Stock Data Analysis

By: Sasha Roberts, Troy Zhongyi Zhang, Parnian Rao, Yuyang Zhang

BUSINESS PROBLEM

The purpose of our project is to look at the historical information of the S&P 500 companies, and build models to predict profitability and bankruptcy risk given the required information. We used a mix of clustering techniques, classification, regression and transfer learning, to explore the data and build models that can help us predict profitability and detect companies that might have bankruptcy risk.

DATASET AND FEATURES

For our project, we used the Kaggle dataset (www.kaggle.com/dgawlik/nyse) which had the S&P 500 companies' historical prices and fundamental data. There were four separate csv's in the dataset and we decided to use the following three: Fundamentals, Spilt Prices and Securities.

Also, we used another Kaggle dataset "bankruptcy_Train.csv" (https://www.kaggle.com/c/companies-bankruptcy-forecast) which contains companies' operating status and financial ratios information on emerging markets around the world. We used this data set for Transfer Learning(explained later).

Data Preparation

In order to apply different techniques learned in class, the first thing we had to do was to prepare the data. We merged the three files: Fundamentals and Split Prices on data column, and the securities on the ticker symbol. Since there were some dates that were present in the Split Prices file, but not in the fundamentals file, it slightly reduced the size of our dataset.

Feature Engineering:

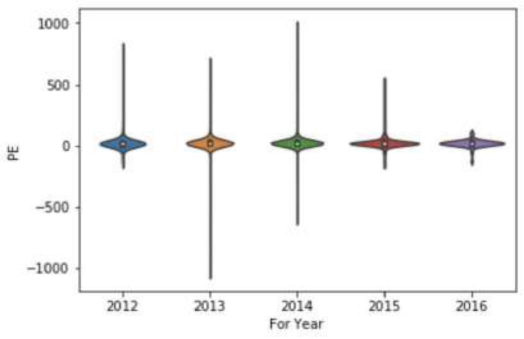
There were a few missing values for Cash Ratio, Current Ratio, Quick Ratio and Shares Outstanding. We decided to use KNN to fill in the missing values of Cash Ratio, Current Ratio and Quick Ratio. To apply KNN, we used the Impyute package Fast KNN. Fast KNN imputes an array with a basic mean and then use the resulting complete array to construct a K-Dimensional Tree; the K-Dimensional Tree is used to compute the nearest neighbors applying Euclidean distance. For Shares Outstanding, we filled the missing values with zeros.

We added three new columns to the dataset, P/E ratio, z-score(added when regression was done), and trend because we are going to use these columns for modelling. We calculated the P/E ratio and z-score based on the information already available in the dataset. For the trend column, we looked at if the Price-to-Earnings (PE) is positive, the trend would be 1 and if PE is negative, trend would be 0. We also dropped a few columns that we thought would not affect our modeling, like Headquarters, SEC filing and Date Added, and categorized GICS Sector and GICS Sub Industry. There were some missing values in the "For Year" column, which we filled by extracting the Year from the Date column.

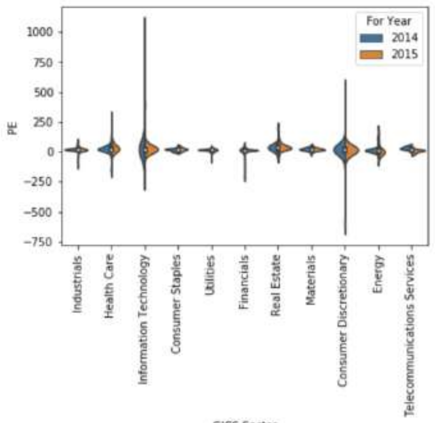
In order to apply the bankruptcy model trained by bankruptcy_Train file to Fundamental file, we need to make sure the features in both files are the same. So we dropped the columns Attr5 , Attr11, Attr21, Attr24, Attr25, Attr27 , Attr29, Attr33, Attr34, Attr35 , Attr38, Attr39, Attr43, Attr54, Attr55 and Attr58 in bankruptcy_Train file since they are not be able to built using columns in Fundamental file. Also, we added new columns Attr1 to Attr 64 (except for the dropped attributes) to Fundamental dataset based on the formulas behind each attributes. For example, "Attr1" equals to "Net profit" column divided by "Total assets" column. bankruptcy_Train file since they are not be able to built using columns in Fundamental file.

Exploratory Analysis

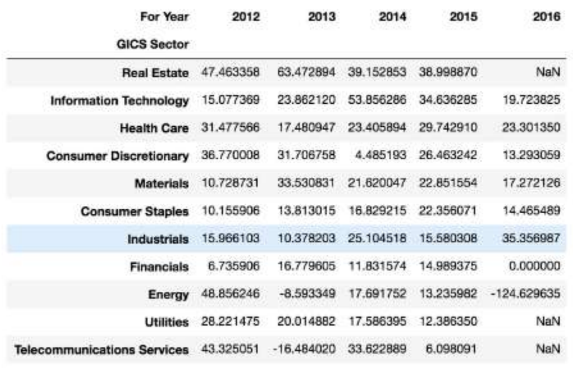
Before we performed any of the supervised or unsupervised learning techniques, we wanted to explore the data of how the companies are performing each year, and how the different sectors and sub industries are distributed based on the PE ratio.
Here are some of the graphs that helped us with understanding the data:



PE ratio by Year



PE ratio for each sector(for 2014&2015)



For Year	2012	2013	2014	2015	2016
GICS Sector					
Real Estate	47.463358	63.472894	39.152853	38.998870	NaN
Information Technology	15.077369	23.862120	53.856286	34.636285	19.723825
Health Care	31.477566	17.480947	23.405894	29.742910	23.301350
Consumer Discretionary	36.770008	31.706758	4.485193	26.463242	13.293059
Materials	10.728731	33.530831	21.620047	22.851554	17.272126
Consumer Staples	10.155906	13.813015	16.829215	22.356071	14.465489
Industrials	15.966103	10.378203	25.104518	15.580308	35.356987
Financials	6.735906	16.779605	11.831574	14.989375	0.000000
Energy	48.856246	-8.593349	17.691752	13.235982	-124.629635
Utilities	28.221475	20.014882	17.586395	12.386350	NaN
Telecommunications Services	43.325051	-16.484020	33.622889	6.098091	NaN

