NYC Open Data Final Report

Jack Wolfgramm

June 2022

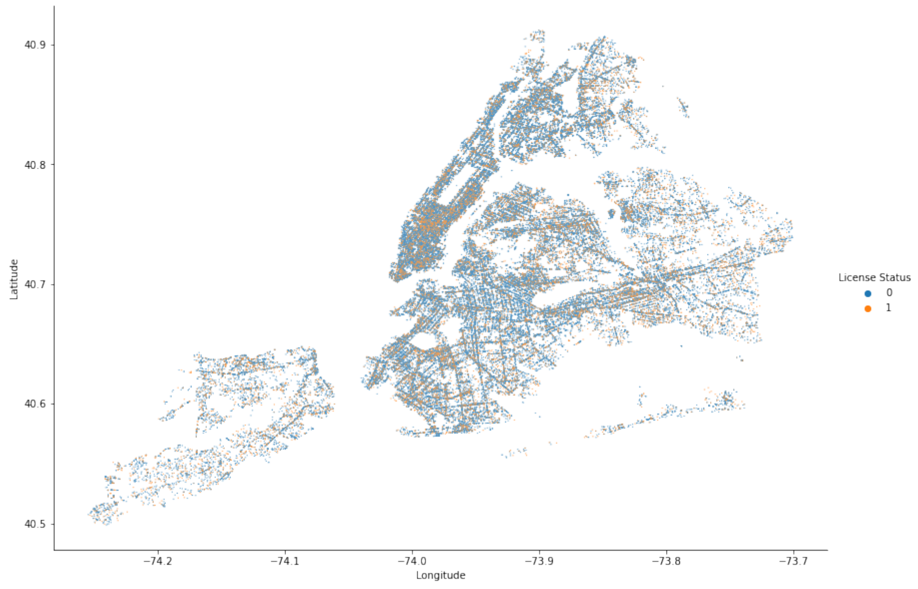


Figure 1: Scatter plot showing the location of business licenses within NYC, the color indicates whether the license is active or not. The outline of Central Park can be seen as there are few businesses inside.

1 Project Idea and Goal

The central plan of the project was to create a model that would predict duration of operation for a given business license by utilizing clustering. This would be useful to any business which was considering a license. For example, the profitability of any business venture depends on the duration of time the venture will be active. Having a more accurate regression model for the duration of license validity would provide value for businesses.
However, the models created without feature analysis (in my experience) often

perform poorly, which drove me to try to use clustering to improve model performance. The key idea is that the duration of business license duration in the nearby vicinity (both in space and in time) should provide good information, and hence be a good feature to add. In addition, a model that was trained using only cluster information should be easier to generalize to other locations, since only local information is being used. A model created using cluster analysis could easily be used on a different city, provided the data was in the same form. Adding the clusters increased the R-squared of the model by .2, which is a rather large increase.

2 Data Sourcing

New York City keeps many datasets on NYC Open Data. The dataset in question is located here, and contains around 270,000 entries. Features include the date the license was given, when the license will expire, whether the license is active, and the type of license. There where 27 columns, or features, total, most of which was data that could not be easily fed in to a machine learning algorithm. Some of the most common licenses, like "Tobacco Retail Dealer," make sense, while others are extremely obscure. For example, there are 27 licenses in the industry called "Commercial Lessors." According to my research, a Commercial Lessor is a business that rents out space or equipment to a *different* business that runs *bingo* operations. Why such businesses would need licenses in order to operate is far beyond my legal knowledge, but into the data set it goes!

