NYC Longevity Analysis & Site Location

Description

We want to use a longitudinal database of retail businesses in New York City built on publicly available data. The directory is 'open' because it aims to include every retail business that has been open for business in NYC in the past decade. It's also 'open' in the sense that the database is built on public, open data, made accessible by city and state regulatory agencies. Our goal is to use this database to inform site location.

Storefront vacancy in New York has received considerable attention in the press. Widespread commercial vacancy constitutes a crisis that is symptomatic of the city's increasing inhospitality to small businesses. Besides disagreement over the consequences of commercial vacancy, there is also disagreement over the precise nature of the phenomenon. The question of whether vacancy has risen in the past two decades is unresolved.

We are considering the creation of a retail site location technology. Combining our database with contextual features, such as proximity to public services, neighborhood demographics, and mobility measures, we hope we can come to an understanding of how a business's surroundings affect vacancy and turnover probabilities. The ultimate goal is to predict, given large-scale economic trends and business data, if a business will survive from one year to the next. Over the course of many years, we may define this survival as business longevity.

Goals

Our first goal is to collect data from public sources. We identified a number of New York City and State governmental sources for this data: the Department of Consumer Affairs (DCA), the Department of Agriculture and Markets (DOA), the Department of Education (DOE), the Department of Finance (DOF), the Department of Health (DOH), the Department of State (DOS), the Department of Transportation (DOT), and the Liquor Authority (LA).

Our second goal is to clean and concatenate the data. Once we collect data from each source, either by scraping it, downloading a CSV file, or downloading a TXT file, we need to translate the data to a Pandas dataframe. This transformation allows the data to be easily manipulated, utilizing Python to organize the data. Using the aforementioned data sources, we are able to access fine-grained details on businesses across all of New York City. The data is often unclean, unclear, and fraught with errors. Often zip codes, addresses, or business names miss a letter or are entirely incorrect. We need to change the names of column headers to achieve greater clarity about the content of each dataset. This standardization

of column headers also allows us to easily concatenate different datasets using like columns. We need to identify which errors we can fix, and write a script to programmatically clean our data.

Our third goal is to perform record linkage across the large dataset. Using the information under each column header, we need to determine which rows pertain to the same business. For example, both the DOF and the DOA may have collected information on a Dunkin franchise on 95th Street, and Columbus Avenue. The DOA may have entered the name as "Dunkin," at an address of "95th Street and Columbus Ave," the DOF may have written an entry using the name "Dunkin Donuts" and the address "730 Columbus Ave." Although the name and the address are represented differently, they reference the same business. We need to write a script to determine the similarity between two address entries, as well as two separate names. We may then assign a unique identification string to each business, to keep track of these matchings.

Our fourth goal, after matching like businesses, is to understand the type for each business. A small coffee shop may have a different vacancy and turnover performance than a large supermarket. We must use the name and business type labels (if applicable) to sort each business into particular, interpretable types.

Our fifth goal, once we have matched like businesses and created an understanding of each business' type, would be to use data from each data source to determine how long each business has existed in a particular location. This allows us to not only understand business trends, but patterns pertaining to building vacancy: if we know where each business is over time, we can identify vacancies. We may use the dates associated with each record to understand when a business begins and ends. We can create a new dataset of observations of each business each year, with a variable indicating whether or not the business survives in the following year.

Our sixth goal would be to use our observation dataset to make predictions about business survival. We may fit a model to the business-year observation data, using the survival variable as the independent variable we intend to predict. Given data on a business at a particular time, we can predict if the business will survive into the following year. With survival results for every business in New York City, and we could create a projection of vacancy.

Our seventh goal would be to determine the accuracy of our probabilistic determinations, of which there are three: record linkage, business type determination, and business survival determination. Given that there is no public and accurate business directory, to understand the accuracy of our record linkage and business type determination, we need to take a sample of businesses and determine the accuracy "by hand." We also want to determine the quality of our survival prediction. We may train the data on survival of a subset of businesses over years 2010-2021. We may take a separate subset of observations as test data, and determine the accuracy of our survival predictions.

Our eighth and final goal is to create a simple demo, displaying building vacancy, current New York City businesses, and our prediction of those that survive (along with the business type of each business).

Software/Approach

Goal One: Collecting data from public sources.