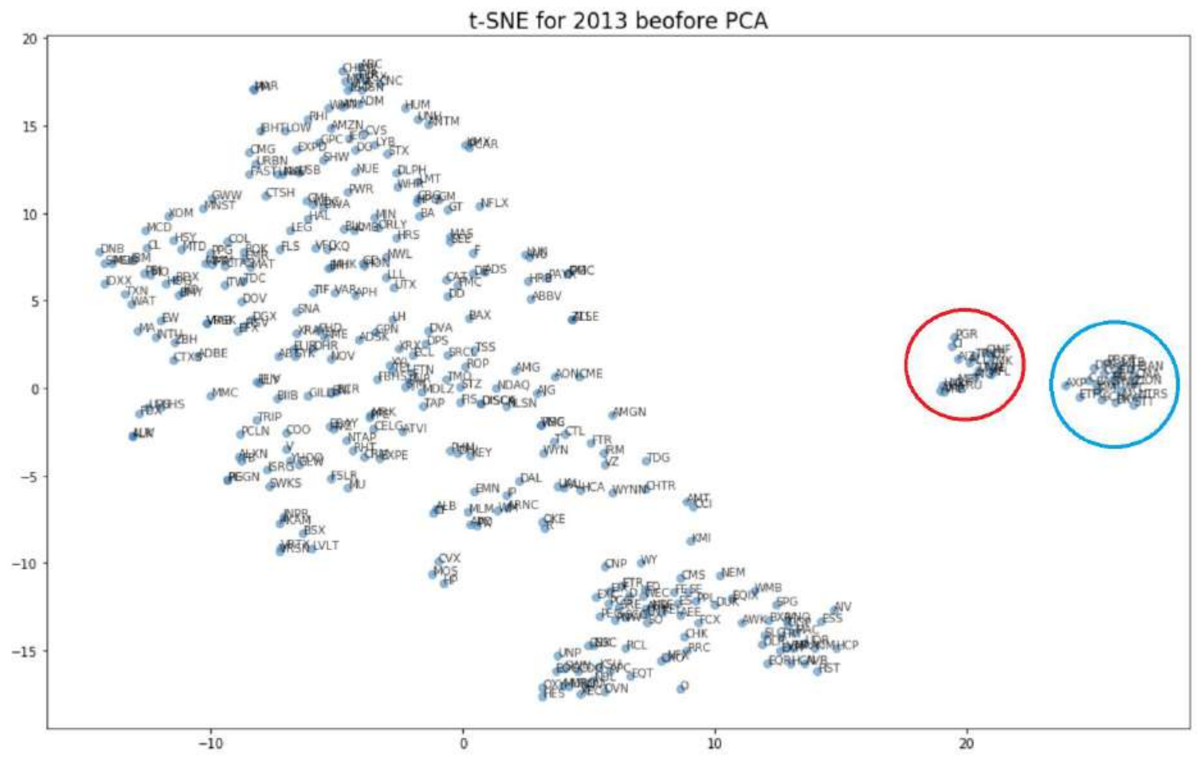
The table shows the PE ratio for each sector, by each year. We can see that Real Estate has the highest PE ratio in 2016, followed by IT and Healthcare.

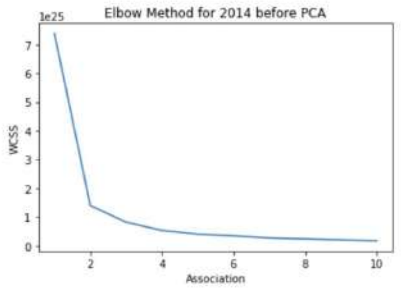
CLUSTERING

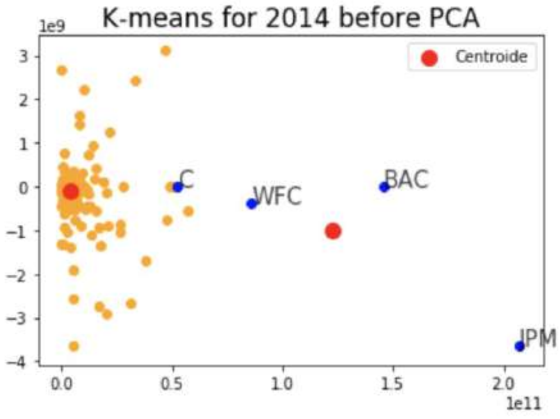
Clustering will group companies in our dataset according to their comprehensive similarities. There were abundant and complicated features in our NYSE dataset. We cannot subjectively filter out features. Since we could not completely read these whole more than 80 columns of data, we needed to process the first step analysis as clustering to treat our dataset as unlabeled data. The clustering algorithm is capable of bringing us intuitive and reasonable visualization and can handle unlabeled data well.

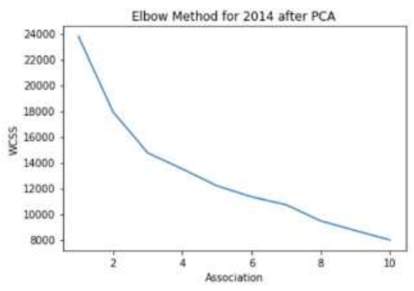
There were 85 columns in total after our first step cleaning. We added another column called "trend" to show these companies either were potentially profitable or had relatively poor market performances. During cluster, we would not need the trend included since it was not an original feature. There were 1357 data rows, but most of them were repeated companies with different years from 2012 to 2016. The first step we did was to separate our dataset to different years because it didn't make sense to plot all the years with the same company in one plot. Our target was to compare the financial conditions of different companies. We retained 20 features after applying PCA by filtering the top twenty features with the highest variances. These 20 features added up to 83.5% of total variances. More than 80% of importance should give us a rational result. (FYI, we named our new dataset as "reduced". However, "reduced" didn't include the year column. We have to re-add the "year" column in order to divide into years and delete it again.) Practically speaking, We had two data frames for each year: one for before PCA and one for after PCA. We removed redundant columns, and finally, we have 82 features before PCA and 20 features for after PCA dataset. Both datasets are ready for clustering.

