Section 1: Week 2: Evaluate Data Mining Techniques

Nate Bachmeier

TIM-8130: Data Mining

February 15th, 2020

North Central University

Evaluate Data Mining Techniques

# Section I: Using Data Mining

OpenNYC offers thousands of data sets containing public information for businesses and researchers to explore. The Department of Health and Mental Hygiene publishes a daily feed of NYC Restaurant Inspection Results that enumerates all citations since July 2015 (DOHMH, 2020). Each establishment receives a grade of A, B, C, P(ending), N(ot available), or Z(oning change) based on the severity of citations. According to NYC Health (2020), "a business that does not maintain a grade of A or B becomes closed until passing a future inspection."

Figure 1 displays the distribution of citations across the different boroughs for January 2019 through Feb 2020. Approximately 87.4% of establishments are in good standing versus 5% receiving a grade of C. There are several outliers, such as Dunkan Donuts consistently receives the most infractions, followed by McDonald's and Star Bucks, in part due to a large number of locations. Specific ethnic categories, such as Chinese in Queens and Indian in the Bronx, have high infractions due to their concentration. Niche categories, such as Cajun-Creole, have an alarming rate of 22% non-compliance due to only 58 businesses in this category.

Figure 1 Citations by Borough

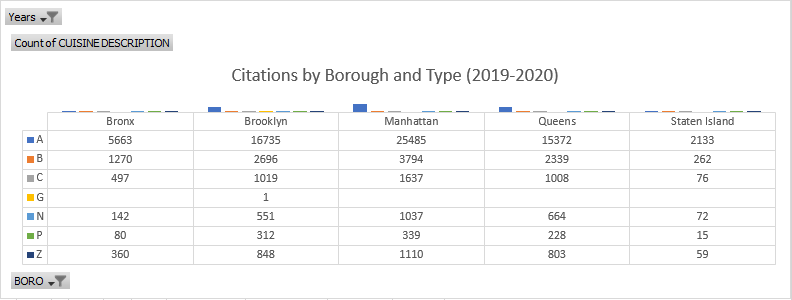


Table : Machine Learning Algorithms Used

|  |  |  |
| --- | --- | --- |
| Algorithm | Strategy Description | Comments |
| Logistic Regression | Maps numeric features to a numeric range | The label-centric dataset was not well suited |
| Decision Tree Regression | Maps vectorized labels to a numeric range | Many predictions in the general ballpark |
| Principal Component Analysis | Uses the correlation between features to combine related features | Improves runtime and accuracy of sparse data |
| Decision Tree Classifier | Maps vectorized labels to finite categories (e.g., value-ranges) | More accurate than DTR as approximate predictions are sufficient |
| Multilayer Perceptron Regressor | Uses backpropagation to build a fitting expression | Comparable to DTR performance |

The feed also details the violation code for the citation along with a numeric score with a mean value of 20.4 and a standard deviation of 14.8, where higher scores represent egregious infractions. A goal of this research is to predict the score that an organization would receive based on the violation code and related features. First, the features set 'Cuisine Description, Critical Flag, Violation Code, Inspection Type' were One Hot Encoded into a 586 by 200730 matrix and fed into Scikit Learn's Logistic Regression algorithm using a 50% split for testing/training data. This initial experiment's accuracy was only 10.8%. Next, the features' Borough, Critical Flag, Grade, Violation Code, and Inspection Type' were One Hot Encoded into a 75 x 200730 matrix as input into the Decision Tree Regressor algorithm (67.8% accuracy). The Multi-Layer Perceptron (MLPRegressor), a neural network-based algorithm, was measured to have a 64.8% accuracy using the same feature set. Next, a Principal Components Analysis (PCA) was able to reduce the features from 75 to 25 while maintaining 95% of the variance. Evaluation of both MLPR and DTR's performance shows no change in accuracy though it did speed up the training a noticeable amount.

