Greg Vargas
2/20/21

Final Report:

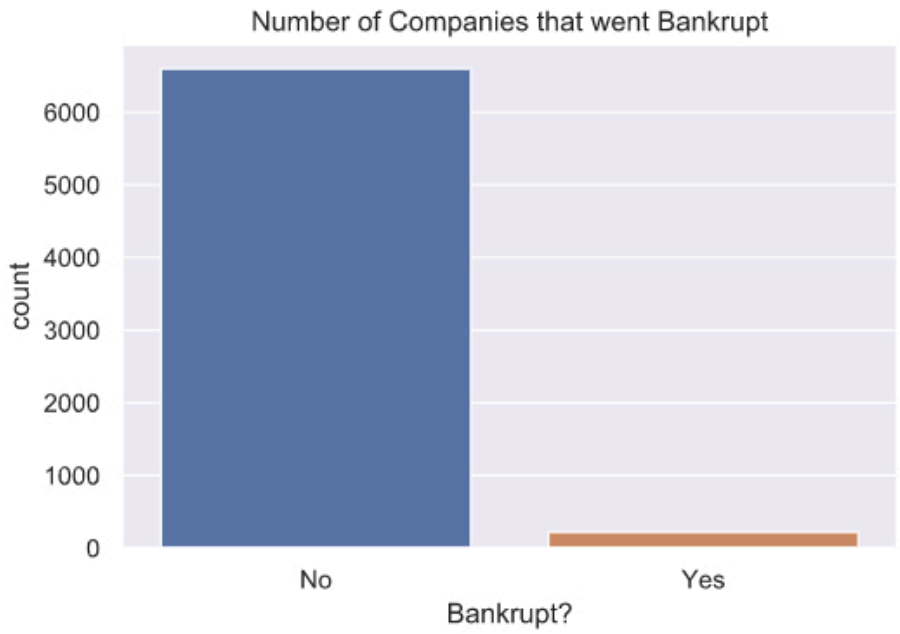
Predicting Company Bankruptcy

Problem Statement

When a company is unable to repay their outstanding debts, they can begin the legal proceedings of filing for bankruptcy. All of the company's assets are measured and evaluated, and they may be used to repay a portion of the outstanding debt. Bankruptcy allows the business a chance to start fresh by forgiving those debts and to give creditors a chance to recoup some of their investment. While in theory bankruptcy helps give money back to investors gives the company a second chance to gain access to credit, it would obviously be preferred to avoid bankruptcy. By looking at common business measures, such as cash flow and the accounts receivable turnover, can a machine learning model predict if a company will go bankrupt? To test this, we will be looking at data from the Taiwan Economic Journal for the years 1999 to 2009. This dataset includes 95 common business measures and nearly 7000 examples of companies. According to the "An Overview of Insolvency Proceedings in Asia", an issue in the *Annual Survey of International & Comparative Law,* Taiwan's bankruptcy laws are heavily influenced by archaic Chinese models and values. In Taiwan, one's "hsin-yung" (business reputation) is so important, many people will commit suiced rather than file bankruptcy. Being able to predict future indicators of bankruptcy, and the features most heavily correlated with it, will help prevent tragedies such as this, as well as give investors and companies an early warning system to focus on important business metrics to improve and hopefully prevent bankruptcy.

Data Wrangling

This dataset was found on Kaggle, but is originally hosted on the UCI Machine Learning Repository. The data was hosted as a CSV file, so it was downloaded and then put into a pandas dataframe. The data was inspected for missing values, but none were found. To make sure that any null values weren't encoded as something like -1, the spread of the data was looked at and nothing was found to be suspect. One column was originally dropped, as all values were 1, and without any exploratory data analysis we can see that the column was not adding any kind of useful information. In this initial data wrangling, it was found that the ratio of companies labeled bankrupt/not bankrupt was very unbalanced. There were 6599 instances of companies not going bankrupt, and only 220 instances of bankruptcy.




```
0    6599
1     220
Name: Bankrupt?, dtype: int64
```