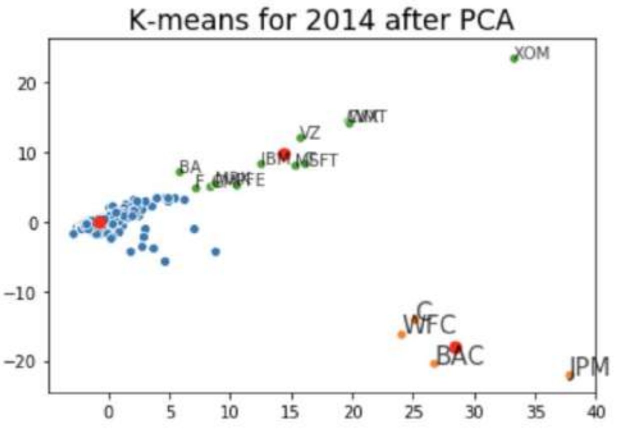
Visualized our data frame by operating t-SNE for each year, we annotated each point on the t-SNE plot with the companies symbol abbreviation list we extracted at the beginning so that we could have a more visualized diagram for every company in different years. Observed through the t-SNE plot for "before PCA", we found that there were two small groups of companies located far from all other enterprises every year except for 2016. Theses two bunches were very close but still had a tiny distance between each other. One small cluster included several banking companies like JP Morgan, American Express bank, State Street Bank, Bank of NY Mellon, BB&T Corporation, etc (in blue circle). Another cluster included various insurance companies like Progressive Corp, Cigna Corp, Assurant, Metlife Inc, Prudential Financial Inc, etc (in red circle). These two sets of companies may have some internal relationships and similarities in their financial situation. According to the after PCA t-SNE plots, we found that only in 2013 and 2015, the plots were divided into two clear clusters. However, according to our observations, each cluster contained different types of companies, for example, one cluster had Nike, Tyson Foods, Cayman Islands Stock Exchange, etc. The points graph were even more chaotic in 2012, 2014, and 2016. We couldn't summarize any relationship for the after PCA t-SNE plots.