Witten (2011) proposes a strategy called binning, which converts a regression problem into a classification scenario by making each bin represent a value range. The score distribution (see Figure 2) suggests that value range partitions between -5 to 15, 16 to 25, 26 to 50, and 50 to 200—but the official documentation suggests that steps of 10 between 0 to 100 *should* be more accurate. The disconnect likely comes from erroneous values during the manual data entry. The Decision Tree Classifier was able to map the features to the correct bucket with a 93.8% accuracy. Additional analysis found that the Grade feature has the most substantial influence (see Figure 3), and its removal decreases the Decision Tree Classifier's accuracy to only 51.7%. Removing both the Grade and PCA preprocessing increases the One Hot Encoded matrix from 10 to 68 features with a total accuracy of only 0.03%. This experiment suggests that there is insufficient data for this strategy, and the boost from using PCA correlations is critical for the sparse data set. The source code for reproducing these measurements is available in the Appendix section.

|  |  |
| --- | --- |
| Figure : Score Distribution | Figure Average Score by Grade |
|  |  |

# Section II: Analyze the Process

## How can businesses use data mining with big data

Before an organization can operationalize predictive capabilities, they need first to identify the specific questions and which facts can support their answers. If the company starts with a sea of data, then how will they know these data points are relevant or fully encompass the problem? For instance, to provide restaurant recommendations requires evidence across different perspectives from the DOHMH to microbloggers (e.g., Twitter and Instagram). A company that only relies on a single source is bound to encounter distortion and bias across its results. While the use of heterogeneous data sources can improve data mining, it also creates specific challenges, such as (1) mapping identifiers between systems, (2) handling inconsistent data schemas, (3) varying data publication cadences, and (4) provider-specific protocols to subscribe for updates. There are many highly reliable data ingestion tools, like Apache Kafka, NiFi, and Flume, to handle these scenarios (Matacuta & Popa, 2018). However, the business needs to consider beyond the data landing in the data store and how best to approach their curation strategy. Perhaps the business also needs Yelp customer reviews and has deployed Flume to load records into their data lake solution (e.g., Apache Hadoop or Amazon S3). Until feature extraction, record transformation, and deduplication processes have normalized these raw unstructured results, it can be challenging to analyze these results. This initial curation process also provides an opportunity to reduce the volume of records through aggregation (e.g., 1000 records a button click happened, versus a single summation record for the period).

After cleaning and curating the ingestion feed, these results need to flow into exploratory programs such as Microsoft Excel, OLAP data stores, Kibana, Jupyter Notebooks, and R Studio. These technologies allow the analysis to review the shape and attribute ranges while minimizing the time investment. Both Witten (2011) and Snee (2015) calls out the criticality of spending this time upfront to understanding (1) what each attribute means and (2) basic descriptive statistics. As the volume of data increases, the variety of examples will also grow and lead to more edge cases. For instance, the documented range of the DOHMH score is between 0 to 100 though the maximum value is 164. Identifying and removing these erroneous records needs to happen before analysis, or it will skew the statistical model. However, merely truncating results outside of the range can cause essential records to be lost, and there needs to be an investigation into the reason (e.g., missing use-cases). After using the exploration tools to determine the filter rules, concise documentation of those decisions needs to exist, so the analysis is reproducible.

Now that each of the heterogeneous data sources is standardized and erroneous records pruned, the organization is ready to begin model training. This step requires choosing one or more categories of machine learning algorithms (e.g., regression and classification) and measuring its performance. Many big data scenarios require machine learning pipelines that combine multiple strategies into a final equation (Garreta, 2013). A data pipeline might include steps to regularize values to improve convergence, then reduce dimensions with using PCA, and finally apply Gaussian noise to prevent over-fitting. Each action within the pipeline has numerous parameters that can influence the accuracy of the results. Algorithms like GridSearch optimize these scenarios through automated parameter tuning, though the engineers need to be aware of these concepts.

Finally, data engineers need to bundle the model into an Application Programming Interface (API) so that it is accessible within the production environment. One approach is to serialize the model into a Scikit Pickle or Hierarchical Data Format (HDF5) and then deploy it into the cloud using Function as a Service (FaaS) technologies (e.g., Amazon Lambda or Azure Functions). FaaS solutions are highly economical and instantly scale to enormous workloads, such as integrating with existing streaming technologies and various edge processing scenarios. However, as the number of input features increases (e.g., speech to text and video processing), commodity FaaS technologies become ineffective for inference. Instead, the use-case needs General Purpose Graphics Processing Units (GPGPU), Field Programmable Gate Arrays (FPGA), or similar Application-specific Integrated Circuits (ASIC) to evaluate the complex equations.

