Deep NLP

3.2 Obtaining Yelp Business and Review Data

Now that we have an understanding of the city's inspection results and have explored that dataset, it is time to pull in data from the crowd-sourced review platform Yelp.

9.3.4 Mean Word Embedding

In [36]:

```

1 #Functionizing the Yelp API Phone Search
2
3 def get_businesses(phone_numbers):
4     """Input a list of formatted phone numbers
5     (must start with + and include the country code, like +14159083801)
6     and returns a corresponding list of Yelp Businesses"""
7
8     biz_list = []
9
10    for number in phone_numbers:
11        url = 'https://api.yelp.com/v3/businesses/search/phone'
12        headers = {'Authorization': 'Bearer ' + creds['api_key']}
13        url_params = {'phone': number}
14        response = requests.get(url, headers=headers, params=url_params)
15        response_json = response.json()
16        biz_list.extend(response_json.get('businesses', []))
17
18    while 'U' in biz_list:
19        biz_list.remove('U')
20
21    return biz_list
22
23 Joining Yelp Reviews
24
25 Joining NYC DOMH
26
27 Scrubbing The Data
28
29 Drop businesses missing `get_business` function
30 Exploring The Data
31 If call_apis == True:
32
33 Preprocessing For Further Analysis
34 biz_list12 = get_businesses(phone_numbers12)
35
36 6.0.1 Lets find most freq
37 Preprocessing For Modeling
38 biz12_df = pd.DataFrame(biz_list12)
39
40 6.0.2 Save returned list as a DataFrame and .csv file
41 biz12_df.to_csv('data/yelp_data/yelp_business/yelp_phone12.csv', index=False)
42
43 Models
44
45 8.1 Bag-of-Words Model
46
47 8.1.1 Model Evaluations
48
49 8.2 Text Preprocessing Pip
50
51 In 8.2.1 Multinomial Naive Bayes
52
53 8.2.2 Tuned Multinomial Naive Bayes
54 8.2.3 Logistic Regression
55 8.2.4 Tuned Logistic Regression
56 8.2.5 Decision Tree Model
57
58 Tuned Decision Tree Model
59
60 8.2.6 Tuned Decision Tree Model
61
62 8.2.7 Support Vector Classification
63
64 In 8.2.8 Tuned Support Vector Classification
65
66 8.2.9 Append saved Yelp Business tables to a dict
67 yelp_tables = {}
68
69 9 Conclusions
70
71 9.1 Best Model Results
72 9.2 Next Steps and Future Work
73
74 9.3 Appendix - Deep NLP
75 fname = f.replace('data/yelp_data/yelp_businesses/yelp_phone', 'df_')
76
77 9.3.1 Word2Vec Embedding
78 yelp_tables[fname] = temp_df
79
80 9.3.2 Classifying With Embeddings
81
82 9.3.3 Pretrained Vector Model
83
84 In 9.3.4 Mean Word Embedding
85 yelp_df_list = [t for t in list(yelp_tables.keys())]
86
87 executed in 7ms, finished 15:02:10 2022-08-12

```

Contents

| |
|--|
| 1 Predicting NYC Restaurant Health Violations |
| 2 Introduction |
| 2.1 Objective |
| 2.1.1 Imports |
| 2.2 Data |
| 2.3 Obtaining Data |
| 2.3.1 Obtaining Restaurant Information |
| 2.3.1.1 Understanding NYC Restaurants |
| 2.3.1.2 Target Variable -- NYC Health Violations |
| 2.3.1.3 Engineering Target Variable |
| 2.3.2 Obtaining Yelp Business Information |
| 2.3.2.1 Yelp Business Search |
| 2.3.2.2 Yelp Business Reviews |
| 2.3.2.3 Joining Yelp Reviews |
| 2.3.3 Joining NYC DOMH Data |
| 2.4 Scrubbing The Data |
| 2.4.1 Drop businesses missing `get_business` function |
| 2.4.2 Exploring The Data |
| 2.4.3 If call_apis == True: |
| 2.5 Preprocessing For Further Analysis |
| 2.5.1 biz_list12 = get_businesses(phone_numbers12) |
| 2.5.2 6.0.1 Lets find most freq |
| 2.5.3 Preprocessing For Modeling |
| 2.5.4 biz12_df = pd.DataFrame(biz_list12) |
| 2.5.5 6.0.2 Save returned list as a DataFrame and .csv file |
| 2.6 Models |
| 2.6.1 8.1 Bag-of-Words Model |
| 2.6.1.1 Model Evaluations |
| 2.6.2 8.2 Text Preprocessing Pipeline |
| 2.6.2.1 In 8.2.1 Multinomial Naive Bayes |
| 2.6.2.2 8.2.2 Tuned Multinomial Naive Bayes |
| 2.6.2.3 8.2.3 Logistic Regression |
| 2.6.2.4 8.2.4 Tuned Logistic Regression |
| 2.6.2.5 8.2.5 Decision Tree Model |
| 2.6.2.6 Tuned Decision Tree Model |
| 2.6.2.7 8.2.7 Support Vector Classification |
| 2.6.3 In 8.2.8 Tuned Support Vector Classification |
| 2.7 Conclusions |
| 2.7.1 9.1 Best Model Results |
| 2.7.2 9.2 Next Steps and Future Work |
| 2.7.3 9.3 Appendix - Deep NLP |
| 2.7.3.1 Word2Vec Embedding |
| 2.7.3.2 Classifying With Embeddings |
| 2.7.3.3 Pretrained Vector Model |
| 2.7.4 In 9.3.4 Mean Word Embedding |
| 2.7.4.1 yelp_df_list = [t for t in list(yelp_tables.keys())] |

executed in 11ms, finished 15:02:09 2022-08-12

In [37]:

```

1 # Call `get_business` function
2 if call_apis == True:
3
4 6 Preprocessing For Further Analysis
5 biz_list12 = get_businesses(phone_numbers12)
6
7 6.0.1 Lets find most freq
8 Preprocessing For Modeling
9 biz12_df = pd.DataFrame(biz_list12)
10
11 6.0.2 Save returned list as a DataFrame and .csv file
12 biz12_df.to_csv('data/yelp_data/yelp_business/yelp_phone12.csv', index=False)
13
14 Models
15
16 8.1 Bag-of-Words Model
17
18 8.1.1 Model Evaluations
19
20 8.2 Text Preprocessing Pipeline
21
22 In 8.2.1 Multinomial Naive Bayes
23
24 8.2.2 Tuned Multinomial Naive Bayes
25 8.2.3 Logistic Regression
26 8.2.4 Tuned Logistic Regression
27 8.2.5 Decision Tree Model
28
29 Tuned Decision Tree Model
30
31 8.2.6 Tuned Decision Tree Model
32
33 8.2.7 Support Vector Classification
34
35 In 8.2.8 Append saved Yelp Business tables to a dict
36 yelp_tables = {}
37
38 9 Conclusions
39
40 9.1 Best Model Results
41 9.2 Next Steps and Future Work
42
43 9.3 Appendix - Deep NLP
44 fname = f.replace('data/yelp_data/yelp_businesses/yelp_phone', 'df_')
45
46 9.3.1 Word2Vec Embedding
47 yelp_tables[fname] = temp_df
48
49 9.3.2 Classifying With Embeddings
50
51 9.3.3 Pretrained Vector Model
52
53 In 9.3.4 Mean Word Embedding
54 yelp_df_list = [t for t in list(yelp_tables.keys())]
55
56 executed in 614ms, finished 15:02:10 2022-08-12

```

executed in 10ms, finished 15:02:09 2022-08-12

In [38]:

```

1 # List of files containing Yelp business data
2 fpath = 'data/yelp_data/yelp_businesses/'
3 os.listdir(fpath)
4 query = fpath+"*.csv"
5 f_list = glob.glob(query)
6
7 8.2.6 Tuned Decision Tree Model
8
9 8.2.7 Support Vector Classification
10
11 In 8.2.8 Append saved Yelp Business tables to a dict
12 yelp_tables = {}
13
14 9 Conclusions
15
16 9.1 Best Model Results
17 9.2 Next Steps and Future Work
18
19 9.3 Appendix - Deep NLP
20 fname = f.replace('data/yelp_data/yelp_businesses/yelp_phone', 'df_')
21
22 9.3.1 Word2Vec Embedding
23 yelp_tables[fname] = temp_df
24
25 9.3.2 Classifying With Embeddings
26
27 9.3.3 Pretrained Vector Model
28
29 In 9.3.4 Mean Word Embedding
30 yelp_df_list = [t for t in list(yelp_tables.keys())]
31
32 executed in 7ms, finished 15:02:10 2022-08-12

```

```
In [41]: 1 # Concatenating all Yelp Businesses responses from the Phone Search
2 yelp_businesses_df = pd.concat(yelp_tables, ignore_index=True)
3 yelp_businesses_df
```

executed in 55ms, finished 15:02:10 2022-08-12

| | | | | |
|---|------------------------|---------------------|--------------------|---|
| 4 | rYI_R-UILYqA6vYsbNwLsg | company-new-york-14 | joe coffee Company | https://s3-media0.fl.yelpcdn.com/bphoto/RHzuMsMC |
|---|------------------------|---------------------|--------------------|---|

Contents ⚙️

| | | | | |
|-------------|---|--|---------------------|---|
| 1 | Predicting NYC Restaurant | | | |
| ▼ 2 | Introduction | | | |
| ▼ 2.1 | Objective | | | |
| 2.1.1 | Imports | ... | ... | ... |
| ▼ 3 | Obtain | | | |
| ▼ 3.1 | Obtaining Restaurant Info | | | |
| 3.1.1 | Understanding NYC | | | |
| 3.1.2 | Target Variable -- NYC | | | |
| 3.1.3 | Engineering Target | | | |
| ▼ 3.2 | Obtaining Yelp Business | loreley-beer-garden-new-york-2 | Loreley Beer Garden | https://media0.fl.yelpcdn.com/bphoto/RHzuMs |
| 3.2.1 | Yelp Business Search | | | |
| 3.2.2 | Yelp Reviews | | | |
| 3.2.3 | Joining Yelp Reviews | | | |
| 3.3 | Joining NYC DOHMH Data | | | |
| ▼ 4 | Scrubbing The Data | | | |
| 4.1 | Drop businesses missing | | | |
| 5 | Exploring The Dataset | | | |
| ▼ 6 | Preprocessing Yelp Businesses Response Data | | | |
| 6.0.1 | Lets find most freq | | | |
| 7 | Preprocessing For Model | yelp_businesses_df.info() | | |
| ▼ 8 | Models | executed in 84ms, finished 15:02:10 2022-08-12 | | |
| ▼ 8.1 | Bag-of-Words Model | | | |
| 8.1.1 | Model Evaluations | <class 'pandas.core.frame.DataFrame'> | | |
| RangeIndex: | 17302 entries, 0 to 17301 | | | |
| ▼ 8.2 | Text Preprocessing Pic | data.columns (total 15 columns): | | |
| 8.2.1 | Multinomial Naive | # Column | Non-Null Count | Dtype |
| 8.2.2 | Tuned Multinomial | ----- | ----- | ----- |
| 8.2.3 | Logistic Regression | 17302 | non-null | object |
| 8.2.4 | Tuned Logistic Reg | 17302 | non-null | object |
| 8.2.5 | Decision Tree Model | 17302 | non-null | object |
| 8.2.6 | Tuned DecisionTree url | 16563 | non-null | object |
| 8.2.7 | Support Vector closed | 17302 | non-null | bool |
| 8.2.8 | Tuned SupportVec | 17302 | non-null | object |
| ▼ 9 | Conclusions | review_count | 17302 | non-null int64 |
| 9.1 | Best Model Results | categories | 17302 | non-null object |
| 9.2 | Next Steps and Future | rating | 17302 | non-null float64 |
| 9.3 | Appendix Deep NLP | coordinates | 17302 | non-null object |
| 9.3.1 | Word2Vec Embedds | transactions | 17302 | non-null object |
| 9.3.2 | Classifying With E | price | 14501 | non-null object |
| 9.3.3 | Pretrained Vector! | location | 17302 | non-null object |
| 9.3.4 | Mean Word Embedds | phone | 17302 | non-null int64 |
| | | display_phone | 17302 | non-null object |
| | | | | dtypes: bool(1), float64(1), int64(2), object(11) |
| | | | | memory usage: 1.9+ MB |

In [43]: 1 yelp_businesses_df['price'].value_counts(normalize=True)

executed in 15ms, finished 15:02:10 2022-08-12

Out[43]:

| | |
|--------|------|
| \$\$ | 0.53 |
| \$ | 0.37 |
| \$\$\$ | 0.07 |
| \$\$ | 0.02 |

Contents

1 Predicting NYC Restaurant Name: price, dtype: float64

▼ 2 Introduction

In [44]: 1 yelp_businesses_df['rating'].value_counts(normalize=True)

2.1.1 Imports

executed in 16ms, finished 15:02:10 2022-08-12

▼ 3 Obtain

Out[44]:

| | |
|------|------|
| 4.00 | 0.30 |
| 3.50 | 0.25 |
| 3.00 | 0.14 |
| 4.50 | 0.12 |
| 2.50 | 0.08 |

3.1.1 Understanding NYC

3.1.2 Target Variable

3.1.3 Engineering Target

3.2.1 Yelp Business Search

3.2.2 Yelp Reviews

3.2.3 Joining Yelp Review

3.3 Joining NYC DOHMH Data

Name: rating, dtype: float64

▼ 4 Scrubbing The Data

In [45]: 1 yelp_businesses_df['categories'].value_counts(normalize=True)

5 Exploring The Dataset

executed in 26ms, finished 15:02:10 2022-08-12

▼ 6 Preprocessing For Further

Out[45]:

| | |
|-----------------------|--------------------------------|
| Let's find alias freq | 'chinese', 'title': 'Chinese'} |
|-----------------------|--------------------------------|

7 Preprocessing For Modelin

▼ 8 Models

| |
|--|
| [{'alias': 'pizza', 'title': 'Pizza'}] |
|--|

8.1 Bag-of-Words Model

8.1.1 Model Evaluations

8.2 Text Preprocessing,Pic

8.2.1 Multinomial Naive

8.2.2 Tuned Multinomial

8.2.3 Logistic Regressio

8.2.4 Tuned Logistic Re

8.2.5 Decision Tree Mod

8.2.6 Tuned Decision Tre

8.2.7 Support Vector Ma

8.2.8 Tuned Support Vec

8.2.9 Conclusions

9.1 Best Model Results

9.2 Next Steps and Future

9.3 Appendix Deep NLP

9.3.1 Word2Vec Embedd

9.3.2 Classifying With E

9.3.3 Pretrained Vecto

9.3.4 Mean Word Embe

```
In [46]: 1 # Duplicates?
          2 yelp_businesses_df[yelp_businesses_df.duplicated(['id'], keep=False)].count()
          executed in 71ms, finished 15:02:10 2022-08-12
```

Out[46]: id 8474
alias 8474
name 8474

Contents

| | |
|-------|-----------------------------------|
| 1 | Predicting NYC Health Violations |
| 2 | Introduction |
| 2.1 | Objective |
| 2.1.1 | Imports |
| 3 | Obtain Yelp Businesses |
| 3.1 | Obtaining Restaurant Transactions |
| 3.1.1 | Understanding NYC Price |
| 3.1.2 | Target Variable -- Location |
| 3.1.3 | Engineering Target Phone |
| 3.2 | Obtaining Yelp Reviews |
| 3.2.1 | Yelp Businesses Data |
| 3.2.2 | Yelp Reviews |
| 3.2.3 | Joining Yelp Reviews |

```
In [47]: 1 yelp_businesses_df.describe()
          3.3 Joining NYC DOHMH
```

4 Scrubbing The Data

```
Out[47]: review_count rating phone
5 Exploring The Dataset
6 Preprocessing For Further
6.0.1 Lets find most freq mean 189.46 3.53 15227799712.25
7 Preprocessing For Modeling std 378.43 0.79 2494930771.02
8 Models
8.1 Bag-of-Words Model min 1.00 1.00 12062250663.00
8.1.1 Model Evaluations 25% 17.00 3.00 12127498940.50
8.2 Text Processing Pip 50% max 67.00 3.50 17182416042.00
8.2.2 Tuned Multinomial 75% 209.00 4.00 17186241444.00
8.2.3 Logistic Regression 8.2.4 Tuned Logistic Reg 19458.00 5.00 19732238527.00
8.2.5 Decision Tree Model
8.2.6 Tuned DecisionTree
8.2.7 Support Vector Clas
8.2.8 Tuned Supportive
```

```
In [48]: 1 yelp_businesses_df['review_count'].sum()
          executed in 15ms, finished 15:02:10 2022-08-12
```

9 Conclusions

- 9.1 Best Model Results
- 9.2 Next Steps and Future

3.2.2 Yelp Reviews

9.3.1 Word2Vec Embedding

9.3.2 Classifying With Embedding

Now that all the restaurants from the NYC DOHMH dataset have been used to search the Yelp API and have been concatenated we can use the return url to gather reviews for each business.

9.3.3 Pretrained Vector

9.3.4 Mean Word Embedding

```
In [49]: 1 # OLD from webscraping
          2 # df_10_2 = df_10.loc[1000:2173]
          3 # df_10_2.to_csv('df_10_2',index=False)
```

executed in 12ms, finished 15:02:10 2022-08-12

```
In [50]: 1 def get_text(url_list):
```

Contents ↗ ↘

- 1 Predicting NYC Restaurant
- 2 Introduction
- 3 Obtain
- 4 Scrubbing The Data
- 5 Exploring The Dataset
- 6 Preprocessing For Further
- 7 Preprocessing For Modelin
- 8 Models
- 9 Conclusion

```
          3     """ Given a list of urls, this function will iterate through the li
          4     extract text from the first page of reviews. The data will be joine
          5     a corpus for each business"""
          6     review_txt = []
          7
          8     for url in url_list:
          9         req = requests.get(url,allow_redirects=False)
         10        soup = bs(req.content)
         11        comments = soup.find_all(class_='raw_09f24_T4Ezm', lang="en")
         12        comment_txt = []
         13
         14        for comment in comments:
         15            comment_txt.append(comment.text)
         16
         17        comment_corp = ('.').join(comment_txt)
         18        review_txt.append(comment_corp)
         19    return review_txt
```

executed in 12ms, finished 15:02:10 2022-08-12

```
In [51]: 1 # Obtaining list of yelp business urls from saved API response
```

```
▼ 2.2 Text Preprocessing
  2.2.1 urlList = list(yelp_tables['df_1']['url'])
          3.1 urlList
```

executed in 10ms, finished 15:02:10 2022-08-12

Out[51]: 859

```
  2.2.2 Tuned Multinomial
  2.2.3 Logistic Regressio
  2.2.4 Tuned Logistic Re
  2.2.5 Decision Tree Mod
In [52]: 1 # Calling `get_text` function to obtain Yelp reviews
          2 if call_apis == True:
          3     review_text = get_text(url_list)
```

executed in 8ms, finished 15:02:10 2022-08-12

9.1 Best Model Results

In[53]: 1 # Saving the Reviews to a csv in the repository

```
▼ 9.2 Deep NLP
  9.2.1 if call_apis == True:
  2     review_text = pd.DataFrame(review_text,columns=['Review_Text'])
  3     review_text.to_csv('rvw_txt1.csv',index=False)
  4
  5 9.3.3 Pretrained Vector
  6
  7 9.3.4 Mean Word Embedding
```

executed in 8ms, finished 15:02:10 2022-08-12

3.2.3 Joining Yelp Reviews to Yelp Business Tables

```
In [54]: 1 # List of files containing Yelp business data
          2 fpath = 'data/yelp_data/yelp_reviews/'
          3 os.listdir(fpath)
          4 query = fpath + "*.csv"
          5 f_list = glob.glob(query)
```

Contents ↗

```
In [55]: 1 Predicting NYC Restaurant Reviews
          2 Introduction
          3 Objective
          4 Obtain Restaurant Data
          5 Obtaining Restaurant Name
          6 Understanding NYC Restaurants
          7 Target Variable
          8 Engineering Target
          9 Obtaining Yelp Business Data
          10 Yelp API
          11 Yelp Business Search
          12 Yelp Reviews
          13 Joining Yelp Reviews
          14 pd.read_csv('data/yelp_data/yelp_reviews/rvw_txt1.csv')
          15 Joining NYC DOHMH Data
          16 Scrubbing The Data
          17 Drop businesses missing
          18 Exploring The Dataset
          19 Preprocessing For Models
          20 Feature Selection
          21 Lets find most freq
          22 Preprocessing For Model
          23 Bag-of-Words Model
          24 Model Evaluations
          25 Text Preprocessing Pip
          26 Multinomial Naive Bayes
          27 Tuned Multinomial
          28 Logistic Regression
          29 Tuned Logistic Regression
          30 Adding review text to the Yelp Business tables
          31 Decision Tree Model
          32 Tuned Decision Tree
          33 Support Vector Model
          34 Tuned Support Vector Model
          35 Conclusions
          36 Best Model Results
          37 Next Steps and Future Work
          38 Appendix - Deep NLP
          39 Word2Vec Embedding
          40 Classifying With Embedding
          41 Pretrained Vector Model
          42 Mean Word Embedding
```

executed in 11ms, finished 15:02:10 2022-08-12

```
In [61]: 1 # Concatenating all Yelp businesses tables with review text included
          2 yelp_df = pd.concat(yelp_tables, ignore_index=True)
          3 yelp_df
```

executed in 54ms, finished 15:02:12 2022-08-12

Contents ⚙

- 1 Predicting NYC Restaurant
- ▼ 2 Introduction
 - ▼ 2.1 Objective
 - 2.1.1 Imports 2 in7QM4vNUTHVZli7Outetg
 - 2.1.2 mcdonalds-new-york-430 McDonald's media0.fl.yelpcdn.com/bphoto/bNmu8D1sj
- ▼ 3 Obtain
 - ▼ 3.1 Obtaining Restaurant Info
 - 3.1.1 Understanding NYC
 - 3.1.2 Target Variable -- NYC
 - 3.1.3 Engineering Target
 - ▼ 3.2 Obtaining Yelp Business
 - 3.2.1 Yelp Business Search
 - 3.2.2 Yelp Reviews
 - 3.2.3 Joining Yelp Reviews
 - 3.3 Joining NYC DOHMH Data
- ▼ 4 Scrubbing The Data
 - 4.1 Drop businesses missing
- 5 Exploring The Dataset
- ▼ 6 Preprocessing For Further Analysis
 - 6.0.1 In [62]: 1 # OLD Yelp Review API
 - 2 # review_df = pd.DataFrame(response_json.get('reviews'))
 - 3 # review_df
- 7 Preprocessing For Modeling
- ▼ 8 Models
 - ▼ 8.1 Bag-of-Words Model
 - 8.1.1 Model Evaluations
 - ▼ 8.2 Text Preprocessing Pipelines
 - 8.2.1 Multinomial Naive Bayes
- ▼ 9 Conclusions
 - 9.1 Best Model Results
 - 9.2 Next Steps and Future
 - ▼ 9.3 Appendix - Deep NLP
 - 9.3.1 Word2Vec Embeddings
 - 9.3.2 Classifying With Embeddings
 - 9.3.3 Pretrained Vector Embeddings
 - 9.3.4 Mean Word Embeddings

3.3 Joining NYC DOHMH & Yelp Datasets

```
In [63]: 1 # Formatting Yelp phone numbers to align with NYC phone numbers to join
          2 response['phone'] = '+' + yelp_df['phone'].apply(str)
```

executed in 5ms, finished 15:02:12 2022-08-12

In [64]: 1 yelp_df.head()

executed in 45ms, finished 15:02:13 2022-08-12

Out[64]:

| id | alias | name |
|-----------|--------------|-------------|
|-----------|--------------|-------------|

Contents ⚙️

| | | | |
|--------------------------------------|---------------------------|------------------|--|
| 1 Predicting NYC Restaurant | | | |
| ▼ 2 Introduction | | | |
| ▼ 2.1 Objective | barrio-chino-new-york | Barrio Chino | media0.fl.yelpcdn.com/bphoto/dpaRqPZD6xq[|
| 2.1.1 Imports RzKR9QcYP03EQ_8DxDxowg | | | |
| ▼ 3 Obtain | | | |
| ▼ 3.1 Obtaining Restaurant Info | | | |
| 3.1.1 Understanding NYC | | | |
| 3.1.2 Target Variable -- NYC | | | |
| 3.1.3 Engineering Target | | | |
| ▼ 3.2 Obtaining Yelp Business | | | |
| 3.2.1 Yelp Business Search | | | |
| 3.2.2 Yelp Reviews | | | |
| 3.2.3 Joining Yelp Reviews | | | |
| 3.3 Joining NYC DOHMH Data | | | |
| ▼ 4 Scrubbing The Data | | | |
| 4.1 Drop businesses missing | | | |
| 5 Exploring The Dataset | reyes-restaurant-woodside | Reyes Restaurant | https://s3-media0.fl.yelpcdn.com/bphoto/ |
| ▼ 6 Preprocessing For Further | 1_hrP3xLWJ8sLwG_XxEgq61Q | | |
| 6.0.1 Lets find most frequent words | | | |
| 7 Preprocessing For Modeling | | | |
| ▼ 8 Models | | | |
| ▼ 8.1 Bag-of-Words Model | | | |
| 8.1.1 Model Evaluations | | | |
| ▼ 8.2 Text Preprocessing Pipeline | | | |
| 8.2.1 Multinomial Naive Bayes | | | |
| 8.2.2 Tuned Multinomial Naive Bayes | | | |
| 8.2.3 Logistic Regression | | | |
| 8.2.4 Tuned Logistic Regression | | | |
| 8.2.5 Decision Tree Model | | | |
| 8.2.6 Tuned DecisionTree Model | | | |
| 8.2.7 Support Vector Machine | mcdonalds-new-york-430 | McDonald's | media0.fl.yelpcdn.com/bphoto/bNmu8D1sjwJz[|
| 8.2.8 Tuned SupportVector Machine | | | |
| ▼ 9 Conclusions | | | |
| 9.1 Best Model Results | | | |
| 9.2 Next Steps and Future | | | |
| ▼ 9.3 Appendix - Deep NLP | | | |
| 9.3.1 Word2Vec Embedding | | | |
| 9.3.2 Classifying With Embedding | | | |
| 9.3.3 Pretrained Vector Embedding | | | |
| 9.3.4 Mean Word Embedding | | | |

| id | alias | name | |
|-------------------------|--|---------------------------|---|
| F-YGKegXRG2TE-Xw1AWeCA | popeyes-louisiana-kitchen-rockaway-beach-2 | Popeyes Louisiana Kitchen | media0.fl.yelpcdn.com/bphoto/A4MR2hDN3mWef |
| 4_FYI_R-ULYqA6vYsbNwLsg | joe-coffee-company-new-york-14 | Joe Coffee Company | https://s3-media0.fl.yelpcdn.co/MQAN4 |

Contents ⚙️ 3

- 1 Predicting NYC Restaurant
- ▼ 2 Introduction
 - ▼ 2.1 Objective
 - 2.1.1 Imports
- ▼ 3 Obtain
 - ▼ 3.1 Obtaining Restaurant Info
 - 3.1.1 Understanding NYC Data
 - 3.1.2 Target Variable -- NYC Health Violations
 - 3.1.3 Engineering Target Variable
 - ▼ 3.2 Obtaining Yelp Business Data
 - 3.2.1 Yelp Business Search
 - 3.2.2 Yelp Reviews
 - 3.2.3 Joining Yelp Reviews
 - 3.3 Joining NYC DOHMH Data
- ▼ 4 Scrubbing The Data
 - 4.1 Drop businesses missing address
- 5 Exploring The Dataset
- ▼ 6 Preprocessing For Further Modeling
 - 6.0.1 Lets find most frequent business names
- 7 Preprocessing For Modeling
- ▼ 8 Models
 - ▼ 8.1 Bag-of-Words Model
 - 8.1.1 Model Evaluations
 - ▼ 8.2 Text Preprocessing Pipeline
 - 8.2.1 Multinomial Naive Bayes
 - 8.2.2 Tuned Multinomial Naive Bayes
 - 8.2.3 Logistic Regression
 - 8.2.4 Tuned Logistic Regression
 - 8.2.5 Decision Tree Model
 - 8.2.6 Tuned DecisionTree Model
 - 8.2.7 Support Vector Classification
 - 8.2.8 Tuned SupportVector Classification
- ▼ 9 Conclusions
 - 9.1 Best Model Results
 - 9.2 Next Steps and Future Work

```
In [65]: # Merging NYC and Yelp datasets
df = pd.merge(nyc_df, yelp_df, left_on='PHONE', right_on='phone', how='left')
9.3.2 Classifying With Embeddings
9.3.3 Pretrained Vector
9.3.4 Mean Word Embedding
```

executed in 71ms, finished 15:02:13 2022-08-12

```
In [66]: 1 # Saving merged dataset to csv
          2 # df_1.to_csv('full_dataset.csv')
          3 df_1
```

executed in 59ms, finished 15:02:13 2022-08-12

Out[66]:

| Contents | CAMIS | DBA | CUISINE DESCRIPTION | BORO | BUILDING | STREET | ZIPCODE |
|-------------------------------------|------------|---------|---------------------|----------|----------|-----------------|-----------------|
| 1 Predicting NYC Restaurant | | | | | | | |
| ▼ 2 Introduction | | | | | | | |
| ▼ 2.1 Objective | | | | | | | |
| 2.1.1 Imports | | | | | | | |
| ▼ 3 Obtain | 0_30112340 | WENDY'S | Hamburgers | Brooklyn | 469 | FLATBUSH AVENUE | 11225.00 +17182 |
| ▼ 3.1 Obtaining Restaurant | | | | | | | |
| 3.1.1 Understanding NYC | | | | | | | |
| 3.1.2 Target Variable -- N | | | | | | | |
| 3.1.3 Engineering Target | | | | | | | |
| ▼ 3.2 Obtaining Yelp Business | | | | | | | |
| 3.2.1 Yelp Business Search | | | | | | | |
| 3.2.2 Yelp Reviews | | | | | | | |
| 3.2.3 Joining Yelp Reviews | | | | | | | |
| 3.3 Joining NYC DOHMH Data | | | | | | | |
| ▼ 4 Scrubbing The Data | | | | | | | |
| 4.1 Drop businesses missing | | RIVIERA | | | | | |
| 5 Exploring The Dataset | | | | | | | |
| ▼ 6 Preprocessing For Further | | | | | | | |
| 6.0.1 Lets find most freq | | | | | | | |
| 7 Preprocessing For Modeling | | | | | | | |
| ▼ 8 Models | | | | | | | |
| ▼ 8.1 Bag-of-Words Model | | | | | | | |
| 8.1.1 Model Evaluations | | | | | | | |
| ▼ 8.2 Text Preprocessing Pictures | | | | | | | |
| 8.2.1 Multinomial Naive Bayes | | | | | | | |
| 8.2.2 Tuned Multinomial | | | | | | | |
| 8.2.3 Logistic Regression | | | | | | | |
| 8.2.4 Tuned Logistic Regression | | | | | | | |
| 8.2.5 Decision Tree Model | | | | | | | |
| 8.2.6 Tuned DecisionTree Model | | | | | | | |
| 8.2.7 Support Vector Classification | | | | | | | |
| 8.2.8 Tuned SupportVectorModel | | | | | | | |
| ▼ 9 Conclusions | | | | | | | |
| 9.1 Best Model Results | | | | | | | |
| 9.2 Next Steps and Future | | | | | | | |
| ▼ 9.3 Appendix - Deep NLP | | | | | | | |
| 9.3.1 Word2Vec Embedding | | | | | | | |
| 9.3.2 Classifying With Embedding | | | | | | | |
| 9.3.3 Pretrained Vector Model | | | | | | | |
| 9.3.4 Mean Word Embedding | | | | | | | |

In [67]:

```
1 df_1.info()
```

executed in 144ms, finished 15:02:13 2022-08-12

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18462 entries, 0 to 18461
Data columns (total 30 columns):
```

| # | Column | Non-Null Count | Dtype |
|-------|-------------------------|---|----------|
| 1 | CAMIS | 18462 | non-null |
| 2 | DBA | 18461 | non-null |
| 2.1 | CUISINE | 18462 | non-null |
| 2.1.1 | BORO | 18462 | non-null |
| 2.1.2 | BUILDING | 18302 | non-null |
| 3 | STREET | 18462 | non-null |
| 3.1 | ZIPCODE | 18168 | non-null |
| 3.1.1 | PHONE | 18462 | non-null |
| 3.1.2 | Latitude | 18420 | non-null |
| 3.1.3 | Longitude | 18420 | non-null |
| 3.2 | Business District | 17992 | non-null |
| 3.2.1 | Census Tract | 17992 | non-null |
| 3.2.2 | Review Count | 17992 | non-null |
| 3.2.3 | Severity | 18462 | non-null |
| 3.3 | id | 18462 | non-null |
| 3.4 | alias | 18462 | non-null |
| 4 | name | 18462 | non-null |
| 4.1 | Drop businesses missing | 17672 | non-null |
| 5 | Exploring The Dataset | 18462 | non-null |
| 6 | is_closed | 18462 | non-null |
| 6.0.1 | Lets find most freq | 18462 | non-null |
| 6.0.2 | Review Count | 18462 | non-null |
| 7 | Categories | 18462 | non-null |
| 8 | rating | 18462 | non-null |
| 8.1 | Coordinates | 18462 | non-null |
| 8.1.1 | Evaluations | 18462 | non-null |
| 8.2 | Text Preprocessing | 15391 | non-null |
| 8.2.1 | Multinomial Naive | 18462 | non-null |
| 8.2.2 | Tuned Multinomial | 18462 | non-null |
| 8.2.3 | Logistic Regression | 18462 | non-null |
| 8.2.4 | Tuned Logistic Reg | 15681 | non-null |
| 8.2.5 | Decision Tree Model | dtypes: bool(1), float64(7), int64(3), object(19) | |
| 8.2.6 | memory usage: | 4.2+ MB | |
| 8.2.7 | Tuned DecisionTree | | |
| 8.2.8 | Support Vector Clas | | |
| 9 | Conclusion | | |

4 Scrubbing The Data

9.1 Best Model Results

9.2 Next Steps and Future