3 Data Cleansing

With 270,000 entries, a few things needed to be cleaned from the data. I only wanted to analyze businesses in NYC proper, so I cut off all entries that were outside. Additionally, many entries had missing longitude and latitude data. I simply dropped these data points because I thought that they were irrelevant to the question at hand. This cutoff a very significant (about 90,000) number of entries. **In retrospect, I believe that this was a significant mistake.** While the longitude and latitude were indeed missing, many of them listed the street address. I could have used the information I had to impute a nearby location for many of the entries. In the future, I would pay much closer attention to this kind of missing data. I also dropped many of features that would be difficult to work with in a machine learning model, like the street and building name (using hot encoding would create far too many features).
Leaving that aside, my first attempt to asses the quality of the data was to try to draw a scatter plot of the location of the licenses. The graph looked like a large indiscriminate blob. After investigating further I found that there was a very large (around 5,000) number of licenses which were all issued with a single point far to the south of the main blob. This can be seen in figure 2. I searched for this point on a map and found that the location was in Pennsylvania, in

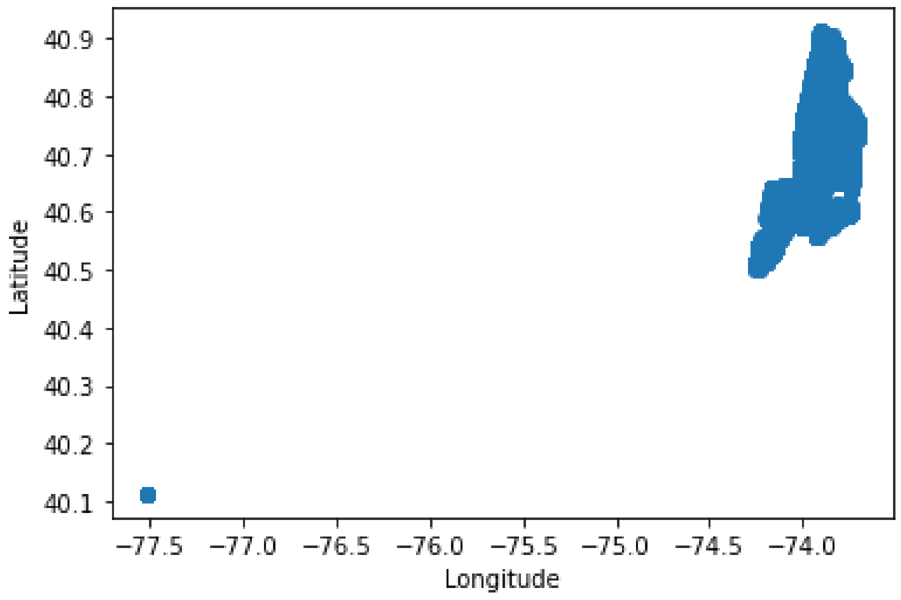


Figure 2: Scatter plot showing the location of business licenses with the state labeled "NY." Note that the there is a large central blob (this is NYC proper) and a single point far away.

a seemingly random location. I could not understand the significance of this (perhaps it's just the location that is defaulted to), so I decided to cut these entries completely (this was only around 5 percent of the data set).
Finally, I began to clean up the data relating to the duration of license validity (which is the target feature for the model). The data is distributed irregularly, has a different distribution depending on whether the license is valid or not (see figure 3). There were around 200 licenses which had an expiration date earlier than the start date, which is not numerically significant but does make me question the quality of the data.With this, I felt relatively comfortable moving