In order to find the pertinent data, we began by researching basic data sources. BetaNYC, a government tech nonprofit, wrote of using barbershop and appearance enhancement licenses to track businesses over time.[1] We were able to accomplish this by scraping the DOS License Center website. We expanded our search to Open Data NYC, and found comprehensive data on NYC businesses, sourced from a number of local government departments.[2] From these data sources, we primarily aim to collect the following information: Dates (Expiration dates, Start dates, Inspection dates), Address(es), Name(s), Phone number(s), License Identifier(s), Inspection Result, Business Type.

Goal Two: Cleaning and concatenating the data.

After the collection of these various TXT and CSV files, we wrote scripts for each data source, to convert each file into a Pandas dataframe. We chose to use Pandas because it allows us to use Python to manipulate data efficiently. We can also append to data frames seamlessly alongside the data scraping process. The Pandas library makes it very easy to convert these native files into the dataframe format.

We needed to create a method to ingest addresses and ensure that we could understand multiple representations of the same address. New York City's borough-block-lot (BBL) system assigns each building to a borough, block, and lot. This means that each building has a unique identifier. In order to convert addresses to this unique ten-digit code, we used a library called GeoSupport for Python. GeoSupport takes address strings and returns an abundance of information about each building, including zip code, census tract, year built, building type, longitude, latitude, and–of course–BBL. We used these tax lots to aggregate license and inspection data at the building level.

---

[1] https://beta.nyc/2018/05/15/scraping-nys-beauty-salon-and-barbershop-data/

[2]
Department of Consumer Affairs (DCA): https://data.cityofnewyork.us/resource/ptev-4hud.csv
Department of Agriculture and Markets (DOA): FOIL on https://openfoil.ny.gov/#/newfoilrequest
Department of Education (DOE): http://www.op.nysed.gov/opsearches#
Department of Finance (DOF): FOIL on https://openfoil.ny.gov/#/newfoilrequest
Department of Health (DOH): https://data.cityofnewyork.us/resource/43nn-pn8j.csv
Department of State (DOS): https://aca.licensecenter.ny.gov/aca/GeneralProperty/PropertyLookUp.aspx
Department of Transportation (DOT): https://data.cityofnewyork.us/resource/4dx7-axux.csv
Liquor Authority (LA): https://www.tran.sla.ny.gov/JSP/query/PublicQueryPremisesSearchPage.jsp

In order to add BBL to each record, we needed to clean each address to make it interpretable by the library. We eliminated non-numeric characters from the zip code and building number entries, and we ensured that the entries for borough had one of the five borough titles. Once the addresses were cleaned, we were able to generate BBLs for over 1.9 million of the nearly 2.1 NYC business records. In addition to adding BBL codes to each business record, we standardized the phone numbers to a ten-digit string and renamed the columns such that the columns that represented the address in one dataframe lined up with those in the others.

Goal Three: Record linkage.

After cleaning the data, we proceeded with record linkage, determining which records referred to the same entity. To start, within a tax lot, for each possible pair of records, we compared the name, phone number, agency-issued ID number, and business type (again, blocking by tax lot). For the name, we calculated the cosine distance between the name values in each record (decimal value between 0 and 1). For the phone number and the agency-issued ID, we looked for an exact match (categorical value of either 0 or 1, although the phone number comparison feature may take the value of .5 if a phone number is not listed). For each business type, we created sets of like business types. If two records had business type values in the same set, we returned a value of 1 (otherwise 0). Within each BBL-block, there are (# number of records per BBL) choose 2 comparisons, and for each comparison, we extract these four features (the comparisons of business name, phone number, agency-issued ID number, and business type, which we will call the comparison features).

We tried using both a deterministic algorithm and an unsupervised clustering algorithm in order to determine if two business records pertained to the same business. We used a Gaussian Mixture Model as our unsupervised probabilistic model: this allowed us to fit multi-dimensional Gaussian distributions to our clusters of comparison features. We fit these distributions across the comparison features of the entire dataset. The Gaussian Mixture Model begins by fitting K (2 clusters in our case: match and no match) distributions to the data with random parameters. We find the optimal values for these parameters, and the solutions will correspond to the Maximum Likelihood Estimates (MLE) for this setting. These distribution parameters are then adjusted in an iterative process called expectation maximization. This iteration occurs until the convergence in the MLE value.