## What challenges does big data create

IoT, Cloud, Mobile, and Big Data (ICBM) increase the complexity to perform these steps as the volume, variety, velocity, and veracity add unique challenges. For instance, production datasets are commonly several factors of magnitude more significant than the local resources of a single computer. Users need to either apply a sampling strategy or rely on distributed data stores (e.g., Azure Data Lake, Amazon DynamoDB, and Apache Hadoop). The time required to train a model can also introduce challenges, as it impacts the user's ability to iterate through experiments. Consider the DOHMH 168mb dataset, where applying Logistic Regression took 2 to 3 minutes to converge, yet many datasets are several terabytes or more. Cloud computing enables instant provisioning of elastic resources, though many instance-based learning algorithms require knowledge about their nearest neighbors to propagate aggregate calculations (Witten, 2011, p. 312). This design requirement limits the scalability of specific training algorithms to expensive vertically-scaled servers instead of more economical horizontally-scaled patterns. Jassy (2019), the CEO of AWS, recently stated that "after creating these models, substantial more processing power is typically necessary to run the inferences." Google is addressing these challenges with custom hardware that runs at the network edge. By running machine learning algorithms at the edge, it removes the need to move these massive workloads across the network and also decreases latency. Minimizing network latency is critical for many machine learning solutions, such as health and safety monitoring or interactive user experiences that need to react in real-time. Even after mitigating the physical resource constraints, other obstacles arise from the data variety. Sun et al. (2018) discuss the challenges of text-mining outside of academic scenarios due to spelling errors and locale-specific terms that are difficult to handle. Similarly, the broad adoption of micro-blogging (e.g., Twitter) requires data practitioners to both determine context and sentiment within 148 characters. These concepts are pushing the limits of machine learning algorithms as the 'exceptions to the rule' are becoming the norm.

## What non-technical constraints exist

Along with technical challenges, concerns exist around (1) the security and privacy of user's data and (2) reasonable use policies. People trust Facebook to facilitate private conversations between a billion people responsibly. When those same users learned how their connections become manipulated to influence elections, there was outrage as this usage did not align with their expectations (Media Buzz, 2018). Equifax collects sensitive financial information about 147 million people, and the breach of their security creates a permanent risk to nearly half of America (FTC, 2020). Countless other instances also exist that introduced a long-term impact on the brand and creates a lasting competitive disadvantage. Both national and international laws, such as the Health Insurance Portability and Accountability Act (HIPAA), the Family Education Rights and Privacy Act (FERPA), and the General Data Protection Regulation (GDPR)—give legislatures the ability to penalize negligence strictly. These legal requirements create both technical and procedural needs, like encrypting data at rest and in transit and limiting access to many exotic big data sets despite their high-value potential to research.

References

DOHMH. (2020, February 15th). *DOHMH New York City Restaurant Inspection Results*. Retrieved from NYC OpenData: https://data.cityofnewyork.us/Health/DOHMH-New-York-City-Restaurant-Inspection-Results/43nn-pn8j

FTC. (2020, January). *Equifax Data Breach Settlement*. Retrieved from Federal Trade Commission: https://www.ftc.gov/enforcement/cases-proceedings/refunds/equifax-data-breach-settlement

Garreta, R. (2013). *Learning scikit-learn: Machine Learning in Python.* Birmingham: Packt Publishing.

Jassy, A. (2019, December 3rd). *AWS re:Invent 2019 - Keynote with Andy Jassy*. Retrieved from YouTube: https://www.youtube.com/watch?v=7-31KgImGgU

Matacuta, A., & Popa, C. (2018). Big Data Analytics: Analysis of Features and Performance of Big Data Ingestion Tools. *Informatica Economică Volume 22, Number 2*, 25-33.

Media Buzz. (2018, April 12th). *Mark Zuckerberg Full Testimony at Senate Hearing On Facebook Data Breach*. Retrieved from YouTube: https://youtu.be/rVfrITX3NfI

NYC Health. (2020, February 15th). *ABCEats-Restaurants*. Retrieved from NYC Health: https://a816-health.nyc.gov/ABCEatsRestaurants/#/faq

Snee, R. (2015). Practical Approach to Data Mining: I Have All These Data; Now What Should I Do? *Quality Engineering, Volume 27*, 477-487.