```
In [68]: 1 # Drop businesses missing reviews
          2 # Drop unnecessary columns
          3 # Check duplicates
          4 # Feature Engineering
          5 # Get coordinates
          6 # Get takeout/delivery
          7 # Get $$$
```

executed in 7ms, finished 15:02:13 2022-08-12

4.1 Drop businesses missing reviews and

duplicates

In [69]: 1 df_1['Reviews'].isna().sum()

executed in 18ms, finished 15:02:13 2022-08-12

Out[69]: 2781

Contents ⚙️

In [70]: 1 NYC_Restaurant.dropna(subset='Reviews', inplace=True)

▼ 2 Introduction 2 df_1['Reviews'].isna().sum()

▼ 2.1 Objectives executed in 78ms, finished 15:02:13 2022-08-12

Out[70]: Imports

▼ 3 Obtain

▼ 3.1 Obtaining Restaurant Info

In [71]: 1 # Checking for duplicated rows

3.1.1 Understanding NYC Restaurants

3.1.2 Target Variable

3.1.3 Engineering Target

Out[71]: Obtaining Yelp Business Data

| CAMIS | DBA | CUISINE DESCRIPTION | BORO | BUILDING | STREET | ZIPCODE | PHONE NUMBER |
|-------|-----|---------------------|------|----------|--------|---------|--------------|
| | | | | | | | |

3.2.1 Yelp Business Seal

3.2.2 Yelp Reviews

3.2.3 Joining Yelp Reviews

3.3 Joining NYC DOHMH Inspections

▼ 4 Scrubbing The Data

4.1 Drop businesses missing CAMIS

5 Exploring The Dataset

▼ 6 Preprocessing For Further Analysis

6.0.1 Lets find most freq

7 Preprocessing For Modeling

▼ 8 Models

▼ 8.1 Bag-of-Words Model

8.1.1 Model Evaluations

▼ 8.2 Text Preprocessing Pipeline

8.2.1 Multinomial Naive Bayes

8.2.2 Tuned Multinomial Naive Bayes

8.2.3 Logistic Regression

8.2.4 Tuned Logistic Regression

In [72]: 1 # Dropping Duplicated Rows

8.2.5 Decision Tree Model

8.2.6 Tuned Decision Tree Model

8.2.7 Support Vector Machine Model

8.2.8 Tuned SupportVector Machine Model

▼ 9 Conclusions

9.1 Best Model Results

9.2 Next Steps and Future Work

▼ 9.3 Appendix - Deep NLP

9.3.1 Word2Vec Embeddings

9.3.2 Classifying With Embeddings

9.3.3 Pretrained Vector Embeddings

9.3.4 Mean Word Embeddings

In [73]: 1 df_1

executed in 55ms, finished 15:02:14 2022-08-12

Contents ⚙️⚙️

- 1 Predicting NYC Restaurant
- ▼ 2 Introduction
 - ▼ 2.1 Objective
 - 2.1.1 Imports
- ▼ 3 Obtain 1 40356018
 - ▼ 3.1 Obtaining Restaurant Info
 - 3.1.1 Understanding NYC Restaurants
 - 3.1.2 Target Variable -- Number of Violations
 - 3.1.3 Engineering Target Variable
 - ▼ 3.2 Obtaining Yelp Business Data
 - 3.2.1 Yelp Business Search
 - 3.2.2 Yelp Reviews
 - 3.2.3 Joining Yelp Reviews
 - 3.3 Joining NYC DOHMH Data
- ▼ 4 Scrubbing The Data *Resetting the index*
 - 4.1 Drop business_id's that have missing values `df.set_index(inplace=True)`
 - 5 Exploring The Dataset `df.drop(columns='index', inplace=True)`
- ▼ 6 Preprocessing For Further Analysis *executed in 30ms, finished 15:02:14 2022-08-12*
 - 6.0.1 Lets find most frequent words
 - 7 Preprocessing For Modeling
- ▼ 8 Models
 - ▼ 8.1 Bag-of-Words Model
 - 8.1.1 Model Evaluations
 - ▼ 8.2 Text Preprocessing Pipeline
 - 8.2.1 Multinomial Naive Bayes
 - 8.2.2 Tuned Multinomial Naive Bayes
 - 8.2.3 Logistic Regression
 - 8.2.4 Tuned Logistic Regression
 - 8.2.5 Decision Tree Model
 - 8.2.6 Tuned DecisionTree Model
 - 8.2.7 Support Vector Classification
 - 8.2.8 Tuned SupportVector Classification
- ▼ 9 Conclusions
 - 9.1 Best Model Results
 - 9.2 Next Steps and Future
 - ▼ 9.3 Appendix - Deep NLP
 - 9.3.1 Word2Vec Embeddings
 - 9.3.2 Classifying With Embeddings
 - 9.3.3 Pretrained Vector Model
 - 9.3.4 Mean Word Embedding

In [75]:

1 df_1.info()

executed in 95ms, finished 15:02:14 2022-08-12

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13061 entries, 0 to 13060
Data columns (total 30 columns):
```

| # | Column | Non-Null Count | Dtype |
|-------|---|----------------|----------|
| 1 | CAMIS | 13061 | non-null |
| 2 | DBA | 13060 | non-null |
| 2.1 | CUISINE | 13061 | non-null |
| 2.1.1 | BORO | 13061 | non-null |
| 2.1.2 | BUILDING | 12976 | non-null |
| 3 | STREET | 13061 | non-null |
| 3.1 | ZIPCODE | 12854 | non-null |
| 3.1.1 | PHONE | 13061 | non-null |
| 3.1.2 | Latitude | 13035 | non-null |
| 3.1.3 | Longitude | 13035 | non-null |
| 3.2 | Business District | 12759 | non-null |
| 3.2.1 | Census Tract | 12759 | non-null |
| 3.2.2 | Severity | 12759 | non-null |
| 3.2.3 | Yelp Review | 13061 | non-null |
| 3.3 | id | 13061 | non-null |
| 3.4 | alias | 13061 | non-null |
| 4 | name | 13061 | non-null |
| 4.1 | businesses missing | 12526 | non-null |
| 5 | Exploring The Dataset | 13061 | non-null |
| 5.1 | is_closed | 13061 | non-null |
| 6 | url | 13061 | non-null |
| 6.0.1 | Lets find most freq | 13061 | non-null |
| 6.0.2 | review_count | 13061 | int64 |
| 7 | categories | 13061 | non-null |
| 8 | rating | 13061 | non-null |
| 8.1 | coordinates | 13061 | non-null |
| 8.1.1 | transactions | 13061 | non-null |
| 8.2 | price | 10965 | non-null |
| 8.2.1 | location | 13061 | non-null |
| 8.2.2 | phone | 13061 | non-null |
| 8.2.3 | display_phone | 13061 | non-null |
| 8.2.4 | Reviews | 13061 | non-null |
| 8.2.5 | dtypes: bool(1), float64(7), int64(3), object(19) | | |
| 8.2.6 | memory usage: 2.9+ MB | | |
| 8.2.7 | Tuned DecisionTree | | |
| 8.2.8 | Support Vector Cl | | |
| 9 | Conclusions | | |
| 9.1 | Best Model Results | | |
| 9.2 | Next Steps and Future | | |
| 9.3 | Appendix - Deep NLP | | |
| 9.3.1 | Word2Vec Embedd | | |
| 9.3.2 | Classifying With Er | | |
| 9.3.3 | Pretrained Vector N | | |
| 9.3.4 | Mean Word Embed | | |

Contents ↗

- 1 Predicting NYC Restaurants
- 2 Introduction
 - 2.1 Objective
 - 2.1.1 Imports
- 3 Obtain
 - 3.1 Obtaining NYC Data
 - 3.1.1 Understanding NYC
 - 3.1.2 Target Variables
 - 3.1.3 Engineering Target
 - 3.2 Obtaining Yelp Data
 - 3.2.1 Yelp Business District
 - 3.2.2 Yelp Reviews
 - 3.2.3 Joining Yelp Review
 - 3.3 Joining NYC DOTMH & Yelp Data
- 4 Scrubbing The Data
 - 4.1 Drop businesses missing
 - 4.2 image_url
- 5 Exploring The Dataset
- 6 Preprocessing For Further
 - 6.0.1 Lets find most freq
 - 6.0.2 review_count
- 7 Preprocessing For Models
- 8 Models
 - 8.1 Bag-of-Words Model
 - 8.1.1 Model Evaluation
 - 8.2 Text Preprocessing
 - 8.2.1 Multinomial Naive Bayes
 - 8.2.2 Tuned Multinomial Naive Bayes
 - 8.2.3 Logistic Regression
 - 8.2.4 Tuned Logistic Regression
 - 8.2.5 Decision Tree Model
 - 8.2.6 Tuned DecisionTree
 - 8.2.7 Support Vector Cl
 - 8.2.8 Tuned SupportVec
- 9 Conclusions
 - 9.1 Best Model Results
 - 9.2 Next Steps and Future
- 9.3 Appendix - Deep NLP
 - 9.3.1 Word2Vec Embedding
 - 9.3.2 Classifying With Embedding
 - 9.3.3 Pretrained Vector Model
 - 9.3.4 Mean Word Embedding

In [76]: 1 df_1[df_1['DBA'].isna()]

executed in 50ms, finished 15:02:14 2022-08-12

Out[76]:

| CAMIS | DBA | CUISINE DESCRIPTION | BORO | BUILDING | STREET | ZIPCODE | PHONE | Lat |
|-------|-----|------------------------|------|----------|--------|---------|-------|-----|
|-------|-----|------------------------|------|----------|--------|---------|-------|-----|

Contents ⚙️

| | | | | | | | | |
|---|------|----------|-----|---------|-----------|-----|-------------------|-----------------------|
| 1 Predicting NYC Restaurant | | | | | | | | |
| ▼ 2 Introduction | | | | | | | | |
| ▼ 2.1 Objective | | | | | | | | |
| 2.1.1 Imports | | | | | | | | |
| ▼ 3 Obtain | 5435 | 41650149 | NaN | Italian | Manhattan | 345 | EAST 83 STREET | 10028.00 +12127378312 |
| ▼ 3.1 Obtaining Restaurant Info | | | | | | | | |
| 3.1.1 Understanding NYC Restaurants | | | | | | | | |
| 3.1.2 Target Variable -- NYC DOHMH Violations | | | | | | | | |
| 3.1.3 Engineering Target | | | | | | | | |
| ▼ 3.2 Obtaining Yelp Business Data | | | | | | | | |
| 3.2.1 Yelp Business Search | | | | | | | | |
| 3.2.2 Yelp Reviews | | | | | | | | |
| 3.2.3 Joining Yelp Reviews | | | | | | | | |
| 3.3 Joining NYC DOHMH & Yelp Data | | | | | | | | |
| ▼ 4 Scrubbing The Data | | | | | | | | |
| 4.1 Drop businesses missing address | | | | | | | | |
| 5 Exploring The Dataset | | | | | | | | |
| ▼ 6 Preprocessing For Further Modeling | | | | | | | | |
| 6.0.1 Lets find most frequent words | | | | | | | | |
| 7 Preprocessing For Modeling | | | | | | | | |
| ▼ 8 Models | | | | | | | | |
| ▼ 8.1 Bag-of-Words Model | | | | | | | | |
| 8.1.1 Model Evaluations | | | | | | | | |
| ▼ 8.2 Text Preprocessing Pipeline | | | | | | | | |
| 8.2.1 Multinomial Naive Bayes | | | | | | | | |
| 8.2.2 Tuned Multinomial Naive Bayes | | | | | | | | |
| 8.2.3 Logistic Regression | | | | | | | | |
| 8.2.4 Tuned Logistic Regression | | | | | | | | |
| 8.2.5 Decision Tree Model | | | | | | | | |
| 8.2.6 Tuned Decision Tree Model | | | | | | | | |
| 8.2.7 Support Vector Machine | | | | | | | | |
| 8.2.8 Tuned Support Vector Machine | | | | | | | | |
| ▼ 9 Conclusion | | | | | | | | |
| 9.1 Best Model Results | | | | | | | | |
| 9.2 Next Steps and Future Work | | | | | | | | |
| ▼ 9.3 Appendix | | | | | | | | |
| 9.3.1 Word2Vec Embedding | | | | | | | | |
| 9.3.2 Classifying With Embeddings | | | | | | | | |
| 9.3.3 Pretrained Vector Embedding | | | | | | | | |
| 9.3.4 Mean Word Embedding | | | | | | | | |

```
In [79]: 1 # Inspecting missing Boro's
2 df_1[df_1['BORO'] == '0']
```

executed in 48ms, finished 15:02:14 2022-08-12

Out[79]:

| CAMIS | DBA | CUISINE DESCRIPTION | BORO | BUILDING | STREET | ZIPCODE | PHONEI |
|-------|-----|------------------------|------|----------|--------|---------|--------|
|-------|-----|------------------------|------|----------|--------|---------|--------|

Contents ⚙

| | | | | | | | |
|---|-----------------------------------|-----------|--|----------|-----|--------------|----------------------|
| 1 Predicting NYC Restaurant | | | | | | | |
| ▼ 2 Introduction | | | | | | | |
| ▼ 2.1 Objective | | | | | | | |
| 2.1.1 Imports | 6780 | 50005134 | KINFOLK | American | 0 | 94 | WYTHE AVE |
| ▼ 3 Obtain | | | | | | | |
| 3.1 Obtaining Restaurant Info | | | | | | | |
| 3.1.1 Understanding NYC | | | | | | | |
| 3.1.2 Target Variable -- NYC | | | | | | | |
| 3.1.3 Engineering Target | | | | | | | |
| 3.2 Obtaining Yelp Businesses | | | | | | | |
| 3.2.1 Yelp Business Search | | | | | | | |
| 3.2.2 Yelp Reviews | | | | | | | |
| 3.2.3 Joining Yelp Reviews | | | | | | | |
| 3.3 Joining NYC DOHMH Data | | | | | | | |
| ▼ 4 Scrubbing The Data | 6781 | 50005134 | KINFOLK | American | 0 | 94 | WYTHE AVE |
| 4.1 Drop businesses missing | | | | | | | |
| 5 Exploring The Dataset | | | | | | | |
| ▼ 6 Preprocessing For Further | | | | | | | |
| 6.0.1 Lets find most freq | | | | | | | |
| 7 Preprocessing For Modeling | | | | | | | |
| ▼ 8 Models | | | | | | | |
| ▼ 8.1 Bag-of-Words Model | | | | | | | |
| 8.1.1 Model Evaluations | | | | | | | |
| ▼ 8.2 Text Preprocessing Pictures | | | POKEWORKS (Entrance is on 41st street between Lexington & Park) | | | | |
| 8.2.1 Multinomial Naive Bayes | 10182 | 500060598 | Hawaiian | 0 | 122 | E 42ND ST | 10168.00 +1646653979 |
| 8.2.2 Tuned Multinomial | | | | | | | |
| 8.2.3 Logistic Regression | | | | | | | |
| 8.2.4 Tuned Logistic Reg | | | | | | | |
| 8.2.5 Decision Tree Model | | | | | | | |
| 8.2.6 Tuned DecisionTree | | | | | | | |
| 8.2.7 Support Vector Class | | | | | | | |
| 8.2.8 Tuned SupportVec | | | | | | | |
| ▼ 9 Conclusions | | | | | | | |
| 9.1 Best Model Results | | | | | | | |
| In [80]: 1 # Imputing missing BORO labels | 2 df_1.loc[6780,3] = 'Brooklyn' | | | | | | |
| 2 Next Steps and Future | 3 df_1.loc[6781,3] = 'Brooklyn' | | | | | | |
| ▼ 9.3 Appendix- Deep NLP | 4 df_1.loc[10182,3] = 'Manhattan' | | | | | | |
| 9.3.1 Word2Vec Embedding | | | | | | | |
| 9.3.2 Classifying With Embed | | | | | | | |
| 9.3.3 Pretrained Vector | | | | | | | |
| 9.3.4 Mean Word Embed | | | | | | | |

```
In [81]: 1 # Looking at the distributions for each Boro
2 df_1['BORO'].value_counts()
```

executed in 21ms, finished 15:02:14 2022-08-12

```
Out[81]: Manhattan      5747
Queens        2954
Brooklyn      2943
Bronx         976
Staten Island  441
```

Contents

- 1 Predicting NYC Restaurants
- 2 Introduction
- 3 Obtaining Data
- 4 Scrubbing The Data
- 5 Exploring The Dataset
- 6 Preprocessing For Further
- 7 Preprocessing For Modeling
- 8 Models
- 9 Conclusions

```
In [82]: imports # Looking at Restaurants with missing zipcodes
```

```
3 Obtain      2 df_1[df_1['ZIPCODE'].isna()]
```

```
3.1 Obtaining Restaurant Info
```

executed in 54ms, finished 15:02:14 2022-08-12

- 3.1.1 Understanding NYC
- 3.1.2 Target Variable -- NYC
- 3.1.3 Engineering Target

- 3.2 Obtaining Yelp Businesses

- 3.2.1 Yelp Business Search

- 3.2.2 Yelp Reviews

- 3.2.3 Joining Yelp Reviews

- 3.3 Joining NYC DOHMH Data

- 4 Scrubbing The Data

- 4.1 Drop businesses missing zip codes

```
5 Exploring The Dataset
```

```
287 40387418
```

| | | | | | | |
|-----------------|-----------------|-----------|--------|--------------|-----|----|
| TAL BAGELS DELI | Bagels/Pretzels | Manhattan | 975979 | FIRST AVENUE | NaN | +1 |
|-----------------|-----------------|-----------|--------|--------------|-----|----|

- 6 Preprocessing For Further

- 6.0.1 Lets find most frequent locations

- 7 Preprocessing For Modeling

- 8 Models

- 8.1 Bag-of-Words Model

- 8.1.1 Model Evaluations

- 8.2 Text Preprocessing Pipeline

- 8.2.1 Multinomial Naive Bayes

```
8.2.2 Tuned Multinomial Naive Bayes
```

- 8.2.3 Logistic Regression

- 8.2.4 Tuned Logistic Regression

- 8.2.5 Decision Tree Model

- 8.2.6 Tuned DecisionTree Model

```
8.2.7 Support Vector Classification
```

- 8.2.8 Tuned SupportVector Classification

- 9 Conclusions

```
Out[83]: Best Model Results
```

- 9.2 Next Steps and Future

- 9.3 Appendix - Deep NLP

- 9.3.1 Word2Vec Embeddings

- 9.3.2 Classifying With Embeddings

- 9.3.3 Pretrained Vector Embeddings

- 9.3.4 Mean Word Embeddings

```
In [85]: 1 # Imputing Zipcodes missing from NYC data with Yelp value  
2 addrss_list = list(df_1['location'])  
3 zip_list = []  
4  
5 for zipcode in addrss_list:  
6     zip_list.append(zipcode['zip_code'])
```

Contents

- 1 Predicting NYC Restaurant Drop(columns=['ZIPCODE'], inplace=True)
 - 2 Introduction
 - executed in 60ms, finished 15:02:16 2022-08-12
 - 2.1 Objective

```
In [211]: # Converting all review text into string objects
```

- ```
In [8]: // Converting all review text into string objects
▼ 3 Obtain 2 df_1['Reviews'].astype(str,errors='raise')
 ▼ 3.1 Obtaining Restaurant Info
 executed in 19ms, finished 15:02:16 2022-08-12
 3.1.1 Understanding NYC
 Out[86]: Target Variable -Forgot my root beer and the burger tasted very weird.If I can give this place no stars I absolutely would. The workers here are in efficient. They do not work the best possible way. They cause th...
 ▼ 3.2 Obtaining Yelp Businesses
 1 I had my wedding here a month ago and I would give Riviera 10 stars if I could. It was such a joy and ease to work with Adam, Tommy, and their staff. A true diamond in the rough of Coney Island a...
 2 We've recently discovered this gem on Uber Eats and they have the best sandwiches!!The broccoli rabe fried eggplant mozzarella sandwich w
 3.3 Joining NYC DOHMH
 ▼ 4 Scrubbing The Data
 1 As amazing. Having a fresh green veggie paired with a light...
 4.1 Drop businesses missing
 5 Exploring The Data
 1 We've ordered from these folks a number of times on Grubhub and have always been extremely satisfied. We've never been to the actual stor
 ▼ 6 Preprocessing For Modeling
 1 But I have called over there a few times with some questi...
 6.0.1 Lets find most freq
 4 We've ordered from these folks a number of times on Grubhub and have always been extremely satisfied. We've never been to the actual stor
 7 Preprocessing For Modeling
 1 e but I have called over there a few times with some questi...
 ▼ 8 Models
```

## ▼ 8.1 Bag-of-Words Model

- 8.1.1 Model Evaluations 13056 This is my go to dessert place since its nearby and affordable.
  - ▼ 8.2 Text Preprocessing Part 1 The Ice-cream is extremely good, it tastes fresh and seems to be made with fresh ingredients as well. The coconut and green tea ...
  - 8.2.1 Multinomial Naive Bayes 13057 My family wanted bo zai fan and decided to come try here for our dinner. It was on 10/3. There was 5 of us (plus baby stroller) sitting outside under the tent in front of the shop (indoor dining had ju...)
  - 8.2.2 Tuned Multinomial Logistic Regression 13058 GREATEST PRICE FOR DIMSUM. You don't find prices like this anymore, not in NYC, not in Bay Area, not in Canada... It's \$3.50 per dish (usually) and is pretty darn good. It hits the spot without b...
  - 8.2.3 Tuned DecisionTreeModel 13059 I really like the drinks here that come with mochi. It's a layer of gooey sticky ricecake that's really fun to drink through a straw, as opposed to individual pieces of mochi. Also the drinks have...
  - 9 Conclusions 13060 Because I've been going here since I was 7 years old And never let me down some of the best food in myOpinion. Service is good but food is EXPENSIVE. We arrived around 9am and ordered f...
  - ▼ 9.3 Appendix: Deep NLP 13061 Word2Vec Embedding Names: Reviews, Length: 13061, dtype: object
  - 9.3.1 Classifying With Embedding
  - 9.3.2 Pretrained Vector Embedding
  - 9.3.3 Mean Word Embedding

## 5 Exploring The Dataset

In [558]: 1 df\_1['Severe'].value\_counts()

executed in 318ms, finished 12:23:11 2022-08-16

Out[558]: 0 9158  
1 3903  
Name: Severe, dtype: int64

## Contents ⚙️

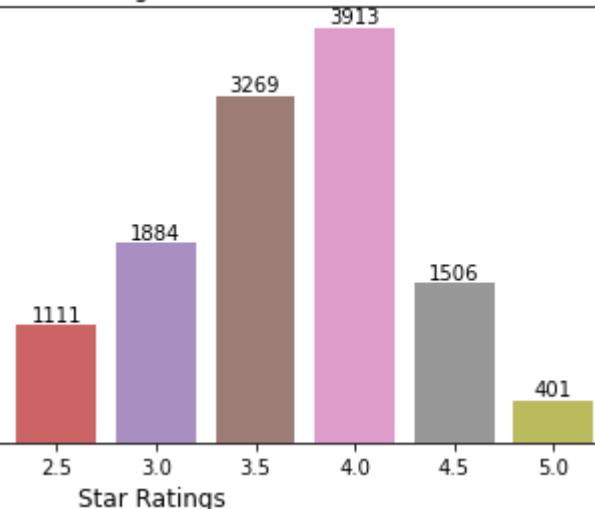
|                                                                                                                                                     |                                        |
|-----------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| In [87]: 1 # Dropping unnecessary columns                                                                                                           | 1 Predicting NYC Restaurant Data       |
| 2 eda_df = df_1.drop(columns=['display_phone', 'phone', 'location', 'coordinates', 'categories', 'is_closed', 'url', 'image_url', 'name', 'alias']) | 2 Introduction                         |
| 3 executed in 13ms, finished 15:02:16 2022-08-12                                                                                                    | 2.1 Objective                          |
| 2.1.1 Imports                                                                                                                                       |                                        |
| In [88]: 1 eda_df['price'].value_counts()                                                                                                           | 3 Obtain                               |
| 2 3.1 Obtaining Restaurant Data                                                                                                                     | 3.1.1 Understanding NYC Restaurants    |
| 3 3.1.3 Engineering Target Variable                                                                                                                 | 3.1.3 Engineering Target Variable      |
| 4 3.2 Obtaining Yelp Business Data                                                                                                                  | 3.2.1 Yelp Business Search             |
| 5 3.2.1.1 Price Categories                                                                                                                          | 3.2.1.1 Price Categories               |
| 6 3.2.2 Yelp Reviews                                                                                                                                | 3.2.2 Yelp Reviews                     |
| 7 3.2.3 Joining Yelp Reviews                                                                                                                        | 3.2.3 Joining Yelp Reviews             |
| In [89]: 1 df_1['rating'].value_counts()                                                                                                            | 8 Joining NYC Data                     |
| 2 4. Scrubbing The Data                                                                                                                             | 4.1 Drop businesses missing            |
| 3 4.00 3913                                                                                                                                         | Out[89]: 4.00 3913                     |
| 4 5 Exploring The Dataset                                                                                                                           | 5 Exploring The Dataset                |
| 5 3.50 3269                                                                                                                                         | 6 Preprocessing For Further Processing |
| 6 3.00 1884                                                                                                                                         | 6.0.1 Lets find most frequent          |
| 7 4.50 1506                                                                                                                                         | 7 Preprocessing For Modeling           |
| 8 2.00 588                                                                                                                                          | 8 Models                               |
| 9 4.01 501                                                                                                                                          | 8.1 Bag-of-Words Model                 |
| 10 5.01 212                                                                                                                                         | 8.1.1 Model Evaluation                 |
| 11 7 7                                                                                                                                              | 8.2 Text Preprocessing                 |
| 12 8.2.1 Multinomial Naive Bayes                                                                                                                    | 8.2.1 Multinomial Naive Bayes          |
| 13 8.2.2 Tuned Multinomial                                                                                                                          | 8.2.2 Tuned Multinomial                |
| 14 8.2.3 Logistic Regression                                                                                                                        | 8.2.3 Logistic Regression              |
| 15 8.2.4 Tuned Logistic Regression                                                                                                                  | 8.2.4 Tuned Logistic Regression        |
| 16 8.2.5 Decision Tree Model                                                                                                                        | 8.2.5 Decision Tree Model              |
| 17 8.2.6 Tuned DecisionTreeModel                                                                                                                    | 8.2.6 Tuned DecisionTreeModel          |
| 18 8.2.7 Support Vector Classifier                                                                                                                  | 8.2.7 Support Vector Classifier        |
| 19 8.2.8 Tuned SupportVectorClassifier                                                                                                              | 8.2.8 Tuned SupportVectorClassifier    |
| 20 9 Conclusions                                                                                                                                    | 9 Conclusions                          |
| 21 9.1 Best Model Results                                                                                                                           | 9.1 Best Model Results                 |
| 22 9.2 Next Steps and Future                                                                                                                        | 9.2 Next Steps and Future              |
| 23 9.3 Appendix - Deep NLP                                                                                                                          | 9.3 Appendix - Deep NLP                |
| 24 9.3.1 Word2Vec Embeddings                                                                                                                        | 9.3.1 Word2Vec Embeddings              |
| 25 9.3.2 Classifying With Embeddings                                                                                                                | 9.3.2 Classifying With Embeddings      |
| 26 9.3.3 Pretrained Vector Embeddings                                                                                                               | 9.3.3 Pretrained Vector Embeddings     |
| 27 9.3.4 Mean Word Embeddings                                                                                                                       | 9.3.4 Mean Word Embeddings             |

```
In [90]: 1 #Get the distribution of the ratings
2 x=eda_df['rating'].value_counts()
3 x=x.sort_index()
4 #plot
5 plt.figure(figsize=(8,4))
6 ax= sns.barplot(x.index, x.values, alpha=0.8)
7 plt.title("Star Rating Distribution")
8 plt.ylabel('# of businesses', fontsize=12)
9 plt.xlabel('Star Ratings ', fontsize=12)
```

## Contents ⚙️

- 1 Predicting NYC Restaurants
- 2 Introduction10
  - 2.1 Objective1 #adding the text labels
  - 2.1.1 Imports12
  - 2.1.2 Rects = ax.patches12
  - 2.1.3 Labels = x.values13
- 3 Obtain14
  - 3.1 Obtaining Restaurant Data15
  - 3.1.1 Understanding NYC15
  - 3.1.2 Target Variable -- N16
  - 3.1.3 Engineering Target17
  - 3.2 Obtaining Yelp Businesses18
  - 3.2.1 Yelp Business Seal18
  - 3.2.2 Yelp Reviews19
  - 3.2.3 Joining Yelp Reviews20
  - 3.3 Joining NYC DOHMH Data21
- 4 Scrubbing The Data22
  - 4.1 Drop businesses missing22
- 5 Exploring The Dataset23
- 6 Preprocessing For Further24
  - 6.0.1 Lets find most freq24
- 7 Preprocessing For Modeling25
- 8 Models26
  - 8.1 Bag-of-Words Model27
  - 8.1.1 Model Evaluations28
  - 8.2 Text Preprocessing Pic29
  - 8.2.1 Multinomial Naive Bayes29
  - 8.2.2 Tuned Multinomial30
  - 8.2.3 Logistic Regression30
  - 8.2.4 Tuned Logistic Regression31
  - 8.2.5 Decision Tree Model32
  - 8.2.6 Tuned DecisionTree32
  - 8.2.7 SupportVectorModel33
  - 8.2.8 Tuned SupportVector33
- 9 Conclusions1 eda\_df['rating'].value\_counts()
- 9.1 Best Model Results33
- 9.2 Next Steps and Future33
- 9.3 Appendix33
  - 9.3.1 Word2Vec Embedding33
  - 9.3.2 Classifying With Embedding33
  - 9.3.3 Pretrained Vector34
  - 9.3.4 Mean Word Embedding34

Star Rating Distribution



```
In [91]: 1 fig=plt.figure()
2 8.2.5 Decision Tree Model
3 executed in 17ms, finished 15:02:17 2022-08-12
4 8.2.6 Tuned DecisionTree
5 <Figure size 432x288 with 0 Axes>
6 8.2.7 SupportVectorModel
7 8.2.8 Tuned SupportVec
```

```
In [92]: 1 eda_df['rating'].value_counts()
```

```
9.1 Best Model Results
executed in 15ms, finished 15:02:17 2022-08-12
```

```
9.2 Next Steps and Future
```

```
Out[92]: 4 00 3913
```

```
3.50 3269
```

```
3.00 1884
```

```
4.50 1506
```

```
2.50 1111
```

```
2.00 588
```

```
5.00 401
```

```
1.50 212
```

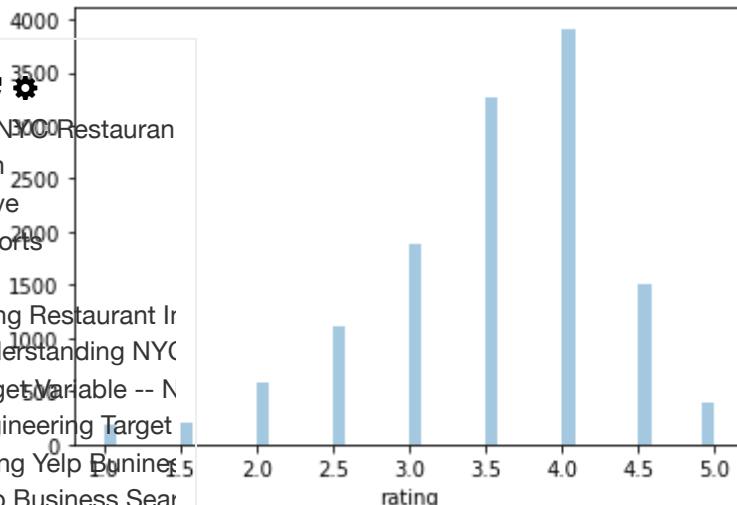
```
1.00 177
```

```
Name: rating, dtype: int64
```

In [93]: 1 sns.distplot(eda\_df.rating, kde=False)

executed in 446ms, finished 15:02:17 2022-08-12

Out[93]: <AxesSubplot:xlabel='rating'>



## Contents ⚙️

- 1 Predicting NYC Restaurant
- 2 Introduction
  - 2.1 Objective
    - 2.1.1 Imports
- 3 Obtain
  - 3.1 Obtaining Restaurant Info
    - 3.1.1 Understanding NYC Data
    - 3.1.2 Target Variable -- NYC Health Violations
    - 3.1.3 Engineering Target Variables
  - 3.2 Obtaining Yelp Businesses
    - 3.2.1 Yelp Business Search
    - 3.2.2 Yelp Reviews
- In [394]: cuisines = pd.DataFrame(eda\_df['CUISINE DESCRIPTION'].value\_counts())
 3.3 Joining NYC DOHMH Data
 cuisines.reset\_index(inplace=True)
- 4 Scrubbing The Data
  - cuisines[0:10]
  - 4.1 Drop businesses missing
- 5 Exploring The Dataset
- Out[394]:

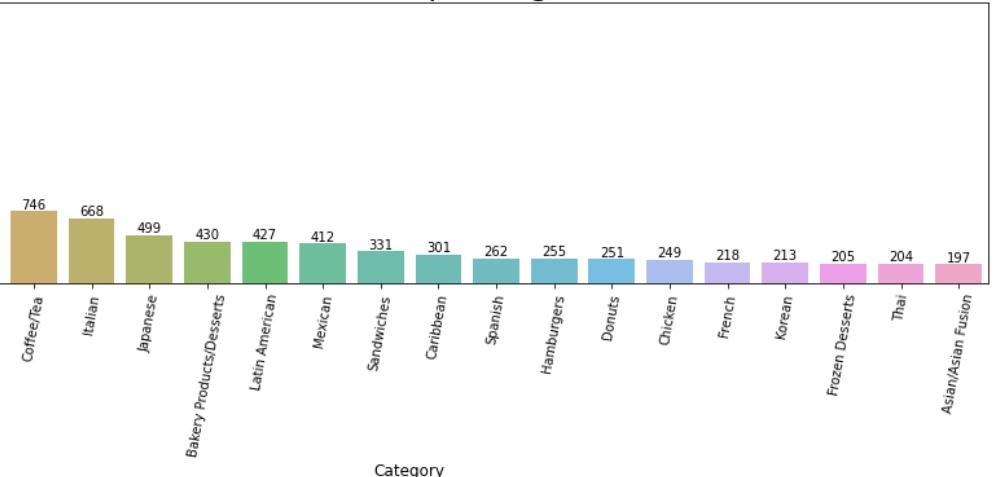
|                               | Index | CUISINE DESCRIPTION      | Count |
|-------------------------------|-------|--------------------------|-------|
| 6.0.1 Lets find most freq     | 0     | American                 | 2775  |
| 7 Preprocessing For Modelin   | 1     | Chinese                  | 1062  |
| 8 Models                      | 2     | Pizza                    | 922   |
| 8.1 Bag-of-Words Model        | 3     | Coffee/Tea               | 746   |
| 8.1.1 Model Evaluations       | 4     | Italian                  | 668   |
| 8.2 Text Preprocessing Pic    | 5     | Japanese                 | 499   |
| 8.2.1 Multinomial Naive Bayes | 6     | Bakery Products/Desserts | 430   |
| 8.2.2 Tuned Multinomial       | 7     | Decision Tree Model      | 427   |
| 8.2.3 Logistic Regression     | 8     | Tuned DecisionTree       | 427   |
| 8.2.4 Tuned Logistic Regre    | 9     | Support Vector Classifi  | 412   |
| 8.2.5 Decision Tree Mod       | 10    | Tuned SupportVec         | 331   |
| 9 Conclusions                 | 11    | Sandwiches               | 331   |
| 9.1 Best Model Results        | 12    |                          |       |
| 9.2 Next Steps and Future     | 13    |                          |       |
| 9.3 Appendix - Deep NLP       | 14    |                          |       |
| 9.3.1 Word2Vec Embedding      | 15    |                          |       |
| 9.3.2 Classifying With Embed  | 16    |                          |       |
| 9.3.3 Pretrained Vector Embed | 17    |                          |       |
| 9.3.4 Mean Word Embedding     | 18    |                          |       |

```
In [95]: 1 # What are the popular Cuisine categories?
2 cuisine_cats = eda_df['CUISINE DESCRIPTION']
3
4 x = cuisine_cats.value_counts()
5
6 print("There are ", len(x), " different types of cuisines in NYC")
7
```

## Contents ⚙️

- 1 Predicting NYC Restaurants
- 2 Introduction
  - 10 x = x.sort\_values(ascending=False)
  - 11 x = x.iloc[0:20]
  - 12 #chart
- 3 Obtain
  - 13 plt.figure(figsize=(16,4))
  - 14 ax = sns.barplot(x.index, x.values, alpha=0.8)*#,color=color[5]*
  - 15 plt.title("What are the top categories?", fontsize=25)
  - 16 locs, labels = plt.xticks()
  - 17 plt.setp(labels, rotation=80)
  - 18 plt.ylabel('# businesses', fontsize=12)
- 4 Scrubbing The Data
  - 21 for rect, label in zip(rects, labels):
  - 22 height = rect.get\_height()
  - 23 ax.text(rect.get\_x() + rect.get\_width()/2, height + 5, label, ha='c')
- 5 Exploring The Dataset
- 6 Preprocessing For Further
  - 27 28 plt.show()
  - 29 6.0.1 Lets find most freq
- 7 Preprocessing For Model
- 8 Models
  - 82 There are 82 different types of cuisines in NYC
  - 8.1 Bag-of-Words Model
    - 8.1.1 Model Evaluations
  - 8.2 Text Preprocessing Pip
    - 2500 275 8.2.1 Multinomial Naive Bayes
    - 2000 250 8.2.2 Tuned Multinomial
    - 1500 150 8.2.3 Logistic Regression
    - 1000 100 8.2.4 Tuned Logistic Regression
    - 500 50 8.2.5 Decision Tree Model
    - 500 50 8.2.6 Tuned DecisionTree
    - 500 50 8.2.7 Support Vector Classifier
    - 500 50 8.2.8 Tuned SupportVector
- 9 Conclusions
  - 9.1 Best Model Results
  - 9.2 Next Steps and Future
  - 9.3 Appendix - Deep NLP
    - 9.3.1 Word2Vec Embeddings
    - 9.3.2 Classifying With Embeddings
    - 9.3.3 Pretrained Vector Model
    - 9.3.4 Mean Word Embedding

What are the top categories?