Exploratory Data Analysis

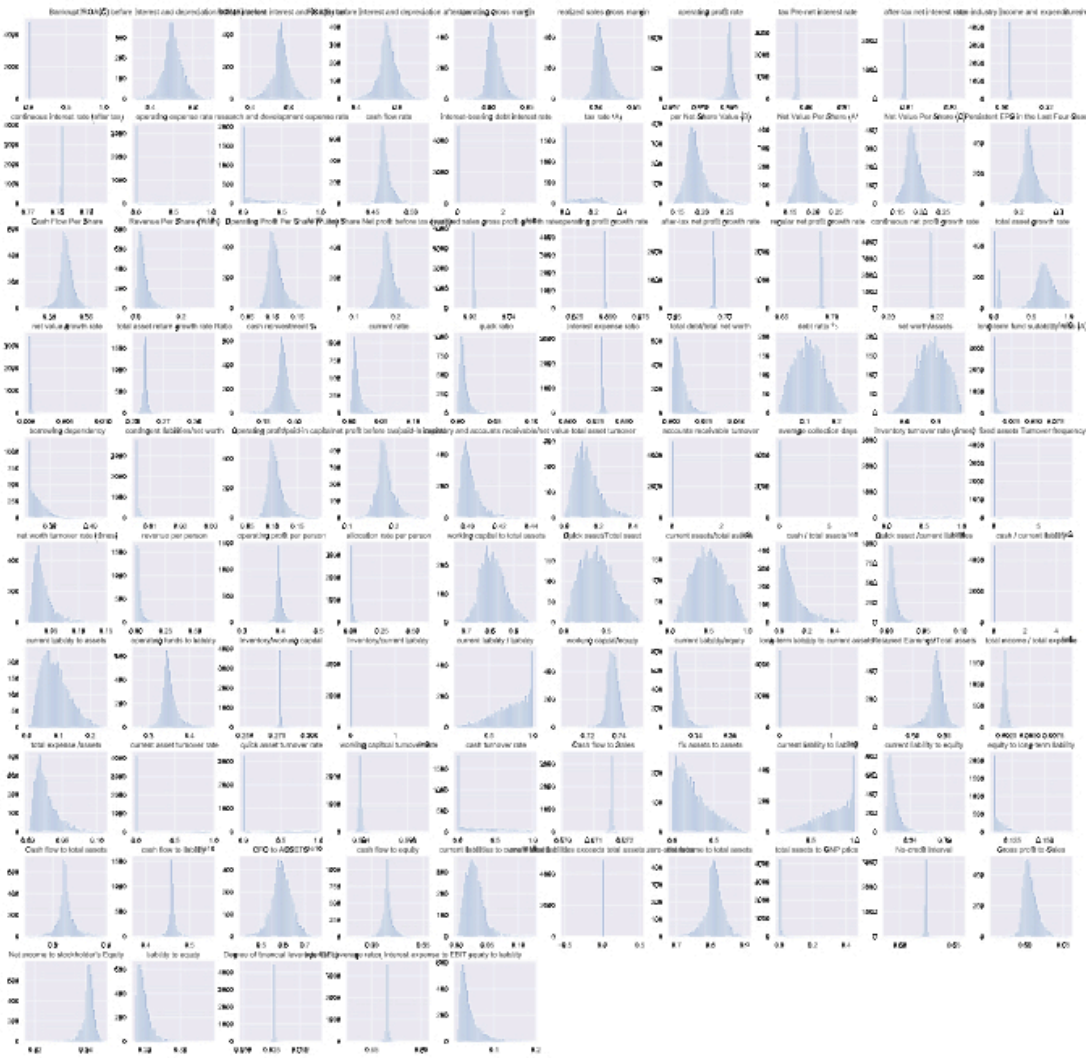
As we saw in the data wrangling phase, there was a column found that only had the value 1 for all instances. This feature had no variance, and therefore was not contributing any meaningful information. To ensure that there were not any other columns like that, but with a value that is not as easy to spot, the dataset was fitted with a Variance Threshold algorithm that would remove any extraneous features. All of the remaining columns had variance, so there were no other features to remove. Any duplicate data was also searched for, but none was found.