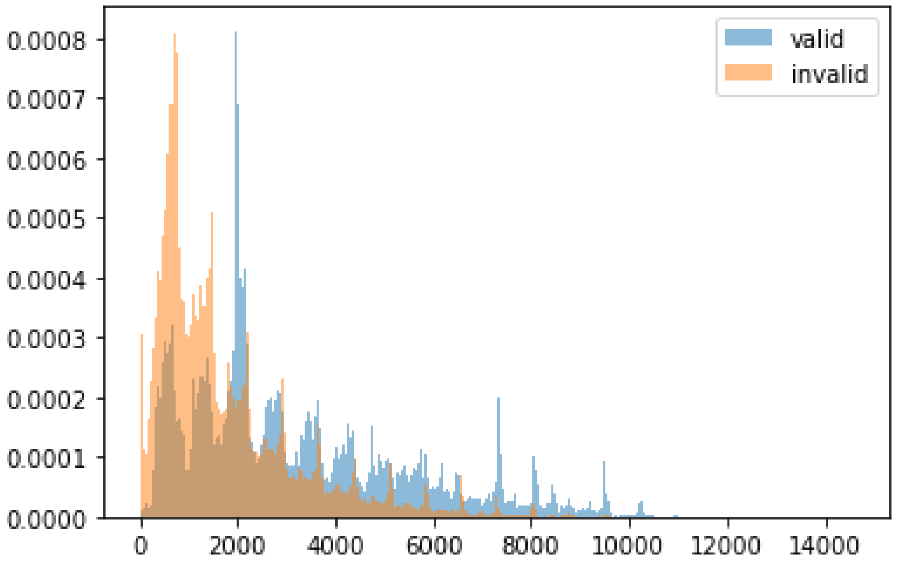


Figure 3: Distribution of valid and invalid licenses. Note that the distribution are different and that there are spikes at one year intervals.

on to other phases of the project.

4 Exploratory Data Analysis

As I explored the data more, I discovered some interesting facts. For example, the average duration of license validity was fairly consistent across the most common types of business licenses, as shown by the box plots in figure 4. Finally, it was time to run a cluster analysis. I decided to only cluster over physical location for a first run through. Exploring running k-means clustering on the businesses yielded the interesting insight that the duration of license validity between the clusters was statistically significant (even if the average cluster size contained 300-500 businesses). This leads credence to the fact that local information about businesses is important, and that the successful or unsuccessful

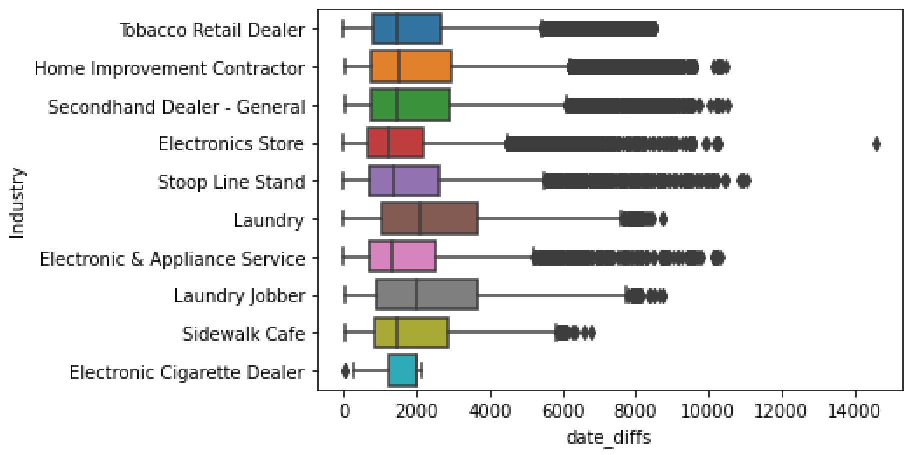


Figure 4: Box plots of the most common business licenses. The means are similar though statistically significant.There seem to be more outliers than expected.

businesses tend to cluster together. On the other hand, there did not appear to be any natural way to cluster businesses together, as can be seen in figure 5. The idea of using such a chart would be that sum of square distances (from the center of the cluster) would initially drop drastically, but that after the true number of clusters was achieved, the sum of squares would fall much more slowly.
Despite this conflicting information, I decided that the results were overall promising and decided to try some modelling. The initial models that I created seemed to show a promising result: by adding the mean of the cluster's business license duration, the root mean squared error of the regression model decreased by 22 percent! ***However, though I didn't realize it at the time, there were very significant issues related to data leakage.*** You see, by adding these cluster means to the data, and only *after* breaking the data in to training and testing sets, I was allowing significant information about the test data to leak in to the training data. This was because the means of were calculated using both the training and the test set. I would confront this issue when I began to try to make better models.