The primary issue with these models is that they expect to fit features in the space of $\mathbb{R}^n$ to the Gaussian distributions. While our name comparison feature (representing cosine similarity) is in $\mathbb{R}$-space, our other comparison features are categorical. We are thus unable to fit Gaussian distributions to these categorical features using the standard methodology, but found a paper and application of Gaussian

Mixture Models on mixed datasets.[3] In short, this algorithm translates our $\mathbb{Z}$-space categorical features to the $\mathbb{R}$-space in a fashion similar to the hidden layers of a neural network. On our new set of $\mathbb{R}^n$ features we fit two Gaussian distributions, one that represents matches, and one that represents mismatches. We used the matches cluster to determine if two rows pertained to the same business.

Goal Four: Assign Business Type

While licenses usually contain some information about the business' industry, the information is not standardized. The North American Industry Classification Scheme (NAICS) is commonly used for industry categorization. Our challenge was to translate the unorganized industry/business type descriptions found in our data into NAICS codes. We used an adjusted version of the Bidirectional Encoder Representations from Transformers or BERT language model, called Sentence-BERT or SBERT to accomplish this. The model looks at words in a bidirectional contest–this suggests that it looks at words both to the right and left to understand their context. SBERT uses this analysis on common Google searches to create semantically meaningful sentence embeddings. This allows us to input two distinct sentences to the model, and quickly derive a value similar to the cosine distance. Rather than using characters to determine this distance, it is based on the semantic embeddings.[4]

We used these embeddings to compare the semantic distance between a business' type, and the list of business types associated with NAICS codes. We assigned the NAICS definition with the highest semantic distance and its code to the business type of each business. An NAICS code has six digits, and, as the place of each digit decreases, the specificity increases. This is to say that we may use the first two digits of the NAICS codes to classify businesses into 100 broad business types, rather than classifying them based on the error-prone string definitions of business type.

Goal Five: Understanding business longevity.

We created an understanding of each business' lifespan based on the associated dates. We took the earliest date that shows up for each business to represent the start date of the business. To understand when the business closed, there may be dates associated with an "out of business" date. Otherwise, we determine how recent the latest recorded dates are. If we determine that the latest recorded dates are recent enough, we decide the business still exists.

From the two dates we create for each business, "start date" and "end date," we create what we call business observations. We create observations at the arbitrary date: January 1 of each year. We record each business that exists at that time, and if they survive into the following year (this is represented by a 1

[3] https://arxiv.org/pdf/2010.06661.pdf
[4] https://arxiv.org/abs/1908.10084

or a 0). At each January 1 date, we add certain useful features: how long the business has survived, the economic trends of the time period, the zoning conditions of the buildings.

Goal Six: Make predictions about business survival.

We used the follow colums as feature values in each model: 'zonedist1', 'bldgclass', 'histdist', 'landmark', 'lotarea', 'bldgarea', 'comarea', 'resarea', 'officearea', 'retailarea', 'garagearea', 'strgearea', 'factryarea', 'otherarea', 'numfloors', 'unitsres', 'unitstotal', 'lotfront', 'lotdepth', 'bldgfront', 'bldgdepth', 'bsmtcode', 'assessland', 'assesstot', 'yearbuilt', 'yearalter1', 'builtfar', 'residfar', 'commfar', 'facilfar',"Months Active", 'GCP (NYC)', 'GDP (USA)', ' Payroll-Jobs Growth, SAAR - NYC', 'Payroll-Jobs Growth, SAAR - USA', 'PIT Withheld, Growth, NSA - NYC', 'PIT Withheld, Growth, NSA - USA', 'Inflation Rate, NSA - NYC', 'Inflation Rate, NSA - USA', 'Unemployment Rate, SA - NYC', 'Unemployment Rate, SA - USA'.

Although we cleaned the data and tried to make sense of address and business type values, given that our data is generated from a wide range of unclean data sources, many of the values of the features were NaN. We looked to pursue one of three options: we could eliminate rows with NaN values in any feature column, we could use a model robust to NaN values, or we could impute the NaN values (using the mode, mean, or a mix of both). Upon elimination of each row with any NaN value, we saw a decrease in data points from 643,597 to just over 60,000. Nearly 70% of these losses were due to a NaN value in just one feature column, and so we lost a lot of data pertaining to changes in the other columns. Also, eliminating all rows with any NaN values would disallow us from making predictions on 90% of businesses. Although a model robust to NaN score may be stronger, as it would not ignore the differences between rows with NaN and rows without, Scikit Learn does not yet allow the use of models robust to NaN values (although XGBoost does).[5] So, we decided to impute the missing values in all columns, using the mode of the feature column. In imputing rather than eliminating NaN, we were seeing higher F1 scores.