In [96]: 1 eda\_df.columns

executed in 12ms, finished 15:02:18 2022-08-12

## Contents

|                                               |                                                                                                                                                                                                                                                        |
|-----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1 Predicting NYC Restaurants                  | 'CAMIS', 'DBA', 'CUISINE DESCRIPTION', 'BORO', 'BUILDING', 'STREET', 'PHONE', 'Latitude', 'Longitude', 'Community Board', 'Council District', 'Census Tract', 'Severe', 'id', 'review_count', 'rating', 'transactions', 'price', 'Reviews', 'zipcode'] |
| 2 Introduction                                |                                                                                                                                                                                                                                                        |
| 2.1 Objective                                 |                                                                                                                                                                                                                                                        |
| 2.1.1 Imports                                 |                                                                                                                                                                                                                                                        |
| 3 Obtain                                      | dtype='object')                                                                                                                                                                                                                                        |
| 3.1 Obtaining Restaurant Data                 |                                                                                                                                                                                                                                                        |
| 3.1.1 Understanding NYC Restaurants           |                                                                                                                                                                                                                                                        |
| 3.1.2 Target Variable -- NYC Restaurant Score |                                                                                                                                                                                                                                                        |
| 3.1.3 Engineering Target Variable             |                                                                                                                                                                                                                                                        |
| 3.2 Obtaining Yelp Business Data              |                                                                                                                                                                                                                                                        |
| 3.2.1 Yelp Business Search                    |                                                                                                                                                                                                                                                        |
| 3.2.2 Yelp Reviews                            |                                                                                                                                                                                                                                                        |
| 3.2.3 Joining Yelp Reviews                    |                                                                                                                                                                                                                                                        |
| 3.3 Joining NYC DOHMH Data                    |                                                                                                                                                                                                                                                        |
| 4 Scrubbing The Data                          |                                                                                                                                                                                                                                                        |
| 4.1 Drop businesses missing data              |                                                                                                                                                                                                                                                        |
| 5 Exploring The Dataset                       |                                                                                                                                                                                                                                                        |
| 6 Preprocessing For Further Modeling          |                                                                                                                                                                                                                                                        |
| 6.0.1 Lets find most frequent words           |                                                                                                                                                                                                                                                        |
| 7 Preprocessing For Modeling                  |                                                                                                                                                                                                                                                        |
| 8 Models                                      |                                                                                                                                                                                                                                                        |
| 8.1 Bag-of-Words Model                        |                                                                                                                                                                                                                                                        |
| 8.1.1 Model Evaluations                       |                                                                                                                                                                                                                                                        |
| 8.2 Text Preprocessing Pipeline               |                                                                                                                                                                                                                                                        |
| 8.2.1 Multinomial Naive Bayes                 |                                                                                                                                                                                                                                                        |
| 8.2.2 Tuned Multinomial Naive Bayes           |                                                                                                                                                                                                                                                        |
| 8.2.3 Logistic Regression                     |                                                                                                                                                                                                                                                        |
| 8.2.4 Tuned Logistic Regression               |                                                                                                                                                                                                                                                        |
| 8.2.5 Decision Tree Model                     |                                                                                                                                                                                                                                                        |
| 8.2.6 Tuned DecisionTree Model                |                                                                                                                                                                                                                                                        |
| 8.2.7 Support Vector Classification           |                                                                                                                                                                                                                                                        |
| 8.2.8 Tuned SupportVectorClassification       |                                                                                                                                                                                                                                                        |
| 9 Conclusions                                 |                                                                                                                                                                                                                                                        |
| 9.1 Best Model Results                        |                                                                                                                                                                                                                                                        |
| 9.2 Next Steps and Future Work                |                                                                                                                                                                                                                                                        |
| 9.3 Appendix - Deep NLP                       |                                                                                                                                                                                                                                                        |
| 9.3.1 Word2Vec Embeddings                     |                                                                                                                                                                                                                                                        |
| 9.3.2 Classifying With Embeddings             |                                                                                                                                                                                                                                                        |
| 9.3.3 Pretrained Vector Embeddings            |                                                                                                                                                                                                                                                        |
| 9.3.4 Mean Word Embeddings                    |                                                                                                                                                                                                                                                        |

In [97]:

```
1 # Most reviewed restaurants
2 eda_df[['CAMIS', 'id', 'DBA', 'review_count', 'rating']].sort_values(ascending=False)
```

executed in 47ms, finished 15:02:18 2022-08-12

Out[97]:

|                                      | CAMIS          | id                        | DBA                                     | review_count | rating |
|--------------------------------------|----------------|---------------------------|-----------------------------------------|--------------|--------|
| <b>Contents ↗</b>                    |                |                           |                                         |              |        |
| 1 Predicting NYC Restaurants         | 1108 50062888  | V7IXZKBDzScDeGB8JmnzSA    | KATZ'S DELICATESSEN (DEKALB MARKET)     | 13458        | 4.00   |
| 2 Introduction                       | 1108 40732665  | V7IXZKBDzScDeGB8JmnzSA    | KATZ'S DELICATESSEN                     | 13458        | 4.00   |
| 2.1 Objective                        | 6845 50005848  | 44SY464xDHbvOcjDzRbKkQ    | IPPUUDO                                 | 10219        | 4.00   |
| 2.1.1 Imports                        | 559 40402172   | Wlhm0W9197f_rRtDziq5qQ    | LOMBARDI'S                              | 6289         | 4.00   |
| 3 Obtain                             | 7658 50017243  | UA2M9QFZghe-9th2KwLoWQ    | BURGER & LOBSTER                        | 5592         | 4.00   |
| 3.1 Obtaining Restaurant Info        | 5036 41012461  | ga6sRtE0l85iftw_5-W84Q    | DOMINIQUE ANSEL BAKERY                  | 4971         | 4.00   |
| 3.1.1 Understanding Data             | 2308 41172734  | U5hCNNyJmb7f3dmC1HTzSQ    | JUNIOR'S RESTAURANT                     | 4971         | 4.00   |
| 3.1.3 Engineering Target             | 2306 41172734  | U5hCNNyJmb7f3dmC1HTzSQ    | JUNIOR'S RESTAURANT                     | 4970         | 4.00   |
| 3.2 Obtaining Yelp Businesses        | 5034 41812461  | ga6sRtE0l85iftw_5-W84Q    | DOMINIQUE ANSEL BAKERY                  | 4969         | 4.00   |
| 3.2.2 Yelp Reviews                   | 5155 41627819  | xt4sa64WOrpJvZBDPNPNYg    | JACOB'S PICKLES                         | 4765         | 4.00   |
| 3.3 Joining NYC POIs                 | 3007 41293936  | HfJhenvKDi-ds8SUJFwrLg    | MOMOFUKU NOODLE BAR                     | 4330         | 3.50   |
| 4 Scrubbing The Data                 | 2250 41159801  | nI1UYDCYUTt23TpGxqnLKg    | BUDDAKAN                                | 4259         | 4.00   |
| 4.1 Drop businesses missing          | 10242 50061233 | vfYhEpp0x-DrNjC6GSjjPQ    | ARTICHOKE BASILLE'S PIZZA               | 4062         | 4.00   |
| 5 Exploring The Data                 | 12978 50100033 | B3_K2kUVbYOU0VaLcj_LTw    | THAI VILLA                              | 4051         | 4.50   |
| 6 Preprocessing For Further Analysis | 9625 50055243  | 6.0.1 Lets find most freq | NAN XIANG XIAO LONG BAO                 | 4034         | 4.00   |
| 7 Preprocessing                      | 2409 4195733   | kBZggrnSP1kcUMnsnfkTaQ    | ESS-A-BAGEL                             | 3950         | 4.00   |
| 8 Models                             | 448 40396464   | c3eMI4_o4dPDDhPV_ibBYQ    | 230 FIFTH                               | 3943         | 3.00   |
| 8.1 Bag-of-Words Model               | 2508 41105733  | V6xIMNLFTsxBy4W0VvcRK_Q   | 230 FIFTH                               | 3942         | 3.00   |
| 8.2 Text Preprocessing Pipeline      | 8911 50045841  | V6xIMNLFTsxBy4W0VvcRK_Q   | BEST BAGEL & COFFEE                     | 3856         | 4.50   |
| 8.2.1 Multinomial Naive Bayes        | 11060 50071474 | j1S3NUrkB3BVT49n_e76NQ    | IPPUUDO NY                              | 3843         | 4.00   |
| 8.2.2 Tuned Multinomial              | 6311 50000923  | C8j0q4Ma_S5hBGuAl-aaww    | DI FARÀ PIZZERIA                        | 3739         | 4.00   |
| 8.2.3 Logistic Regression            | 4561 41543887  | pfmAcS-g6SiNG0KILvrqnA    | BEAUTY AND ESSEX                        | 3641         | 4.00   |
| 8.2.4 Tuned Logistic Regression      | 4554 41543410  | fxGpXRxFUDzIU3Cyszu4uQ    | RED ROOSTER HARLEM, GINNY'S SUPPER CLUB | 3499         | 3.50   |
| 9 Conclusions                        | 9910 50058139  | uoT2_Wrt1noD6kZOId7tUg    | JANE                                    | 3330         | 4.00   |
| 9.1 Best Model Results               | 7641 50017190  | IWOkeS-wV4no8qqA9OwwEg    | DOUGHNUT PLANT                          | 3312         | 4.50   |
| 9.3 Appendix - Deep NLP              | 7647 50079524  | IWOkeS-wV4no8qqA9OwwEg    | DOUGHNUT PLANT                          | 3312         | 4.50   |
| 9.3.1 Word2Vec Embedding             | 8689 50043740  | QVWmn5jmy_TPmg0G71Dk5g    | SEA THAI                                | 3290         | 3.50   |
| 9.3.3 Pretrained Vector Library      | 996 40685858   | b8a-8u_A51v2lzyjLVsx6w    | DEL FRISCO'S STEAKHOUSE                 | 3209         | 4.00   |
| 9.3.4 Mean Word Embedding            | 4013 41453428  | ikazsJps1k-Br2FbunwCtg    | SPOT                                    | 3200         | 4.00   |
|                                      | 3193 50012117  | FhXjAc6nKLf414KxYujUhw    | ROBERTA'S PIZZA & BAKERY                | 3077         | 4.00   |
|                                      | 3191 41307383  | FhXjAc6nKLf414KxYujUhw    | ROBERTA'S                               | 3077         | 4.00   |

| CAMIS                                           |                                                                    | id                     | DBA                    | review_count          | rating |      |
|-------------------------------------------------|--------------------------------------------------------------------|------------------------|------------------------|-----------------------|--------|------|
| 3192                                            | 50001089                                                           | FhXjAc6nKlf414KxYujUhw | BLANCA                 | 3077                  | 4.00   |      |
| 3407                                            | 41351524                                                           | A_YpTLbAlEqeLVSS9bxbeA | CLUB A STEAKHOUSE      | 3076                  | 4.50   |      |
| 2259                                            | 41162796                                                           | dMhRafXdr765DHe0k-QfaQ | ABC KITCHEN            | 3072                  | 4.00   |      |
| 5564                                            | 41664978                                                           | veq1Bl1DW3UWMekZJUsG1Q | GRAMERCY TAVERN        | 3072                  | 4.50   |      |
| 5563                                            | 41664978                                                           | veq1Bl1DW3UWMekZJUsG1Q | GRAMERCY TAVERN        | 3071                  | 4.50   |      |
| 1 Predicting NYC Restaurant                     |                                                                    |                        |                        |                       |        |      |
| ▼ 2 Introduction                                | 404                                                                | 40395055               | QObHX0yR6zd0WfksRDbJTA | BECCO                 | 2936   | 3.50 |
| ▼ 2.1 Objective                                 | 405                                                                | 40395055               | QObHX0yR6zd0WfksRDbJTA | BECCO                 | 2936   | 3.50 |
| 2.1.1 Imports                                   |                                                                    |                        |                        |                       |        |      |
| ▼ 3 Obtain                                      | 5095                                                               | 41620680               | C-8mGN7lt5rlraO5n-15ug | THE SMITH             | 2871   | 4.00 |
| ▼ 3.1 Obtaining Restaurant Jr.                  | 5094                                                               | 41620680               | C-8mGN7lt5rlraO5n-15ug | THE SMITH             | 2870   | 4.00 |
| 3.1.1 Understanding NYC                         |                                                                    |                        |                        |                       |        |      |
| 3.1.2 Target Table                              | 252                                                                | 40384528               | CVxz9B5ncpMwVK1bk_c1YA | VESELKA RESTAURANT    | 2805   | 4.00 |
| 3.1.3 Engineering Target                        | 250                                                                | 40384528               | CVxz9B5ncpMwVK1bk_c1YA | VESELKA RESTAURANT    | 2803   | 4.00 |
| ▼ 3.2 Obtaining Yelp Business                   | 5040                                                               | 41614161               | tsiC3VpO-hZEb98Qk3p-DQ | AMY RUTH'S RESTAURANT | 2793   | 4.00 |
| 3.2.1 Yelp Business Seal                        |                                                                    |                        |                        |                       |        |      |
| 3.2.2 Yelp Reviews                              | 164                                                                | 40373656               | pQuZDnefX038xurT1BhDXA | KEENS STEAKHOUSE      | 2791   | 4.00 |
| 3.2.3 Joining Yelp Review                       |                                                                    |                        |                        |                       |        |      |
| 3.3 Joining NYC DOHMH                           | 765                                                                | 40575863               | uc5qQMzs96rzjK27epDCug | FAMOUS JOE'S PIZZA    | 2780   | 4.00 |
| ▼ 4 Scrubbing The Data                          | 5351                                                               | 41644048               | ETgJqJHV7BW6plr9Ox74sA | AMELIE                | 2779   | 4.50 |
| 4.1 Drop businesses missing                     |                                                                    |                        |                        |                       |        |      |
| 5 Exploring The Dataset                         | 2957                                                               | 41280221               | ogCC-IJJYnwXDvKGmKZ6Sw | WOORIJIP              | 2726   | 4.00 |
| ▼ 6 Preprocessing                               | 214                                                                | 40795021               | TU_BU9HLfYI2xlyqVQdRA  | TAO RESTAURANT        | 2713   | 3.50 |
| 6.0.1 Lets find most freq                       | 1215                                                               | 40795021               | TU_BU9HLfYI2xlyqVQdRA  | TAO RESTAURANT        | 2713   | 3.50 |
| 7 Preprocessing For Modelin                     |                                                                    |                        |                        |                       |        |      |
| ▼ 8 Models                                      | 4356                                                               | 41508109               | MhceelrM2_3mcKw5EGLY6A | PURE THAI RESTAURANT  | 2684   | 4.00 |
| ▼ 8.1 Bag-of-Words Model                        |                                                                    |                        |                        |                       |        |      |
| 8.1.1 Model Evaluations                         |                                                                    |                        |                        |                       |        |      |
| In [98]:                                        | 1 # Mapping prices to ordinal categories                           |                        |                        |                       |        |      |
| ▼ 8.2 Text Preprocessing                        | 2 price_dict = { '\$\$\$\$': 4, '\$\$\$': 3, '\$\$': 2, '\$': 1 }  |                        |                        |                       |        |      |
| 8.2.1 Multinomial Naive                         |                                                                    |                        |                        |                       |        |      |
| 8.2.2 Tuned Multinomial                         | 3 eda_df['prices'] = eda_df['price'].map(price_dict)               |                        |                        |                       |        |      |
| 8.2.3 Logistic Regression                       |                                                                    |                        |                        |                       |        |      |
| executed in 27ms, finished 15:02:18 2022-08-12  |                                                                    |                        |                        |                       |        |      |
| 8.2.4 Tuned Logistic Reg                        |                                                                    |                        |                        |                       |        |      |
| 8.2.5 Decision Tree Mod                         |                                                                    |                        |                        |                       |        |      |
| In [99]:                                        | 1 # Imputing a 1 for NAN values                                    |                        |                        |                       |        |      |
| 8.2.6 Tuned DecisionTree                        | 2 eda_df['prices'].fillna(1, inplace=True)                         |                        |                        |                       |        |      |
| 8.2.7 Support Vector Clas                       |                                                                    |                        |                        |                       |        |      |
| executed in 10ms, finished 15:02:18 2022-08-12  |                                                                    |                        |                        |                       |        |      |
| 8.2.8 Tuned SupportVec                          |                                                                    |                        |                        |                       |        |      |
| ▼ 9 Conclusions                                 |                                                                    |                        |                        |                       |        |      |
| In [100]:                                       | 1 transactions = eda_df['transactions']                            |                        |                        |                       |        |      |
| 9.1 Best Model Results                          |                                                                    |                        |                        |                       |        |      |
| 9.2 Next Steps and Future                       |                                                                    |                        |                        |                       |        |      |
| ▼ 9.3 Appendix - Deep NLP                       |                                                                    |                        |                        |                       |        |      |
| In [101]:                                       | Word2Vec# Embed module to convert string objects into list         |                        |                        |                       |        |      |
| 9.3.2 Classifying Transactions                  | 2 transactions = transactions.apply(lambda x: ast.literal_eval(x)) |                        |                        |                       |        |      |
| 9.3.3 Pretrained Vectors                        |                                                                    |                        |                        |                       |        |      |
| executed in 295ms, finished 15:02:19 2022-08-12 |                                                                    |                        |                        |                       |        |      |
| 9.3.4 Mean Word Embed                           |                                                                    |                        |                        |                       |        |      |

In [102]:

```

1 # Change list into dicts
2 dcts = transactions.apply(lambda x: {c: 1 for c in x})
3
4 # Create new dataframe based on the list of dictionaries.
5 ohe_df = pd.DataFrame(dcts.tolist()).fillna(0)
6

```

## Contents

executed in 52ms, finished 15:02:19 2022-08-12

1 Predicting NYC Restaurant

In [103]: 1 ohe\_df

executed in 24ms, finished 15:02:19 2022-08-12

Out[103]:

|                                          | delivery | pickup | restaurant_reservation |
|------------------------------------------|----------|--------|------------------------|
| ▼ 3 Obtain                               |          |        |                        |
| ▼ 3.1 Obtaining Restaurant               | 1.00     | 0.00   | 0.00                   |
| 3.1.1 Understanding NYC                  | 1.00     | 0.00   | 0.00                   |
| 3.1.2 Target Variable -- N               | 1.00     | 0.00   | 0.00                   |
| 3.1.3 Engineering Target                 | 1.00     | 1.00   | 0.00                   |
| ▼ 3.2 Obtaining Yelp Business            | 1.00     | 1.00   | 0.00                   |
| 3.2.1 Yelp Business Search               | 1.00     | 0.00   | 0.00                   |
| 3.2.2 Yelp Reviews                       | 1.00     | 1.00   | 0.00                   |
| 3.2.3 Joining Yelp Review                | ...      | ...    | ...                    |
| 3.3 Joining NYC DOHMH Data               | 1.00     | 0.00   | 0.00                   |
| ▼ 4 Scrubbing The Data                   | 1.00     | 0.00   | 0.00                   |
| 4.1 Drop businesses missing              | 1.00     | 0.00   | 0.00                   |
| 5 Exploring The Dataset                  | 1.00     | 0.00   | 0.00                   |
| ▼ 6 Preprocessing For Further            | 1.00     | 0.00   | 1.00                   |
| 6.0.1 Lets find the most freq            | 1.00     | 0.00   | 0.00                   |
| 7 Preprocessing For Modeling             | 1.00     | 0.00   | 0.00                   |
| ▼ 8 Models                               | 1.00     | 1.00   | 0.00                   |
| ▼ 8.1 Bag-of-Words Model                 | 1.00     | 0.00   | 0.00                   |
| 8.1.1 Model Evaluations                  | 1.00     | 0.00   | 0.00                   |
| ▼ 8.2 Text Preprocessing Pip             | 1.00     | 0.00   | 0.00                   |
| 8.2.1 Multinomial Naive Bayes            | 1.00     | 0.00   | 0.00                   |
| 8.2.2 Tuned Multinomial Naive Bayes      | 1.00     | 0.00   | 0.00                   |
| 8.2.3 Logistic Regression                | 1.00     | 0.00   | 0.00                   |
| 8.2.4 Tuned Logistic Regression          | 1.00     | 0.00   | 0.00                   |
| 8.2.5 Decision Tree Model                | 1.00     | 0.00   | 0.00                   |
| 8.2.6 Tuned DecisionTree Model           | 1.00     | 0.00   | 0.00                   |
| 8.2.7 Support Vector Classification      | 1.00     | 0.00   | 0.00                   |
| 8.2.8 Tuned SupportVector Classification | 1.00     | 0.00   | 0.00                   |
| ▼ 9 Conclusions                          |          |        |                        |
| 9.1 Best Model Results                   |          |        |                        |
| 9.2 Next Steps and Future                |          |        |                        |
| ▼ 9.3 Appendix - Deep NLP                |          |        |                        |
| 9.3.1 Word2Vec Embedding                 |          |        |                        |
| 9.3.2 Classifying With Embedding         |          |        |                        |
| 9.3.3 Pretrained Vector Embedding        |          |        |                        |
| 9.3.4 Mean Word Embedding                |          |        |                        |

```
In [104]: 1 # Concatenate dummy variables with the full dataset
2 eda_df = pd.concat([eda_df,ohe_df],axis=1)
3 eda_df.head()
```

executed in 39ms, finished 15:02:19 2022-08-12

Out[104]:

| CAMIS | DBA | CUISINE DESCRIPTION | BORO | BUILDING | STREET | PHONE | Latitude |
|-------|-----|---------------------|------|----------|--------|-------|----------|
|-------|-----|---------------------|------|----------|--------|-------|----------|

## Contents

- 1 Predicting NYC Restaurant
- ▼ 2 Introduction
  - ▼ 2.1 Objective
    - 2.1.1 Imports
  - ▼ 3 Obtain
    - 3.1 Obtaining Restaurant Info
      - 3.1.1 Understanding NYC Restaurants
      - 3.1.2 Target Variable -- Number of Violations
      - 3.1.3 Engineering Target Variable
    - 3.2 Obtaining Yelp Business Data
      - 3.2.1 Yelp Business Search
      - 3.2.2 Yelp Reviews
      - 3.2.3 Joining Yelp Reviews
      - 3.3 Joining NYC DOHMH Data
  - ▼ 4 Scrubbing The Data
    - 4.1 Drop businesses missing address
    - 5 Exploring The Dataset
  - ▼ 6 Preprocessing For Further Modeling
    - 6.0.1 Lets find most frequent words
    - 7 Preprocessing For Modeling
  - ▼ 8 Models
    - ▼ 8.1 Bag-of-Words Model
      - 8.1.1 Model Evaluations
    - ▼ 8.2 Text Preprocessing Pipeline
      - 8.2.1 Multinomial Naive Bayes
      - 8.2.2 Tuned Multinomial Naive Bayes
      - 8.2.3 Logistic Regression
      - 8.2.4 Tuned Logistic Regression
      - 8.2.5 Decision Tree Model
      - 8.2.6 Tuned DecisionTree Model
      - 8.2.7 Support Vector Classification
      - 8.2.8 Tuned SupportVectorClassification
  - ▼ 9 Conclusions
    - 9.1 Best Model Results
    - 9.2 Next Steps and Future
  - ▼ 9.3 Appendix - Deep NLP
    - 9.3.1 Word2Vec Embeddings
    - 9.3.2 Classifying With Embeddings
    - 9.3.3 Pretrained Word2Vec Model
    - 9.3.4 Mean Word Embedding

|   |          |                    |                 |          |      |                  |              |       |
|---|----------|--------------------|-----------------|----------|------|------------------|--------------|-------|
| 0 | 30112340 | WENDY'S            | Hamburgers      | Brooklyn | 469  | FLATBUSH AVENUE  | +17182875005 | 40.66 |
| 1 | 40356018 | RIVIERA CATERERS   | American        | Brooklyn | 2780 | STILLWELL AVENUE | +17183723031 | 40.58 |
| 2 | 40356483 | WILKEN'S FINE FOOD | Sandwiches      | Brooklyn | 7114 | AVENUE U         | +17184443838 | 40.62 |
| 3 | 40356739 | TROPICS ICE CREAM  | Frozen Desserts | Brooklyn | 1839 | NOSTRAND AVENUE  | +17188560821 | 40.64 |

| CAMIS | DBA | CUISINE<br>DESCRIPTION | BORO | BUILDING | STREET | PHONE | Latitude | Longitude |
|-------|-----|------------------------|------|----------|--------|-------|----------|-----------|
|-------|-----|------------------------|------|----------|--------|-------|----------|-----------|

## Contents ⚙️

1 Predicting NYC Restaurant 40356731

2 Introduction

2.1 Objective

2.1.1 Imports

3 Obtain

3.1 Obtaining Restaurant Info

3.1.1 Understanding NYC Data

3.1.2 Target Variable -- NYC Violations

3.1.3 Engineering Target Variable

In [105]: `eda_df.drop(columns=['price', 'transactions'], inplace=True)`

3.2.1 Yelp Business Search

3.2.2 Yelp Reviews

3.2.3 Joining Yelp Reviews

In [106]: `# Look at all the features compared to the 0 class and 1 class`

3.3 Joining NYC DOHMH Data

4 Scrubbing The Data

4.1 Drop businesses missing data

In [107]: `labels_df = eda_df['Severe']`

5 Exploring The Dataset

6 Preprocessing For Further Analysis

6.0.1 Let's explore the data

7 Preprocessing For Modeling

out[107]: `(13061, 1)`

8 Models

8.1 Bag-of-Words Model

8.1.1 Model Evaluations

8.2 Text Preprocessing Pipeline

8.2.1 Multinomial Naive Bayes

8.2.2 Tuned Multinomial Naive Bayes

8.2.3 Logistic Regression

8.2.4 Tuned Logistic Regression

8.2.5 Decision Tree Model

8.2.6 Tuned DecisionTree Model

8.2.7 Support Vector Classification

8.2.8 Tuned SupportVector Classification

9 Conclusions

9.1 Best Model Results

9.2 Next Steps and Future Work

9.3 Appendix - Deep NLP

9.3.1 Word2Vec Embeddings

9.3.2 Classifying With Embeddings

9.3.3 Pretrained Vector Embeddings

9.3.4 Mean Word Embeddings

TASTE THE  
TROPICS  
ICE  
CREAM

Frozen  
Desserts

Brooklyn

1839

NOSTRAND  
AVENUE

+17188560821

40.64

```
In [108]: 1 fig, ax = plt.subplots(figsize=(9.2, 5))
2
3 n_obs = labels_df.shape
4
5 (eda_df['Severe']
6 .value_counts()
7 .plot.barh(title="Proportion of Restaurants with Severe Violations"
8))
9
```

## Contents ⚙️

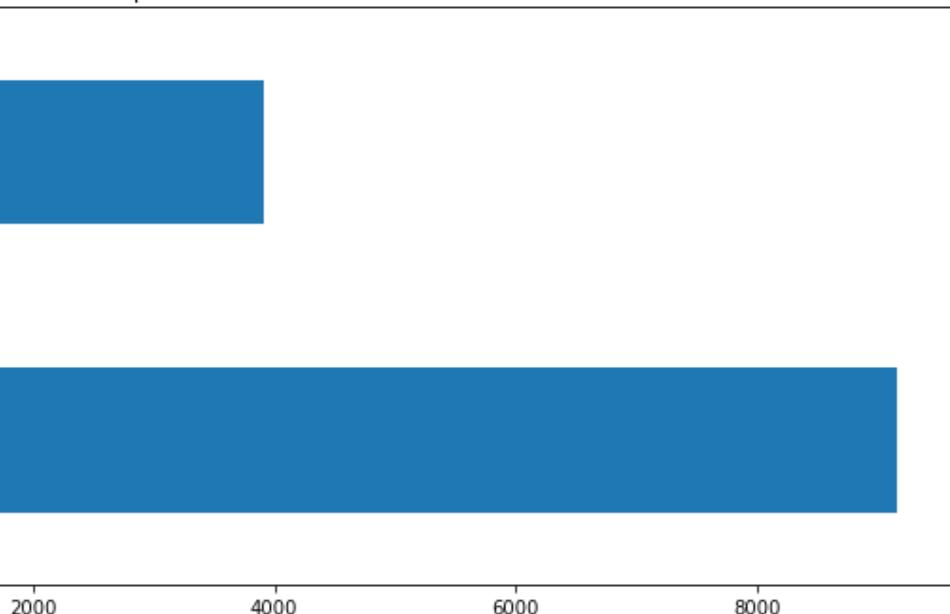
- 1 Predicting NYC Restaurants with Severe Violations
- 2 Introduction
- 2.1 Objective
- 2.1.1 Imports

Executed in 303ms, finished 15:02:19 2022-08-12

## 3 Obtain

- 3.1 Obtaining Restaurant Data
- 3.1.1 Understanding NYC Restaurants
- 3.1.2 Target Variable -- NYC DOHMH Violations
- 3.1.3 Engineering Target Variable
- 3.2 Obtaining Yelp Business Data
- 3.2.1 Yelp Business Search
- 3.2.2 Yelp Reviews
- 3.2.3 Joining Yelp Reviews
- 3.3 Joining NYC DOHMH Violations
- 4 Scrubbing The Data
- 4.1 Drop businesses missing reviews
- 5 Exploring The Dataset
- 6 Preprocessing For Further Modeling
- 6.0.1 Lets find the most frequent business names
- 7 Preprocessing For Modeling
- 8 Models