5 Modelling

5.1 Model Pipeline

Using several different regression models, I found that XGBoost generally performed the best. Random forests were a close second, with other models rather far behind. When it came time to actually build the XGBoost regression model, it was important to tune not just the hyperparameters of the model, but also the

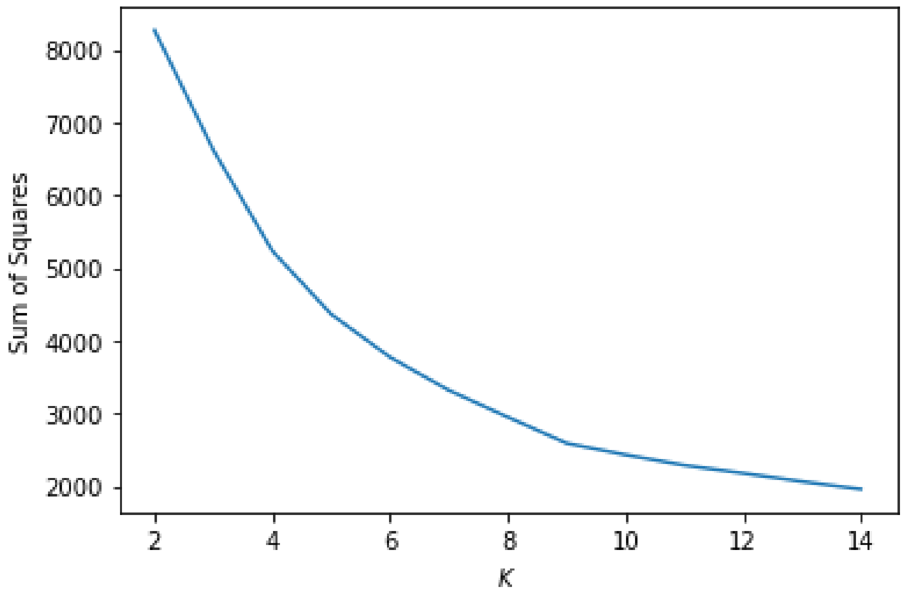
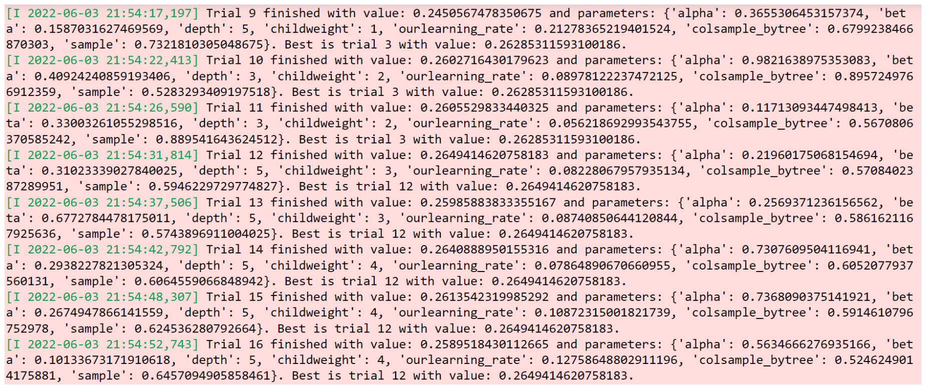


Figure 5: Basic sum-of-square-distance vs number of clusters chart. I also created an extended chart with a greater number of clusters, which simply looked like this one (consistently decreasing but never having a real kink).

number of clusters, and the way that the clusters were created (i.e. how heavily to weigh the distance vs the time since the license was created). This presents its own technical issues, since the number of clusters used should depend on the size of the data set. However, finding some relationship between the size of the dataset and the ideal number of clusters seems difficult to do without some extra steps that would require finding the ideal k for several different data sized. Since I was using training sets of size 0.8, I made the assumption that using the same k would be close enough to ideal.
Using this approximation, I could find good hyperparameters directly. **I also decided to vary the weight of the time vs. distance factors.**Scaling both time and distance to be equally large would be standard clustering practice, but, A priori, it is unclear whether we should consider businesses which got their licenses at a closer time, or businesses which got their licenses at a closer place as more important. Which should you take in to account more - the successful business that opened a block away just last month, or the unsuccessful license that opened at the neighboring location a year ago? This type of question had to be answered by tuning the hyperparameters. Since creating clusters can take quite a while, I used **Optuna** to help find the ideal set of paramters. Optuna