We transformed both the target column (survival value) and each feature column into NumPy arrays. Of the feature columns, some were categorical ('zonedist1', 'bldgclass', 'histdist', 'landmark'), so we needed to binarize the column data. The other columns were numerical, so we did not apply any mappings to them during the transformation to a NumPy array.[6]

We tested out seven different classifiers in order to see which model would best fit the data and have the closest, most accurate results. The seven we worked with were MLP Classifier, SVM SVC, Stochastic Gradient Descent, Decision Tree, Random Forest, Cox's proportional hazards model, Random Survival Forests.

[5] https://stackoverflow.com/questions/30317119/classifiers-in-scikit-learn-that-handle-nan-null
[6] https://dunyaoguz.github.io/my-blog/dataframemapper.html

MLP Classification. This is a multi-layer Perceptron classifier that optimizes the log-loss function using LBFGS or stochastic gradient descent. We may specify the hidden layer size, the solver, and other parameters.[7] The MLP classification is very flexible, and well suited to generally map input to output classification variables.

C-Support Vector Classification (SVM SVC). This classifier utilizes the flexibility of kernel functions to separate a vector space into the target classification. We may specify the kernel type, along with other parameters.[8] This support vector classifier works best on linearly separable data, and the runtime scales quadratically given the number of samples.

Stochastic Gradient Descent Classification. This classifier determines a loss function of which is estimated the gradient at a given point. We may specify the loss function type, penalty, and other parameters. The gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule until convergence.[9] Although computationally fast, the classifier steps toward the minima are noisy, and may lean the gradient descent in different directions.[10]

Decision Tree Classification. Decision trees split data points by features, often beginning with those most deterministic of effects in the target variable (may make this selection random). It may do this to a max depth, which we may specify in order to prevent overfitting.[11] Although decision trees often require relatively minimal preprocessing, small changes in data may yield large changes in predictions, as changes propagate through the tree.[12]

Random Forest Classification. This classifier fits a number of decision tree classifiers on different sub-samples of the data. This allows the random forest classifier to improve predictive accuracy and control for overfitting. We may specify the number of trees, the criterion to determine the quality of a split, and the max depth for each tree, along with other parameters.[13]

Cox's proportional hazards model. In order to predict survival after multiple years, we need to understand the conditions of a business in the future. We may use an above model on 2021 economic and location condition data to predict the survival of businesses in 2022. In order to then make predictions about 2023 business survival, we need to make assumptions about the economic and location data of 2022, and so on. There are multiple sources of data on predictions of economic trends, and there are building permits that may assist in the prediction of location data. Retrieving and organizing the data may be too complicated, so we can use the confidence of our prediction (if the prediction indicates that a

[7] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
[8] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[9] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
[10] https://www.asquero.com/article/advantages-and-disadvantages-of-stochastic-gradient-descent/
[11] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
[12] https://dhirajkumarblog.medium.com/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a
[13] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

business will survive) as an analog for longer-term survival predictions. This is to say that a business that is predicted to survive into the following year with low confidence, will likely not survive as long as a business with high confidence. We may test the correlation between confidence and business longevity by taking our 2015 predictions, measuring their confidence values, and retrieving the correlation between those confidence values and the survival time after 2015. Given a positive correlation, we may assume that the confidence values are an acceptable, although imprecise, measure of longevity.

Although this may work, Cox's proportional hazards model is built to predict survival times and thus may be used to circumvent the use of a confidence measure. The model identifies entities at a point in time and, along with important features, two variables are collected for each observation: "status" and "survival time." Status measures whether or not a record was censored–that is to say that the collection of data stopped at a point, and so the survival time listed serves as a minimum value. This may occur for businesses that are open during the time of observation, so we do not have a sense of their full survival time, only how long they have survived. When the "status" variable is false, however, the record was not censored, so the survival time is as recorded. For the purpose of this project, this means that the business has closed. Consider the graph in Appendix A. Entity A was lost, and so we know that A survived for a minimum of three months. Similarity B dropped out, and E survived until the end of the study. We understand fully the survival of only C and D and so, at the end of the study, the "status" variable would be true for only those two entities.[14]

The survival function, $S(t)=P(T>t)$, returns the probability of survival beyond time t, where T denotes a continuous non-negative random variable corresponding to survival time. We may estimate the value of the function: *S(t)=number of patients surviving beyond t / total number of patients*. Given censored values, however, we may not understand exactly how many patients survive beyond a certain time. The Kaplan-Meier Estimator can, however, make these estimations given censored information.