t-SNE for 2013 beofore PCA


Elbow Method for 2014 before PCA
1e25
WCSS
Association

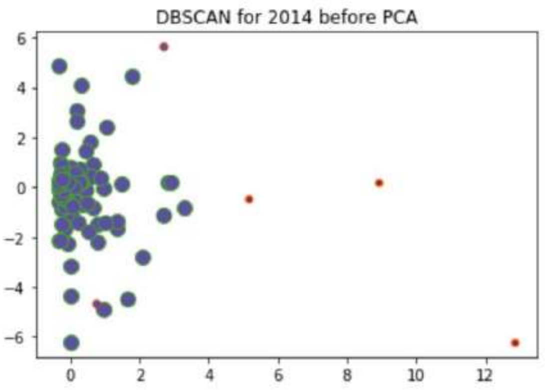

K-means for 2014 before PCA
1e9
1e11
Centroide
C
WFC
BAC
JPM

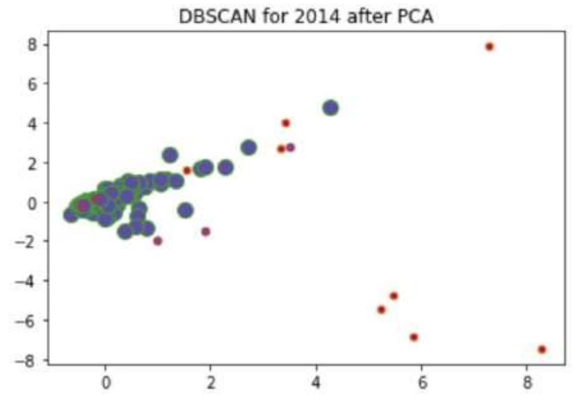

Elbow Method for 2014 after PCA
WCSS
Association

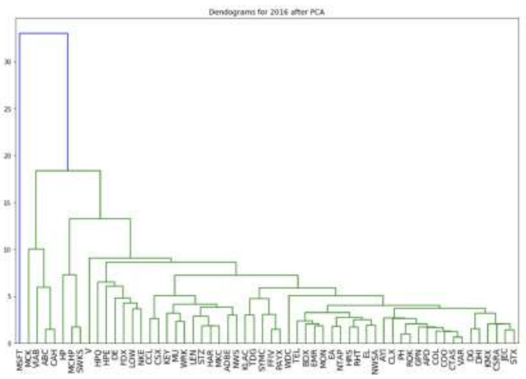

K-means for 2014 after PCA
XOM
VZ
IBM
MSFT
BA
C
WFC
BAC
JPM

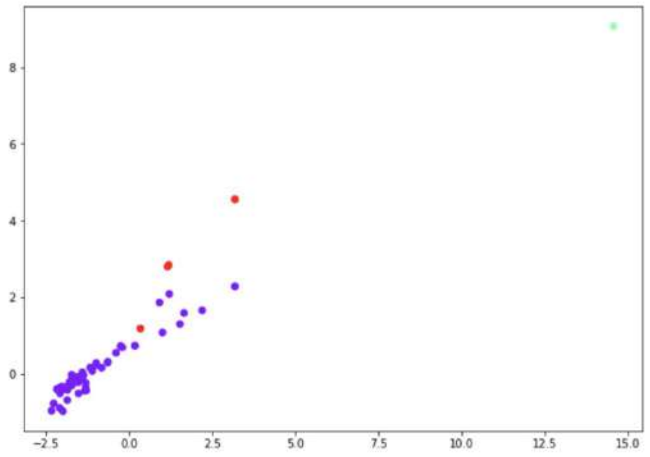
Tried to perform the best plot distribution, we actually did clusters for all 5 years before and after PCA with all the type of clustering method we learned through the class. Now we step forward to process the three types of clustering we learned, K-means, DBSCAN, and hierarchical clustering. We plotted almost 40 plots in total with changing different parameters. In K-means clustering, interpreted the results from the Elbow method plot, we found most of the marginal critical points located at k=2 for the plots before PCA and k = 3 or k = 2 for plots after PCA. Before PCA, the dots of different companies were not uniformly presented in our diagram. A big cluster of points located close to the y-axis with sporadic points distributed far from most of the points. The marginal point located at K=2 obviously obtained from Elbow method plot. We finally defined 2 clusters, which one big cluster included all the points that were close to the y-axis and another small cluster contained the points were far from the main cluster. These points could be concluded as outliers or noises. However, k-means clustering wouldn't separate outliers. All the points would be classified into the nearest cluster by comparing the Euclidean distance between the point and the cluster's centroid. However, for after-PCA plots, we nearly acquired a curve line from Elbow method plot. We would have to make a decision on choosing the most appropriate number of clusters according to our eye observations. We could make 2 clusters since most of the after-PCA clustering plots have 4 points at the right bottom corner and other points presented to be a linear line, but finally, we mostly chose 3 clusters for after-PCA plots since we also wanted to separate the "tail" (right side) of the linear shape dots plot into two clusters (one cluster for the 4 points in bottom right corner and two clusters for the linear shape distribution). We also extracted the symbol of the company for small clusters and annotated onto next to the points. According to DBSCAN, the cluster at the bottom right corner is counted as outliers. The four "special noises" companies are "JP Morgan, Wells Fargo, BOA, and Citibank". Surprisingly, we found our outliers are all banking corporations. Our clustering might make sense since all four banks are leading banking companies in the US. They will have huge similarities. We will pay more attention to these four companies in the further in-depth analysis of our dataset.

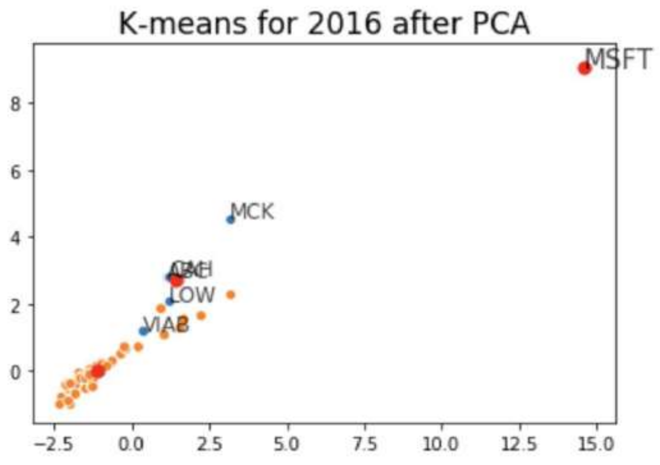
After PCA, the points were distributed relatively more even. The points in our graph presented a linear relationship with ascending order. DBSCAN worked on these same plots but defined the remote points as outliers. For hierarchical clustering, we would pay more attention to the year 2016 since the number of companies in 2016 is not too many. The name could be clearly labeled at the bottom of the dendrogram. We chose the 'ward' method to plot our dendrogram onto the x-y coordinate axis since ward method will compare the deviation for the distance between a point and the centroid, then pull each point into the nearest cluster with the lowest deviation until combining into one cluster. The ward method made sense since it was more similar to K-means algorithms. We could find the specific company for points by comparing our dendrogram with k-means clustering. The after PCA hierarchical cluster dendrogram gave a very distinct classification, which brought us two clusters, one cluster for only one point located at the right top corner and all other points for another cluster. The one outlier point is MSFT, which is Microsoft. We would also keep an eye on Microsoft's financial situation during further analysis. In conclusion, by checking an overall survey of our clustering plot performances, we found some enterprises with similar characteristics or from one industry aggregating together into a small cluster but far from most other points through the t-SNE plot. The after-PCA plots gave a better-organized, plain, and unambiguous classification for these companies. We would pay more attention to the market performances on these "suspicious" companies like Citibank, Wells Fargo, JP Morgan, BOA, and Microsoft. Considered on how to utilize our clustering, we could remove the outlier points and cluster our dataset again. We would obtain a cleaner result for deeper interpretations.











METHODS

METHOD	PROS	CONS
Logistic Regression	- easy to interpret - robust to noise - seldom overfits	- does not handle categorical variables well - suffer multicollinearity - bad if correlations are mostly nonlinear - can suffer from outliers
K-Nearest Neighbors	- fast training - can naturally handle extreme multiclass problems	- can fail to predict correctly due to the curse of dimensionality
Gaussian Naive Bayes	- easy to understand - requires less training data - converges faster than logistic regression - good for few categorical variables	- suffer multicollinearity - suffers from irrelevant features - fails estimating rare occurrences
Support Vector Classifier (SVC)	- flexible selection of kernels for nonlinear correlation - does not suffer multicollinearity - handles high dimensional data well	- hard to interpret - takes a long time to train
Decision Trees	- easy to interpret and explain - non-parametric--no need to worry about outliers or whether the data is linearly separable - good for categorical	- can easily overfit

	variables	
AdaBoost	- doesn’t overfit easily - few parameters to tweak - boosts weak learners	- sensitive to noisy data and outliers
Gradient Tree Boosting	- each new tree corrects some errors made by the previous trees - can approximate most nonlinear function - robust to outliers	- more difficult to tune and more prone to overfitting compared to random forest - training takes longer since trees are built sequentially - sensitive to noisy data and outliers - doesn’t work well without parameter tuning
Random Forest	- good for many types of problems - more robust than a single decision tree - less likely to overfit than other models - perform as well as or better than svms; trains faster - calculates feature importance	- difficult to interpret - weaker on regression when estimating values at the extremities of the distribution of response values - biased in multiclass problems toward more frequent classes
Stochastic Gradient Descent (SGD)	- efficiency - lots of opportunities for code tuning	- many parameters makes it difficult to effectively tune - sensitive to feature scaling
Perceptron (single layer)	- averaged perceptron improves generalization - fast to train - usually has high accuracy	- single layer perceptrons can learn only linearly separable patterns - difficult to explain - “black box”