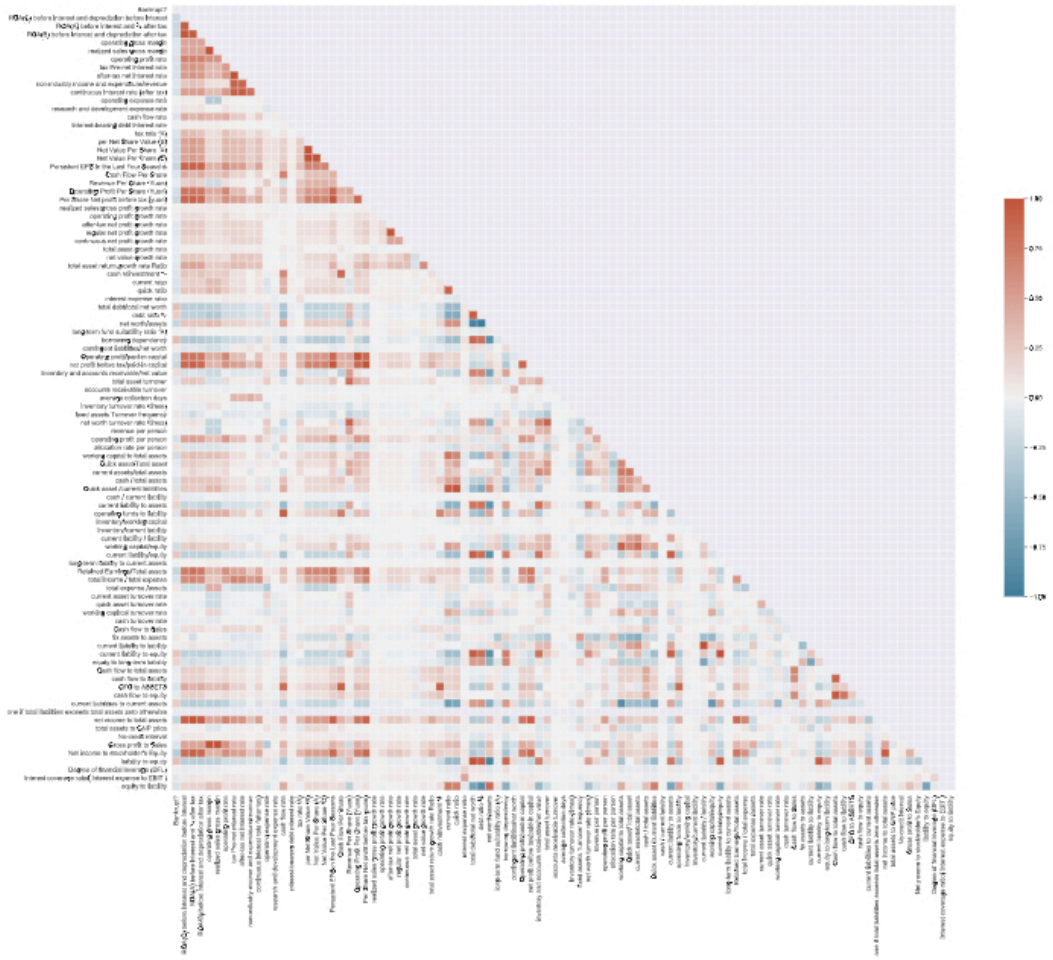
Now that we had a dataset with no duplicates, missing data, or useless features, it was time to look at the distribution of the dataset. The initial histogram showed a few features with a normal distribution, but the rest looked like all of the data was in one bin. To have a usable histogram, the bin number was then increased to 50 to allow the single peak to spread out. This provided more features that looked to have normal distribution, but the location of the peak implied that there were some extreme values that were the initial cause of the binning problem. Before looking into those extreme values, a correlation heatmap was made to see what features correlated with the target feature. There did not seem to be any strong correlations with the target feature, but the extreme values may be hiding some of the relationships in the data.

To begin the search for outliers, the scipy z-score function was applied to each instance in its column. By looking at the z-score, we can see how many standard deviations each instance is from the mean of that specific feature. The highest z-score in this dataset was nearly 83 standard deviations from the mean. A z-score of +/- 3 was then used as a threshold for outliers, and the dataset filtered using this threshold. Out of the original 6819 instances, 1910 were removed because they contained outliers. Of

these 1910 outliers, 1809 were of the majority class(not bankrupt) and represented 27% reduction in the majority class instances. The remaining 101 outliers were in the minority class(bankrupt), and represented a reduction of 46 instances of bankrupt companies.