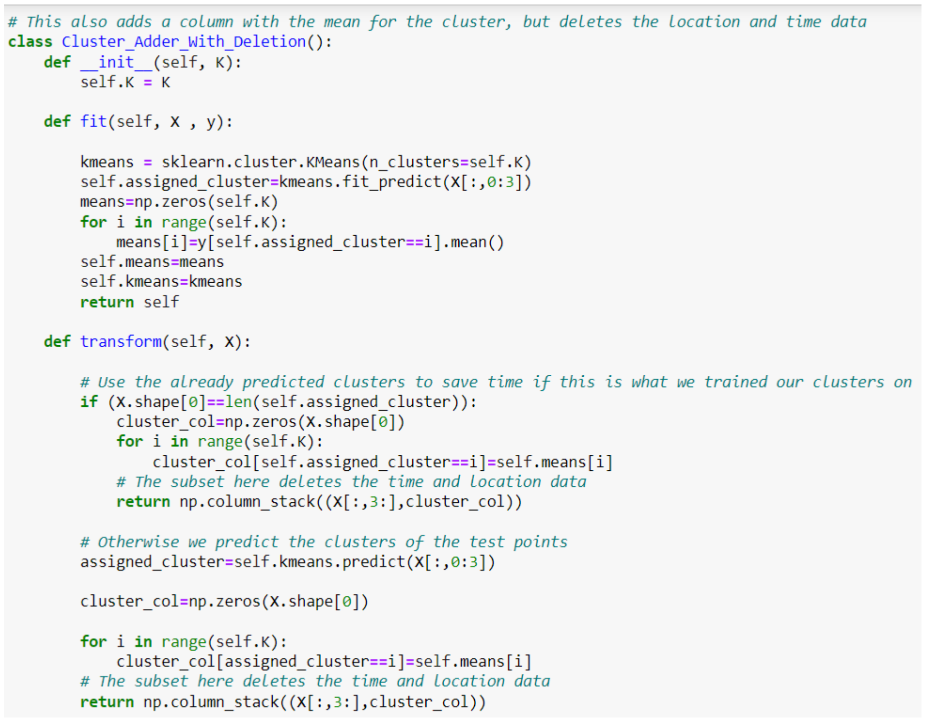


```
[I 2022-06-03 21:54:17,197] Trial 9 finished with value: 0.2450567478350675 and parameters: {'alpha': 0.3655306453157374, 'bet
a': 0.1587031627469569, 'depth': 5, 'childweight': 1, 'ourlearning_rate': 0.21278365219401524, 'colsample_bytree': 0.6799238466
870303, 'sample': 0.7321810305048675}. Best is trial 3 with value: 0.26285311593100186.
[I 2022-06-03 21:54:22,413] Trial 10 finished with value: 0.2602716430179623 and parameters: {'alpha': 0.9821638975353083, 'bet
a': 0.40924240859193406, 'depth': 3, 'childweight': 2, 'ourlearning_rate': 0.08978122237472125, 'colsample_bytree': 0.895724976
6912359, 'sample': 0.5283293409197518}. Best is trial 3 with value: 0.26285311593100186.
[I 2022-06-03 21:54:26,590] Trial 11 finished with value: 0.2605529833440325 and parameters: {'alpha': 0.11713093447498413, 'be
ta': 0.33003261055298516, 'depth': 3, 'childweight': 2, 'ourlearning_rate': 0.056218692993543755, 'colsample_bytree': 0.5670806
370585242, 'sample': 0.889541643624512}. Best is trial 3 with value: 0.26285311593100186.
[I 2022-06-03 21:54:31,814] Trial 12 finished with value: 0.2649414620758183 and parameters: {'alpha': 0.21960175068154694, 'be
ta': 0.31023339027840025, 'depth': 5, 'childweight': 3, 'ourlearning_rate': 0.08228067957935134, 'colsample_bytree': 0.57084023
87289951, 'sample': 0.5946229729774827}. Best is trial 12 with value: 0.2649414620758183.
[I 2022-06-03 21:54:37,506] Trial 13 finished with value: 0.25985883833355167 and parameters: {'alpha': 0.2569371236156562, 'be
ta': 0.6772784478175011, 'depth': 5, 'childweight': 3, 'ourlearning_rate': 0.08740850644120844, 'colsample_bytree': 0.586162116
7925636, 'sample': 0.5743896911004025}. Best is trial 12 with value: 0.2649414620758183.
[I 2022-06-03 21:54:42,792] Trial 14 finished with value: 0.2640888950155316 and parameters: {'alpha': 0.7307609504116941, 'bet
a': 0.2938227821305324, 'depth': 5, 'childweight': 4, 'ourlearning_rate': 0.07864890670660955, 'colsample_bytree': 0.6052077937
560131, 'sample': 0.6064559066848942}. Best is trial 12 with value: 0.2649414620758183.
[I 2022-06-03 21:54:48,307] Trial 15 finished with value: 0.2613542319985292 and parameters: {'alpha': 0.7368090375141921, 'bet
a': 0.2674947866141559, 'depth': 5, 'childweight': 4, 'ourlearning_rate': 0.10872315001821739, 'colsample_bytree': 0.5914610796
752978, 'sample': 0.624536280792664}. Best is trial 12 with value: 0.2649414620758183.
[I 2022-06-03 21:54:52,743] Trial 16 finished with value: 0.2589518430112665 and parameters: {'alpha': 0.563466627693516б, 'bet
a': 0.10133673171910618, 'depth': 5, 'childweight': 4, 'ourlearning_rate': 0.12758648802911196, 'colsample_bytree': 0.524624901
4175881, 'sample': 0.6457094905858461}. Best is trial 12 with value: 0.2649414620758183.
```