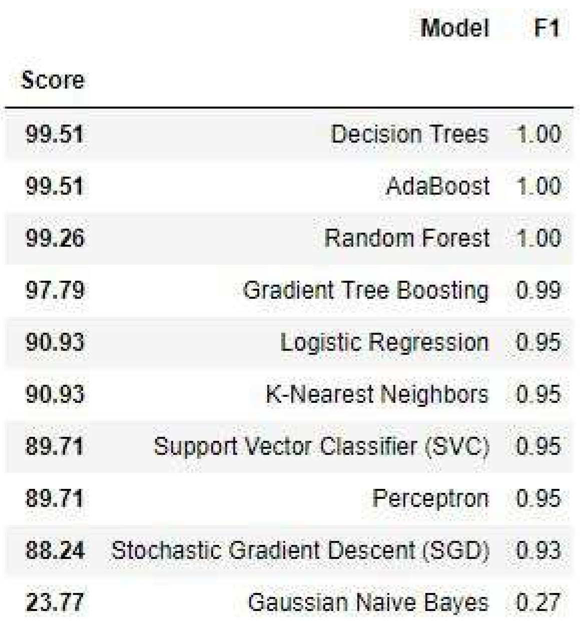
PREDICTIONS: PROFITABILITY

To create a measure of profitability we calculated the price-earnings ratio (PE). PE is the ratio of the company's share price to the company's earnings per share (EPS). We are using PE as a measure of profitability. Before calculating PE, missing EPS values were assigned zero. It is possible that missing EPS values meant the company wasn't public, but since this was a small dataset we did not want to eliminate rows unnecessarily. Using “trend" as our Y-variable, we then applied classification algorithms to predict profitability.

For this problem, we choose 10 methods to test. Methods were chosen based on (1) what was discussed in class and (2) what we discovered researching similar problems. Advantages and disadvantages of each method is briefly discussed under METHODS. For many methods, we utilised scikit-learn’s GridSearchCV to experiment with parameters. The best set of parameters were selected using mean cross-validated score of the best estimator. For all others, we choose parameters that felt reasonable considering similar examples found during our research.

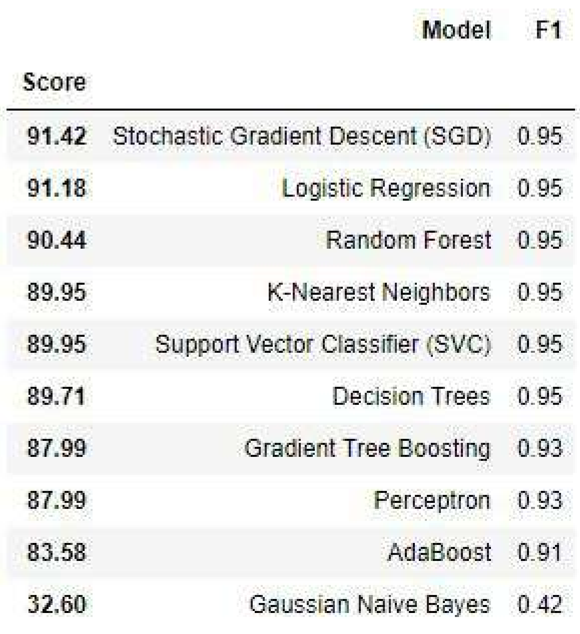
In order to explore a breadth of topics covered in lecture, we decided to apply methods on the dataset with and without principal component analysis (PCA). PCA uses orthogonal transformation to convert potentially correlated variables to a set of linearly uncorrelated variables called principal components. PCA was applied to scaled data to see if scaling boosted the performance of any of the models. After plotting, we discovered that 20 principal components explained close to 80% of the variance in the dataset. These 20 components were used as X-variables for model train/test.

Without PCA



Score	Model	F1
99.51	Decision Trees	1.00
99.51	AdaBoost	1.00
99.26	Random Forest	1.00
97.79	Gradient Tree Boosting	0.99
90.93	Logistic Regression	0.95
90.93	K-Nearest Neighbors	0.95
89.71	Support Vector Classifier (SVC)	0.95
89.71	Perceptron	0.95
88.24	Stochastic Gradient Descent (SGD)	0.93
23.77	Gaussian Naive Bayes	0.27

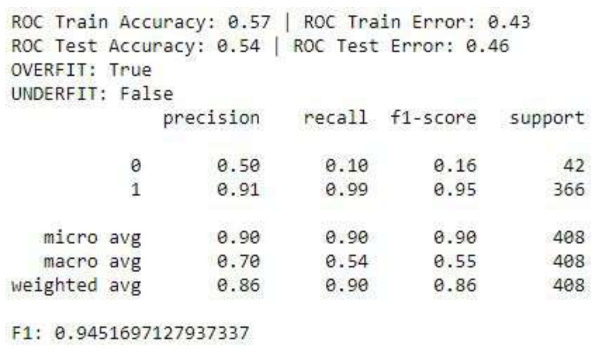
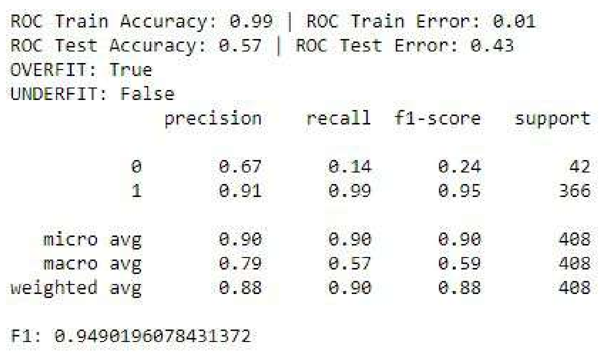
With PCA