We must consider that there may be differences between the survival of different entities, given different features (for buildings, imagine traffic, rent, zoning, etc). Although we can divide data by features, this grows unwieldy as feature counts increase. Cox's proportional hazards model is a multivariate survival model that estimates the impact of each feature on survival.

Traditionally, we make one observation of each data point and do not capture time-dependent covariates. Between changes in building data and economic trends, however, we think our predictions are impacted by changes in these trends, and so we need to adapt Cox's model to take into account these time-dependent values. In order to account for these values, we set twelve yearly observation points (from 2010 to 2021), where each business observed has a start date before the observation date. If it exists after the end of the observation period, we mark that record as censored. To these observations, we apply the

[14] https://scikit-survival.readthedocs.io/en/stable/user_guide/00-introduction.html

model, which fits a line to the features and returns a set of coefficients, which serve to estimate the effect of each covariate.

Although we are able to use time-dependent variables as features in our model, we still struggle to use them in predictions for many years into the future, as changes in the features are often unpredictable. As such, a time-dependent survival model with unpredictable time-dependent variables relies on the "martingale theory, a mathematical construct which arose out of the study of games of chance. A key underlying condition for a martingale-like game is that present actions depend only on the past." This is to say that our understanding of the future of a time-dependent feature depends only on the past/present values of that feature.[15]

Once the feature set is property formatted. Cox's model fits a line to the data, and assigns coefficient values, which allows us to understand the impact of the marginal increase of each covariate. The default method, Breslow's method, assumes that each event at a given time is distinct. When this number of ties is low, Breslow's method will often return similar coefficient values to Efron's method, which works better with a high number of ties.[16]

Random Survival Forest. Although not a linear model, we can train a random survival forest model on the same dataset as the Cox proportional hazards model. Unlike the Cox proportional hazards model, however, random survival forest models allow risk estimates to vary over time. This flexibility allows for the understanding that the risk factors that affect business survival are not constant year to year. We must still use our understanding of the martingale theory to use past data to make assumptions about unpredictable features. Also unlike Cox's model, rather than a linear model, random survival forests fit individual de-correlated trees to bootstrap samples of the original data. Rather than selecting features by those that are most decisive of the independent variable, this model makes those selections randomly.[17]

Evaluation

Goal Seven: Evaluation.

Record linkage evaluation. There are nearly 50,000 retail businesses in New York City during a given year.[18] With a high turnover estimate of 20%, we may assume that we capture an additional 100,000 businesses over the course of ten years. Given the high estimate of 150,000 businesses captured, if we identify the linkage success rate of just 385 businesses, we can be 95% certain that our found linkage

[15] https://cran.r-project.org/web/packages/survival/vignettes/timedep.pdf
[16] https://scikit-survival.readthedocs.io/en/stable/api/generated/sksurv.linear_model.CoxPHSurvivalAnalysis.html
[17] https://scikit-survival.readthedocs.io/en/stable/user_guide/random-survival-forest.html
[18] https://www.osc.state.ny.us/reports/osdc/retail-sector-new-york-city-recent-trends-and-impact-covid-19#:~:text=In%202019%2C%20New%20York%20City's,sales%20to%20the%20City's%20economy.

success rate is within 5% of the true linkage success rate of the data.[19] In our evaluation of the probabilistic record linkage model, we took entries associated with a random sample of 385 businesses. Using our probabilistic model, we correctly linked 310 businesses across the dataset. This yields an 80% success rate, so we may be 95% sure that we have a linkage success rate between 75% and 85%.

We found that we could use the same comparison features to generate a simple deterministic model to link records across the dataset. We found that a deterministic approach to linkage captures 351 of the 385 businesses correctly. This yields a 91% success rate, so we may be 95% sure that we have a linkage success rate between 86% and 96%. We are also more than 95% confident that the deterministic approach performed better than the unsupervised probabilistic approach. Although this is likely a result of some sort of misspecification of the model, we proceeded to use the deterministic linkage approach.