Proportion of Restaurants with Severe Violations



```
In [109]: 1 boro_counts = (eda_df[['BORO', 'Severe']]
2 .groupby(['BORO', 'Severe'])
3 .size()
4 .unstack('Severe'))
5
```

Executed in 27ms, finished 15:02:19 2022-08-12

```
Out[109]: Severe 0 1
```

## 9 Conclusions

### BORO

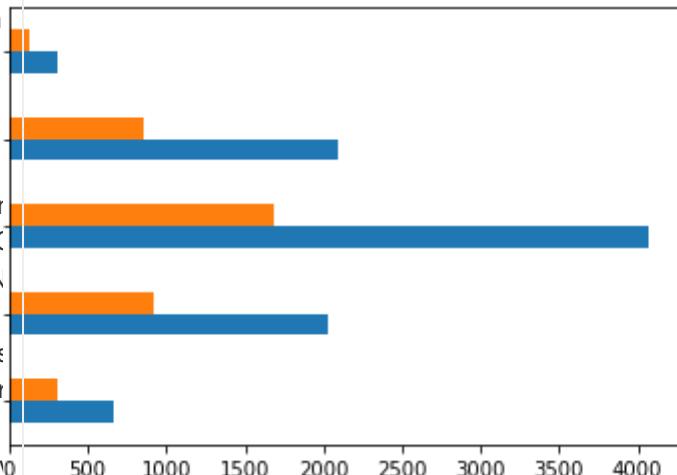
- 9.1 Best Model Results
  - 9.2 Next Steps and Future Work
  - 9.3 Appendix - Deep NLP
  - 9.3.1 Word2Vec Embedding
  - 9.3.2 Classifying NYC Restaurants
  - 9.3.3 Pretrained Vector Embedding
  - 9.3.4 Mean Word Embedding
- | BORO          | 0    | 1    |
|---------------|------|------|
| Bronx         | 664  | 312  |
| Brooklyn      | 2025 | 918  |
| Manhattan     | 4064 | 1683 |
| Queens        | 2092 | 862  |
| Staten Island | 313  | 128  |

```
In [110]: 1 ax = boro_counts.plot.barh()
2 ax.legend(
3 loc='center right',
4 bbox_to_anchor=(1.3, 0.5),
5 title='Severe Violations'
6);
```

executed in 393ms, finished 15:02:20 2022-08-12

## Contents

- 1 Predicting NYC Restaurant
- 2 Introduction Staten Island
  - 2.1 Objective
  - 2.1.1 Imports Queens
- 3 Obtain
  - 3.1 Obtaining Restaurant Info Manhattan
    - 3.1.1 Understanding NYC
    - 3.1.2 Target Variable -- NYC
    - 3.1.3 Engineering Target
  - 3.2 Obtaining Yelp Businesses Bronx
    - 3.2.1 Yelp Business Search Bronx
    - 3.2.2 Yelp Reviews
    - 3.2.3 Joining Yelp Reviews
  - 3.3 Joining NYC DOHMH Data
- 4 Scrubbing The Data
  - 4.1 Drop businesses missing
- 5 Exploring The Dataset
- 6 Preprocessing For Further
  - 6.0.1 Lets find most frequent boroughs
  - 6.0.2 Severe\_boro\_counts = boro\_counts.sum(axis='columns')



```
In [111]: 1 severe_boro_counts = boro_counts.sum(axis='columns')
2 severe_boro_counts
```

executed in 10ms, finished 15:02:20 2022-08-12

## 8 Models

- 8.1 Bag-of-Words Model
- 8.1.1 Model Evaluations
 

| Borough       | Score |
|---------------|-------|
| Bronx         | 976   |
| Brooklyn      | 2943  |
| Manhattan     | 5747  |
| Queens        | 2954  |
| Staten Island | 441   |
- 8.2 Text Preprocessing Pipeline
  - 8.2.1 Multinomial Naive Bayes
  - 8.2.2 Tuned Multinomial
  - 8.2.3 Logistic Regression
  - 8.2.4 Tuned Logistic Regression
  - 8.2.5 Decision Tree Model
  - 8.2.6 Tuned DecisionTree

```
In [112]: 1 boro_props = boro_counts.div(severe_boro_counts, axis='index')
2 boro_props
```

executed in 20ms, finished 15:02:20 2022-08-12

```
Out[112]: Best Model Results
Severe 0 1
9.2 Next Steps and Future
BORO
9.3 Appendix - Deep NLP
9.3.1 Word2Vec Embedding
9.3.2 Classifying With Embedding
9.3.3 Pretrained Vector Model
9.3.4 Mean Word Embedding
Queens 0.71 0.29
Staten Island 0.71 0.29
```

In [113]:

```

1 # Prototyping Stack Barh plot
2 ax = boro_props.plot.banh(stacked=True)
3 ax.set_title('Severe Violations by Boro')
4 ax.set_xlabel('Proportion of Restaurants w/Severe Violations ')
5 ax.legend(
6 loc='center left',
7 bbox_to_anchor=(1.05, 0.5),
8 title='Severe Violations'

```

## Contents ⚙️

- 1 Predicting NYC Restaurant
- ▼ 2 Introduction      executed in 365ms, finished 15:02:20 2022-08-12
  - ▼ 2.1 Objective
  - 2.1.1 Imports

### ▼ 3 Obtain

- ▼ 3.1 Obtaining Restaurant Info

3.1.1 Understanding NYC Data

3.1.2 Target Variable

3.1.3 Engineering Target

- ▼ 3.2 Obtaining Yelp Businesses

3.2.1 Yelp Business Search

3.2.2 Yelp Reviews

3.2.3 Joining Yelp Reviews

3.3 Joining NYC DOHMH Data

- ▼ 4 Scrubbing The Data

4.1 Drop businesses missing

- 5 Exploring The Dataset

- ▼ 6 Preprocessing For Further

6.0.1 Lets find most freq

- 7 Preprocessing For Modeling

- ▼ 8 Models:

```
1 def violation_rate_plot(col, target, data, ax=None):
```

```
2 """Stacked bar chart of severe health and safety violations rate against
```

8.1.1 Model Evaluations

- ▼ 8.2 Text Preprocessing Pic

8.2.1 Multinomial Naive Bayes

8.2.2 Tuned Multinomial

8.2.3 Logistic Regression

8.2.4 Tuned Logistic Reg

8.2.5 Decision Tree Mod

8.2.6 Tuned DecisionTree

8.2.7 Support Vector Clas

8.2.8 Tuned SupportVec

**Args:**

```

col (string): column name of feature variable
target (string): column name of target variable
data (pandas DataFrame): dataframe that contains columns
 `col` and `target`
ax (matplotlib axes object, optional): matplotlib axes
 object to attach plot to

```

- ▼ 9 Conclusions

9.1 Best Model Results

9.2 Next Steps and Future

- ▼ 9.3 Appendix - Deep NLP

9.3.1 Word2Vec Embedding

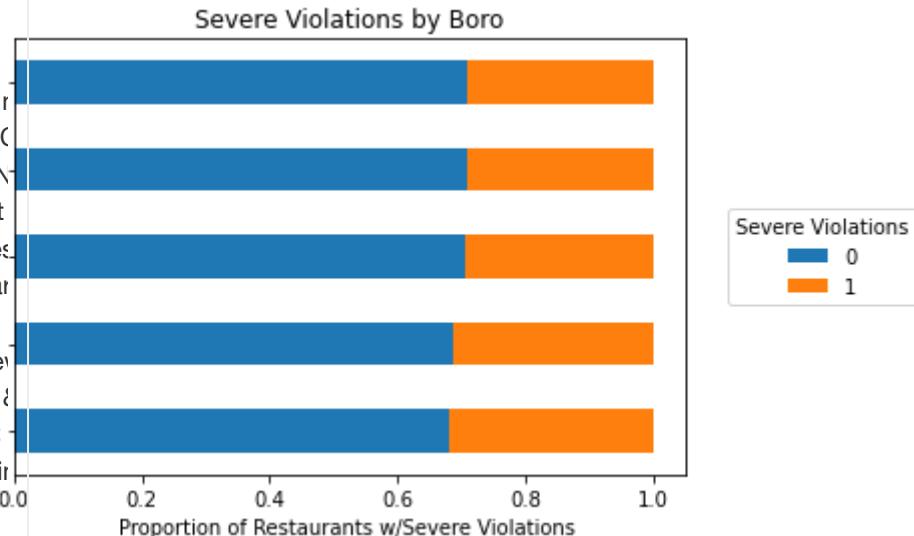
9.3.2 Classifying With Embed

9.3.3 Pretrained Vector Embed

9.3.4 Mean2Word Embedding

```
23 ax.legend().remove()
```

executed in 11ms, finished 15:02:20 2022-08-12



```
In [115]: 1 eda_df.columns
```

executed in 11ms, finished 15:02:20 2022-08-12

```
Out[115]: Index(['CAMIS', 'DBA', 'CUISINE DESCRIPTION', 'BORO', 'BUILDING', 'STREET',
 'PHONE', 'Latitude', 'Longitude', 'Community Board', 'Council Dist
```

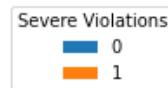
## Contents ⚙️

- 1 Predicting NYC Restaurant Data
- 2 Introduction
  - 2.1 Objective, Scope
  - 2.1.1 Imports: pd.read\_csv('nyc\_open\_data.csv', dtype='object')
- 3 Obtain
  - 3.1 Obtaining Restaurant Data
    - 3.1.1 Understanding NYC Restaurants
    - 3.1.2 Target Variable -- NYC Health Violations
    - 3.1.3 Engineering Target Variable
  - 3.2 Obtaining Yelp Business Data
    - 3.2.1 Yelp Business Search
    - 3.2.2 Yelp Reviews
    - 3.2.3 Joining Yelp Reviews
    - 3.3 Joining NYC DOHMH Data
- 4 Scrubbing The Data
  - 4.1 Drop businesses missing address
- 5 Exploring The Dataset
- 6 Preprocessing For Further Modeling
  - 6.0.1 Lets find most frequent business types
  - 6.0.2 Let's look at the distribution of violations
- 7 Preprocessing For Modeling
  - 7.1 Text Preprocessing Pipeline
  - 7.2 Feature Engineering
- 8 Models
  - 8.1 Bag-of-Words Model
    - 8.1.1 Model Evaluations
  - 8.2 Text Preprocessing Pipeline
    - 8.2.1 Multinomial Naive Bayes
    - 8.2.2 Tuned Multinomial Naive Bayes
    - 8.2.3 Logistic Regression
    - 8.2.4 Tuned Logistic Regression
    - 8.2.5 Decision Tree Model
    - 8.2.6 Tuned DecisionTree Model
    - 8.2.7 Support Vector Classification
    - 8.2.8 Tuned SupportVectorClassification
- 9 Conclusions
  - 9.1 Best Model Results
  - 9.2 Next Steps and Future Work
  - 9.3 Appendix - Deep NLP
    - 9.3.1 Word2Vec Embeddings
    - 9.3.2 Classifying With Embeddings
    - 9.3.3 Pretrained Vector Model
    - 9.3.4 Mean Word Embedding

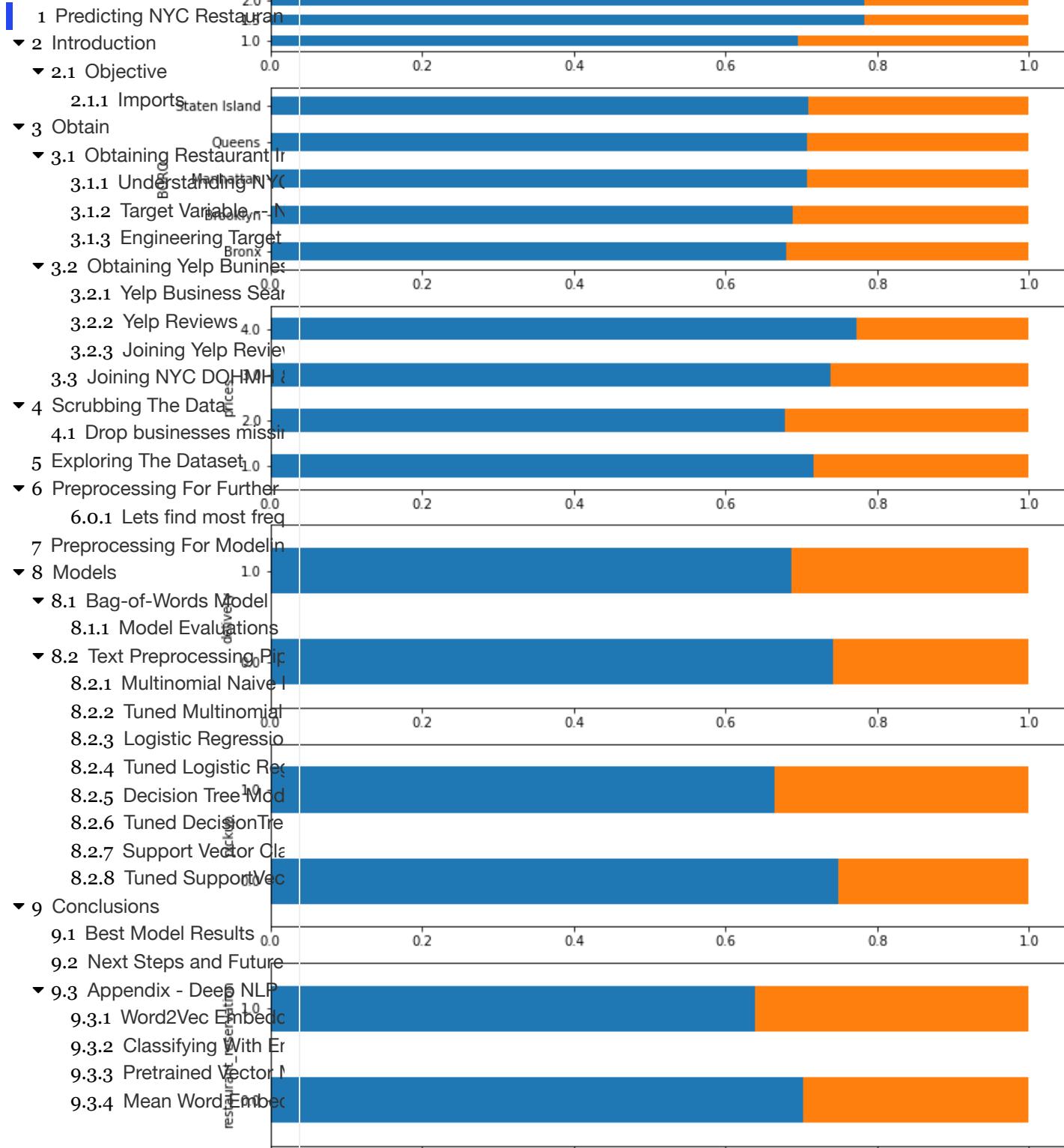
```
In [212]: 1 # Loop through several columns and plot against both severe violations.
2
3 cols_to_plot = [
4 'rating',
5 'BORO',
6 'prices',
7 'delivery',
8 'pickup',
9 'restaurant_reservation'
10]
11
12 fig, ax = plt.subplots(
13 len(cols_to_plot), figsize=(10, len(cols_to_plot)*2.5)
14)
15 for idx, col in enumerate(cols_to_plot):
16 understanding_NYC(
17 violation_rate_plot(
18 col, 'Severe', eda_df, ax=ax[idx]
19)
20)
21
22 YelpBusinessSearch(
23 YelpReviews(
24 YelpReviewLegend(
25 NYCDOHMHq='lower center', bbox_to_anchor=(0.5, 1.05), title='Severe Violations'
26)
27)
28)
29
30 DropBusinessesMissing()
31
32 ExploringTheDataset[1.44s, finished 19:05:19 2022-08-12]
```

## Contents ⚙️

- 1 Predicting NYC Restaurants
- 2 Introduction
  - 10 1 Objective
    - 11 2.1 Imports
    - 12 12 fig, ax = plt.subplots(
    - 13 13 len(cols\_to\_plot), figsize=(10, len(cols\_to\_plot)\*2.5)
    - 14 14 )
    - 15 15 for idx, col in enumerate(cols\_to\_plot):
    - 16 16 understanding\_NYC(
    - 17 17 violation\_rate\_plot(
    - 18 18 col, 'Severe', eda\_df, ax=ax[idx]
  - 19 3.1 Obtaining Restaurant Information
  - 20 3.2 Obtaining Yelp Businesses
  - 21 3.3 Joining NYC DOHMH Data
  - 22 4 Scrubbing The Data
  - 23 5 Exploring The Dataset
  - 24 6 Preprocessing For Further Analysis
  - 25 7 Preprocessing For Modeling
  - 26 8 Models
    - 27 8.1 Bag-of-Words Model
      - 28 8.1.1 Model Evaluations
    - 29 8.2 Text Preprocessing Pipelines
      - 30 8.2.1 Multinomial Naive Bayes
      - 31 8.2.2 Tuned Multinomial Naive Bayes
      - 32 8.2.3 Logistic Regression
      - 33 8.2.4 Tuned Logistic Regression
      - 34 8.2.5 Decision Tree Model
      - 35 8.2.6 Tuned DecisionTree Model
      - 36 8.2.7 Support Vector Classification
      - 37 8.2.8 Tuned SupportVectorClassification
  - 38 9 Conclusions
    - 39 9.1 Best Model Results
    - 40 9.2 Next Steps and Future
    - 41 9.3 Appendix - Deep NLP
      - 42 9.3.1 Word2Vec Embeddings
      - 43 9.3.2 Classifying With Embeddings
      - 44 9.3.3 Pretrained Vector Embeddings
      - 45 9.3.4 Mean Word Embeddings



## Contents ⚙️



## Contents ⚙️

- 1 Predicting NYC Restaurant
- 2 Introduction
  - 2.1 Objective
 

The plots above provide very little to no insight from plotting the target variable against the categorical variables.
- 3 Obtain
  - 3.1 Obtaining Restaurant I
- In [309]: Understanding NYC Scale('log')
 

```
1 TargetVariable = 1.15** (np.arange(0,50))
2 EngineeringTarget =eda_df[eda_df['Severe']==0]['review_count'],bins=bins,alpha=0.
3 plt.hist(eda_df[eda_df['Severe']==1]['review_count'],bins=bins,alpha=0.
4 plt.legend(('Passed','Failed'))
5 plt.show()
```

executed in 1.32s, finished 19:02:39 2022-08-12
- 3.3 Joining NYC DOHMH
- 4 Scrubbing The Data
  - 4.1 Drop businesses missing
- 5 Exploring The Dataset
- 6 Preprocessing For Further
  - 6.0.1 Lets find most freq
- 7 Preprocessing For Modelin
- 8 Models
  - 8.1 Bag-of-Words Model
    - 8.1.1 Model Evaluations
  - 8.2 Text Preprocessing Pip
    - 8.2.1 Multinomial Naive Bayes
    - 8.2.2 Tuned Multinomial
    - 8.2.3 Logistic Regressio
    - 8.2.4 Tuned Logistic Re
    - 8.2.5 Decision Tree Model
    - 8.2.6 Tuned DecisionTree
    - 8.2.7 Support Vector Cl
    - 8.2.8 Tuned SupportVec
- In [117]: Conclusions
  - 1 # import street map
  - 2 # https://data.cityofnewyork.us/City-Government/Borough-Boundaries/tqmj
  - 3 9.1 Best Model Results
  - 4 9.2 Next Steps and Future
  - 5 9.3 Appendix
- In [118]: Deep NLP
  - 1 street\_map = gpd.read\_file('data/Borough\_Boundaries/geo\_export\_392103e77f1b\_15000000000000000.shp')
  - 2 9.3.1 Word2Vec Embedd
  - 3 9.3.2 Classifying With Er
  - 4 9.3.3 Pretrained Vector
  - 5 9.3.4 Mean Word Embed
- In [119]: Dropna
  - 1 geo\_eda = eda\_df.copy()
  - 2 9.3.4 Mean Word Embed
  - 3 geo\_eda.dropna(axis=0, subset=['Latitude','Longitude'], inplace=True)



In [126]:

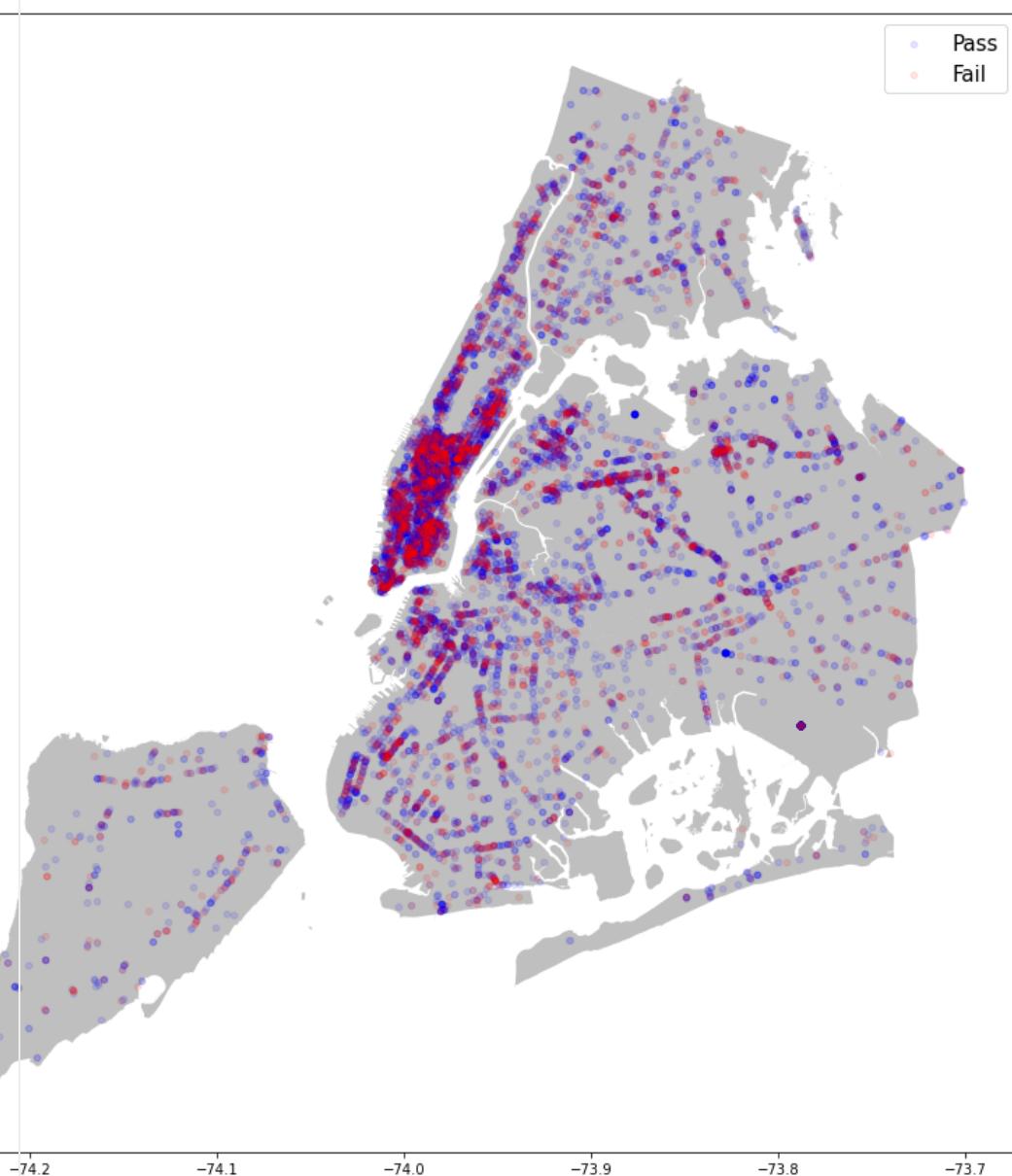
```

1 # Plotting map
2 fig, ax = plt.subplots(figsize=(15,15))
3 street_map.plot(ax=ax, alpha=.5,color='gray')
4 geo_df[geo_df['Severe'] == 0].plot(ax=ax, alpha=.1, markersize=20,color='blue')
5 geo_df[geo_df['Severe'] == 1].plot(ax=ax, alpha=.1, markersize=20,color='red')
6 plt.legend(prop={'size':15});

```

## Contents

|       |                                          |
|-------|------------------------------------------|
| 1     | Predicting NYC Restaurant                |
| ▼ 2   | Introduction                             |
| ▼ 2.1 | Objective                                |
| 2.1.1 | Imports                                  |
| ▼ 3   | Obtain                                   |
| ▼ 3.1 | Obtaining Restaurant Info                |
| 3.1.1 | Understanding NYC Restaurants            |
| 3.1.2 | Target Variable -- NYC Health Violations |
| 3.1.3 | Engineering Target Variable              |
| ▼ 3.2 | Obtaining Yelp Business Data             |
| 3.2.1 | Yelp Business Search API                 |
| 3.2.2 | Yelp Reviews                             |
| 3.2.3 | Joining Yelp Reviews                     |
| 3.3   | Joining NYC DOHMH & Yelp Data            |
| ▼ 4   | Scrubbing The Data                       |
| 4.1   | Drop businesses missing address          |
| ▼ 5   | Exploring The Dataset                    |
| ▼ 6   | Preprocessing For Further Modeling       |
| 6.0.1 | Lets find most frequent words            |
| ▼ 7   | Preprocessing For Modeling               |
| ▼ 8   | Models                                   |
| ▼ 8.1 | Bag-of-Words Model                       |
| 8.1.1 | Model Evaluations                        |
| ▼ 8.2 | Text Preprocessing Pictures              |
| 8.2.1 | Multinomial Naive Bayes                  |
| 8.2.2 | Tuned Multinomial Naive Bayes            |
| 8.2.3 | Logistic Regression                      |
| 8.2.4 | Tuned Logistic Regression                |
| 8.2.5 | Decision Tree Model                      |
| 8.2.6 | Tuned DecisionTreeClassifier             |
| 8.2.7 | Support Vector Classification            |
| 8.2.8 | Tuned SupportVectorClassification        |
| ▼ 9   | Conclusions                              |
| 9.1   | Best Model Results                       |
| 9.2   | Next Steps and Future                    |
| ▼ 9.3 | Appendix - Deep NLP                      |
| 9.3.1 | Word2Vec Embeddings                      |
| 9.3.2 | Classifying With Embeddings              |
| 9.3.3 | Pretrained Vector Embeddings             |
| 9.3.4 | Mean Word Embedding                      |



We already learned from the plot above there is very little we can infer from the restaurant location by each borough. And the map plot above confirms there is no insight to be gained from location data in all. There are clearly a much higher density of restaurants in Manhattan, especially Midtown and Lower Manhattan as well as the neighborhoods along the East River in Brooklyn and Queens.

## Contents

and Lower Manhattan as well as the neighborhoods along the East River in Brooklyn and Queens. But the data points don't tell us where a higher portion of restaurants fail their health inspections.

1 Predicting NYC Restaurant

2 Introduction

2.1 Objective

2.1.1 Imports

3 Obtain

# 6 Preprocessing For Further EDA

3.1 Obtaining Restaurant Info

3.1.1 Understanding NYC

3.1.2 Target Variables

3.1.3 Engineering Target

3.2 Obtaining Yelp Businesses

Basically, we will find most frequent words in reviews to get an overview of why users gave low ratings. These words could be related to those business attributes or services about which users are most unhappy.

3.2.1 Yelp Business Search

3.2.2 Yelp Reviews

3.2.3 Joining Yelp Reviews

3.3 Joining NYC DOHMH Data

4 Scrubbing The Data `txt_eda = df_1.copy()`

4.1 Drop businesses missing

executed in 17ms, finished 15:02:29 2022-08-12

5 Exploring The Dataset

6 Preprocessing For Further EDA `passed_df = txt_eda.loc[txt_eda['Severe'] == 0]`

6.0.1 Lets find most freq words

7 Preprocessing For Modeling

executed in 130ms, finished 15:02:29 2022-08-12

8 Models

8.1 Bag-of-Words Model

8.1.1 Model Evaluations

13058 50113198

SENG  
SEAFOOD  
RESTAURANT

Seafood Manhattan

39

EAST BROADWAY

+164678

8.2 Text Preprocessing Pip

8.2.1 Multinomial Naive Bayes

8.2.2 Tuned Multinomial

8.2.3 Logistic Regression

8.2.4 Tuned Logistic Regression

8.2.5 Decision Tree Model

8.2.6 Tuned DecisionTree Model

8.2.7 Support Vector Classification

8.2.8 Tuned SupportVector Classification

9 Conclusions `13059 50117350`

9.1 Best Model Results

9.2 Next Steps and Future

9.3 Appendix - Deep NLP

9.3.1 Word2Vec Embedding

9.3.2 Classifying With Embedding

9.3.3 Pretrained Vector Embedding

9.3.4 Mean Word Embedding

SONG TEA

Coffee/Tea Manhattan

488

7 AVENUE

+164666

```
In [129]: 1 failed_df = txt_eda.loc[txt_eda['Severe']==1]
2 failed_df
```

executed in 92ms, finished 15:02:29 2022-08-12

Out[129]:

| CAMIS | DBA | CUISINE DESCRIPTION | BORO | BUILDING | STREET | PHONE | La |
|-------|-----|---------------------|------|----------|--------|-------|----|
|-------|-----|---------------------|------|----------|--------|-------|----|

## Contents ⚙

- 1 Predicting NYC Restaurant
- ▼ 2 Introduction
  - ▼ 2.1 Objective
    - 2.1.1 Imports
- ▼ 3 Obtain
  - 5 40360076 CARVEL Frozen Desserts Brooklyn 203 CHURCH AVENUE +17184389501
  - ▼ 3.1 Obtaining Restaurant Info
    - 3.1.1 Understanding NYC
    - 3.1.2 Target Variable -- NYC
    - 3.1.3 Engineering Target
  - ▼ 3.2 Obtaining Yelp Business
    - 3.2.1 Yelp Business Search
    - 3.2.2 Yelp Reviews
    - 3.2.3 Joining Yelp Reviews
    - 3.3 Joining NYC DOHMH Data
- ▼ 4 Scrubbing The Data
- In 4[1] D0qjbusinespassed\_corpus = passed\_df['Reviews'].to\_list()
 5 Exploring The Dataset passed\_corpus[:5]
- ▼ 6 Preprocessing For Further Analysis
  - 6.0.1 Let's take a look at the progression, reception outline, and meal-music timing, the coordination was flawless. I, as the bride, didn't have to worry about a thing on my wedding day. Everything was perfect because of the Riviera team. Many of our guests stated that it was the best wedding they had ever attended. Couldn't have wished for a more memorable night. We would recommend the Riviera team in a heartbeat! Tommy is awesome! My fiancé and I met with him in hopes of having our wedding here. Out of the 15 venues we visited, Tommy was the most down to earth and relatable person we met with. The venue is gorgeous and the menu is jam packed with loads of options. The Logistic Regression Backyard is so stunning and will make for great photo backgrounds. We ended up going with a venue outside of Brooklyn but we will definitely keep this in mind for future events. I hope to use this beautiful venue at a later date..Magical!
  - 6.0.2 Tuned Decision Tree No need to say more!!! My cousin got married on 6/23/18Wow Riviera! You really did an amazing job! From the food to the smallest details! Like a bottle of champagne in the bathrooms so the guest don't go thirsty.
  - 6.0.3 Support Vector Classification
    - 6.0.3.1 Best Model Results lol The food was outstanding! I had the rack of lamb! This was the best lamb I ever had! And the potato purée!!!! Mouth watering! Great job Riviera!
    - 6.0.3.2 Next Steps and Future Staff was outstanding!!!! The drinks were served all night! And the service was excellent! The staff kept on coming to our table without asking! Which was a great!!
  - 6.0.4 Appendix - Short NLP
    - 6.0.4.1 Word2Vec Embedding
    - 6.0.4.2 Classifying With Embedding
    - 6.0.4.3 Pretrained Vector Model
    - 6.0.4.4 Mean Word Embedding

```
In [131]: 1 failed_corpus = failed_df['Reviews'].to_list()
2 failed_corpus[:5]
```

executed in 23ms, finished 15:02:29 2022-08-12

Out[131]: ["I love carvel there used to be carvel at Nostrand Avenue and Church Ave  
nue back in the nineties that one closed down, but this location is locat  
ed in the Kensington section of Brooklyn .there ice cream is great, I lov

## Contents

- 1 Predicting NYC Staff members are cool too! \xa0(their delivery here as well I'm not  
sure if it's from Grubhub, uber eats, door dash, postmates, etc )- call i  
n double check P.S. something about Carvel ice cream that does it, no off  
ense Baskin Robbins .....Lol !!!!( all dough I could go to the shop and s  
top supermarket and get a carvel ice cream cake box I prefer to come her  
e, instead ).Very convenient by the MTA transportation:Subway:F, G-Train  
3.1 Obtaining Restaurant! S. Church avenue. Train station.Bus:B35 ,B67,B103.Had the vanilla ice cre  
3.1.1 Understanding NY am it tasted off ... Now I'm home at 3am in the morning with food poisoni  
3.1.2 Target Variable N ng. Be wary..Cousin & I went to polish off our dinner with some ice crea  
3.1.3 Engineering Target m.Best sundaes we ever had. The amount of fudge was nothing I'd ever see  
3.2 Obtaining Yelp Business who made it put my peanuts on first, then the hot fudge. At f  
3.2.1 Yelp Business was confused but it made sense. Instead of the little suckers esca  
3.2.2 Yelp Review and falling off, they were mounted underneath the mountain of fudge,  
guaranteed to be eaten with no escapees..Since the ice cream is/should be  
the same in all locations the 3 is a review of the store. The store is a  
3.3 Joining NYC DOHMH . . . . .
- 4 Scrubbing The Data

In [132]: # Tokenizing the corpus of review text from restaurants that have sever  
5 Exploring The Dataset failed\_tokens = word\_tokenize(', '.join(failed\_corpus))