After removal of the outliers, the correlation heatmap and histogram were created again. The new correlation map showed stronger correlations between the various features, proving the outliers were obscuring the underlying relationships in the data. The new histogram also gave a better understanding of the data, as the normal distribution of more features was uncovered.





Preprocessing Data

After the exploratory data analysis and feature selection, the data had 4909 instances and 95 features. Of the instances, 4790 companies had not gone bankrupt, and 119 had gone bankrupt. The problem with working with unbalanced is that most machine learning techniques will ignore and perform poorly on the minority class. To handle this, Synthetic Minority Oversampling Technique (SMOTE) was used to balance the minority class. SMOTE synthesizes new examples of the minority class to bring the number of samples to the same number as the majority class. After using SMOTE, there were an equal number of 4790 bankrupt and non-bankrupt company instances. To create a baseline and understand the benefits of using SMOTE on the dataset, the original data and SMOTE data were both split into train and test sets to run through machine learning models. The each data set was split into 67% training data and 33% test data and stratified by the class labels to ensure each split had equal proportions of the bankrupt class.

Model Selection/ Further Considerations

To create a baseline and understand the benefits of SMOTE, logistic regression was run on both datasets and compared against a dummy classifier. Even though it was a baseline, a grid search for hyperparameter tuning was used to ensure the best baseline before a more exhaustive search. This grid search was done using 5-fold cross validation and by tuning 4 hyperparameters. These hyperparameters included the penalty term, the C ratio, whether a warm start was used, and the l1 ratio. The penalty is the regularization penalty used, and l2, elastic net, and no regularization were tested. The C value was a range from 0 to 1, and represents the strength of regularization. Warm start is if the model uses the previous call to fit as initialization or to erase the previous solution. True and False were used to see the previous solution would increase the performance of the models.

The cross validation mean accuracy score of the base data was ~98% and ~93% for the SMOTE data. These would be impressive results, but the dummy classifier also had an accuracy score of 98%. With the dummy model just using the “Most Frequent” approach, which by just guessing the majority class, it can achieve an accuracy of 98%. The SMOTE also seemed to do worse. Because of the nature of this problem, accuracy is not the best metric to understand the performance of the models, and looking at other scoring metrics shows this. To best capture the efficacy of a binary classifier, balanced accuracy is the best metric because it shows the combined accuracy of both classes. Balanced accuracy is defined as the average of recall obtained on each class. Using a classification report, we see that the balanced accuracy of the original data is 61% and 91% for the SMOTE data. We can see that the SMOTE data has a higher recall score in regards to both classes. In regards to bankruptcy, it is better to incorrectly classify a company as bankrupt than to miss a classification of a company that went bankrupt. This is because incorrectly labeling a company allows for additional scrutiny, but missing a classification means the opportunity was missed to lessen the blow of bankruptcy. With that in mind, the best metric to prevent missing a classification is recall to prevent false negatives. To compare other models, we will use recall to compare to the baseline.

Unbalanced Classification Report

	precision	recall	f1-score	support
0	0.98	0.99	0.99	1581
1	0.45	0.23	0.31	39
accuracy			0.97	1620
macro avg	0.72	0.61	0.65	1620
weighted avg	0.97	0.97	0.97	1620

SMOTE Classification report

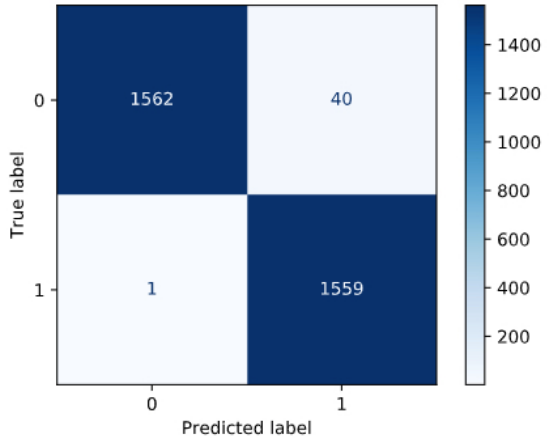
	precision	recall	f1-score	support
0	0.95	0.88	0.91	1602
1	0.88	0.95	0.91	1560
accuracy			0.91	3162
macro avg	0.91	0.91	0.91	3162
weighted avg	0.91	0.91	0.91	3162