Business type classification evaluation. We approached the evaluation for our business type determination using sentence embeddings in a similar manner to the linkage evaluation. We took a sample of 385 businesses, found them in our dataset, and decided if the business type was correctly specified for each business. Determining the correctness of these classifications is difficult, as it will often over-specify: a cafe may be classified as a bakery, an Asian fusion restaurant will likely be classified to a specific cuisine. We did not identify a successful classification one that was exactly correct. Given this criterion, we determined that of the 385 businesses, we correctly classified 189. This is a 49% success rate, so we may be 95% sure that we have a business type classification success rate between 44% and 54%. The correct classification of businesses to business types is especially useful insofar as we can use the types to generate predictions about each business. Given that only approximately one-half of the businesses were correctly classified, we may need to add more information about each business or remove unnecessary characters to make the classification a useful predictor.

Survival model prediction evaluation. We split the data into training and test data (a 70-30 split). We began by identifying three metrics by which we would compare models' predictions on the test data: accuracy, F1 score, and ROC AUC. The accuracy score is the percent of correctly classified businesses. That may seem an effective strategy for understanding prediction quality, but it does not serve as a very useful metric to understand prediction quality. Our data mostly shows that businesses survive into the following year. In fact, our data shows that 72.4% of businesses survive year to year, so the survival target variable consists mostly of ones. This is to say that if we had a prediction of all ones, we would see an accuracy score near 72% (given a naive model). Of course, the prediction accuracy on non-surviving businesses would be 0%, but that is not captured in the accuracy score.

The F1 score metric attempts to get around this issue. The F1 score is at best 1, and at worst 0. It can be calculated using the following equation: 2 * (precision * recall) / (precision + recall), where

[19] https://www.surveysystem.com/sscalc.htm

precision may be defined as (true positive) / (true positive + false positive) and recall may be defined as (true positive) / (true positive + false negative). As discussed above, a naive classifier may yield an accuracy score near 72%. A similar classifier would have a precision of .72, a recall of 1, and so an F1 score of .42, approximately. F1 score penalizes this naive approach.

The ROC AUC score goes one step further, to calculate the area under the receiver operating characteristic (ROC) curve. The ROC curve represents the true positive rate and false positive rate of a classifier, across a range of thresholds. The curve compares the Sensitivity vs (1-Specificity), and at any one point on the ROC curve, we may calculate an F score. Although the calculation for ROC AUC generally performs better for binary data, when there is an imbalance between positive and negative samples, as we have in our data, the F1 score performs better. This is because ROC AUC averages over all possible thresholds, which overlooks the possibility of imbalanced data. Although each metric is important in its own right, we specifically look at the F1 score in our comparison of each model.[20]

Naive classifiers. One naive classifier predicted that each business would close. It had an accuracy score of .7256, an F1 score of .4205, an ROC AUC score of .5, a precision score of 0 and a recall score of 0. Another naive classifier predicted that each business would survive. It had an accuracy score of .2744, an F1 score of .2153, an ROC AUC score of .5, a precision score of .7256 and a recall score of 1.

MLP Classification. Based on a parameter grid, we resolved to use LBFGS as the solver, with an alpha value of 1e-5, and a hidden layer size of (5,2). Although the classifier resulted in an accuracy score of .7115, the F1 and ROC AUC scores performed worse than random selection. The F1 score was .4157, and the ROC AUC score was .5. This classifier had a precision score of .7254 and a recall score of 1.

C-Support Vector Classification (SVM SVC). Based on a parameter grid, we resolved to use an RBF kernel. The support vector classifier performed similarly to the MLP classifier. While the accuracy was .7114, the F1 score was .4166 and the ROC AUC score was .5002. Similar to the MLP classifier, we did not exceed the performance of a random classifier. This classifier had a precision score of .7254 and a recall score of 1.

Stochastic Gradient Descent Classification. Based on a parameter grid, we resolved to use a hinge loss function, with an l2 penalty. The SGD classifier showed slightly more promising results than MLP and SVM SVC. Although the accuracy was low, with a value of .5667, the F1 and ROC AUC scores were higher than those of the previous two classifiers, at .5192 and .5282, respectively. Adding polynomial transformations of the features did not markedly increase these scores. This classifier had a precision score of .7255 and a recall score of .9998.

---

[20] https://stackoverflow.com/questions/44172162/f1-score-vs-roc-auc#:~:text=F1%20score%20is%20applicable%20for,and%20recall%20should%20be%20high.