- 6 Preprocessing For Further

6.0.1 Lets find most freq

executed in 32ms, finished 15:03:22 2022-08-12

### 7 Preprocessing For Modelin

- 18 Models: 1 # Tokenizing the corpus of review text from restaurants that have not s  
8.1 Bag-of-Words passed\_tokens = word\_tokenize(', '.join(passed\_corpus))  
8.1.1 Model Evaluations

executed in 1m 58.1s, finished 15:05:20 2022-08-12

### 8.2 Text Preprocessing Pic

```
In [134]: 1 # Importing English stop words
2 Tuned Multinomial stopwords_list = stopwords.words('english')
3 Logistic Regression punctuation marks to the stopwords_list
4 Tuned Logistic regression stopwords_list.extend(string.punctuation)
5 Decision Tree additional_punc = [",", "!", "...", "!!!", "'", "``"]
6 stopwords_list.extend(additional_punc)
7 Support Vector Machine restaurant_words = ['restaurant', 'place', 'food', 'drink']
8 stopwords_list.extend(restaurant_words)
```

- 9 Conclusion

executed in 33ms, finished 15:05:20 2022-08-12

### 9.1 Best Model Results

### 9.2 Next Steps and Future

- 9.3 Appendix - Deep NLP

9.3.1 Word2Vec Embedds

9.3.2 Classifying With Er

9.3.3 Pretrained Vector N

9.3.4 Mean Word Embed

```
In [135]: 1 # Removing stopwords from failed_corpus
2 stopped_failed_tokens = [w.lower() for w in failed_tokens if w.lower()
3 stopped_failed_tokens[:10]
```

Out[135]: ['love',  
          'carvel',

# Contents

- 1 Predicting NYC restaurant
  - 2 Introduction 'nostrand'
  - 2.1 Objective 'avenue',  
    2.1.1 Imports 'church',  
    2.1.1 Exports 'avenue',
  - 3 Obtain 'back',  
  3.1 Obtaining Restaurant in nineties]  
    3.1.1 Understanding NYC

```
In [136]: # Removing stopwords from passed_corpus
3.1.2 target_variable -->
3.1.3 Engineering Target
3.2 Obtaining Yelp Businesses
 2 stopped_passed_tokens = [w.lower() for w in passed_tokens if w.lower() not in STOPWORDS]
 3 stopped_passed_tokens[:10]
3.2.1 Yelp Business Search
3.2.2 Yelp Reviews
```

Out[326]: Joining Yelp Review

- 3.3 Joining NYCOODOHMH & 'been', 'businesses', 'tasted', 'weird', 'if', 'give', 'find most freq stars', 'absolutely', 'would']
    - ▼ 4 Scrubbing The Data
    - 4.1 Drop businesses missin
  - 5 Exploring The Dataset
  - ▼ 6 Preprocessing For Further
  - 6.0.1 Lets find most freq stars
  - 7 Preprocessing For Modelin
  - ▼ 8 Models
  - ▼ 8.1 Bag-of-Words Model

- Out [8.274]: Tuned('Logistic Reg', 371),  
8.2.5 DecisionStep(16690),  
8.2.6 Tuned('Decision', 22707),  
8.2.7 SupportVectorCle('good', 20802),  
8.2.8 Tuned('SupportVec', 13681),  
8.2.9 ('great', 12793),  
▼ 9 Conclusions  
9.1 Best Model Results('one', 11126),  
9.2 Next Steps and Future('order', 11084),  
9.3 Appendix - Deep NLP  
9.3.1 Word2Vec Embeds('chicken', 10993),  
9.3.2 Classifying With F1('get', 10465),  
9.3.3 Pretrained Model('service', 10413),  
9.3.4 Mean Word Embed('love', 10159),  
9.3.5 ('really', 9961)]

```
In [138]: 1 # Creating FreqDist from stopped_passed_tokens
2 freq_passed = FreqDist(stopped_passed_tokens)
3 passed_bar = freq_passed.most_common(15)
```

executed in 4.40s, finished 15:06:10 2022-08-12

```
In [139]: 1 # Creating FreqDist from stopped_passed_tokens
```

**Contents** ↗  

- 1 Predicting NYC Restaurant
- 2 Introduction
- 3 2.1 Objective
- 4 2.2 Obtaining Data
- 5 2.3 Yelp Business
- 6 3.1 Obtaining Restaurant
- 7 3.2 Yelp Reviews
- 8 3.3 Joining NYC DOHMH Dataset
- 9 3.4 Scrubbing The Data
- 10 3.5 Exploring The Dataset
- 11 3.6 Preprocessing For Further
- 12 3.7 Models
- 13 3.8 Bag-of-Words Model
- 14 3.9 Text Preprocessing
- 15 3.10 Conclusions
- 16 3.11 Best Model Results
- 17 3.12 Next Steps and Future
- 18 3.13 Appendix
- 19 3.14 Deep NLP
- 20 3.15 Word2Vec Embed
- 21 3.16 Classifying With E
- 22 3.17 Pretrained Vector
- 23 3.18 Mean Word Embed

executed in 663ms, finished 15:06:10 2022-08-12

## 2.1 Objective

```
In [140]: 1 # Creating FreqDist from stopped_failed_tokens
```

- 2 Obtain
- 3 3.1 Obtaining Restaurant
- 4 3.2 Yelp Business
- 5 3.3 Joining NYC DOHMH Dataset
- 6 3.4 Scrubbing The Data
- 7 3.5 Exploring The Dataset
- 8 3.6 Preprocessing For Further
- 9 3.7 Models
- 10 3.8 Bag-of-Words Model
- 11 3.9 Text Preprocessing
- 12 3.10 Conclusions
- 13 3.11 Best Model Results
- 14 3.12 Next Steps and Future
- 15 3.13 Appendix
- 16 3.14 Deep NLP
- 17 3.15 Word2Vec Embed
- 18 3.16 Classifying With E
- 19 3.17 Pretrained Vector
- 20 3.18 Mean Word Embed

executed in 13ms, finished 22:31:05 2022-08-10

### 3.1.1 Understanding NYC

#### 3.1.2 Target Variable -- N

#### 3.1.3 Engineering Target

## 3.2 Obtaining Yelp Business

### 3.2.1 Yelp Business Sear

```
In [341]: 1 new_fig = plt.figure(figsize=(16, 4))
```

#### 3.2.2 Joining Yelp Review

3.3 Joining NYC DOHMH Dataset

```
4 new_fig.add_subplot(121)
```

```
5 new_fig.add_subplot(122)
```

#### 4.1 Drop businesses missin

5 Exploring The Dataset

```
6 ax1.bar(passed_bar_words, passed_bar_count)
```

```
7 ax2.bar(failed_bar_words, failed_bar_count)
```

#### 6.0.1 Lets find most freq

9 ax1.title.set\_text('Passed')

```
10 ax2.title.set_text('Failed')
```

```
11 8 Models
```

#### 8.1 Bag-of-Words Model

#### 8.2 Text Preprocessing

#### 8.3 Model Evaluations

#### 8.4 Tuned Model

#### 8.5 Multi-class Layout

#### 8.6 Logistic Regression

#### 8.7 Tuned Logistic Reg

#### 8.8 Tuned Decision Tree

#### 8.9 Tuned DecisionTree

#### 8.10 Support Vector Clas

#### 8.11 Tuned SupportVec

```
12 for ax in new_fig.axes:
```

#### 8.1.1 Model Evaluations

#### 8.2.1 Multinomial Naive

#### 8.2.2 Tuned Multinomial

#### 8.2.3 Logistic Regressio

#### 8.2.4 Tuned Logistic Reg

#### 8.2.5 Decision Tree Model

#### 8.2.6 Tuned DecisionTre

#### 8.2.7 Support Vector Clas

#### 8.2.8 Tuned SupportVec

```
13 9 Conclusions
```

#### 9.1 Best Model Results

#### 9.2 Next Steps and Future

```
14 9.3 Appendix
```

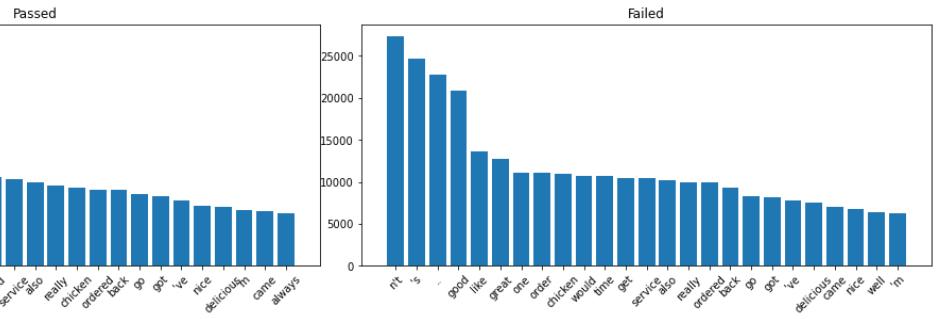
#### 9.3.1 Deep NLP

#### 9.3.2 Word2Vec Embed

#### 9.3.3 Classifying With E

#### 9.3.4 Pretrained Vector

#### 9.3.5 Mean Word Embed



In [142]:

```
1 # Functionizing wordcloud generator
2 def wordcloud_generator(tokens, collocations=False, background_color='b
3 colormap='Reds', display=True):
4
5
6 # Initialize a WordCloud
7 wordcloud = WordCloud(collocations=collocations,
8 background_color=background_color,
9 colormap=colormap,
10 width=500, height=300)
11
12 # Generate wordcloud from tokens
13 wordcloud.generate(', '.join(tokens))
14
15 # Plot with matplotlib
16 if display:
17 plt.figure(figsize = (12, 15), facecolor = None)
18 plt.imshow(wordcloud)
19 plt.axis('off');
20
21 return wordcloud
22
23 # Joining Yelp Reviews
```

```
In 4:1 Drop businesses missin
In 4:1 # Generating a WordCloud for reviews from restaurants that have failed
5 Exploring The Dataset
6 Preprocessing For Further
 failed_cloud = wordcloud_generator(stopped_failed_tokens, collocations=
```



```
In [144]: 1 # Generate a WordCloud from reviews of restaurants that have not severe
2 passed_cloud = wordcloud_generator(stopped_passed_tokens, colormap='Gr
3 collocations=True)
```

executed in 43.0s, finished 15:07:13 2022-08-12



## Contents

- 1 Predicting NYC Restaurant
  - 2 Introduction
  - 2.1 Objective
    - 2.1.1 Impacts
  - 3 Obtain
    - 3.1 Obtaining Restaurant Information
      - 3.1.1 Understanding NYC Data
      - 3.1.2 Target Variable -- Is it a restaurant?
      - 3.1.3 Engineering Target Variable instead of business
    - 3.2 Obtaining Yelp Business Reviews
      - 3.2.1 Yelp Business Search
      - 3.2.2 Yelp Reviews
      - 3.2.3 Joining Yelp Reviews
    - 3.3 Joining NYC DOHMH Data
      - take into account missing values
  - 4 Scrubbing The Data
    - 4.1 Drop businesses missing data
  - 5 Exploring The Dataset
  - 6 Preprocessing For Further Analysis
    - 6.0.1 Lets find most frequent categories
  - 7 Preprocessing For Modeling
  - 8 Models

# 7 Preprocessing For Modeling

Topic 8.244 Tuned Logistic Regression

8.2.5 Decision Tree Mod executed in 33ms, finished 21:02:57 2022-08-15

## 8.2.6 Tuned Decision Tree

```
Out[384]: Index(['CAMIS', 'DBA', 'CUISINE DESCRIPTION', 'BORO', 'BUILDING', 'STREET',
8.2.7 Support Vector Cls
8.2.8 TunedSupportVec
 'PHONE', 'Latitude', 'Longitude', 'Community Board', 'Council Dist']
```

## ▼ 9 Conclusions . . .

### **1. Best Model Results**

## 9.1 Best Model Results

<sup>2</sup> Next Steps and Future closed url review count categories rating

## 2.2 Appendix: Deep NLP

3 Appendix - Deep Network Coordinates', 'transactions', 'price', 'location', 'phone',

### 9.3.1 Word2Vec Embedding

0.3.3 Classifying With `Type='object'`

### 9.3.2 Classifying Data

### 9.3.3 Pretrained Vector Λ

### 9.3.4 Mean Word Em

#### 9.3.4 Mean Word Ent.

```
In [385]: 1 # Dropping unnecessary columns
2 model_df = df_1.drop(columns=['CAMIS', 'DBA', 'CUISINE DESCRIPTION', 'B
3 'PHONE', 'Latitude', 'Longitude', 'Community Board', 'Council Di
4 'Census Tract', 'alias', 'name', 'image_url',
5 'is_closed', 'url', 'review_count', 'categories', 'rating',
6 'coordinates', 'transactions', 'price', 'location', 'phone',
7 'display_phone', 'zipcode'])
```

## Contents ⚙

executed in 135ms, finished 21:02:58 2022-08-15

1 Predicting NYC Restaurant

▼ 2 Introduction

In [386]: 1 model\_df.head(3)

executed in 27ms, finished 21:02:58 2022-08-15

▼ 3 Obtain:

|                               | id                       | Reviews                                                                                                                                                                                                 |
|-------------------------------|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3.1.1 Understanding NYC       |                          | Forgot my root beer and the burger tasted very weird. If I can give this place no stars I absolutely would. The workers here are in efficient. They do not work the best possible way. They cause th... |
| 3.1.2 Target Variable         | -AMxMPBkWi20dn_1BRalahA  |                                                                                                                                                                                                         |
| 3.1.3 Engineering Target      |                          | I had my wedding here a month ago and I would give Riviera 10 stars if I could. It was such a joy and ease to work with Adam, Tommy, and their staff. A true diamond in the rough of Coney Island a...  |
| 3.2 Obtaining Yelp Businesses |                          | We've recently discovered this gem on Uber Eats and they have the best sandwiches!!The broccoli rabe fried eggplant mozzarella sandwich was amazing. Having a fresh green veggie paired with a light... |
| 3.2.1 Yelp Business Search    | 10g8tAhVqBsC-BN6otcn85OA |                                                                                                                                                                                                         |
| 3.2.2 Yelp Reviews            |                          |                                                                                                                                                                                                         |
| 3.2.3 Joining Yelp Reviews    |                          |                                                                                                                                                                                                         |
| 3.3 Joining NYC DOHMH         |                          |                                                                                                                                                                                                         |
| 4 Scrubbing The Data          | 0 Pe6MsH2DW0CXjvUtxUpI4A |                                                                                                                                                                                                         |
| 4.1 Drop businesses missing   |                          |                                                                                                                                                                                                         |
| 5 Exploring The Dataset       |                          |                                                                                                                                                                                                         |
| 6 Preprocessing For Further   |                          |                                                                                                                                                                                                         |

In [387]: 1 model\_df['Reviews'].isna().sum()

7 Preprocessing For Further

executed in 10ms, finished 21:02:59 2022-08-15

▼ 8 Models

Out[387]: 0

▼ 8.1 Bag-of-Words Model

8.1.1 Model Evaluations

In [388]: 1 # Checking Class Balance

▼ 8.2 Text Preprocessing Pip

8.2.1 Multinomial Naïve

executed in 14ms, finished 21:03:00 2022-08-15

Out[388]: Logistic Regression

8.2.4 Tuned Logistic Reg

8.2.5 DecisionTreeSevere, dtype: int64

8.2.6 Tuned DecisionTre

In [389]: 1 # Make X and y

8.2.8 Tuned SupportVec

▼ 9 Conclusions 3 X = model\_df['Reviews'].copy()

9.1 Best Model Results

executed in 11ms, finished 21:03:02 2022-08-15

9.2 Next Steps and Future

In [390]: 1 # Train Test Split

9.3.1 Word2Vec Embedd

9.3.2 Classifying With Er

9.3.3 Pretrained Vector N

9.3.4 Mean Word Embed

executed in 36ms, finished 21:08:59 2022-08-15

## 8 Models

## 8.1 Bag-of-Words Model

### 8.1.1 Model Evaluations

|                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Contents</b>                         | Because the objective of this project is to flag restaurants that are likely to fail their health inspections the metric we will be optimizing for is Recall because we need our model to find all of the restaurants that pose a risk to public health. This metric is defined by the number of true positives divided by true positives and false negatives. If our model mistakenly classifies a clean kitchen as unsafe, the consequence of sending an inspector to check on it is much less worse than the alternative of classifying an unsafe establishment as clean. The Accuracy measurement, scoring how many times the model correctly predicted is a good indicator of performance but not as suitable as recall because of this business case AND because of our class imbalance. That is to say since only 30% of our businesses are flagged. If our model predicts every restaurant will pass its inspection, it will be correct 70%. |
| 1 Predicting NYC Restaurant             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 2 Introduction                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 2.1 Objective                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 2.1.1 Imports                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 3 Obtain                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 3.1 Obtaining Restaurant                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 3.1.1 Understanding NYC                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 3.1.2 Target Variable                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 3.1.3 Engineering Target                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 3.2 Obtaining Yelp Business             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 3.2.1 Yelp Business Search              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 3.2.2 Yelp Reviews                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 3.2.3 Joining Yelp Reviews              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 3.3 Joining NYC DOHMH                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 4 Scrubbing The Data                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 4.1 Drop businesses missing             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 5 Exploring The Dataset                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 6 Preprocessing For Further             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 6.0.1 Lets find most freq               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 7 Preprocessing For Modeling            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 8 Models                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 8.1 Bag-of-Words Model                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 8.1.1 Model Evaluations                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 8.2 Text Preprocessing Pictures         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 8.2.1 Multinomial Naive Bayes           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 8.2.2 Tuned Multinomial                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 8.2.3 Logistic Regression               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 8.2.4 Tuned Logistic Regression         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 8.2.5 Decision Tree Model               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 8.2.6 Tuned DecisionTree Model          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 8.2.7 Support Vector Classification     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 8.2.8 Tuned SupportVectorClassification |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 9 Conclusions                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 9.1 Best Model Results                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 9.2 Next Steps and Future               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 9.3 Appendix - Deep NLP                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 9.3.1 Word2Vec Embedding                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 9.3.2 Classifying With Embedding        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 9.3.3 Pretrained Vector Model           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 9.3.4 Mean Word Embedding               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

```
In [381]: 1 def evaluate_classification(model, X_test,y_test,cmap='Blues',
2 normalize=None,classes=['Passed','Failed'],
3 X_train = None, y_train = None,):
4 """Evaluates a scikit-learn binary classification model.
5
6 Args:
7 model (classifier): any sklearn classification model.
8 X_test (Frame or Array): X data
9 y_test (Series or Array): y data
10 cmap (str, optional): Colormap for confusion matrix. Defaults to 'Blues'.
11 normalize (str, optional): normalize argument for plot_confusion_matrix.
12 Defaults to 'true'.
13 classes (list, optional): List of class names for display. Defaults to ['Passed', 'Failed'].
14 figsize (tuple, optional): figure size. Defaults to (8,4).
15
16 X_train (Frame or Array, optional): If provided, compare model.
17 for train and test. Defaults to None.
18 y_train (Series or Array, optional): If provided, compare model.
19 for train and test. Defaults to None.
20
21 # Get Predictions and Classification Report
22
23 3.3 Joining NYC DOHMH Data
24 print(metrics.classification_report(y_test, y_hat_test,target_names=)
25
26 # Plot Confusion Matrix and roc curve
27 fig,ax = plt.subplots(ncols=2, figsize=figsize)
28 metrics.plot_confusion_matrix(model, X_test,y_test,cmap=cmap,
29 normalize=normalize,display_labels=cl
30 ax=ax[0])
31
32 # if roc curve errors, delete second ax
33
34 8.1.1 ModelEvaluation:
35
36 8.2 Text Preprocessing Pip
37 curve = metrics.plot_roc_curve(model,X_test,y_test,ax=ax[1])
38 curve.ax_.grid()
39 curve.ax_.plot([0,1],[0,1],ls=':')
40 fig.tight_layout()
41 except:
42 fig.delaxes(ax[1])
43 plt.show()
44
45 # Add comparing Scores if X_train and y_train provided.
46
47 8.2.8 Tuned SupportVec
48 if (X_train is not None) & (y_train is not None):
49 print(f"Training Score = {model.score(X_train,y_train):.2f}")
50 print(f"Test Score = {model.score(X_test,y_test):.2f}")
51
52
53 9.3 Appendix - Deep NLP
54
55 9.3.1 Word2Vec Embedd
56 9.3.2 Classifying With Er
57 9.3.3 Pretrained Vector M
58 9.3.4 Mean Word Embedd
```



We'll be dealing with the text classification problem with a Naive Bayes Model, which typically does well with text classifications.

```
In [293]: 1 ## Make a full pipeline with the DecisionTree model as the second step
2 mnb_pipe = Pipeline([('text_pipe',text_pipe),
3 ('clf',MultinomialNB())])
4 mnb_pipe
```

## Contents

1 Predicting NYC Restaurants, finished 18:26:36 2022-08-15

2 Introduction

3 Obtain

4 Scrubbing The Data

5 Exploring The Dataset

6 Preprocessing For Further

7 Preprocessing For Modelin

8 Models

9 Conclusions

10 Appendix - Deep NLP

### Pipeline

2.1 Objective

2.1.1 Imports

**text\_pipe: Pipeline**

3.1 Obtaining Restaurant

3.1.1 Understanding NY

**RestaurantTransformer**

3.1.2 Target Variable -- N

**MultinomialNB**

3.2 Obtaining Yelp Business

3.2.1 Yelp Business Sear

In [324]: Yelp Reviews ##Modeling with full pipeline

3.2.3 Joining Yelp Review

3.3 Joining NYC Restaurant

4.1 Drop businesses missin

|  | precision | recall | f1-score | support |
|--|-----------|--------|----------|---------|
|--|-----------|--------|----------|---------|

5 Exploring The Dataset

|   |      |      |      |      |
|---|------|------|------|------|
| 0 | 0.69 | 1.00 | 0.82 | 2257 |
| 1 | 0.00 | 0.00 | 0.00 | 1009 |

6.0.1 Lets find most freq

|   |      |      |      |      |
|---|------|------|------|------|
| 0 | 0.35 | 0.50 | 0.41 | 3266 |
| 1 | 0.48 | 0.69 | 0.56 | 3266 |

8.1 Bag-of-Words Model

8.1.1 Model Evaluations

8.2 Text Preprocessing Pic

8.2.1 Multinomial Naive

8.2.2 Tuned Multinomial

8.2.3 Logistic Regressio

8.2.4 Tuned Logistic Reg

8.2.5 Decision Tree Mod

8.2.6 Tuned DecisionTre

8.2.7 Support Vector Cla

8.2.8 Tuned SupportVec

9.1 Best Model Results

9.2 Next Steps and Future

9.3 Appendix - Deep NLP

9.3.1 Word2Vec Embedd

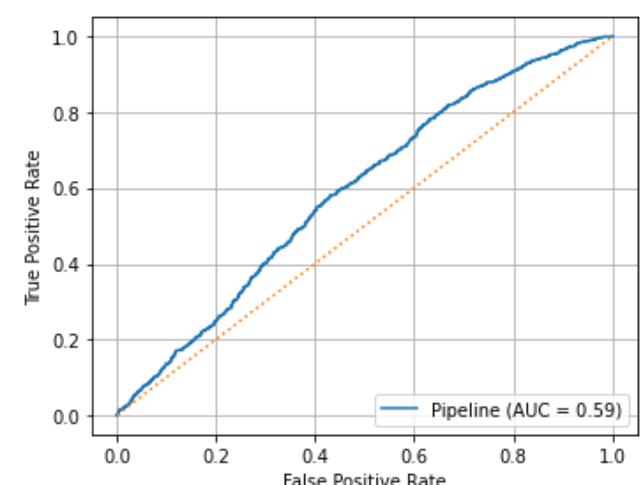
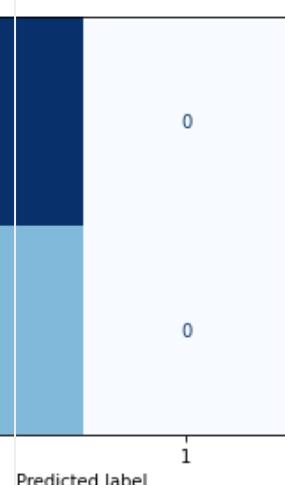
9.3.2 Classifying With Er

9.3.3 Pretrained Vector

9.3.4 Mean Word Embedd

Training Score = 0.70

Test Score = 0.69



The MultinomialNB model has a 70% accuracy, but did not predict any restaurants failed their inspection. This is not helpful for the objective of this project. We'll try tuning the hyperparameters with Gridsearch.

## ▼ Tuning with GridsearchCV

```
In [258]: 1 params = {'clf_alpha': [0.001, 0.01, 0.1, 1]}
2
3 gs = GridSearchCV(mnb_pipe, param_grid=params,
scoring='recall')
```

### Contents ⚙ \*

- 1 Predicting NYC Restaurant
- 2 Introduction
- 3 Objective
- 4 Imports
- 5 Executed in 3m 46s, finished 17:23:25 2022-08-15
- 6 Obtaining Restaurant Data
- 7 gs.fit(X\_train, y\_train)
- 8 gs.best\_params\_

Out[258]: {'clf\_alpha': 0.001}

#### 3.1 Obtaining Restaurant Data

- 3.1.1 Understanding NYC
- 3.1.2 Target Variable - NYC DOHMH
- 3.1.3 Engineering Target

#### 3.2 Obtaining Yelp Business Data

```
In [259]: 1 tuned_mnb_pipe = Pipeline([('text_pipe',text_pipe),
2 ('clf',MultinomialNB(alpha=0.001))])
3
4 tuned_mnb_pipe.fit(X_train,y_train)
5
6 evaluate_classification(tuned_mnb_pipe,X_test,y_test,X_train=X_train,
y_train=y_train)
```

#### 4 Scrubbing The Data

##### 4.1 Drop businesses missing

Executed in 37.0s, finished 17:25:12 2022-08-15

#### 5 Exploring The Dataset

#### 6 Preprocessing For Further

##### 6.0.1 Lets find most freq

#### 7 Preprocessing For Modelin

##### 7.0.1 Bag-of-Words Model

##### 7.1.1 Model Evaluations

#### 8 Models

##### 8.1 Bag-of-Words Model

##### 8.1.1 Model Evaluations

##### 8.2 Text Preprocessing Pip

##### 8.2.1 Multinomial Naive

##### 8.2.2 Tuned Multinomial

##### 8.2.3 Logistic Regressio

##### 8.2.4 Tuned Logistic Re

##### 8.2.5 Decision Tree Mod

##### 8.2.6 Tuned DecisionTre

##### 8.2.7 Support Vector Cl

##### 8.2.8 Tuned SupportVec

#### 9 Conclusions

##### 9.1 Best Model Results

##### 9.2 Next Steps and Future

#### 9.3 Appendix - Deep NLP

##### 9.3.1 Word2Vec Embedc

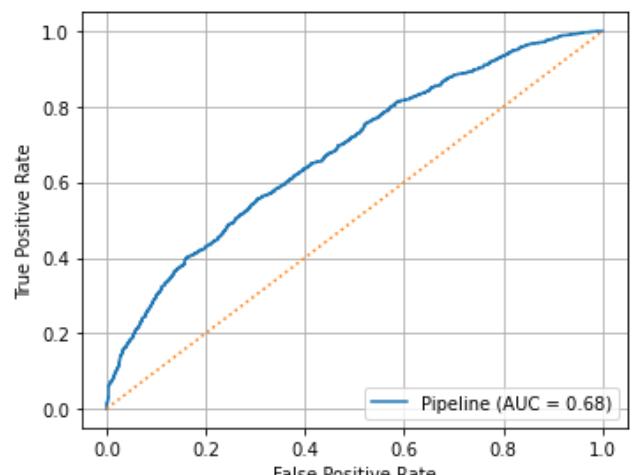
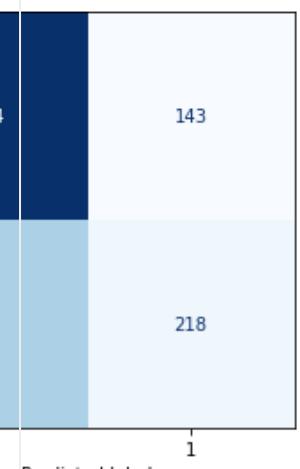
##### 9.3.2 Classifying With Fr

##### 9.3.3 Pretrained Vector N

##### 9.3.4 Mean Word Embed

Training Score = 0.96

Test Score = 0.71



Tuning the alpha smoothing parameter did improve the models accuracy from 70% to 96% in the training data, but only slightly improved the score for the validation data. This model it overfit to the

training data and will perform poorly at classifying new reviews it hasn't seen. It did however predict some violations but the recall score of 22% is below satisfactory.

## 8.2.3 Logistic Regression Model

### Contents

The next classifier we'll fit to the data is a Logistic Regression model. A logistic regression can provide a useful tool to our stakeholders by not just predicting whether or not a restaurant will fail its health and safety inspection but can model the probability of a restaurant failing its inspection.