Figure 6: Some runs that Optuna generated.

uses more advanced algorithms to find the maximum in the parameter space, and tends to greatly outperform not just grid search, but also random search. I found in my experience that Optuna would always find a very good set of parameters after 25 runs, and that model performance would not appreciably increase after this (even after 300+ runs).
Finally, I returned to the issue of building a pipeline without data leakage. This was an unforeseen but highly instructive difficulty of the modelling process. Luckily, sklearn has a fairly intuitive way to do this - one can add in a preprocessing step to the pipeline. This typically allows one to scale the data and use one-hot-encoding with no data leakage (for an explanation and solution to OneHotEncoder data leakage see this data science stack exchange question). Finally, I had to create my own function that would cluster only over the train-

ing data, and when running over the test data assign the new data the means of the clusters generated with the test set. This required building my own function that had both a fit() and transform(), so that it would function within the sklearn "columntransformer" preprocessing step. I created a rough-and-ready function that worked for my purposes, though it generalizes from my own use case very poorly. Putting all of this together allowed me to build good models



```
# This also adds a column with the mean for the cluster, but deletes the location and time data
class Cluster_Adder_With_Deletion():
    def __init__(self, K):
        self.K = K

    def fit(self, X , y):

        kmeans = sklearn.cluster.KMeans(n_clusters=self.K)
        self.assigned_cluster=kmeans.fit_predict(X[:,0:3])
        means=np.zeros(self.K)
        for i in range(self.K):
            means[i]=y[self.assigned_cluster==i].mean()
        self.means=means
        self.kmeans=kmeans
        return self

    def transform(self, X):

        # Use the already predicted clusters to save time if this is what we trained our clusters on
        if (X.shape[0]==len(self.assigned_cluster)):
            cluster_col=np.zeros(X.shape[0])
            for i in range(self.K):
                cluster_col[self.assigned_cluster==i]=self.means[i]
            # The subset here deletes the time and location data
            return np.column_stack((X[:,3:],cluster_col))

        # Otherwise we predict the clusters of the test points
        assigned_cluster=self.kmeans.predict(X[:,0:3])

        cluster_col=np.zeros(X.shape[0])

        for i in range(self.K):
            cluster_col[assigned_cluster==i]=self.means[i]
        # The subset here deletes the time and location data
        return np.column_stack((X[:,3:],cluster_col))
```