Decision Tree Classification. Based on a parameter grid, we resolved to use the Gini criterion with a "best" splitter. The decision tree classifier performed better than the previous classifiers by all three metrics. With an accuracy score of .7343, an F1 score of .6781, and a ROC AUC score of .6783. This classifier had a precision score of .8138 and a recall score of .8027. The tree first split on Months Active, then, on one branch, Unemployment Rate, and Inflation Rate, as shown in Appendix B.

Random Forest Classification. Based on a parameter grid, we resolved to use the Gini criterion with a max depth of 19. Although based on decision tree classification, the random forest classification performed worse than the above classifier, with an accuracy score of .7514, an F1 score of .5754, and a ROC AUC score of .5829. This classifier had a precision score of .7509 and a recall score of .9832. Given that the random forest classifier uses multiple decision trees on subsets of the data, we were surprised to see a lower F1 score. This may be a result of limitations in the range of our parameter grid.

Cox's Proportional Hazards Model. Given the linear nature of the model, we are able to extract coefficient values for each covariate, which measures the impact each covariate has on the survival term of each business. While certain building classes seem to have an impact on business survival, unit changes in inflation rate and job growth seem to be drivers of survival. Meanwhile, GCP and unemployment rate have negative coefficient values. Coefficient values may be found in Appendix C.

Harrell's concordance index is often used to score the proportional hazards model, by determining the correlation between the determined risk scores and observed time points (where risk score is higher for scores with lower predicted survival time). Our initial model saw a Harrell's concordance index value of .5676. This score allows us to understand the discriminatory power of the features in the model Although easy to compute, Harrell's concordance index is biased upward as censoring increases and overestimates the performance of models with high censoring. Given that we make twelve observations (one per year), our dataset contains a high amount of censoring, and will likely overestimate the performance of our model. It also does not measure well the performance of a model given a specific time range.

Uno's concordance index, based on the inverse probability of censoring weighting, limits the distorting effect of censoring on model scoring. Although Uno's concordance index actually underestimates the performance, it is significantly more stable than Harrell's concordance index, given increases in censoring.

We may also use the time-dependent ROC AUC measure, which, similar to Uno's estimator, is based on the inverse probability of censoring weighting. Unlike Harrell's estimator, this is a useful measure of performance given a certain time frame. As discussed above, this measure works especially well on binary classification, so we classify our businesses into two categories: those that survive beyond time t, and those that do not. We can select a time t, such that there is a balance between businesses that

do and do not survive before and after t; this balance makes ROC AUC a better measure of success for the model.

Given a list of time points, the Scikit Learn library offers an implementation of a dynamic ROC, which takes into account survival before and after each time t, in the given set of times. In addition to understanding the discriminatory power of each feature, we may use this time-dependent ROC to understand the prediction performance of our model. While the mean ROC AUC score is .5892, the AUC scores over time are listed below in Appendix D.

Although not an exact measure of the predictive capacity of a model, and only measurable for models that estimate a survival function, the time-dependent Brier Score measures not only discriminatory power (as a concordance index), but also the calibration of a model. A well-calibrated model would observe survival that agrees with the frequency of its predicted risk.[21] The Cox proportional hazards model achieves a Brier Score of .6228.

Random Survival Forest. Similar to the Cox proportional hazards model, in addition to understanding the discriminatory power of each feature, we may use this time-dependent ROC to understand the prediction performance of our model. While the mean ROC AUC score is .6208, the AUC scores over time are listed below in Appendix E. Again, we may measure the time-dependent Brier Score measures not only discriminatory power (as a concordance index), but also the calibration of a model. The random survival forest model achieves a Brier Score of .6451. Although both perform better than a random model (which would likely have a Brier Score near .5), compared to the Cox proportional hazard model, the random survival forest model had a better ROC AUC score and Brier Score.

Although limited in its predictive capacity, the decision tree seemed an effective model for predicting business survival in the following year, with an F1 score of .6781, and a ROC AUC score of .6783. The random survival forest model seemed to show a lower value for ROC AUC. This ROC AUC score, however, was measured on predictions beyond just the following year, and thus is not comparable. The survival model uses censoring and past survival to generate a more nuanced understanding of the future, and is useful in its own right.

Goal Eight: Visualization.