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>1 Predicting NYC Restaurant</li> <li>2 Introduction</li> <li>2.1 Objective</li> <li>2.1.1 Imports</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | <pre>In [442]: 1 ## Make a full pipeline with the LogisticRegression model as the second 3 Obtain 2 lr_pipe = Pipeline([('text_pipe',text_pipe),                                 ('clf',LogisticRegression(solver='sag',class_weight='balanced'))]) 3.1 Obtaining Restaurant Data 3.1.1 Understanding NYC Restaurants 3.1.2 Target Variable -- Number of Violations 3.1.3 Engineering Target</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <ul style="list-style-type: none"> <li>3.2 Obtaining Yelp Businesses</li> <li>3.2.1 Yelp Business Search Pipeline</li> <li>3.2.2 Yelp Reviews</li> <li>3.2.3 Joining Yelp Reviews Vectorizer</li> <li>3.3 Joining NYC DOHMH Data</li> <li>4 Scrubbing The Data</li> <li>4.1 Drop businesses missing address</li> <li>5 Exploring The Dataset</li> <li>6 Preprocessing For Further Modeling</li> <li>6.0.1 Lets find most frequent words</li> <li>7 Preprocessing For Modeling</li> <li>8 Models</li> <li>8.1 Bag-of-Words Model</li> <li>8.1.1 Model Evaluations</li> <li>8.2 Text Preprocessing Pipeline</li> <li>8.2.1 Multinomial Naive Bayes</li> <li>8.2.2 Tuned Multinomial Naive Bayes</li> <li>8.2.3 Logistic Regression</li> <li>8.2.4 Tuned Logistic Regression</li> <li>8.2.5 Decision Tree Model</li> <li>8.2.6 Tuned DecisionTree Model</li> <li>8.2.7 Support Vector Classification</li> <li>8.2.8 Tuned SupportVector Classification</li> <li>9 Conclusions</li> <li>9.1 Best Model Results</li> <li>9.2 Next Steps and Future</li> <li>9.3 Appendix - Deep NLP</li> <li>9.3.1 Word2Vec Embeddings</li> <li>9.3.2 Classifying With Embeddings</li> <li>9.3.3 Pretrained Vector Embeddings</li> <li>9.3.4 Mean Word Embeddings</li> </ul> | <pre>Out[442]: 3.2 Obtaining Yelp Businesses Pipeline           3.2.1 Yelp Business Search Pipeline           3.2.2 Yelp Reviews           3.2.3 Joining Yelp Reviews Vectorizer           3.3 Joining NYC DOHMH Data           4 Scrubbing The Data           4.1 Drop businesses missing address           5 Exploring The Dataset           6 Preprocessing For Further Modeling           6.0.1 Lets find most frequent words           7 Preprocessing For Modeling           8 Models           8.1 Bag-of-Words Model           8.1.1 Model Evaluations           8.2 Text Preprocessing Pipeline           8.2.1 Multinomial Naive Bayes           8.2.2 Tuned Multinomial Naive Bayes           8.2.3 Logistic Regression           8.2.4 Tuned Logistic Regression           8.2.5 Decision Tree Model           8.2.6 Tuned DecisionTree Model           8.2.7 Support Vector Classification           8.2.8 Tuned SupportVector Classification           9 Conclusions           9.1 Best Model Results           9.2 Next Steps and Future           9.3 Appendix - Deep NLP           9.3.1 Word2Vec Embeddings           9.3.2 Classifying With Embeddings           9.3.3 Pretrained Vector Embeddings           9.3.4 Mean Word Embeddings</pre> |
| <ul style="list-style-type: none"> <li>9.4 Summary</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | <pre>9.4 Summary</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

In [443]:

```

1 ## Modeling with full pipeline
2 lr_pipe.fit(X_train,y_train)
3 evaluate_classification(lr_pipe,X_test,y_test,X_train=X_train, y_train=y_train)

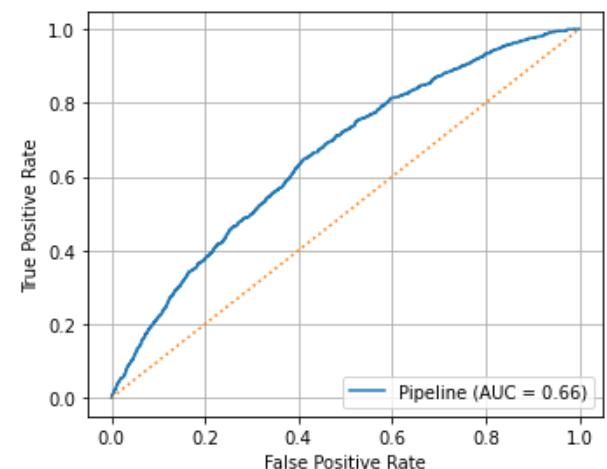
```

executed in 1m 1.91s, finished 22:13:26 2022-08-15

|        | precision | recall | f1-score | support |
|--------|-----------|--------|----------|---------|
| Passed | 0.77      | 0.67   | 0.71     | 2708    |
| Failed | 0.42      | 0.54   | 0.47     | 1211    |

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| macro avg    | 0.59      | 0.60   | 0.59     | 3919    |
| weighted avg | 0.66      | 0.63   | 0.64     | 3919    |

| Contents                                                                                                           | Passed | Failed |
|--------------------------------------------------------------------------------------------------------------------|--------|--------|
| 1 Predicting NYC Restaurants                                                                                       | 1805   | 903    |
| 2 Introduction                                                                                                     |        |        |
| 2.1 Objective accuracy                                                                                             |        |        |
| 2.1.1 Imports                                                                                                      |        |        |
| 2.1.2 Macro avg                                                                                                    |        |        |
| 2.1.3 Weighted avg                                                                                                 |        |        |
| 2.2 Engineering Target                                                                                             |        |        |
| 2.3 Obtaining Yelp Business Data                                                                                   |        |        |
| 2.3.1 Yelp Business Search API                                                                                     |        |        |
| 2.3.2 Yelp Reviews                                                                                                 |        |        |
| 2.3.3 Joining Yelp Reviews                                                                                         |        |        |
| 2.3.4 Joining NYC DOHMH Data                                                                                       |        |        |
| 2.4 Scrubbing The Data                                                                                             |        |        |
| 2.4.1 Drop businesses missing address                                                                              |        |        |
| 2.5 Exploring The Dataset                                                                                          |        |        |
| 2.6 Preprocessing For Further Analysis                                                                             |        |        |
| 2.6.1 Lets find most frequent words                                                                                |        |        |
| 2.7 Preprocessing For Model Training                                                                               |        |        |
| 2.8 Models                                                                                                         |        |        |
| 2.8.1 Training Score = 0.78                                                                                        |        |        |
| 2.8.2 Test Score = 0.63                                                                                            |        |        |
| 2.8.3 Bag-of-Words Model                                                                                           |        |        |
| 2.8.3.1 Model Evaluations                                                                                          |        |        |
| 2.8.3.2 Text Preprocessing Pitfall                                                                                 |        |        |
| The baseline Logistic Regression shows more promising results than the tuned Naive Bayes model.                    |        |        |
| 2.8.3.3 Its definitely overfitting. Lets try to get more performance from the model by tuning the hyperparameters. |        |        |
| 2.8.3.4 Logistic Regression                                                                                        |        |        |
| 2.8.3.5 Tuned Logistic Regression                                                                                  |        |        |
| 2.8.3.6 Decision Tree Model                                                                                        |        |        |
| Hyperparameter Tuning With GridsearchCV                                                                            |        |        |
| 2.8.3.7 Tuned DecisionTreeClassifier                                                                               |        |        |
| 2.8.3.8 Support Vector Classifier                                                                                  |        |        |
| 2.8.3.9 Tuned SupportVectorClassifier                                                                              |        |        |
| 2.9 Conclusions                                                                                                    |        |        |
| 2.9.1 Best Model Results                                                                                           |        |        |
| 2.9.2 Next Steps and Future                                                                                        |        |        |
| 2.9.3 Appendix - Deep NLP                                                                                          |        |        |
| 2.9.3.1 Word2Vec Embeddings                                                                                        |        |        |
| 2.9.3.2 Classifying With Embeddings                                                                                |        |        |
| 2.9.3.3 Pretrained Vector Embeddings                                                                               |        |        |
| 2.9.3.4 Mean Word Embedding                                                                                        |        |        |



```
In [447]: 1 lr_pipe.get_params().keys()
```

executed in 64ms, finished 22:45:32 2022-08-15

```
Out[447]: dict_keys(['memory', 'steps', 'verbose', 'text_pipe', 'clf', 'text_pipe__memory', 'text_pipe_steps', 'text_pipe_verbose', 'text_pipe_count_vectorizer', 'text_pipe_tf_transformer', 'text_pipe_count_vectorizer_analyzer', 'text_pipe_count_vectorizer_binary', 'text_pipe_count_vectorizer_decoder_error', 'text_pipe_count_vectorizer_dtype', 'text_pipe_count_vectorizer_encoding', 'text_pipe_count_vectorizer_input', 'text_pipe_count_vectorizer_lowercase', 'text_pipe_count_vectorizer_max_df', 'text_pipe_count_vectorizer_max_features', 'text_pipe_count_vectorizer_min_n_df', 'text_pipe_count_vectorizer_ngram_range', 'text_pipe_count_vectorizer_preprocessor', 'text_pipe_count_vectorizer_stop_words', 'text_pipe_count_vectorizer_strip_accents', 'text_pipe_count_vectorizer_toke
```

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |  |  |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| <b>Contents</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |  |  |
| <ul style="list-style-type: none"> <li>1 Predicting NYC Restaurant Encoding</li> <li>2 Introduction</li> <li>3 Obtain</li> <li>4 Scrubbing The Data</li> <li>5 Exploring The Dataset</li> <li>6 Preprocessing For Further</li> <li>7 Preprocessing For Modeling</li> <li>8 Models</li> <li>9 Conclusions</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |  |  |
| <ul style="list-style-type: none"> <li>2.1 Objective</li> <li>2.1.1 Imports</li> <li>2.1.2 Target Variable</li> <li>2.1.3 Engineering Target</li> <li>2.2 Obtaining Yelp Businesses</li> <li>2.2.1 Yelp Business Star</li> <li>2.2.2 Yelp Reviews</li> <li>2.2.3 Joining Yelp Reviews</li> <li>2.3 Joining NYC DOHMH</li> <li>3.1 Obtaining Restaurant Information Pattern</li> <li>3.1.1 Understanding NYC Vocabulary</li> <li>3.1.2 Target Variable</li> <li>3.1.3 Engineering Target</li> <li>3.2 Obtaining Yelp Businesses</li> <li>3.2.1 Yelp Business Star</li> <li>3.2.2 Yelp Reviews</li> <li>3.2.3 Joining Yelp Reviews</li> <li>3.3 Joining NYC DOHMH</li> <li>4.1 Drop businesses missing</li> <li>5.1 Exploring The Dataset</li> <li>6.0.1 Lets find most freq</li> <li>7.1 Preprocessing For Modeling</li> <li>8.1 Bag-of-Words Model</li> <li>8.1.1 Model Evaluations</li> <li>8.2 Text Preprocessing Pipeline</li> <li>8.2.1 Multinomial Naive Bayes</li> <li>8.2.2 Tuned Multinomial</li> <li>8.2.3 Logistic Regression</li> <li>8.2.4 Tuned Logistic Regression</li> <li>8.2.5 Decision Tree Model</li> <li>8.2.6 Tuned DecisionTree</li> <li>8.2.7 Support Vector Classification</li> <li>8.2.8 Tuned SupportVector</li> <li>9.1 Best Model Results</li> <li>9.2 Next Steps and Future</li> <li>9.3 Appendix - Deep NLP</li> <li>9.3.1 Word2Vec Embedding</li> <li>9.3.2 Classifying With Embedding</li> <li>9.3.3 Pretrained Vector Model</li> <li>9.3.4 Mean Word Embedding</li> </ul> |  |  |

## 8.2.4 Tuned Logistic Regression Model

```
In [301]: 1 tuned_lr_pipe = Pipeline([('text_pipe',text_pipe),
2 ('clf',LogisticRegression(C=0.1,max_iter=100
3 solver='saga',
4 class_weight='bala
5
6 tuned_lr_pipe.fit(X_train,y_train)
7 evaluate_classification(tuned_lr_pipe,X_test,y_test,X_train=X_train,
y_train=y_train)
```

## Contents ⚙

1 Predicting NYC Restaurant

executed in 1m 15.2s, finished 19:06:08 2022-08-15

### 2 Introduction

#### 2.1 Objective

##### 2.1.1 Imports

|                                      | precision | recall | f1-score | support |
|--------------------------------------|-----------|--------|----------|---------|
| 0                                    | 0.75      | 0.57   | 0.65     | 2257    |
| 1                                    | 0.38      | 0.58   | 0.46     | 1009    |
| 3.1.1 Understanding NYC accuracy     |           |        | 0.57     | 3266    |
| 3.1.2 Target Variable - Macro Avg    | 0.57      | 0.58   | 0.55     | 3266    |
| 3.1.3 Engineering Target Weights Avg | 0.64      | 0.57   | 0.59     | 3266    |

#### 3.2 Obtaining Yelp Businesses

##### 3.2.1 Yelp Business Search

##### 3.2.2 Yelp Reviews

##### 3.2.3 Joining Yelp Reviews

#### 3.3 Joining NYC DOHMH Data

### 4 Scrubbing The Data

#### 4.1 Drop businesses missing

### 5 Exploring The Dataset

### 6 Preprocessing For Further

#### 6.0.1 Lets find most freq

### 7 Preprocessing For Modeling

### 8 Models

#### 8.1 Bag-of-Words Model

##### 8.1.1 Model Evaluations

#### 8.2 Text Preprocessing Pipeline

##### 8.2.1 Multinomial Naive Bayes

Training Score = 0.63

Test Score = 0.57

##### 8.2.3 Logistic Regression

##### 8.2.4 Tuned Logistic Regression

##### 8.2.5 Decision Tree Model

##### 8.2.6 Tuned DecisionTree Model

##### 8.2.7 Support Vector Classifier

##### 8.2.8 Tuned SupportVector Model

### 9 Conclusions

#### 9.1 Best Model Results

#### 9.2 Next Steps and Future

#### 9.3 Appendix - Deep NLP

##### 9.3.1 Word2Vec Embedding

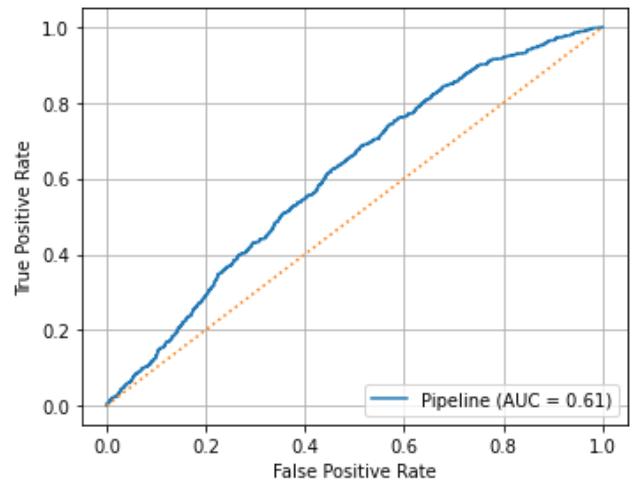
##### 9.3.2 Classifying With Embedding

##### 9.3.3 Pretrained Vector Model

##### 9.3.4 Mean Word Embedding

Predicted label

0 1



```
In [446]: 1 params = {'text_pipe__tf_transformer__use_idf':[True, False],
 2 'text_pipe__count_vectorizer__token_pattern':[None,r"([a-zA-Z]+(?:'[a-z]+)?)"],
 3 'text_pipe__count_vectorizer__stop_words':[None,stopwords_list],
 4 'clf__class_weight': ['balanced'],
 5 'clf__max_iter': [100, 500],
 6 'clf__C': [0.1, 1],
 7 'clf__solver': ['lbfqgs', 'sag', 'saga']}
```

## Contents ⚙

1 Predicting NYC Restaurants GridSearchCV(estimator=lr\_pipe, param\_grid = params, scoring='recall', verbose=3, n\_jobs=-1, cv=3)

2 Introduction 10

- 2.1 Objective 11
- 2.1.1 Imports 12

3 Obtain 13

- 3.1 Obtaining Data 13
- 3.1.1 Understanding NYC Fitting 3 folds for each of 96 candidates, totalling 288 fits
- 3.1.2 Target Variable -- N
- 3.1.3 Engineered Features [ParallelBackend(n\_jobs=-1)]: Using backend LokyBackend with 4 concurrent worker
- 3.2 Obtaining Yelp Businesses [Parallel(n\_jobs=-1)]: Done 24 tasks | elapsed: 2.7min
- 3.2.1 Yelp Business Details [Parallel(n\_jobs=-1)]: Done 120 tasks | elapsed: 12.5min
- 3.2.2 Yelp Reviews [Parallel(n\_jobs=-1)]: Done 280 tasks | elapsed: 28.1min
- 3.2.3 Joining Yelp Reviews [Parallel(n\_jobs=-1)]: Done 288 out of 288 | elapsed: 29.5min finished
- 3.3 Joining NYC DOHMH Data

4 Scrubbing The Data [clf\_\_C': 0.1,

- 4.1 Drop business miss\_weight': 'balanced',
- 4.2 Exploring The Data max\_iter': 100,
- 4.3 Preprocessing [clf\_\_solver': 'lbfqgs',
- 4.4 Text Pipeline [text\_pipe\_\_count\_vectorizer\_\_stop\_words': None,
- 4.5 Preprocessing [text\_pipe\_\_count\_vectorizer\_\_token\_pattern': "([a-zA-Z]+(?:'[a-z]+)?)",

5 Models

- 8.1 Bag-of-Words Model

```
In [448]: ModelEvaluation.text_pipe2 = Pipeline(steps=[
```

- 8.2 Text Preprocessing [Pipeline(count\_vectorizer', CountVectorizer(min\_df=.01, max\_df=.99, token\_pattern=r"([a-zA-Z]+(?:'[a-z]+)?)", lowercase=True, stop\_words=None, ngram\_range=(1, 2))),
- 8.2.1 Multinomial Naive Bayes
- 8.2.2 Tuned Multinomial
- 8.2.3 Logistic Regression
- 8.2.4 Tuned Logistic Regression
- 8.2.5 Decision Tree Model
- 8.2.6 Tuned DecisionTreeClassifier
- 8.2.7 Support Vector Classifier
- 8.2.8 Tuned SupportVectorClassifier

Out[448]: Pipeline

- 9.1 Best Model Results
- 9.2 Next Steps [CountVectorizer]
- 9.3 Appendix Deep NLP [TfidfTransformer]
- 9.3.1 Word2Vec Embedding
- 9.3.2 Classifying With Embedding
- 9.3.3 Pretrained Vector Model
- 9.3.4 Mean Word Embedding

```
In [449]: 1 tuned_lr_pipe2 = Pipeline([('text_pipe2',text_pipe2),
2 ('clf',LogisticRegression(C=0.1,max_iter=100
3 solver='lbfgs',
4 class_weight='balanced'))
5
6 tuned_lr_pipe2.fit(X_train,y_train)
7 evaluate_classification(tuned_lr_pipe2,X_test,y_test,X_train=X_train,
y_train=y_train)
```

## Contents ⚙️

1 Predicting NYC Restaurant  
executed in 1m 29.1s, finished 22:52:11 2022-08-15

### 2 Introduction

#### 2.1 Objective

##### 2.1.1 Imports

3 Obtain  
Passed  
Failed

#### 3.1 Obtaining Restaurant Info

##### 3.1.1 Understanding NYC Accuracy

##### 3.1.2 Target Variable

##### 3.1.3 Engineering Target

#### 3.2 Obtaining Yelp Business Data

##### 3.2.1 Yelp Business Search

##### 3.2.2 Yelp Reviews

##### 3.2.3 Joining Yelp Reviews

#### 3.3 Joining NYC DOHMH Data

#### 4 Scrubbing The Data

##### 4.1 Drop businesses with missing data

#### 5 Exploring The Dataset

#### 6 Preprocessing For Further Analysis

##### 6.0.1 Lets find most frequent words

#### 7 Preprocessing For Model Training

#### 8 Models

##### 8.1 Bag-of-Words Model

##### 8.1.1 Model Evaluations

##### 8.2 Text Preprocessing Pipeline

##### 8.2.1 Multinomial Naive Bayes

##### 8.2.2 Tuned Multinomial Naive Bayes

##### 8.2.3 Logistic Regression

##### 8.2.4 Tuned Logistic Regression

##### 8.2.5 Decision Tree Model

##### 8.2.6 Tuned DecisionTree Model

##### 8.2.7 Support Vector Classifier

##### 8.2.8 Tuned SupportVector Classifier

#### 9 Conclusions

##### Test Score = 0.57

##### 9.1 Best Model Results

##### 9.2 Next Steps and Future

##### 9.3 Appendix - Deep NLP

##### By tuning the Logistic Regression model we were able to increase Recall by 6% but this cost our

##### accuracy to drop 6% on the validation data. Which is a good trade-off for the business problem.

##### 9.3.3 Pretrained Vector Model

```
In [952]: MeanWordFeature = y_test.reset_index(drop=True)
```

executed in 14ms, finished 22:54:42 2022-08-15

|        | precision | recall | f1-score | support |
|--------|-----------|--------|----------|---------|
| Passed | 0.76      | 0.56   | 0.64     | 2708    |
| Failed | 0.38      | 0.60   | 0.46     | 1211    |

&lt;/div

In [453]: 1 lr\_probas = tuned\_lr\_pipe2.predict\_proba(X\_test)

executed in 8.41s, finished 22:54:52 2022-08-15

In [454]: 1 probs\_df = pd.DataFrame(lr\_probas, columns= ['Pass', 'Fail'])  
2 probs\_df = pd.concat([probs\_df,y\_actual],axis=1)  
3 probs\_df

## Contents

executed in 26ms, finished 22:54:52 2022-08-15

1 Predicting NYC Restaurant

Out[454]:

|       |                            | Pass | Fail     | Severe  |
|-------|----------------------------|------|----------|---------|
| ▼ 2   | Introduction               | 0    | 0.60     | 0.40    |
| ▼ 2.1 | Objective                  | 0    | 0        | 0       |
| 2.1.1 | Imports                    | 0    | 0.60     | 0.40    |
| ▼ 3   | Obtain                     | 1    | 0.48     | 0.52    |
| ▼ 3.1 | Obtaining Restaurant Ir    | 2    | 0.57     | 0.43    |
| 3.1.1 | Understanding NYC          | 2    | 0.57     | 0.43    |
| 3.1.2 | Target Variable            | 3    | 0.57     | 0.43    |
| 3.1.3 | Engineering Target         | 4    | 0.45     | 0.55    |
| ▼ 3.2 | Obtaining Yelp Business    | 4    | 0.45     | 0.55    |
| 3.2.1 | Yelp Business Search       | ...  | ...      | ...     |
| 3.2.2 | Yelp Reviews               | 3914 | 0.43     | 0.57    |
| 3.2.3 | Joining Yelp Review        | 3914 | 0.43     | 0.57    |
| 3.3   | Joining Yelp Data          | 3915 | 0.40     | 0.54    |
| ▼ 4   | Scrubbing The Data         | 3916 | 0.49     | 0.51    |
| 4.1   | Drop businesses missing    | 3916 | 0.49     | 0.51    |
| 5     | Exploring The Dataset      | 3917 | 0.47     | 0.53    |
| ▼ 6   | Preprocessing For Further  | 3918 | 0.45     | 0.55    |
| 6.0.1 | Lets find most freq        | 3918 | 0.45     | 0.55    |
| 7     | Preprocessing For Modeling | 3919 | rows x 3 | columns |
| ▼ 8   | Models                     | 3919 | rows x 3 | columns |

▼ 8.1 Bag-of-Words Model

Here above the model returns the probability a restaurant will pass or fail its inspection. This can be useful to inspectors who will be visiting all businesses but can prioritize those most at risk.

▼ 8.2 Text Preprocessing Pipeline

8.2.1 Multinomial Naive Bayes

8.2.2 Tuned Multinomial

8.2.3 Logistic Regression

8.2.4 Tuned Logistic Reg

8.2.5 Decision Tree Model

In [330]: 1 # Make a full pipeline with the DecisionTree model as the second step

8.2.6 Tuned DecisionTree

dt\_pipe = Pipeline([('text\_pipe',text\_pipe),

('clf',DecisionTreeClassifier(max\_depth=6,class\_weight='balanced'))]

8.2.8 Tuned Support Vector

8.2.9 Tuned SVC

▼ 9 Conclusions

executed in 86ms, finished 19:25:24 2022-08-15

9.1 Best Model Results

Out[330]: Pipeline

9.2 Next Steps and Future Work

▼ 9.3 Appendix - text\_pipe: Pipeline

9.3.1 Word2Vec Embedding

9.3.2 Classifying With TFIDF

9.3.3 Pretrained Vectorizer

9.3.4 Mean Word Embedding

DecisionTreeClassifier

In [333]:

```

1 ## Modeling with full pipeline
2 dt_pipe.fit(X_train,y_train)
3 evaluate_classification(dt_pipe,X_test,y_test,X_train=X_train, y_train=y_train)

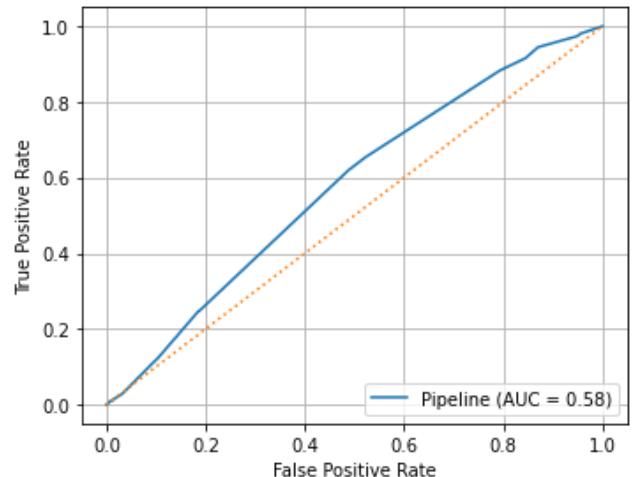
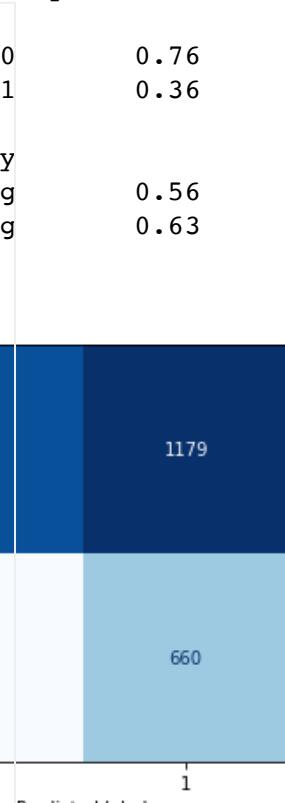
```

executed in 55.8s, finished 19:30:48 2022-08-15

|   | precision | recall | f1-score | support |
|---|-----------|--------|----------|---------|
| 0 | 0.76      | 0.48   | 0.59     | 2257    |
| 1 | 0.36      | 0.65   | 0.46     | 1009    |

## Contents ⚙️

- 1 Predicting NYC Restaurant
- ▼ 2 Introduction
  - ▼ 2.1 Objective accuracy
    - 2.1.1 Imports macro avg
    - 2.1.2 weighted avg
- ▼ 3 Obtain
  - ▼ 3.1 Obtaining Restaurant Data
    - 3.1.1 Understanding NYC
    - 3.1.2 Target Variable -- NYC
    - 3.1.3 Engineering Target
  - ▼ 3.2 Obtaining Yelp Business Data
    - 3.2.1 Yelp Business Search
    - 3.2.2 Yelp Reviews
    - 3.2.3 Joining Yelp Reviews
  - 3.3 Joining NYC DOHMH & Yelp Data
- ▼ 4 Scrubbing The Data
  - 4.1 Drop businesses missing
- 5 Exploring The Dataset
- ▼ 6 Preprocessing For Further Analysis
  - 6.0.1 Lets find most frequent words
- 7 Preprocessing For Modeling
- ▼ 8 Models
  - ▼ 8.1 Bag-of-Words Model
    - 8.1.1 Model Evaluations
  - ▼ 8.2 Text Preprocessing Pipeline
    - Training Score = 0.57
    - Test Score = 0.53
    - 8.2.1 Multinomial Naive Bayes
    - 8.2.2 Tuned Multinomial Naive Bayes
    - 8.2.3 Logistic Regression
    - 8.2.4 Tuned Logistic Regression
    - 8.2.5 Decision Tree Model
    - 8.2.6 Tuned DecisionTreeClassifier



In [466]:

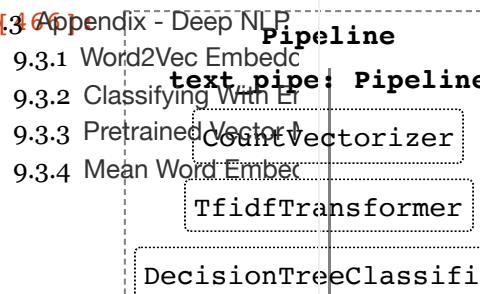
```

1 ## Make a full pipeline with the DecisionTree model as the second step
2 dt_pipe = Pipeline([('text_pipe',text_pipe),
3 ('clf',DecisionTreeClassifier(max_depth=6,class_weight='balanced'))]
4)

```

executed in 478ms, finished 01:25:20 2022-08-16

Out[466]:



In [467]:

```

1 ## Modeling with full pipeline
2 dt_pipe.fit(X_train,y_train)
3 evaluate_classification(dt_pipe,X_test,y_test,X_train=X_train, y_train=y_train)

```

executed in 1m 21.3s, finished 01:26:43 2022-08-16

|        | precision | recall | f1-score | support |
|--------|-----------|--------|----------|---------|
| Passed | 0.73      | 0.71   | 0.72     | 2708    |
| Failed | 0.39      | 0.41   | 0.40     | 1211    |

## Contents ↗

- 1 Predicting NYC Restaurants

- 2 Introduction

- 2.1 Objective accuracy

- 2.1.1 Imports macro avg  
2.1.2 weighted avg

- 3 Obtain

- 3.1 Obtaining Restaurant Data

- 3.1.1 Understanding NYC

- 3.1.2 Target Variable -- N

- 3.1.3 Engineering Target

- 3.2 Obtaining Yelp Business Data

- 3.2.1 Yelp Business Search

- 3.2.2 Yelp Reviews

- 3.2.3 Joining Yelp Reviews

- 3.3 Joining NYC DOHMH Data

- 4 Scrubbing The Data

- 4.1 Drop businesses missing

- 5 Exploring The Dataset

- 6 Preprocessing For Further

- 6.0.1 Lets find most freq

- 7 Preprocessing For Modeling

- 8 Models

- 8.1 Bag-of-Words Model

- 8.1.1 Model Evaluations

- 8.2 Text Preprocessing Pipeline

- 8.2.1 Multinomial Naive Bayes

- 8.2.2 Tuned Multinomial

- 8.2.3 Logistic Regression

- 8.2.4 Tuned Logistic Reg

- 8.2.5 Decision Tree Model

- 8.2.6 Tuned Decision Tree

In [468]:

```
Make a full pipeline with the DecisionTree model as the second step
```

```
text_pipe2 = Pipeline([('text_pipe2',text_pipe),
```

```
('clf',DecisionTreeClassifier(max_depth=6,class_weight='balanced'))
```

```
dt_pipe2 = Pipeline([('text_pipe2',text_pipe2),
```

```
('clf',DecisionTreeClassifier(max_depth=6,class_weight='balanced'))]
```

```
dt_pipe2.fit(X_train,y_train)
```

```
evaluate_classification(dt_pipe2,X_test,y_test,X_train=X_train, y_train=y_train)
```

executed in 180ms, finished 01:27:15 2022-08-16

Out[468]:

Pipeline

Support Vector Classifier

TfidfTransformer

CountVectorizer

- 9 Conclusions

- 9.1 Best Model Results

- 9.2 Next Steps and Future

- 9.3 Appendix - Deep NLP

- 9.3.1 Word2Vec Embedding

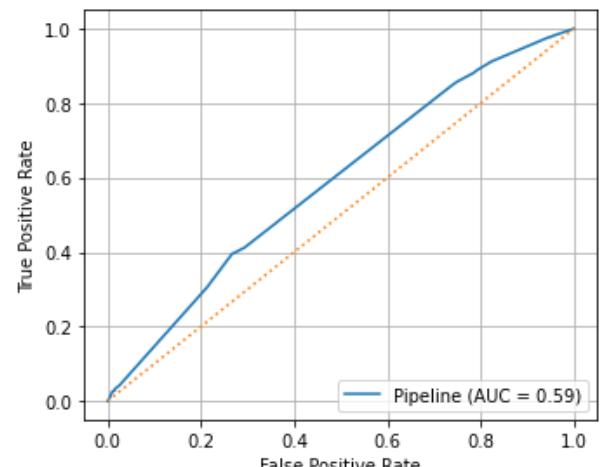
- 9.3.2 DecisionTreeClassifier

- 9.3.3 Classifying With Embeddings

- 9.3.4 Pretrained Vector Model

- 9.3.5 Mean Word Embedding

|        | precision | recall | f1-score | support |
|--------|-----------|--------|----------|---------|
| Passed | 0.73      | 0.71   | 0.72     | 2708    |
| Failed | 0.39      | 0.41   | 0.40     | 1211    |



In [469]:

```

1 ## Modeling with full pipeline
2 dt_pipe2.fit(X_train,y_train)
3 evaluate_classification(dt_pipe2,X_test,y_test,X_train=X_train, y_train=y_train)

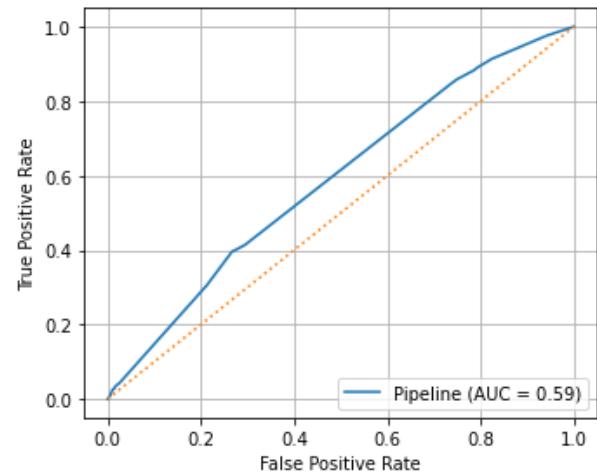
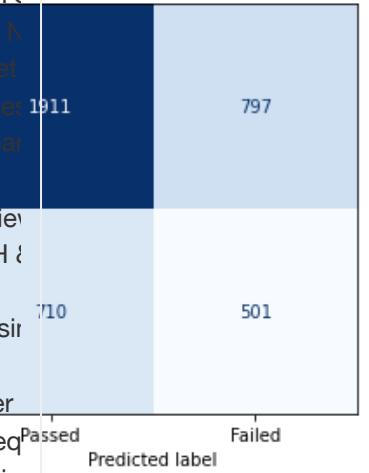
```

executed in 1m 6.02s, finished 01:28:23 2022-08-16

|        | precision | recall | f1-score | support |
|--------|-----------|--------|----------|---------|
| Passed | 0.73      | 0.71   | 0.72     | 2708    |
| Failed | 0.39      | 0.41   | 0.40     | 1211    |

## Contents ⚙️ Passed

- 1 Predicting NYC Restaurants
- 2 Introduction
  - 2.1 Objective accuracy
    - 2.1.1 Imports macro avg
    - 2.1.2 weighted avg
- 3 Obtain
  - 3.1 Obtaining Restaurant Data
    - 3.1.1 Understanding NYC Data
    - 3.1.2 Target Variable -- NYC Data
    - 3.1.3 Engineering Target Variable
  - 3.2 Obtaining Yelp Business Data
    - 3.2.1 Yelp Business Search API
    - 3.2.2 Yelp Reviews
    - 3.2.3 Joining Yelp Reviews
  - 3.3 Joining NYC DOHMH & Yelp Data
- 4 Scrubbing The Data
  - 4.1 Drop businesses missing data
- 5 Exploring The Dataset
- 6 Preprocessing For Further Analysis
  - 6.0.1 Lets find most frequent words
- 7 Preprocessing For Modelin
- 8 Models
  - 8.1 Bag-of-Words Model
    - 8.1.1 Model Evaluations
  - 8.2 Text Preprocessing Pipeline
    - 8.2.1 Multinomial Naive Bayes
    - 8.2.2 Tuned Multinomial Naive Bayes
    - 8.2.3 Logistic Regression score = 0.66
    - 8.2.4 Tuned Logistic Regressor score = 0.62
    - 8.2.5 Decision Tree Model
    - 8.2.6 Tuned DecisionTreeClassifier
    - The DecisionTree model has shown the best Recall score yet. Not a great accuracy but catching 65% of the businesses that failed their inspections is a 7% improvement from the Logistic Regression classifier.
    - 8.2.7 Support Vector Classifier
    - 8.2.8 Tuned SupportVectorClassifier
- 9 Conclusions
  - 9.1 Best Model Results
  - 9.2 Next Steps and Future Work
  - 9.3 Appendix - Deep NLP
    - 9.3.1 Word2Vec Embeddings
    - 9.3.2 Classifying With Embeddings
    - 9.3.3 Pretrained Vector Model
    - 9.3.4 Mean Word Embedding



In [336]: 1 dt\_pipe.get\_params().keys()

executed in 10ms, finished 19:36:04 2022-08-15

Out[336]: dict\_keys(['memory', 'steps', 'verbose', 'text\_pipe', 'clf', 'text\_pipe\_memory', 'text\_pipe\_steps', 'text\_pipe\_verbose', 'text\_pipe\_count\_vectorizer', 'text\_pipe\_tf\_transformer', 'text\_pipe\_count\_vectorizer\_analyzer', 'text\_pipe\_count\_vectorizer\_binary', 'text\_pipe\_count\_vectorizer\_decode\_error', 'text\_pipe\_count\_vectorizer\_dtype', 'text\_pipe\_count\_vectorizer\_encoding', 'text\_pipe\_count\_vectorizer\_input', 'text\_pipe\_count\_vectorizer\_lowercase', 'text\_pipe\_count\_vectorizer\_max\_df', 'text\_pipe\_count\_vectorizer\_max\_features', 'text\_pipe\_count\_vectorizer\_min\_df', 'text\_pipe\_count\_vectorizer\_ngram\_range', 'text\_pipe\_count\_vectorizer\_nlp\_processor', 'text\_pipe\_count\_vectorizer\_stop\_words', 'text\_pipe\_count\_vectorizer\_strip\_accents', 'text\_pipe\_count\_vectorizer\_tokenize', 'text\_pipe\_count\_vectorizer\_tokenizer', 'text\_pipe\_count\_vectorizer\_tokenizer\_norm', 'text\_pipe\_tf\_transformer\_smooth\_idf', 'text\_pipe\_tf\_transformer\_sublinear\_tf', 'text\_pipe\_target\_transformer\_use\_idf', 'clf\_ccp\_alpha', 'clf\_class\_weight', 'clf\_max\_depth', 'clf\_max\_features', 'clf\_max\_leaf\_n', 'clf\_min\_impurity\_decrease', 'clf\_min\_impurity\_split', 'clf\_min\_samples\_leaf', 'clf\_min\_samples\_split', 'clf\_min\_weight\_fraction\_leaf', 'clf\_presort', 'clf\_random\_state', 'clf\_splitter'])