Figure 7: Function that adds cluster means column to data. This version deletes the actual longitude and latitude, and creation date in order to create a model that does not depend on them.

that were well optimized and suffered from minimal data leakage.

5.2 Modelling Findings

The pipeline and Optuna optimization process allowed me to test various subsets and possibilities. **The results that I got were confusing and unclear.** In general, the clustering model (with deletion) was better than the non-clustering model by a very small margin (R-Squared improved by around 1 to 10 percent). The models themselves were rarely stellar - the best one achieved an R-Squared of around .25, which may be useful if a business were to consider and pursue many different licenses, but would rarely help for a single license. Additionally, I ran in to the strange issue that using more data produced a worse model.

The R-Squared that my model got by running on 10 percent (randomly chosen) of the data was much better than the R-Squared that I got by running it on the full data set, dropping from 0.2591 to -0.05. The reason for this is unclear to me, though I did manage to find one hint. By using data relating to only a single industry (say cigarette selling licenses), the model was able to achieve and R-Squared of 0.35. This would seem to hint that the quality and predictability of the data could vary greatly depending on what subset one looks at.
Finally, the optimal number of clusters does not seem to make sense! Using 10 percent of the data, the optimal number of clusters to use was 33 - this corresponds to an average cluster size of 280. However, when using the full dataset, the ideal number of clusters was only 10! While this may cause one to believe that the clusters were unimportant, the opposite also seems to be the case - using neither clusters nor the location/time results in models that are significantly worse. These deep mysteries could not be answered in the time that I had, perhaps they can be left as exercises to the reader.

6 Future Directions and Possible Uses

Again, the final model used a cluster of size 10. The weighting of the time and distance scaling was a ratio of about 4:1. This indicates that businesses which obtained licenses at similar times (on a 50 year scale) were more important indicators of performance than businesses which opened at similar locations (on the scale of NYC). The model had a cross-validated R-Squared of -0.051. Compare this to the model obtained with no clusters (but including time and location): -0.057. Thus, the models seem to be almost identical in performance. This still leaves open the avenue that the the clustering model generalizes better to other cities and times. ***The following is my suggestion for how the model could be used:*** a business would like get a new licenses (it could be anywhere and at any time) for a number of different locations, for example, they would like to begin selling e-cigarettes at 10 different locations across a city. They would like to know which of their 100 different store locations would be the best candidates. They would begin by gathering information about how long the licenses of stores near each one of their own stores were lasting, as a rough approximation of the cluster mean. The beauty of this model is that they could use it find the stores which were most likely to have long-lasting licenses, without having to gather data on the whole city! An entry to the model would consist only of the industry and cluster mean, and the model would produce a fairly accurate prediction of the duration of license duration. Since the model was not trained to depend on the specific time and location, this prediction should at least be more accurate than random. Thus, by making informed decisions over long enough periods of time, the business should be able to increase its profits.
In the future, I would like to improve the model significantly. I believe, as detailed in the modelling section, that there are some deep mysteries as to which subsets of the data are most amenable to modelling. It is possible that

truly exemplary models could be created on certain special subsets of the data. For the number of clusters, its importance and its relationship to the size of the dataset could be much better understood. Finally, I would like to find a way to include the licenses that lacked exact longitude and latitude data, since this would increase the size of the dataset by 30 percent. In addition, businesses which did not list Latitude and Longitude may have similarities with eachother, which introduces a correlation that means that they cannot be dropped from the model without consequences. In total, this project has been a rich learning experience and I can imagine picking it up at some other point.