We wanted to visualize our data, so we can see which businesses are surviving, and where. We considered the possibility of stating the mean median and mode of businesses that survived, divided by different boroughs or neighborhoods. Thanks to pre-existing implementations of tools like Mapbox, we were easily able to create a display of the businesses on a map of New York City. We not only wanted to display building vacancy, but an understanding of our existing businesses. Finally, we wanted to add an

[21] https://scikit-survival.readthedocs.io/en/stable/user_guide/evaluating-survival-models.html

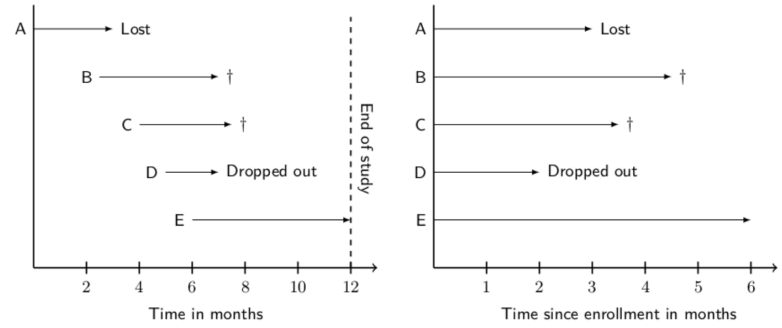
ability to view which current businesses we expect to fail to survive in the following year. This makes it easy to see patterns in survival and understand key traits of each business and building. Images from the visualization are shown in Appendix F.

Appendix

Link to video:

https://drive.google.com/file/d/1YLBE6aoWibAqHLqOQz5HAvM2aooM5_Xx/view?usp=sharing

(A)



(B)

```
|--- Months Active >  11.50
|   |--- Unemployment Rate, SA - NYC <= 0.06
|   |   |--- Months Active <= 17.50
|   |   |   |--- Inflation Rate, NSA - USA <= 0.02
|   |   |   |   |--- Inflation Rate, NSA - NYC <= 0.02
|   |   |   |   |   |--- bldgfront <= 246.50
|   |   |   |   |   |   |--- truncated branch of depth 15
|   |   |   |   |   |--- bldgfront >  246.50
|   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |--- Inflation Rate, NSA - NYC >  0.02
|   |   |   |   |   |--- lotarea <= 6716.00
|   |   |   |   |   |   |--- truncated branch of depth 8
|   |   |   |   |   |--- lotarea >  6716.00
|   |   |   |   |   |   |--- truncated branch of depth 7
```

* Added after the presentation following a question by Professor Selman

(C)

bldgclass=Z1	1.032099e+00
bldgclass=Z2	1.473641e+00
bldgclass=Z3	1.447877e+00
bldgclass=Z4	8.477341e-01
bldgclass=Z8	1.412394e+00
bldgclass=Z9	6.536553e-01
lotarea	-2.183483e-10
garagearea	-2.824758e-08
strgearea	8.090647e-08
numfloors	3.180696e-03
unitstotal	2.085759e-05
bldgdepth	-1.997813e-05
yearbuilt	-5.252067e-05
builtfar	-5.052990e-05
GCP (NYC)	-7.064690e-03
Payroll-Jobs Growth, SAAR - NYC	2.570808e+00
Inflation Rate, NSA - NYC	4.324422e+00
Unemployment Rate, SA - NYC	-2.679454e+00

(D)

```
[0.5120703  0.55425557 0.54642726 0.54151516 0.53854325 0.5402588
 0.54244017 0.54231709 0.54279186 0.54070118 0.59591941 0.59543025
 0.59482486 0.59445149 0.59390656 0.59314383 0.59224471 0.59771064
 0.59752896 0.59698829 0.59681465 0.59718155 0.59585119 0.59537223
 0.59434147 0.59419773 0.59326492 0.59338654 0.59351565 0.59354881
 0.59357098]
0.589160079571479
```

Cox proportional hazard model ROC AUC scores measured for each month, over three years.

(E)

```
[0.61715724 0.60952515 0.59663358 0.60329186 0.59783486 0.59264557
 0.60111442 0.60103771 0.6039998  0.59823898 0.62710435 0.62109849
 0.61714501 0.61495876 0.61226576 0.61180016 0.61079823 0.6047301
 0.60327266 0.6015725  0.60138187 0.59903539 0.59736957 0.59560898
 0.5940529  0.59330231 0.5917129  0.59116819 0.5901231  0.58737547
 0.58587082]
0.6208198770222908
```

Random survival forest model ROC AUC scores measured for each month, over three years.

(F)