## Contents

- 1 Predicting NYC Restaurant Encoding
- 2 Introduction to Count Vectorizer Lowercase
- 2.1 Objective of Count Vectorizer Max Features
- 2.1.1 Imports
- 3 Obtain the Count Vectorizer Strip Accents
- 3.1 Obtaining Restaurant Data Pattern
- 3.1.1 Understanding NYC Vocabulary
- 3.1.2 Target Variable
- 3.1.3 Engineering Target Transformer Use Idf
- 3.2 Obtaining Yelp Business Data
- 3.2.1 Yelp Business Data Min Impurity Decrease
- 3.2.2 Yelp Business Data Min Samples Leaf
- 3.2.3 Joining Yelp Review Data
- 3.3 Joining NYC DOHMH Data
- 4 Scrubbing The Data
- 4.1 Drop businesses missing address
- 5 Exploring The Dataset
- 6 Preprocessing For Further Analysis
- 6.0.1 Lets Find most frequent words
- 7 Preprocessing For Modelin
- 8 Models
- 8.1 Bag-of-Words Model
- 8.1.1 Model Evaluations
- 8.2 Text Preprocessing Pipeline
- 8.2.1 Multinomial Naive Bayes
- 8.2.2 Tuned Multinomial Naive Bayes
- 8.2.3 Logistic Regression
- 8.2.4 Tuned Logistic Regression
- 8.2.5 Decision Tree Model
- 8.2.6 Tuned DecisionTreeClassifier
- 8.2.7 Support Vector Classifier
- 8.2.8 Tuned SupportVectorClassifier
- 9 Conclusions
- 9.1 Best Model Results
- 9.2 Next Steps and Future Work
- 9.3 Appendix - Deep NLP
- 9.3.1 Word2Vec Embedding
- 9.3.2 Classifying Words
- 9.3.3 Pretrained Vector Model
- 9.3.4 Mean Word Embedding

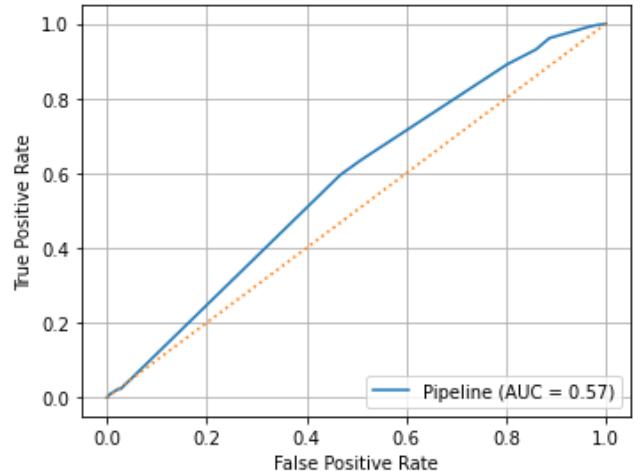
## 8.2.6 Tuned DecisionTree Model

```
In [340]: 1 tuned_dt_pipe = Pipeline([('text_pipe',text_pipe),
2 ('clf',DecisionTreeClassifier(max_depth=4,
3 criterion='gini',
4 class_weight='balanced')),
5])
6
```

**Contents** ↗ 

- 1 Predicting NYC Restaurant
- 2 Introduction executed in 56.3s, finished 19:46:11 2022-08-15
- 3 Obtain
- 4 Scrubbing The Data
- 5 Exploring The Dataset
- 6 Preprocessing For Further
- 7 Preprocessing For Modelin
- 8 Models
- 9 Conclusions

|                                          |                                                                                              | precision | recall | f1-score | support |
|------------------------------------------|----------------------------------------------------------------------------------------------|-----------|--------|----------|---------|
| 2.1.1 Imports                            | 0                                                                                            | 0.75      | 0.53   | 0.62     | 2257    |
| 3.1 Obtaining Restaurant Information     | 1                                                                                            | 0.36      | 0.60   | 0.45     | 1009    |
| 3.1.1 Understanding NYC DOHMH Data       |                                                                                              |           |        | 0.55     | 3266    |
| 3.1.2 Target Variable                    |                                                                                              |           |        | 0.54     | 3266    |
| 3.1.3 Engineering Target Variable        |                                                                                              | 0.55      | 0.56   | 0.54     | 3266    |
| 3.2 Obtaining Yelp Business Data         |                                                                                              | 0.63      | 0.55   | 0.57     | 3266    |
| 3.2.1 Yelp Business Search               |                                                                                              |           |        |          |         |
| 3.2.2 Yelp Reviews                       |                                                                                              |           |        |          |         |
| 3.2.3 Joining Yelp Reviews               |                                                                                              |           |        |          |         |
| 3.3 Joining NYC DOHMH Data               |                                                                                              |           |        |          |         |
| 4.1 Drop businesses missing              | 1291                                                                                         |           |        |          |         |
| 5 Exploring The Dataset                  |                                                                                              |           |        |          |         |
| 6 Preprocessing For Further              |                                                                                              |           |        |          |         |
| 6.0.1 Lets find most frequent words      |                                                                                              |           |        |          |         |
| 7 Preprocessing For Modeling             |                                                                                              |           |        |          |         |
| 8 Models                                 | 1                                                                                            |           |        |          |         |
| 8.1 Bag-of-Words Model                   | 408                                                                                          |           |        |          |         |
| 8.1.1 Model Evaluations                  |                                                                                              |           |        |          |         |
| 8.2 Text Preprocessing Pipeline          | 0                                                                                            |           |        |          |         |
| 8.2.1 Multinomial Naive Bayes            | 1                                                                                            |           |        |          |         |
| 8.2.2 Tuned Multinomial Naive Bayes      |                                                                                              |           |        |          |         |
| 8.2.3 Logistic Regression                |                                                                                              |           |        |          |         |
| 8.2.4 Tuned Logistic Regression          |                                                                                              |           |        |          |         |
| 8.2.5 Decision Tree Model                |                                                                                              |           |        |          |         |
| 8.2.6 Tuned DecisionTree Model           |                                                                                              |           |        |          |         |
| 8.2.7 Support Vector Classification      |                                                                                              |           |        |          |         |
| 8.2.8 Tuned SupportVector Classification |                                                                                              |           |        |          |         |
| 9 Conclusions                            | Training Score = 0.58                                                                        |           |        |          |         |
| 9.1 Best Model Results                   | Test Score = 0.55                                                                            |           |        |          |         |
| 9.2 Next Steps and Future                |                                                                                              |           |        |          |         |
| 9.3 Appendix - Deep NLP                  | Surprisingly, the cross-validation GridSearch did not improve the models Recall performance. |           |        |          |         |
| 9.3.1 Word2Vec Embedding                 |                                                                                              |           |        |          |         |
| 9.3.2 Classifying With Embedding         |                                                                                              |           |        |          |         |



```
In [931]: 1 dt_pipe_proba = dt_pipe.predict_proba(X_test)
2
```

```
In [373]: 1 probs_dt = pd.DataFrame(dt_probas, columns= ['Pass', 'Fail'])
2 probs_dt = pd.concat([probs_dt,y_actual],axis=1)
3 probs_dt
```

executed in 28ms, finished 20:51:41 2022-08-15

Out[373]:

**Contents** ⚙ Pass Fail Severe

|       |                            |      |      |      |     |
|-------|----------------------------|------|------|------|-----|
| 1     | Predicting NYC Restaurant  | 0    | 0.59 | 0.41 | 0   |
| ▼ 2   | Introduction               | 1    | 0.33 | 0.67 | 0   |
| ▼ 2.1 | Objective                  | 1    | 0.33 | 0.67 | 0   |
| 2.1.1 | Imports                    | 2    | 0.59 | 0.41 | 0   |
| ▼ 3   | Obtain                     | 3    | 0.36 | 0.64 | 0   |
| ▼ 3.1 | Obtaining Restaurant       | 11   | 0.36 | 0.64 | 0   |
| 3.1.1 | Understanding NY           | 4    | 0.36 | 0.64 | 0   |
| 3.1.2 | Target Variable -- N       | 1    | 0.36 | 0.64 | 0   |
| 3.1.3 | Engineering Target         | 1    | 0.36 | 0.64 | 0   |
| ▼ 3.2 | Obtaining Yelp Business    | 26   | 0.46 | 0.54 | 0   |
| 3.2.1 | Yelp Business Search       | 3262 | 0.46 | 0.54 | 0   |
| 3.2.2 | Yelp Reviews               | 1    | 0.46 | 0.54 | 0   |
| 3.2.3 | Joining Yelp Reviews       | 3263 | 0.25 | 0.75 | 0   |
| 3.3   | Joining NYC DOHMH          | 3264 | 0.59 | 0.41 | 0   |
| ▼ 4   | Scrubbing The Data         | 3265 | 0.46 | 0.54 | 1   |
| 4.1   | Drop business misspells    | 3265 | 0.46 | 0.54 | 1   |
| 5     | Exploring The Dataset      |      |      |      |     |
| ▼ 6   | Preprocessing              | 3266 | 0.60 | 0.39 | 0.8 |
| 6.0.1 | Lets find most freq        |      |      |      |     |
| 7     | Preprocessing For Modeling |      |      |      |     |
| ▼ 8   | Models                     |      |      |      |     |

```
In [374]: 8.1 Bag-of-Words Model
 pipe.named_steps['clf'].feature_importances_
```

8.1.1 Model Evaluations

executed in 54ms, finished 21:41:26 2022-08-15

▼ 8.2 Text Preprocessing Pipeline

Out[402]: array([0., 0., ..., 0., 0., 0.])

8.2.1 Multinomial Naive

8.2.2 Tuned Multinomial

8.2.3 Logistic Regression

8.2.4 Tuned Logistic Reg

8.2.5 Decision Tree Model

**8.2.7 Support Vector Classifier Model**

In [375]: Tuned Decision Tree to a full pipeline with the Support Vector Classifier model as the

8.2.7 Support Vector Pipe = Pipeline([('text\_pipe',text\_pipe),

('clf',SVC(class\_weight='balanced'))])

▼ 9 Conclusions 4 svc\_pipe

9.1 Best Model Results

executed in 282ms, finished 22:58:52 2022-08-15

9.2 Next Steps and Future

Out[458]: Pipeline

▼ 9.3 Appendix - Deep NLP

9.3.1 Word Embedding Pipeline

9.3.2 Classifying With CountVectorizer

9.3.3 Pretrained Vector

9.3.4 Mean Word Embedding Transformer

SVC

In [459]:

```

1 ## Modeling with full pipeline
2 svc_pipe.fit(X_train,y_train)
3 evaluate_classification(svc_pipe,X_test,y_test,X_train=X_train, y_train=y_train)

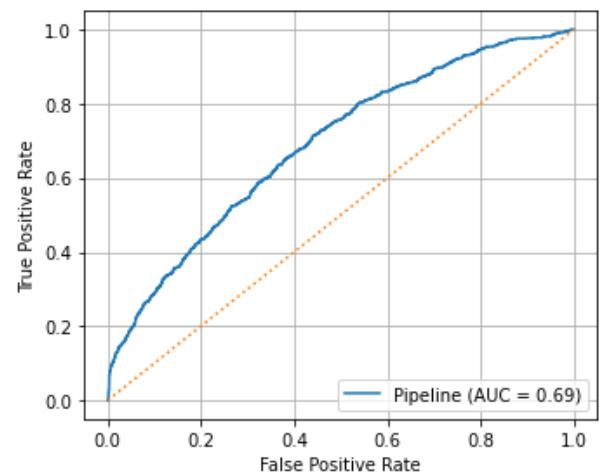
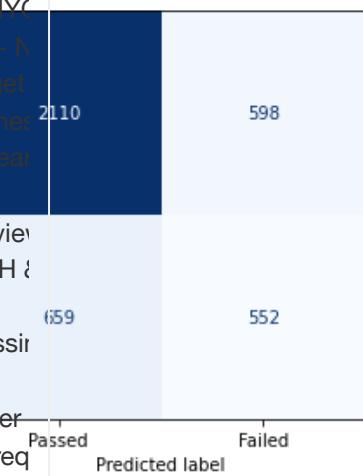
```

executed in 19m 45s, finished 23:18:38 2022-08-15

|        | precision | recall | f1-score | support |
|--------|-----------|--------|----------|---------|
| Passed | 0.76      | 0.78   | 0.77     | 2708    |
| Failed | 0.48      | 0.46   | 0.47     | 1211    |

## Contents Passed

- 1 Predicting NYC Restaurants
- ▼ 2 Introduction
  - 2.1 Objective accuracy
    - 2.1.1 Imports
    - 2.1.2 weighted avg
- ▼ 3 Obtain
  - 3.1 Obtaining Restaurant Info
    - 3.1.1 Understanding NYC Data
    - 3.1.2 Target Variable -- NYC Health Violations
    - 3.1.3 Engineering Target Variable
  - 3.2 Obtaining Yelp Business Reviews
    - 3.2.1 Yelp Business Search API
    - 3.2.2 Yelp Reviews API
    - 3.2.3 Joining Yelp Reviews with NYC DOHMH Data
- ▼ 4 Scrubbing The Data
- 5 Exploring The Dataset
- ▼ 6 Preprocessing For Further Analysis
  - 6.0.1 Lets find most frequent words
- 7 Preprocessing For Modelin
- ▼ 8 Models
  - 8.1 Bag-of-Words Model
  - 8.1.1 Model Evaluation Scores = 0.68
  - 8.2 Text Preprocessing Pipeline
    - 8.2.1 Multinomial Naive Bayes
    - 8.2.2 Tuned Multinomial Naive Bayes
    - 8.2.3 Logistic Regression Model
      - Not a very good baseline from the SVC model. Recall score of 46% and 68% accuracy. We'll see if tuned parameters make an improvement.**
    - 8.2.4 Tuned Logistic Regression Model
    - 8.2.5 Decision Tree Model
    - 8.2.6 Tuned DecisionTree Model
    - 8.2.7 Support Vector Classification
    - 8.2.8 Tuned SupportVector Classification
- ▼ 9 Conclusions
  - 9.1 Best Model Results
  - 9.2 Next Steps and Future
- ▼ 9.3 Appendix - Deep NLP
  - 9.3.1 Word2Vec Embedding
  - 9.3.2 Classifying With Embedding
  - 9.3.3 Pretrained Vector Model
  - 9.3.4 Mean Word Embedding



## Hyperparameter Tuning With GridsearchCV

```
In [460]: 1 svc_pipe.get_params().keys()
```

executed in 31ms. finished 23:18:38 2022-08-15

```
Out[460]: dict_keys(['memory', 'steps', 'verbose', 'text_pipe', 'clf', 'text_pipe_
 memory', 'text_pipe_steps', 'text_pipe_verbose', 'text_pipe_count_vect
 orizer', 'text_pipe_tf_transformer', 'text_pipe_count_vectorizer_analy
 zer', 'text pipe count vectorizer binary', 'text pipe count vectorizer'])
```

## Contents

In [463]: 1 text\_pipe2

executed in 194ms, finished 00:32:52 2022-08-16

Out[463]:

### Pipeline

#### CountVectorizer

```
CountVectorizer(max_df=0.99, min_df=0.01, ngram_range=(1, 2),
 token_pattern='([a-zA-Z]+(?:'[a-z]+)?)')
```

#### TfidfTransformer

```
TfidfTransformer(use_idf=False)
```

## Contents

|                                                |
|------------------------------------------------|
| 1 Predicting NYC Restaurant                    |
| ▼ 2 Introduction                               |
| ▼ 2.1 Objective                                |
| 2.1.1 Imports                                  |
| ▼ 3 Obtain                                     |
| ▼ 3.1 Obtaining Restaurant Info                |
| 3.1.1 Understanding NYC Restaurants            |
| 3.1.2 Target Variable -- NYC Health Violations |
| 3.1.3 Engineering Target Variable              |
| ▼ 3.2 Obtaining Yelp Business Data             |
| 3.2.1 Yelp Business Search API                 |
| 3.2.2 Yelp Reviews                             |
| 3.2.3 Joining Yelp Reviews                     |
| 3.3 Joining NYC DOHMH Data                     |
| ▼ 4 Scrubbing The Data                         |
| 4.1 Drop businesses missing address            |
| 5 Exploring The Dataset                        |
| ▼ 6 Preprocessing For Further Modeling         |
| 6.0.1 Lets find most frequent words            |
| 7 Preprocessing For Modeling                   |
| ▼ 8 Models                                     |
| ▼ 8.1 Bag-of-Words Model                       |
| 8.1.1 Model Evaluations                        |
| ▼ 8.2 Text Preprocessing Pipeline              |
| 8.2.1 Multinomial Naive Bayes                  |
| 8.2.2 Tuned Multinomial Naive Bayes            |
| 8.2.3 Logistic Regression                      |
| 8.2.4 Tuned Logistic Regression                |
| 8.2.5 Decision Tree Model                      |
| 8.2.6 Tuned DecisionTree Model                 |
| 8.2.7 Support Vector Classifier                |
| 8.2.8 Tuned SupportVector Classifier           |
| ▼ 9 Conclusions                                |
| 9.1 Best Model Results                         |
| 9.2 Next Steps and Future Work                 |
| ▼ 9.3 Appendix - Deep NLP                      |
| 9.3.1 Word2Vec Embeddings                      |
| 9.3.2 Classifying With Embeddings              |
| 9.3.3 Pretrained Vector Embeddings             |
| 9.3.4 Mean Word Embedding                      |

## 8.2.8 Tuned SupportVectorClassifier Model

```
In [465]: 1 tuned_svc_pipe = Pipeline([('text_pipe2',text_pipe2),
2 ('clf',SVC(gamma='scale',kernel='sigmoid',
3 class_weight='balanced'))
4])
5
6 tuned_svc_pipe.fit(X_train,y_train)
7 evaluate_classification(tuned_svc_pipe,X_test,y_test,X_train=X_train,
8 y_train=y_train)
```

## Contents ⚙️

1 Predicting NYC Restaurant  
executed in 40m 2s finished 01:14:34 2022-08-16

### 2 Introduction

#### 2.1 Objective

##### 2.1.1 Imports

|          |        | precision | recall | f1-score | support |
|----------|--------|-----------|--------|----------|---------|
| 3 Obtain | Passed | 0.74      | 0.61   | 0.67     | 2708    |
|          | Failed | 0.37      | 0.51   | 0.43     | 1211    |

#### 3.1 Obtaining Restaurant Info

##### 3.1.1 Understanding NYC Accuracy

##### 3.1.2 Target Variable - Macro Avg

##### 3.1.3 Engineering Target Weights Avg

#### 3.2 Obtaining Yelp Business Data

##### 3.2.1 Yelp Business Search

##### 3.2.2 Yelp Reviews

##### 3.2.3 Joining Yelp Reviews

#### 3.3 Joining NYC DOHMH Data

#### 4 Scrubbing The Data

##### 4.1 Drop businesses with missing data

#### 5 Exploring The Dataset

#### 6 Preprocessing For Further Analysis

##### 6.0.1 Lets find most frequent words

#### 7 Preprocessing For Modeling

#### 8 Models

##### 8.1 Bag-of-Words Model

##### 8.1.1 Model Evaluations

##### 8.2 Text Preprocessing Pictures

##### 8.2.1 Multinomial Naive Bayes

##### 8.2.2 Tuned Multinomial Test Score = 0.61

##### 8.2.3 Logistic Regression

##### 8.2.4 Tuned Logistic Regression Model results

##### 8.2.5 Decision Tree Model

##### 8.2.6 Tuned DecisionTree Model

##### 8.2.7 Support Vector Classification

##### 8.2.8 Tuned SupportVectorModel

## 9 Conclusions

#### 9.1 Best Model Results

#### 9.2 Next Steps and Future Work

#### 9.3 Appendix - Deep NLP

##### 9.3.1 Word2Vec Embedding

##### In 9.3.2 Classifying With Embeddings

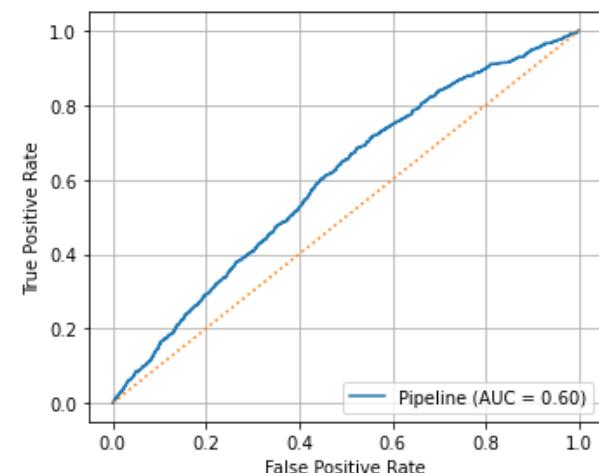
##### 9.3.3 Pretrained Vector Embedding

##### 9.3.4 Mean Word Embedding

#### 9.2 Next Steps and Future Work

## Data Source

## 9.1 Best Model Results



Hyperparameter Tuning

Neural Network Training

## 9.3 Appendix - Deep NLP Models

### Contents ⚙️

- 1 Predicting NYC Restaurant Violations After fitting and tuning a handful of bag-of-n-grams, we can try solving the business problem with a simple neural network. First we will need to generate word embeddings. Word embeddings map each word to one vector, which can then be learned in a way which resembles a neural network.
- 2.1 Objective 2.1.1 Imports Embedding vectors differ from bag-of-n-grams because they try to capture various characteristics
- 2.2 Introduction
- 3 Obtain 3.1 Obtaining Restaurant Reviews 3.1.1 Understanding NYC
- 3.1.2 Target Variable -- NYC
- 3.1.3 Engineered Features The simple bag-of-words featurized models above summarize a document, in this case restaurant reviews, by the words present in that review and ignores their context. Below we will attempt to take context into account using Recurrent Neural Networks (RNNs) that treat the document as a sequence, taking account of all the words in the context of those preceded and following.
- 3.2 Obtaining Yelp Reviews 3.2.1 Yelp Business Sequences
- 3.2.2 Yelp Reviews
- 3.2.3 Joining Yelp Reviews
- 3.3 Joining NYC DOHMH Data
- 4 Scrubbing The Data 4.1 Drop businesses missing
- 5 Exploring The Dataset
- 6 Preprocessing For Further Word2Vec CBOW 6.0.1 Lets find most freq
- 7 Preprocessing For Model In [477]: 1 # Minimally preprocessing for Deep NLP with all lower case
- 8 Models 2 # Keeping stopwords to help provide further word context
- 8.1 Bag-of-Words Model 3 lower\_data = model\_df['Reviews'].map(lambda x: simple\_preprocess(x.lower()))  
4 deacc=
- 8.2 Text Preprocessing Pipeline  
executed in 1m 14s, finished 08:09:36 2022-08-16
- 8.2.1 Multinomial Naive Bayes
- 8.2.2 Tuned Multinomial Logistic Regression
- In [478]: 1 # CBOW (Continuous Bag of Words)
- 8.2.3 Logistic Regression 2 w2v\_cbow\_model = Word2Vec(lower\_data, size=50, window=3, min\_count=3, sg=0)  
executed in 5m 12s, finished 08:14:58 2022-08-16
- 8.2.4 Tuned Logistic Regression
- 8.2.5 Decision Tree Model
- 8.2.6 Tuned DecisionTreeClassifier
- In [480]: 1 w2v\_cbow\_model
- 8.2.7 Support Vector Classifier  
executed in 1m 14s, finished 08:15:25 2022-08-16
- 8.2.8 Tuned SupportVectorClassifier

Out[480]: <gensim.models.word2vec.Word2Vec at 0x7fc8a5ee8340>

- 9.1 Best Model Results
- 9.2 Next Steps and Future
- In [481]: 1 examples = w2v\_cbow\_model.corpus\_count
- 9.3 Appendix - Deep NLP
- 9.3.1 Word2Vec Embedding  
executed in 1m 26s, finished 08:15:54 2022-08-16
- 9.3.2 Classifying With Embedding

In [482]: 1 # Training Word2Vec Model

9.3.3 Pretrained Vector 2 w2v\_cbow\_model.train(lower\_data, total\_examples=examples, epochs=50)  
executed in 9m 44s, finished 08:25:42 2022-08-16

Out[482]: (464260994, 610253300)

```
In [196]: 1 # Get the keyed vector
 2 wv = w2v_cbow_model.wv
 3 len(wv.vocab)
```

executed in 34ms, finished 13:31:11 2022-08-12

Out[196]: 33298

## Contents ⚙️⚙️

1 Predicting NYC Restaurants  
We have 33,298 trained words.

▼ 2 Introduction

```
In [197]: 1 # Cosine similarity of embedded word vectors
 2.1.1 Imports wv.most_similar(negative=['sick'])
```

▼ 3 Obtain  
executed in 69ms, finished 13:31:11 2022-08-12

▼ 3.1 Obtaining Restaurant Info  
Out[197]: Understanding NYC Restaurants  
('crudités', 0.5211197733879089),  
('offset', 0.5062964558601379),  
('target variable', 0.49563947319984436),  
('engineering target', 0.4898544251918793),  
▼ 3.2 Obtaining Yelp Business Offers  
('Yelp Business Search', 0.48499834537506104),  
('albeit', 0.4783792197704315),  
('Yelp Reviews', 0.4731408953666687),  
('joining', 0.47181084752082825),  
('joining NYC Guidelines', 0.47130322456359863),  
▼ 4 Scrubbing The Data  
('namely', 0.4589795768260956)]

4.1 Drop businesses missing

```
In [198]: 1 wv.most_similar(negative=['poison'])
```

▼ 6 Preprocessing For Further  
executed in 16ms, finished 13:31:11 2022-08-12

6.0.1 Lets find most freq  
Out[198]: Preprocessing For Modeling  
('pore', 0.5644559860229492),  
▼ 8 Models  
('yogurt', 0.5012726783752441),  
('doughier', 0.4855877161026001),  
('waaayyy', 0.48042428493499756),  
8.1.1 Model Evaluations  
('slightly', 0.47897931933403015),  
▼ 8.2 Text Preprocessing Pic  
('crispy', 0.47088611125946045),  
8.2.1 Multinomial Naive  
('crisp', 0.4691073000431061),  
8.2.2 Tuned Multinomial  
('deconstructed', 0.4550844132900238),  
8.2.3 Logistic Regression  
('bucky's', 0.45102351903915405),  
8.2.4 Tuned Logistic Reg  
('lentil', 0.4506188631057739)]

```
In [199]: 1 wv.most_similar(negative=['health'])
```

8.2.7 Support Vector Classification  
executed in 14ms, finished 13:31:11 2022-08-12

8.2.8 Tuned Support Vector Classification  
Out[199]: ('skinny', 0.5029502511024475),  
9.1 Best Model  
('goodworth', 0.4860212206840515),  
9.2 Next Steps  
('petite', 0.4592294991016388),  
▼ 9.3 Appendix  
('plastics', 0.44577306509017944),  
9.3.1 Word2Vec Embedding  
('sweet', 0.44104284048080444),  
9.3.2 Classifying With It  
('towellettes', 0.43190431594848633),  
9.3.3 Pretrained Vector  
('bedrock', 0.42919042706489563),  
9.3.4 Mean Word Embedding  
('tart', 0.4178142547607422),  
9.3.5 Mean Word Embedding  
('goood', 0.4148668050765991),  
('discombobulated', 0.41178783774375916)]

## Word2Vec Skip-Gram

```
In [477]: 1 # Minimally preprocessing for Deep NLP with all lower case
 2 # Keeping stopwords to help provide further word context
 3 lower_data = model_df['Reviews'].map(lambda x: simple_preprocess(x.lower()))
 4 deacc=
```

```
In [483]: 1 # SG (Skip Gram)
```

**Contents** ↗ ↘ w2v\_sg\_model = Word2Vec(lower\_data, size=50, window=3, min\_count=3, sg=1)

1 Predicting NYC Restaurant Violations, finished 08:33:06 2022-08-16

▼ 2 Introduction

In [484]: Objective1 w2v\_sg\_model

2.1.1 Imports

Executed in 16ms, finished 08:33:06 2022-08-16

▼ 3 Obtain

Out[484]: <gensim.models.word2vec.Word2Vec at 0x7fc9bc8f96a0>

3.1 Obtaining Restaurant Data

3.1.1 Understanding NYC Restaurants

In [485]: 1 examples = w2v\_sg\_model.corpus\_count

3.1.2 Target Variable

3.1.3 Engineering Target

Executed in 21ms, finished 08:33:06 2022-08-16

▼ 3.2 Obtaining Yelp Businesses

In [486]: # Training Word2Vec Model

3.2.2 Yelp Reviews

w2v\_sg\_model.train(lower\_data, total\_examples=examples, epochs=50)

3.2.3 Joining Yelp Reviews

Executed in 21ms, finished 08:54:38 2022-08-16

3.3 Joining NYC DOHMH Data

Out[487]: (464235054, 610253300)

▼ 4 Scrubbing The Data

4.1 Drop businesses missing

In [488]: 1 # Get the keyed vector

5 Exploring The Dataset

wv = w2v\_sg\_model.wv

▼ 6 Preprocessing For Further Processing

6.0.1 Lets find most freq

len(wv.vocab)

7 Preprocessing For Models

Executed in 17ms, finished 08:54:39 2022-08-16

▼ 8 Models

Out[488]: 33298

▼ 8.1 Bag-of-Words Model

8.1.1 Model Evaluations

Again, there are 33,298 trained words.