To find the next models to try and beat the baseline, a library called lazy predict with an ability to use a lazy classifier and run multiple classifiers at once. A custom metric is allowed to be added, so a recall score was added for the LazyClassifier. This library ran 27 classifiers, such as Random Forest, XG Boost, SVC, and KNeighbors.

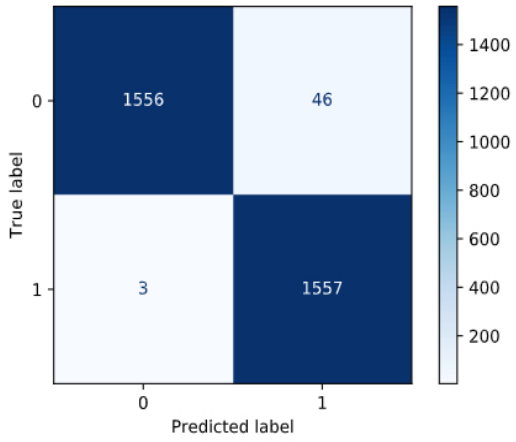
```
100%|████████| 29/29 [00:39<00:00,  1.38s/it]                          Accuracy  Balanced Accuracy  ROC AUC  F1 Score
\
Model
ExtraTreesClassifier               0.99               0.99     0.99      0.99
LGBMClassifier                     0.99               0.99     0.99      0.99
XGBClassifier                      0.99               0.99     0.99      0.99
RandomForestClassifier             0.98               0.98     0.98      0.98
BaggingClassifier                  0.98               0.98     0.98      0.98
LabelPropagation                   0.98               0.98     0.98      0.98
LabelSpreading                     0.98               0.98     0.98      0.98
SVC                                0.98               0.98     0.98      0.98
DecisionTreeClassifier             0.96               0.96     0.96      0.96
AdaBoostClassifier                 0.96               0.96     0.96      0.96
ExtraTreeClassifier                0.96               0.96     0.96      0.96
QuadraticDiscriminantAnalysis      0.95               0.95     0.95      0.95
KNeighborsClassifier               0.94               0.94     0.94      0.94
LinearSVC                          0.92               0.92     0.92      0.92
CalibratedClassifierCV             0.92               0.92     0.92      0.92
LogisticRegression                 0.91               0.91     0.91      0.91
SGDClassifier                      0.91               0.91     0.91      0.91
PassiveAggressiveClassifier        0.90               0.90     0.90      0.90
LinearDiscriminantAnalysis         0.90               0.90     0.90      0.90
RidgeClassifierCV                  0.90               0.90     0.90      0.90
RidgeClassifier                    0.89               0.90     0.90      0.89
NuSVC                              0.89               0.89     0.89      0.89
Perceptron                         0.85               0.85     0.85      0.85
NearestCentroid                    0.85               0.85     0.85      0.85
BernoulliNB                        0.83               0.83     0.83      0.83
GaussianNB                         0.77               0.77     0.77      0.76
DummyClassifier                    0.50               0.50     0.50      0.50

                           recall_score  Time Taken
Model
ExtraTreesClassifier               1.00        0.85
LGBMClassifier                     1.00        1.27
XGBClassifier                      1.00        1.63
RandomForestClassifier             1.00        4.22
BaggingClassifier                  0.99        4.78
LabelPropagation                   1.00        3.29
LabelSpreading                     1.00        3.84
```

The ExtraTrees, LGBM, XGBoost, and RandomForest Classifiers had the best recall scores, so they were chosen to be investigated further. The models were run without any grid search, to see which one performed best before running a grid search to further tune the model. However, before tuning any hyperparameters, the models were able to achieve recall scores of 99% and 98%. Therefore, any of the 4 models from the Lazy Classifier can be used to correctly predict when a company will go bankrupt.