Score	Model	F1
91.42	Stochastic Gradient Descent (SGD)	0.95
91.18	Logistic Regression	0.95
90.44	Random Forest	0.95
89.95	K-Nearest Neighbors	0.95
89.95	Support Vector Classifier (SVC)	0.95
89.71	Decision Trees	0.95
87.99	Gradient Tree Boosting	0.93
87.99	Perceptron	0.93
83.58	AdaBoost	0.91
32.60	Gaussian Naive Bayes	0.42

Scores were lower with PCA for many models. PCA will treat a feature that has large variance as important, but a feature with large variance can have nothing to do with the prediction target. There is a risk that, with PCA, we eliminate useful features or keep too many less-useful ones. We found evidence that PCA-based feature transformation usually does not improve classification performance (this was discussed in “PCA-based Feature Transformation for Classification: Issues in Medical Diagnostics”).

We used several ensemble methods to create models. With ensemble methods and PCA accuracy gains will almost always be minimal because the model already deals well with correlated predictors and high dimensional data sets. We saw no gains by using PCA with ensembles for this particular dataset. In fact, the only model with PCA that showed significant improvement was Gaussian Naives Bayes (GNB). GNB accuracy may have been improved further by tuning the parameters, boosting/bagging, or refining the data fed to the classifier (feature selection, adding more data, etc.). We choose not to because (1) GNB is typically applied to spam filtering instead of stock analysis and (2) because other models performed better with minimal tuning.

Decision Tree without PCA	Random Forest without PCA
	

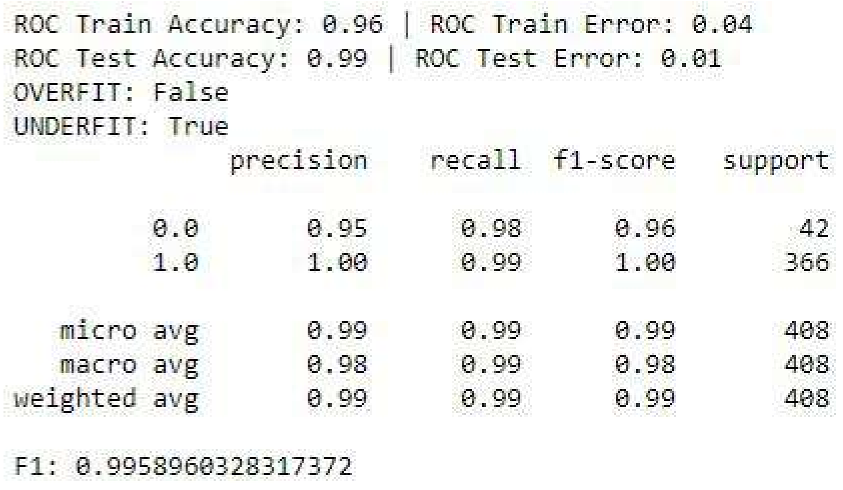
```
ROC Train Accuracy: 0.57 | ROC Train Error: 0.43
ROC Test Accuracy: 0.54 | ROC Test Error: 0.46
OVERFIT: True
UNDERFIT: False
              precision    recall  f1-score   support

           0       0.50      0.10      0.16        42
           1       0.91      0.99      0.95       366

   micro avg       0.90      0.90      0.90       408
   macro avg       0.70      0.54      0.55       408
weighted avg       0.86      0.90      0.86       408

F1: 0.9451697127937337
```

```
ROC Train Accuracy: 0.99 | ROC Train Error: 0.01
ROC Test Accuracy: 0.57 | ROC Test Error: 0.43
OVERFIT: True
UNDERFIT: False
              precision    recall  f1-score   support

           0       0.67      0.14      0.24        42
           1       0.91      0.99      0.95       366

   micro avg       0.90      0.90      0.90       408
   macro avg       0.79      0.57      0.59       408
weighted avg       0.88      0.90      0.88       408

F1: 0.9490196078431372
```

We tried to improve the accuracy, ROC accuracy, and F1 scores by generating more samples using bootstrapping, KDE, and SMOTE. Bootstrapping did not improve F1 scores. KDE did not make a significant improvement in class “0” (not profitable) precision and recall scores. SMOTE, on the other hand, dramatically improved precision and recall for class “0”. In addition, SMOTE improved ROC train and test errors. In this situation, classes were unbalanced but both were important. If a class distribution is highly skewed the classifier can get a low misclassification rate simply by choosing the majority class. For that reason, we choose the classifier that got high F1 scores on both classes, as well as a low misclassification rate. We discarded versions of Random Forest that had low F1 scores.

There is slight underfitting with this model. A good model has low bias and low variance. High bias (underfitting) can be identified when there is (1) high training error and (2) validation error or test error is same as training error. High variance (overfitting) can be identified when (1) there is low training error and (2) high validation error or high test error. Train error alone is misleading because it always delivers an overly optimistic estimation about model accuracy.

Classification Matrix for Random Forest w/ SMOTE

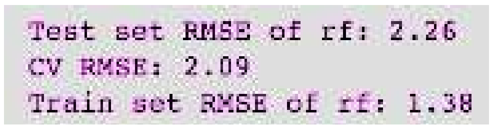


```
ROC Train Accuracy: 0.96 | ROC Train Error: 0.04
ROC Test Accuracy: 0.99 | ROC Test Error: 0.01
OVERFIT: False
UNDERFIT: True
              precision    recall  f1-score   support

         0.0       0.95      0.98      0.96        42
         1.0       1.00      0.99      1.00       366

   micro avg       0.99      0.99      0.99       408
   macro avg       0.98      0.99      0.98       408
weighted avg       0.99      0.99      0.99       408

F1: 0.9958960328317372
```

In our opinion, the best model for this dataset is Random Forest trained with SMOTE (no PCA). Random Forests require almost no input preparation, perform implicit feature selection, and are not hypersensitive to the parameters used. These qualities made it easier to produce a good model. We choose Random Forest in particular because of the high test accuracy, high precision, and satisfactory recall. In this model F1 scores were high for both classes. We could have reduced training time by reducing the number of features; this would be performed by analyzing the “importances” attribute of the model.