▼ 8.2 Text Preprocessing Pip

8.2.1 Multinomial Naive Bayes

In [489]: Tuned MultinomialSimilar(negative=['sick'])

8.2.3 Logistic Regression

Executed in 219ms, finished 08:54:39 2022-08-16

8.2.4 Tuned Logistic Regression

Decision Tree Model

(aquatic, 0.2592369616031647),

Decision Tree Model

(coffeehouse', 0.24427442252635956),

Support Vector Classification

(becherovka', 0.24270877242088318),

Support Vector Classification

(reduction', 0.24244828522205353),

▼ 9 Conclusions

('beaubourg', 0.24152667820453644),

Best Model Results

('beaubourg', 0.2410590648651123),

Next Steps

('beaubourg', 0.2288820892572403),

Appendix

(Deep NLP, 0.22802232205867767),

Word2Vec Embeddings

('contains', 0.22213214635849)]

Classifying With Embeddings

Pretrained Vector Model

Mean Word Embedding

```
In [490]: 1 wv.most_similar(negative=['poison'])
```

executed in 25ms, finished 08:54:39 2022-08-16

```
Out[490]: [('formally', 0.3024853765964508),
('slightly', 0.2836840748786926),
('zabzi', 0.28303104639053345),
('hilltop', 0.27307450771331787),
('brat', 0.2700038254261017),
```

## Contents ↗

- 1 Predicting NYC Restaurant Deconstructed
- 2 Introduction
  - 2.1 Objective ('tad', 0.25589582324028015),
    - 2.1.1 Imports ('dolce', 0.2506220042705536),
- 3 Obtain
  - 3.1 Obtaining Restaurant Irl
    - 3.1.1 Understanding NYC Restaurants ('published', 0.29327139258384705),
      - 3.1.2 Target Variable ('start', 0.2162439078092575),
      - 3.1.3 Engineering Target ('wrongly', 0.24508269131183624)]

```
In [491]: 1 wv.most_similar(negative=['health'])
```

executed in 28ms, finished 08:54:39 2022-08-16

```
Out[491]: [('yelp', 0.29327139258384705),
('business', 0.25872737169265747),
('yelp_review', 0.24874328076839447),
('joining_japan', 0.24648892879486084),
('joining_nyc', 0.21890901029109955),
('scrubbing_the_data', 0.21743138134479523),
('snickers', 0.21641209721565247),
('drop_businesses_missed', 0.2162439078092575),
('start', 0.2162439078092575),
('exploring_the_dataset', 0.21612727642059326),
('fella', 0.21612727642059326),
('kunafa', 0.20877067744731903)]
```

- 4 Scrubbing The Data
  - 4.1 Drop businesses missed
- 5 Exploring The Dataset
- 6 Preprocessing For Further
  - 6.0.1 Lets find most freq
- 7 Preprocessing For Modelin
- 8 Models
  - 8.1 Bag-of-Words Model

### 9.3.2 Classifying With Embeddings

```
In [494]: 1 X = model_df['Reviews']
```

2 y = model\_df['Severe']

3 y.tail()

4 y.value\_counts()

executed in 33ms, finished 09:09:45 2022-08-16

```
Out[494]: 13056
```

2.5 Decision Tree Mod

2.6 Tuned Decision Tree

2.7 Support Vector Cla

2.8 Tuned SupportVec

- 9 Conclusions Name: Severe, dtype: int64

#### 9.1 Best Model Results

```
In [495]: # Future Test Split before any pre-processing
```

```
9.3 Appendix2 X_train_wv, X_test_wv, y_train_wv, y_test_wv = train_test_split(X,y,random_state=42)
```

9.3.1 Word2VecEmbedding(X\_train\_wv.shape, y\_train\_wv.shape)

9.3.2 Classifying With Embedding

executed in 56ms, finished 09:09:47 2022-08-16

```
Out[495]: ((9795, 3266,))
```

9.3.3 Pretrained Vector

9.3.4 Mean Word Embedding

In [496]: 1 pd.Series(y\_test).value\_counts(normalize=True)

executed in 332ms, finished 09:09:49 2022-08-16

Out[496]: 0 0.69

1 0.31

Name: Severe, dtype: float64

## Contents ⚙

In [497]: 1 # Class imbalance

1 Predicting NYC Restaurant  
2 y\_train.value\_counts(1)

▼ 2 Introduction

▼ 2.1 Objective  
executed in 24ms, finished 09:09:50 2022-08-16

Out[497]: 1 imports 0.71

▼ 3 Obtain  
1 0.29

▼ 3.1 Obtaining Restaurant  
Name: Severe, dtype: float64

3.1.1 Understanding NYC

In [314]: Target Variable  
# Calculating class weights

3.1.3 Engineering Target

▼ 3.2 Obtaining Yelp Businesses  
3.2.1 weights = compute\_class\_weight('balanced', np.unique(y\_train), y\_train)

3.2.1.1 weights\_dict = dict(zip(np.unique(y\_train), weights))

3.2.2 Yelp Reviews

3.2.3 Joining Yelp Reviews  
executed in 18:39:52 2022-08-13

3.3 Joining NYC DOHMH

Out[234]: {0: 0.7096797565570208, 1: 1.692294402211472}

▼ 4 Scrubbing The Data

4.1 Drop businesses missing

In [235]: 1 # OneHotEncoding target variable for NN

5 Exploring The Dataset

▼ 6 Preprocessing For Further

6.0.1 Lets find most freq  
3 y\_train\_seq = to\_categorical(y\_train)

7 Preprocessing For Modelin

▼ 8 Models  
executed in 28ms, finished 18:39:52 2022-08-13

Out[235]: (97.85 Model)

8.1 Model Evaluations

In [8.2.9]: 1 # Tokenizing text with Keras tokenizer

8.2.1 Multinomial Naive  
max\_words = 30000

8.2.2 Tuned MultiTokenizer = text.Tokenizer(num\_words=max\_words)

8.2.3 Logistic Regressor.fit\_on\_texts(X\_train)

8.2.4 Tuned Logistic Regressor.train\_sequences = tokenizer.texts\_to\_sequences(X\_train)

8.2.5 Decision Tree Model  
test\_sequences = tokenizer.texts\_to\_sequences(X\_test)

8.2.6 Tuned DecisionTree  
executed in 09:17:06 2022-08-16

8.2.7 Support Vector Clas

In [8.2.18]: Tuned Support Vector Classifier  
# Tuning maximum sequence length for optimizaition

▼ 9 Conclusions  
2 seq\_lengths = list(map(lambda x: len(x), [\*train\_sequences, \*test\_sequences]))

9.1 Best Model Results

9.2 Next Steps and Future  
executed in 20ms, finished 09:18:14 2022-08-16

Out[301]: 10700

9.3.1 Word2Vec Embedds

9.3.2 Classifying With Er

In [505]: 1 # Finding average sequence length for optimizaition

2 import statistics

9.3.4 MeanWordEmbed  
3 statistics.mean(seq\_lengths)

executed in 59ms, finished 09:22:17 2022-08-16

Out[505]: 991.4490467804916

In [506]: 1 max\_seq\_length = 2000

executed in 5ms, finished 09:23:14 2022-08-16

In [507]: 1 # Padding sequences because NN require same sizes

```
2 X_train_seq = keras.preprocessing.sequence.pad_sequences(train_sequence
 maxlen=max_sequence_length)
3 X_test_seq = keras.preprocessing.sequence.pad_sequences(test_sequences,
 maxlen=max_sequence_length)
```

## Contents ↗

1 Predicting NYC Restaurant

▼ 2 Introduction

executed in 2.88s, finished 09:23:28 2022-08-16

  ▼ 2.1 Objective

In [508]: 1 len(tokenizer.index\_word)

▼ 3 Obtain

executed in 24ms, finished 09:23:50 2022-08-16

  ▼ 3.1 Obtaining Restaurant Data

Out[508]: 1 75799 Understanding NYC

  3.1.2 Target Variable -- N

In [509]: 1 # Adapted from https://nbviewer.org/github/learn-co-curriculum/aug-ds-f

  ▼ 3.2 Obtaining Yelp Business Data

```
2 def get_earlystop(monitor='val_accuracy', patience=3, restore_best_weights=False):
3 """
```

  3.2.1 Yelp Business Seal

  3.2.2 Yelp Reviews args = locals()

  3.2.3 Joining Yelp Reviews

```
5 return EarlyStopping(**args)
```

  3.3 Joining NYC DOHMH Data

  ▼ 4 Scrubbing The Data

```
7 get_earlystop.__doc__ += EarlyStopping.__doc__
```

  4.1 Drop businesses missing

5 Exploring The Data

  ▼ 6 Preprocessing For Further Processing

Out[509]: 1 Function \_\_main\_\_.get\_earlystop(monitor='val\_accuracy', patience=3, rest

  6.0.1 Lets find most frequent words

  6.0.2 One best weights=False)

7 Preprocessing For Modelin

▼ 8 Models

In [243]: 1 def make\_model(EMBEDDING\_SIZE = 128):

  ▼ 8.1 Bag-of-Words Model

    model=Sequential()

      8.1.1 Model Evaluations

  ▼ 8.2 Text Preprocessing Pipeline

    model.add(Embedding(max\_words, EMBEDDING\_SIZE))

      8.2.1 Multinomial Naive Bayes

        model.add(LSTM(50,return\_sequences=False))

      8.2.2 Tuned Multinomial Bayes

        model.add(Dropout(0.5))

      8.2.3 Logistic Regression

        model.add(Dense(25, activation='relu'))

      8.2.4 Tuned Logistic Regression

        model.add(Dropout(0.5))

      8.2.5 Decision Tree Model

        model.add(Dense(2, activation='softmax'))

      8.2.6 Tuned DecisionTree Model

        model.compile(loss='categorical\_crossentropy',

          8.2.7 Support Vector Classifier

            optimizer='adam',

          8.2.8 Tuned SupportVector

            metrics=['accuracy'])

▼ 9 Conclusions

14 # display(model.summary())

9.1 Best Model Results

15 return model

9.2 Next Steps and Future

executed in 24ms, finished 18:40:28 2022-08-13

▼ 9.3 Appendix - Deep NLP

  9.3.1 Word2Vec Embedds

  9.3.2 Classifying With Embedding

  9.3.3 Pretrained Vector Embedding

  9.3.4 Mean Word Embedding

```
In [245]: 1 model = make_model()
2 history = model.fit(X_train_seq, y_train_seq, epochs=5,
3 batch_size=32, validation_split=0.2, callbacks=callbacks,
4 class_weight=weights_dict)
```

executed in 5h 44m 56s, finished 00:25:42 2022-08-14

Epoch 1/5

**Contents** ↗ 245/245 [=====] - 4003s 16s/step - loss: 0.6945

- 1 Predicting NYC Restaurant Accuracy: 0.4777 - val\_loss: 0.6907 - val\_accuracy: 0.5666
- ▼ 2 Introduction Epoch 2/5
  - ▼ 2.1 Objective 245/245 [=====] - 4307s 18s/step - loss: 0.6707
    - 2.1.1 Imports accuracy: 0.5972 - val\_loss: 0.7343 - val\_accuracy: 0.4844
- ▼ 3 Obtain Epoch 3/5
  - ▼ 3.1 Obtaining Restaurant Information Accuracy: 0.7389 - val\_loss: 0.7619 - val\_accuracy: 0.5426
    - 3.1.1 Understanding NYC Epoch 4/5
    - 3.1.2 Target Variable Accuracy: 0.8647 - val\_loss: 0.8504 - val\_accuracy: 0.6064
- ▼ 3.2 Obtaining Yelp Business Data
  - 3.2.1 Yelp Business Search Accuracy: 0.9343 - val\_loss: 1.1274 - val\_accuracy: 0.6110
  - 3.2.2 Yelp Reviews Accuracy: 0.9343 - val\_loss: 1.1274 - val\_accuracy: 0.6110
  - 3.2.3 Joining Yelp Reviews
- 3.3 Joining NYC DOHMH Data

**In [251]:** 1 model.summary()

4.1 Drop business data

executed in 126ms, finished 15:55:08 2022-08-14

5 Exploring The Dataset Model: "sequential\_1"

▼ 6 Preprocessing For Further

|                           | 6.0.1 Lets find most frequent words (type) | Output Shape      | Param # |
|---------------------------|--------------------------------------------|-------------------|---------|
| 7                         | Preprocessing For Model                    | (None, None, 128) | 6400000 |
| ▼ 8 Models                | embedding_1 (Embedding)                    | (None, None, 128) | 6400000 |
| ▼ 8.1                     | Bag-of-Words Model                         | (None, 50)        | 35800   |
| 8.1.1                     | Model Evaluations                          | (None, 50)        | 0       |
| ▼ 8.2                     | Text Preprocessing Pipeline                | (None, 25)        | 1275    |
| 8.2.1                     | Multinomial Naïve Bayes                    | (None, 25)        | 0       |
| 8.2.2                     | Tuned Multinomial Naïve Bayes              | (None, 25)        | 0       |
| 8.2.3                     | Logistic Regression                        | (None, 25)        | 0       |
| 8.2.4                     | Tuned Logistic Regression                  | (None, 25)        | 0       |
| 8.2.5                     | Decision Tree Model                        | (None, 25)        | 0       |
| 8.2.6                     | Tuned Decision Tree Model                  | (None, 2)         | 52      |
| 8.2.7                     | Support Vector Classification              | (None, 2)         | 52      |
| 8.2.8                     | Tuned Support Vector Classification        | (None, 2)         | 52      |
| ▼ 9 Conclusions           | Total params: 6,437,127                    |                   |         |
| 9.1 Best Model Results    | Trainable params: 6,437,127                |                   |         |
| 9.2 Next Steps and Future | Non-trainable params: 0                    |                   |         |

▼ 9.3 Appendix - Deep NLP

9.3.1 Word2Vec Embeddings

**In [252]:** 1 # Collapsing y\_test data down to single column

2 y\_test\_seq.argmax(axis=1)

3 Pretrained Vector

4 Measured Model Errors

executed in 104ms, finished 09:42:51 2022-08-16

**In [252]:** 1 # model.save('First\_LSTM\_Model.h5')

executed in 631ms, finished 16:03:15 2022-08-14

```
In [519]: 1 new_model = load_model('First_LSTM_Model.h5')
```

executed in 2.03s, finished 09:52:03 2022-08-16

```
In [520]: 1 new_model
```

executed in 11ms, finished 09:52:10 2022-08-16

## Contents ↗

**Out[520]:** <keras.engine.sequential.Sequential at 0x7fc8c31af340>

1 Predicting NYC Restaurant

▼ 2 Introduction

In [524]: 1 # Collapsing down to which class is predicted

  2.1 Objective

    2 y\_hat\_test = new\_model.predict(X\_test\_seq).argmax(axis=1)

      2.1.1 Imports

        3 y\_hat\_test[:5]

▼ 3 Obtain

  executed in 3m 33s, finished 09:59:48 2022-08-16

  ▼ 3.1 Obtaining Restaurant If

    3.1.1 Under 1000 Violations NYC [=====] - 212s 2s/step

    3.1.2 Target Variable -- N

Out[524]: array([0, 0, 0, 1, 1])

    3.1.3 Engineering Target

  ▼ 3.2 Obtaining Yelp Businesses

In [349]: 1 # Tokenizing text with Keras tokenizer

  2 # max\_words2 = 25000

  3 # tokenizer2 = text.Tokenizer(num\_words=max\_words)

  3.3 Joining Yelp Reviews

  executed in 10m 10s, finished 15:12:22 2022-08-14

▼ 4 Scrubbing The Data

In [425]: 1 # def make\_model2(EMBEDDING\_SIZE = 128):

  2 # model=Sequential()

  ▼ 6 Preprocessing For Further

    6.0.1 Lets find most freq

    5 # model.add(Embedding(max\_words, EMBEDDING\_SIZE))

  7 Preprocessing For Modeling

    6 # model.add(LSTM(50,return\_sequences=False))

  ▼ 8 Models

    7 # model.add(GlobalMaxPool1D())

  ▼ 8.1 Bag-of-Words Model

    8 # model.add(Dense(25, activation='relu'))

      8.1.1 Model Evaluations

      # model.add(Dropout(0.5))

  ▼ 8.2 Text Preprocessing Pipeline

    9 # model.add(Dense(2, activation='softmax'))

      8.2.1 Multinomial Naive I

      8.2.2 Tuned Multinomial

      8.2.3 Logistic Regression

      8.2.4 Tuned Logistic Reg

      10 # model.compile(loss='categorical\_crossentropy',

          optimizer='adam',

          metrics=['accuracy'])

      11 # model.summary()

      12 # display(model.summary())

      13 # return model

      8.2.6 Tuned DecisionTree

      8.2.7 SupportVectorM

      8.2.8 Tuned SupportVec

▼ 9 Conclusions

  9.1 Best Model Results

  9.2 Next Steps and Future

  ▼ 9.3 Appendix - Deep NLP

    9.3.1 Word2Vec Embedds

    9.3.2 Classifying With Er

    9.3.3 Pretrained Vector N

    9.3.4 Mean Word Embed

```
In [245]: 1 # model2 = make_model2()
2 # history = model.fit(X_train_seq, y_train_seq, epochs=10,
3 # batch_size=32, validation_split=0.2, callbacks=get
4 # class_weight=weights_dict)
```

executed in 5h 44m 56s, finished 00:25:42 2022-08-14

Epoch 1/5

**Contents** ↗ 245/245 [=====] - 4003s 16s/step - loss: 0.6945

1 Predicting NYC Restaurant Accuracy: 0.4777 - val\_loss: 0.6907 - val\_accuracy: 0.5666

▼ 2 Introduction Epoch 2/5

▼ 2.1 Objective 245/245 [=====] - 4307s 18s/step - loss: 0.6707

2.1.1 Imports accuracy: 0.5972 - val\_loss: 0.7343 - val\_accuracy: 0.4844

▼ 3 Obtain Epoch 3/5

▼ 3.1 Obtaining Restaurant Accuracy: 0.7389 - val\_loss: 0.7619 - val\_accuracy: 0.5426

3.1.1 Understanding NYC Epoch 4/5

3.1.2 Target Variable Accuracy: 0.8647 - val\_loss: 0.8504 - val\_accuracy: 0.6064

▼ 3.2 Obtaining Yelp Business

3.2.1 Yelp Business Search Accuracy: 0.9343 - val\_loss: 1.1274 - val\_accuracy: 0.6110

3.2.2 Yelp Reviews Accuracy: 0.9343 - val\_loss: 1.1274 - val\_accuracy: 0.6110

3.2.3 Joining Yelp Reviews

3.3 Joining NYC DOHMH Data

▼ 4 Scrubbing The Data

61% Validation accuracy score is not much better performance than the Bag-of-N-gram models we tried first

5 Exploring The Dataset

▼ 6 Preprocessing For Further

In [601]: 1 from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

7 Preprocessing For Modeling

executed in 42ms, finished 10:01:19 2022-08-16

▼ 8 Models

▼ 8.1 Bag-of-Words Model

In [533]: 1 cm = confusion\_matrix(y\_test\_seq.argmax(axis=1), y\_hat\_test)

8.1.1 Model Evaluations

▼ 8.2 Text Preprocessing

executed in 14ms, finished 10:02:54 2022-08-16

8.2.1 Multinomial Naive Bayes

8.2.2 Tuned Multinomial Naive Bayes

8.2.3 Logistic Regression

8.2.4 Tuned Logistic Regression

8.2.5 Decision Tree Model

8.2.6 Tuned DecisionTree Model

8.2.7 Support Vector Classification

8.2.8 Tuned SupportVector Classification

▼ 9 Conclusions

9.1 Best Model Results

9.2 Next Steps and Future

▼ 9.3 Appendix - Deep NLP

9.3.1 Word2Vec Embedding

9.3.2 Classifying With Embedding

9.3.3 Pretrained Vector Embedding

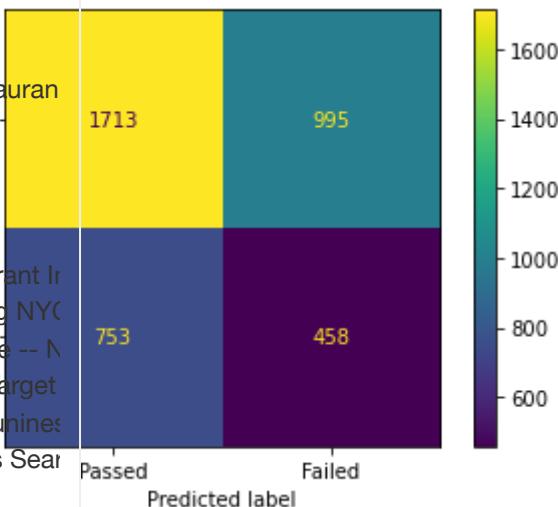
9.3.4 Mean Word Embedding

```
In [540]: 1 disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=['Pass'
2 disp.plot()
3 plt.show()
4 print(classification_report(y_test_seq.argmax(axis=1),y_hat_test))
```

executed in 733ms, finished 10:08:00 2022-08-16

## Contents ⚙️

- 1 Predicting NYC Restaurant
- ▼ 2 Introduction Passed
  - ✓ 2.1 Objective
  - 2.1.1 Imports
- ▼ 3 Obtain
  - ✓ 3.1 Obtaining Restaurant Info
    - 3.1.1 Understanding NYC
    - 3.1.2 Target Variable -- NYC
    - 3.1.3 Engineering Target
  - ✓ 3.2 Obtaining Yelp Business Data
    - 3.2.1 Yelp Business Search
    - 3.2.2 Yelp Reviews
    - 3.2.3 Joining Yelp Reviews
    - 3.3 Joining NYC DOHMH Data
- ▼ 4 Scrubbing The Data
  - 4.1 Drop businesses missing
  - 4.2 Exploring The Dataset
- ▼ 5 Exploring The Dataset
- ▼ 6 Preprocessing For Further Accuracy
  - 6.0.1 Lets find most frequent business category
  - 6.0.2 Weighted average
- ▼ 7 Preprocessing For Modeling
- ▼ 8 Models



|   | precision | recall | f1-score | support |
|---|-----------|--------|----------|---------|
| 0 | 0.69      | 0.63   | 0.66     | 2708    |
| 1 | 0.32      | 0.38   | 0.34     | 1211    |
|   |           |        | 0.55     | 3919    |
|   | 0.50      | 0.51   | 0.50     | 3919    |
|   | 0.58      | 0.55   | 0.56     | 3919    |

## 9.3.3 Pretrained Vector Models

### Our Embeddings

```
In [542]: Tuned Support Vector Model keras_embedding()
```

executed in 116ms, finished 10:11:24 2022-08-16

#### 9.1 Best Model Results

```
Out[542]: <keras.layers.core.embedding.Embedding at 0x7fc8640ecf40>
```

#### 9.2 Next Steps and Future

- ▼ 9.3 Appendix - Deep NLP
  - 9.3.1 Word2Vec Embeddings
  - 9.3.2 Classifying With Embeddings
  - 9.3.3 Pretrained Vector Models
  - 9.3.4 Mean Word Embeddings

```
In [545]: 1 def make_model_wv(wv):
2 model=Sequential()
3
4 model.add(wv.get_keras_embedding())
5 model.add(LSTM(50,return_sequences=False))
6 model.add(Dense(25, activation='relu'))
7 model.add(Dropout(0.5))
8 model.add(Dense(2, activation='softmax'))
9
10 model.compile(loss='categorical_crossentropy',
11 optimizer='adam',
12 metrics=['accuracy'])
13
14 return model
```

## Contents ⚙️

- 1 Predicting NYC Restaurant
- ▼ 2 Introduction10
  - ▼ 2.1 Objective11
    - 2.1.1 Imports12
  - ▼ 3 Obtain13
    - 3.1 Obtaining Data1214, finished 10:15:39 2022-08-16
      - 3.1.1 Understanding NYC
      - 3.1.2 Target Variable -- N
    - In 3.1.3 Engineering Target
    - ▼ 3.2 Obtaining Yelp Businesses14

```
In [341]: 1 model3 = make_model_wv(wv)
2
3.2.2 Yelp Reviews history = model3.fit(X_train_seq, y_train_seq, epochs=5,
4
5 batch_size=32, validation_split=0.2,
6 callbacks=get_earlystop(), class_weight=weights_dict)
```

- ▼ 4 Scrubbing The Data15, finished 1h 50m 41s, finished 12:06:22 2022-08-16
  - 4.1 Drop businesses missing
  - Epoch 1/5
  - 5 Exploring The Dataset229/229 [=====] - 1215s 5s/step - loss: 0.6946 -
  - ▼ 6 Preprocessing For Further accuracy: 0.5105 - val\_loss: 0.6881 - val\_accuracy: 0.6320
    - 6.0.1 Lets find most freq
    - Epoch 2/5
    - 7 Preprocessing For Model229/229 [=====] - 1300s 6s/step - loss: 0.6938 -
    - ▼ 8 Models accuracy: 0.5143 - val\_loss: 0.6983 - val\_accuracy: 0.3461
      - ▼ 8.1 Bag-of-Words Model229/229 [=====] - 1534s 7s/step - loss: 0.6924 -
      - accuracy: 0.4925 - val\_loss: 0.6835 - val\_accuracy: 0.6402
      - ▼ 8.2 Text Preprocessing: Pip229/229 [=====] - 1327s 6s/step - loss: 0.6907 -
      - accuracy: 0.5263 - val\_loss: 0.6785 - val\_accuracy: 0.6337
      - 8.2.3 Logistic Regression229/229 [=====] - 1264s 6s/step - loss: 0.6881 -
      - accuracy: 0.5479 - val\_loss: 0.6792 - val\_accuracy: 0.5992
      - 8.2.6 Tuned DecisionTree
      - 8.2.7 Support Vector Clas229/229 [=====] - 123s 2s/step
      - 8.2.8 Tuned SupportVec

```
In [551]: 1 y_hat_test = model.predict(X_test_seq).argmax(axis=1)
```

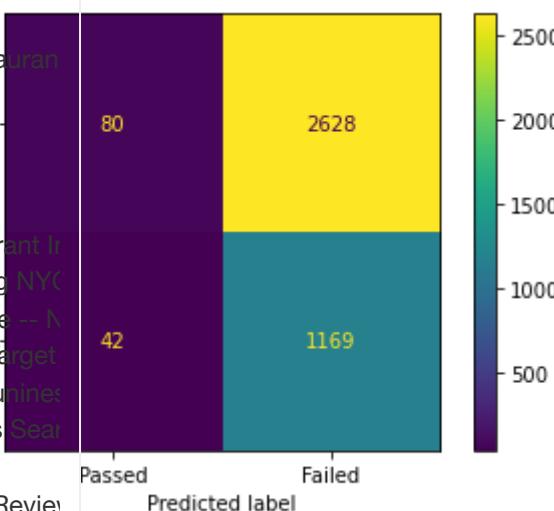
- ▼ 9 Conclusionsexecuted in 3m 48s, finished 12:15:37 2022-08-16
  - 9.1 Best Model Results123/123 [=====] - 224s 2s/step
  - 9.2 Next Steps and Future
  - ▼ 9.3 Appendix - Deep NLP
    - 9.3.1 Word2Vec Embedding
    - 9.3.2 Classifying With Embedding
    - 9.3.3 Pretrained Vector Embedding
    - 9.3.4 Mean Word Embedding

```
In [556]: 1 cm = confusion_matrix(y_test_seq.argmax(axis=1),y_hat_test)
2 disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Passed', 'Failed'])
3 disp.plot()
4 plt.show()
5 print(classification_report(y_test_seq.argmax(axis=1),y_hat_test))
```

executed in 432ms, finished 12:21:16 2022-08-16

## Contents ⚙️

- 1 Predicting NYC Restaurant Inspections
- ▼ 2 Introduction
  - ▼ 2.1 Objective Passed - Failed
  - 2.1.1 Imports
- ▼ 3 Obtain
  - ▼ 3.1 Obtaining Restaurant Inspection Data
    - 3.1.1 Understanding NYC DOHMH Data
    - 3.1.2 Target Variable -- Passed / Failed
    - 3.1.3 Engineering Target Variable
  - ▼ 3.2 Obtaining Yelp Business Reviews
    - 3.2.1 Yelp Business Search
    - 3.2.2 Yelp Reviews
    - 3.2.3 Joining Yelp Reviews
  - 3.3 Joining NYC DOHMH & Yelp Data
- ▼ 4 Scrubbing The Data
  - 4.1 Drop businesses missing reviews
- 5 Exploring The Dataset
- ▼ 6 Preprocessing For Further Modeling
  - 6.0.1 Lets find most frequent words
- 7 Preprocessing For Modeling
- ▼ 8 Models
  - ▼ 8.1 Bag-of-Words Model
    - 8.1.1 Model Evaluations
  - ▼ 8.2 Text Preprocessing Pipelines
    - 8.2.1 Multinomial Naive Bayes
    - 8.2.2 Tuned Multinomial Naive Bayes
    - 8.2.3 Logistic Regression
    - 8.2.4 Tuned Logistic Regression
    - 8.2.5 Decision Tree Model
    - 8.2.6 Tuned Decision Tree Model
    - 8.2.7 Support Vector Machine
    - 8.2.8 Tuned SupportVector Machine
- ▼ 9 Conclusions
  - 9.1 Best Model Results
  - 9.2 Next Steps and Future Work



|                                   | precision | recall | f1-score | support |
|-----------------------------------|-----------|--------|----------|---------|
| 8.2.1 Multinomial Naive Bayes     | 0.66      | 0.03   | 0.06     | 2708    |
| 8.2.3 Logistic Regression         | 0.31      | 0.97   | 0.47     | 1211    |
| 8.2.4 Tuned Logistic Regression   |           |        |          |         |
| 8.2.5 Decision Tree Model         |           |        | 0.32     | 3919    |
| 8.2.6 Tuned Decision Tree Model   | 0.48      | 0.50   | 0.26     | 3919    |
| 8.2.7 Support Vector Machine      | 0.55      | 0.32   | 0.18     | 3919    |
| 8.2.8 Tuned SupportVector Machine |           |        |          |         |

## Pretrained Embeddings With GLOVE

```
In [947]: Word2VecEmbeds = model_df['Severe']
```

9.3.2 Classifying With Embedding  
executed in 193ms, finished 12:06:22 2022-08-16

```
In [948]: MeanWordEmbeds = model_df['Reviews'].map(word_tokenize).values
```

executed in 3m 8s, finished 12:09:30 2022-08-16

In [549]: 1 total\_vocabulary = set(word for review in data for word in review)

executed in 2.19s, finished 12:09:32 2022-08-16

In [550]: 1 len(total\_vocabulary)  
2 print('There are {} unique tokens in the dataset.'.format(len(total\_vocabulary)))

executed in 36ms, finished 12:09:32 2022-08-16

## Contents

There are 180157 unique tokens in the dataset.

1 Predicting NYC Restaurant

▼ 2 Introduction

In [243]: 1 glove = {}  
2 with open('glove.6B.50d.txt', 'rb') as f:  
3 for line in f:  
4 parts = line.split()  
5 word = parts[0].decode('utf-8')  
6 if word in total\_vocabulary:  
7 vector = np.array(parts[1:], dtype=np.float32)  
8 glove[word] = vector

▼ 3 Obtain

3.1 Obtaining Restaurant Info

3.1.1 Understanding NYC

3.1.2 Target Variable -- NYC

3.1.3 Engineering Target

▼ 3.2 Obtaining Yelp Business

3.2.1 Yelp Business Search

executed in 2.43s, finished 14:50:31 2022-08-12

In [244]: 1 # testing vecotrizer  
2 3.2.2 Yelp Reviews  
3 3.2.3 Joining Yelp Reviews

2 glove['food']

3.3 Joining NYC DOHMH

▼ 4 Scrubbing The Data

Out[244]: array([0.7222, -0.44545, -0.51833, -0.26818, 0.44427, 0.25108, -0.99282, -0.90198, 1.8729, 0.039081, 0.14284, 0.074878, 1.0543, -0.3203, 1.0722, 0.44323, 0.0099484, 0.15754, 0.51399, -0.77668, 0.924, 0.010958, 0.58815, 0.23078, -0.34281, -0.88444, -0.31492, 0.12661, 1.1445, 0.60775, 3.4344, 0.63561, -0.13832, 0.28045, -0.16181, 0.77541, -0.49888, 0.4602, 0.91799, 0.29007, 0.06884, 0.59978, 0.53967, -0.061752, 1.2975, 0.2323, -0.80945, 0.34932, 0.33934, 0.25499], dtype=float32)

8.2.1 Multinomial Naive Bayes

8.2.2 Tuned Multinomial

8.2.3 Logistic Regression

8.2.4 Tuned Logistic Reg

8.2.5 Decision Tree Model

8.2.6 Tuned DecisionTree

8.2.7 Support Vector Class

8.2.8 Tuned SupportVec

▼ 9 Conclusions

9.1 Best Model Results

9.2 Next Steps and Future

▼ 9.3 Appendix - Deep NLP

9.3.1 Word2Vec Embedding

9.3.2 Classifying With Embed

9.3.3 Pretrained Vector Embed

9.3.4 Mean Word Embedding

## 9.3.4 Mean Word Embeddings

```
In [245]: 1 # # Adapted from https://github.com/learn-co-curriculum/dsc-classificat
 2 # class W2vVectorizer(object):
 3 #
 4 # def __init__(self, w2v):
 5 # #
 6 # self.w2v = w2v
 7 # if len(w2v) == 0:
 8 # self.dimensions = 0
 9 # else:
 10 # self.dimensions = len(w2v[next(iter(glove))])
 11 #
 12 # def fit(self, X, y):
 13 # return self
 14 #
 15 # def transform(self, X):
 16 # return np.array([
 17 # np.mean([self.w2v[w] for w in words if w in self.w2v]
 18 # or [np.zeros(self.dimensions)], axis=0) for words
 19 #])
 20
```

## Contents ⚙️

- 1 Predicting NYC Restaurant Health Violations
- ▼ 2 Introduction
  - 10 #
  - ▼ 2.1 Objective
    - 11 #
    - 2.1.1 Imports
      - 12 #
      - def fit(self, X, y):
      - return self
  - ▼ 3 Obtain
    - 13 #
    - 14 #
    - ▼ 3.1 Obtaining Restaurant Info
      - 15 #
      - 3.1.1 Understanding NYC DOHMH Data
        - 16 #
        - 3.1.2 Target Variable -- Number of Violations
          - 17 #
        - 3.1.3 Engineering Target Variable
          - 18 #
      - ▼ 3.2 Obtaining Yelp Business Data
        - executed in 8ms, finished 14:50:35 2022-08-12
        - 3.2.1 Yelp Business Seal
        - 3.2.2 Yelp Reviews
        - 3.2.3 Joining Yelp Reviews with NYC DOHMH Data
          - Passing the mean vectorizer class created above in the first step of pipeline. To be followed it up with the model classifying the data fed in.
    - ▼ 4 Scrubbing The Data
      - 4.1 Drop businesses missing

```
In [246]: 1 # Creating pipeline objects that make use of the mean embedding vectorizer
 2 Pipeline([('Word2Vec Vectorizer', W2vVectorizer(glove)),
 3 ('DecisionTree', DescisionTreeClassifier() verbose=True))
 4 svc = Pipeline([('Word2Vec Vectorizer', W2vVectorizer(glove)),
 5 ('Support Vector Machine', SVC())])
 6 lr = Pipeline([('Word2Vec Vectorizer', W2vVectorizer(glove)),
 7 ('Logistic Regression', LogisticRegression())])
 8
```

```
In [247]: 1 # Saving a list for each pipeline name and object
 2 models = [('Random Forest', rf),
 3 ('Support Vector Machine', svc),
 4 ('Logistic Regression', lr)]
```

- ▼ 9 Conclusions
  - 9.1 Best Model Results
  - 9.2 Next Steps and Future
  - ▼ 9.3 Appendix - Deep NLP
    - 9.3.1 Word2Vec Embeddings
    - 9.3.2 Classifying With Embeddings
    - 9.3.3 Pretrained Vector Embeddings
    - 9.3.4 Mean Word Embeddings

In [248]:

```

1 # Storing cross validation scores with sklearn's cross_val_score() func
2 cv_scores = [(name, cross_val_score(model, data, target, cv=2).mean())
3 cv_scores

```

executed in 1m 39.0s, finished 14:52:19 2022-08-12

[Parallel(n\_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

## Contents

- 1 Predicting NYC Restaurant Inspections
- 2 Introduction to NYC Health Data
- 3 Obtain Data
  - 3.1 Obtaining Restaurant Data
  - 3.2 Obtaining Yelp Business Reviews
  - 3.3 Joining NYC DOHMH Data
- 4 Scrubbing The Data
- 5 Exploring The Dataset
- 6 Preprocessing For Further Analysis
  - 6.0.1 Lets find most frequent words
- 7 Preprocessing For Modeling
- 8 Models
  - 8.1 Bag-of-Words Model
    - 8.1.1 Model Evaluations
  - 8.2 Text Preprocessing Pipelines
    - 8.2.1 Multinomial Naive Bayes
    - 8.2.2 Tuned Multinomial Naive Bayes
    - 8.2.3 Logistic Regression
    - 8.2.4 Tuned Logistic Regression
    - 8.2.5 Decision Tree Model
    - 8.2.6 Tuned DecisionTree Model
    - 8.2.7 Support Vector Classification
    - 8.2.8 Tuned SupportVectorClassification
- 9 Conclusions
  - 9.1 Best Model Results
  - 9.2 Next Steps and Future Work
  - 9.3 Appendix - Deep NLP
    - 9.3.1 Word2Vec Embeddings
    - 9.3.2 Classifying With Embeddings
    - 9.3.3 Pretrained Vector Embeddings
    - 9.3.4 Mean Word Embeddings