Sun et al. (2018). Data Processing and Text Mining Technologies on Electronic Medical Records: A Review. *Journal Of Healthcare Engineering 2018 April*.

Witten, I. (2011). *Data Mining: Practical Machine Learning Tools and Techniques, 3rd Edition.* Amsterdam: Morgan Kaufmann.

# Appendix: Python Code

######################################

# (c) Nate Bachmeier – 2020.02.16

# TIM-8130 Data Mining

# North Central University (NCU Phd CS)

##########################

# Load the CSV file into Pandas

# This creates a DataFrame that allows efficient exploration

import pandas as pd

csv = pd.read\_csv('DOHMH\_New\_York\_City\_Restaurant\_Inspection\_Results.csv')

# This snippet creates the one-hot encoded data frame

# Garreta, R (2013). Learning sckit-learn machine learning in python

from sklearn import feature\_extraction

def one\_hot\_dataframe(data, cols, replace=False):

    vec = feature\_extraction.DictVectorizer()

    mkdict = lambda row: dict((col, row[col]) for col in cols)

    vecData = pd.DataFrame(vec.fit\_transform(

    data[cols].apply(mkdict, axis=1)).toarray())

    vecData.columns = vec.get\_feature\_names()

    vecData.index = data.index

    if replace:

        data = data.drop(cols, axis=1)

        data = data.join(vecData)

    return (data, vecData)

# Split the test and training data

train = csv[:200000][["BORO","CRITICAL FLAG","GRADE","VIOLATION CODE","INSPECTION TYPE","SCORE"]].dropna()

test = csv[200000:][["BORO","CRITICAL FLAG","GRADE","VIOLATION CODE","INSPECTION TYPE","SCORE"]].dropna()

# Build the train set

X\_train = train.drop(columns=['SCORE'])

Y\_train = train[["SCORE"]]

# Build the test set

X\_test = test.drop(columns=['SCORE'])

Y\_test = test[["SCORE"]]

# One Hot encode

x\_train\_hot = one\_hot\_dataframe(X\_train, ["BORO","CRITICAL FLAG","GRADE","VIOLATION CODE"], replace=True)

x\_test\_hot = one\_hot\_dataframe(X\_test, ["BORO","CRITICAL FLAG","GRADE","VIOLATION CODE"], replace=True)

# Drop these columns as they do not exist in both halfs

x\_train\_data = x\_train\_hot[1].drop(columns=["VIOLATION CODE=06H"])

x\_test\_data = x\_test\_hot[1].drop(columns=['VIOLATION CODE=02E','VIOLATION CODE=04I'])

from sklearn import tree

clf = tree.DecisionTreeRegressor()

clf.fit(x\_train\_data, Y\_train['SCORE'])

clf.score(x\_test\_data, Y\_test['SCORE'])

# 67%

from sklearn.decomposition import PCA

pca = PCA(n\_components=10)

pca = pca.fit(x\_train\_data, Y\_train['SCORE'])

x\_train\_pca = pca.transform(x\_train\_data)

x\_test\_pca = pca.transform(x\_test\_data)

from sklearn import tree

clf = tree.DecisionTreeRegressor()

clf.fit(x\_train\_pca, Y\_train['SCORE'])

clf.score(x\_test\_pca, Y\_test['SCORE'])

# 67%

bins = [-5, 15, 25,50,200]

Y\_train\_bins = pd.cut(Y\_train['SCORE'], bins=bins, labels=bins[0:-1])

Y\_test\_bins =  pd.cut(Y\_test['SCORE'], bins=bins, labels=bins[0:-1])

from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()

clf.fit(x\_train\_pca, Y\_train\_bins)

clf.score(x\_test\_pca, Y\_test\_bins)

# 93.8%

import numpy as np

Y\_rand\_bins = pd.cut((np.random.random(Y\_test\_bins.shape)\*100), bins=bins, labels=bins[0:-1])

clf.score(x\_test\_pca, Y\_rand\_bins)

# 10.7%