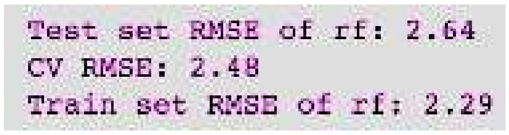
PREDICTIONS: ALTMAN Z-SCORE

Z-score is a common measure used to find out companies that have the risk of going bankrupt. A score of less than 1.8 indicates that there is a chance based on the information available, the company can go bankrupt. A score in the range 1.8-3.0 indicates the company is in the caution zone, and a score of more than 3 generally means that the company is in the safe zone.
We decided to apply different regression models to predict the z-score of a company, given the available fundamental information, and compare the RMSE scores.
Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors); it is a better measure of goodness of fit than a correlation coefficient because it is directly interpretable in terms of measurement units. If RMSE of test is greater than RMSE of train there is overfitting of the data. If RMSE of test is less than RMSE of train there is underfitting of the data.
We divided the dataset into train and test, and ran the Random Forest Regression model first. The results are shown below:



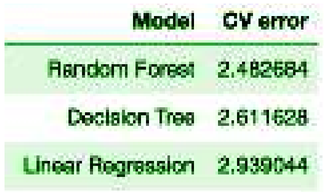
```
Test set RMSE of rf: 2.26
CV RMSE: 2.09
Train set RMSE of rf: 1.38
```

The model is overfitting because the CV RMSE is greater than the Train RMSE. We tried a couple of things to eliminate that problem. The ideal situation would have been to gather more observations to increase the data size. However, that was not possible for our dataset. So we decided to generate more data points using KDE and SMOTE, and compare the results.
We added 500 new rows to the Train data set by using KDE, which helped reduce the difference between CV RMSE and Train RMSE. Another thing we did was to reduce the complexity of the model, by reducing the number of trees and increasing the min_samples_leaf. The results of the random Forest after doing these are shown below:



```
Test set RMSE of rf: 2.64
CV RMSE: 2.48
Train set RMSE of rf: 2.29
```

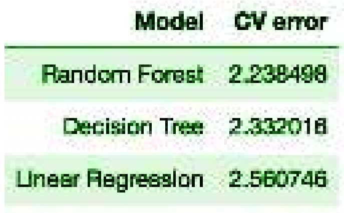
Although the model is still overfitting, the difference between the CV RMSE and Train RMSE has decreased, which means that there was some improvement provided by KDE and reducing complexity.
We also ran two additional models using the new train dataset: Decision Tree Regressor and Linear Regression.
Comparing the CV scores of three models, Random Forest is performing the best because it has the lowest CV RMSE score.



Model	CV error
Random Forest	2.482684
Decision Tree	2.611628
Linear Regression	2.939044

The above results were when data was generated using KDE.
We also tried SMOTE to generate more Training data and do the same process as above. Models work well when data is generated using SMOTE. Although we were not able to completely eliminate the overfitting problem, the difference between CV RMSE and Train RMSE

reduced after more training data was generated. Again, Random Forest worked the best, with the lowest CV RMSE error.



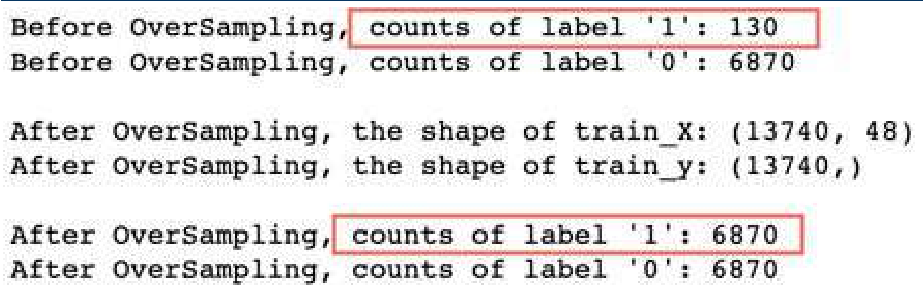
Model	CV error
Random Forest	2.238498
Decision Tree	2.332018
Linear Regression	2.560746

PREDICTIONS: Transfer Learning (Bankruptcy)

Transfer learning is a machine learning technique that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. The reason why we apply transfer learning method here is that our fundamental dataset only includes financial metrics and doesn't include companies' operating status, that is to say, we couldn't tell which companies were bankrupt. So we decided to train different classification models using an external dataset "bankruptcy_Train" which has 10,000 records.

We observed that only 203 (2.03%) went bankrupt among 10,000 records. After splitting 70% training dataset and 30% testing dataset, we applied the SMOTE technique to generate more samples in the training dataset to balance the number of bankruptcy records and non-bankruptcy record. Bankruptcy records amount increased from 130 to 6870 (as below), which is equal to the non-bankruptcy records.