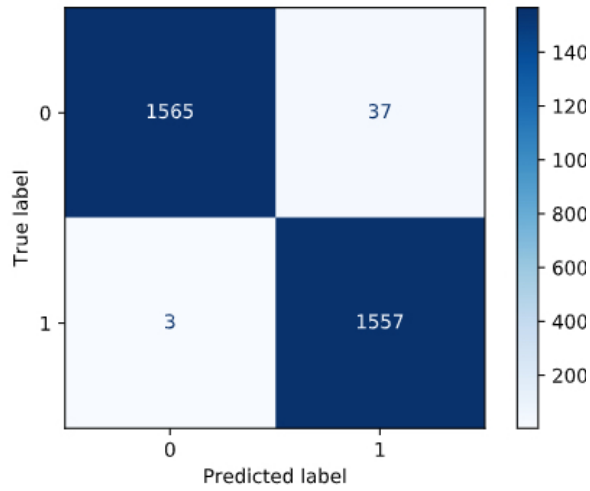
ExtraTreeClassifier



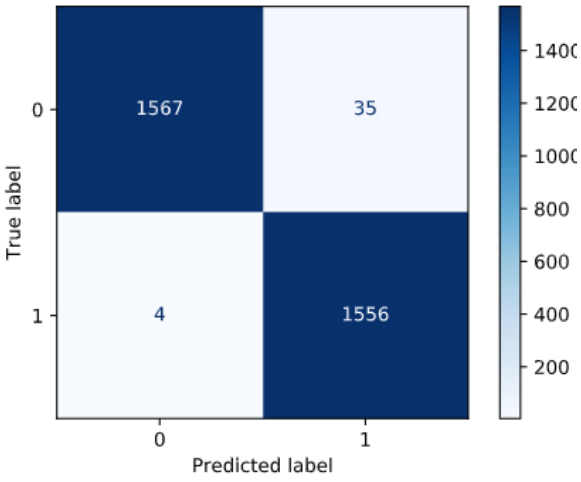
RandomForestClassifier



LGBMClassifier



XGBClassifier



Takeaways/Further Research

` When working with unbalanced dataset, the importance of highlighting the minority class is apparent. By synthetically generating the minority class samples, each model tested was able to prove that it could find the major features that lead to a company going bankrupt. While using Logistic Regression, it was unable to perform better than just guessing that a company had not gone bankrupt. The LazyClassifier provided a great starting point, and each of the top models performed well. There are a multitude of models able to correctly predict if a company would go bankrupt.

To see if the same performance could be attained with less computational cost, feature reduction was performed on the original dataset. Using sklearn's SelectFromModel with a RandomForest Classifier, there were 60 features removed from the data set. The same preprocessing was performed on the data, followed by classification by the same 4 chosen models. The XGBoost Classifier was able to achieve the same recall score, but improperly labeled more companies as being bankrupt. The stakeholders using this model would need to decide if the reduction in

computational cost is worth the extra work of verifying the additional false positive bankrupt companies.

Selected Features

```
' ROA(C) before interest and depreciation before interest',
' ROA(B) before interest and depreciation after tax',
' non-industry income and expenditure/revenue',
' continuous interest rate (after tax)', ' operating expense rate',
' interest-bearing debt interest rate', ' per Net Share Value (B)',
' Net Value Per Share (C)', ' Persistent EPS in the Last Four Seasons',
' Per Share Net profit before tax (yuan)', ' net value growth rate',
' quick ratio', ' interest expense ratio',
' total debt/total net worth', ' debt ratio %', ' net worth/assets',
' borrowing dependency', ' net profit before tax/paid-in capital',
' accounts receivable turnover', ' average collection days',
' fixed assets Turnover frequency', ' working capital to total assets',
'cash / total assets', 'cash / current liability',
'Inventory/working capital', 'working capital/equity',
'current liability/equity', 'net income to total assets',
'total assets to GNP price', 'Net income to stockholder's Equity',
'liability to equity', 'Degree of financial leverage (DFL)',
'Interest coverage ratio( Interest expense to EBIT )',
'equity to liability'],
```