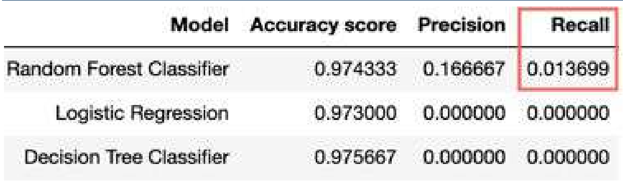
Resampling training dataset using SMOTE technique



```
Before OverSampling, counts of label '1': 130
Before OverSampling, counts of label '0': 6870

After OverSampling, the shape of train_X: (13740, 48)
After OverSampling, the shape of train_y: (13740,)

After OverSampling, counts of label '1': 6870
After OverSampling, counts of label '0': 6870
```

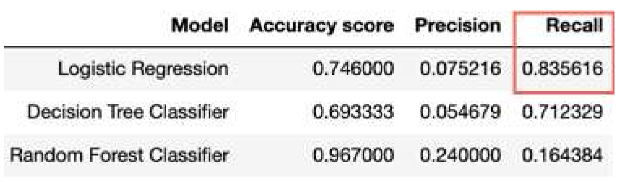
We ran the Logistic Regression, Decision Tree Classifier and Random Forest Classifier models on both original training dataset (no SMOTE) and the resampled training dataset (SMOTE), and observed the models were improved a lot after SMOTE technique applied. The accuracy scores before SMOTE were extremely high, but with low precision and recall. The models just predict all (or the majority) records as non-bankruptcy to obtain a high accuracy score. The accuracy scores after SMOTE were lower, however, both precision and recall score are higher, especially recall score.

NO SMOTE



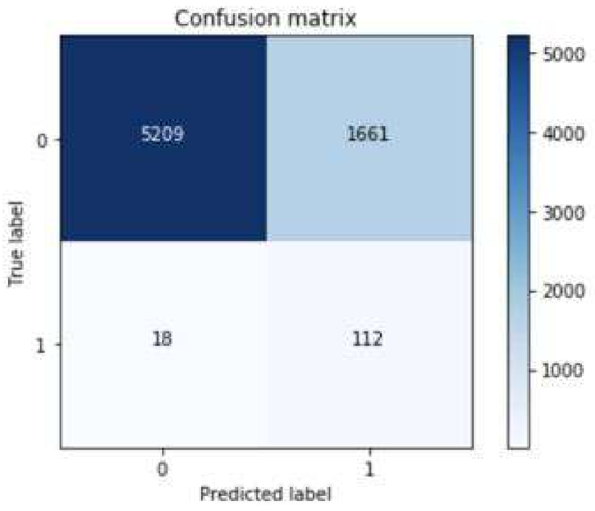
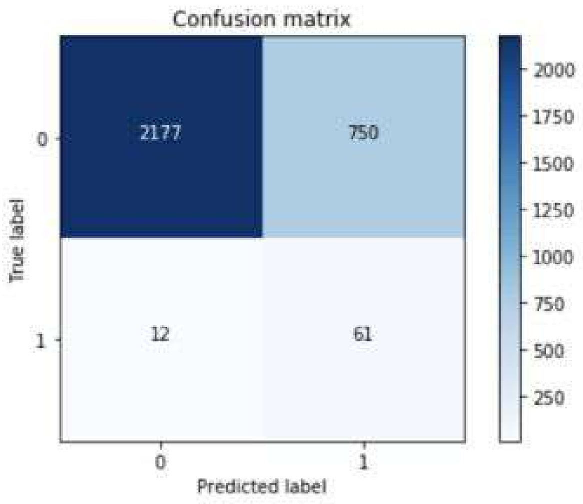
Model	Accuracy score	Precision	Recall
Random Forest Classifier	0.974333	0.166667	0.013699
Logistic Regression	0.973000	0.000000	0.000000
Decision Tree Classifier	0.975667	0.000000	0.000000

SMOTE



Model	Accuracy score	Precision	Recall
Logistic Regression	0.746000	0.075216	0.835616
Decision Tree Classifier	0.693333	0.054679	0.712329
Random Forest Classifier	0.967000	0.240000	0.164384

"Recall" is true positive divided by the sum of true positive and false positive, while "Precision" is true positive divided by the sum of true positive and false negative. We believe "Recall" is more important than "Precision" and "Accuracy" in this case because our purpose is to narrow down the list of companies which might be bankruptcy. We would like to identify the companies which need further track or investigation. It is fine to include some companies which actually will not be the bankruptcy (false positive) in our narrow down list, however, we would like to avoid missing any companies which actually will be the bankruptcy(false negative). Although the "Random Forest Classifier" has the highest accuracy score, its recall score is low. After comparing recall score, we believe the best model for this dataset is a Logistic Regression model which has the highest recall score (83.56%).

Logistic Regression Recall metric in the train dataset: 86.15%	Logistic Regression Recall metric in the test dataset: 83.56%
	

Finally, we applied our logistic regression model to the Fundamental file, which has 1781 records. We narrowed down the list of companies (as below) which might be bankruptcy and need further track or investigation:

KMX (CarMax)	CVX (Chevron Corporation)	BDX (Becton Dickinson)	KR (Kroger)	TJX (TJX Companies)	SYY (Sysco)
MKC (McCormick & Company)	WFM (Whole Foods Market)	PWR (Quanta Services)	BBY (Best Buy)	SWK (Stanley Black & Decker, Inc.)	PSX (Phillips 66)
GWW (W. W. Grainger)	XRAY (DENTSPLY SIRONA)	MO (Altria Group)	GIS (General Mills)	XOM (ExxonMobil)	WMT (Walmart Inc)
CTAS (Cintas)	JBHT (J. B. Hunt)	HD (The Home Depot)			

CONCLUSION

To summarize, random forest produced the best models for predicting profitability and z-score. The best model for bankruptcy classification was logistic regression due to particularly high recall score. We found out that it’s really easy for models to overfit. Although we were able to reduce the overfitting problem by generating more data points, in order to eliminate the overfitting completely and make sure our models are generalizable, we would have to add more actual data. PCA was situationally useful--especially for creating interpretable clusters. Most of our clustering attempts resulted in two clusters. We think that this is because of outliers. In order to improve the clusters and get more interpretable results, we could have removed the outliers and performed clustering again.

REFERENCES

PCA-based Feature Transformation for Classification:
Issues in Medical Diagnostics:
https://www.win.tue.nl/~mpechen/publications/pubs/PechenizkiyCBMS04.pdf

SMOTE for Regression in R/Python:
https://medium.com/@srujana.rao2/oversampling-for-regression-model-7677ecd93195