Machine Learning Engineer Nanodegree

Capstone Project

Satish Irrinki

I. Definition

Project Overview

Corporate bankruptcy prediction is a very beneficial tool to have for wide range of practitioners in the financial services industry. Investors and investment portfolio mangers can better manage their funds and portfolio allocations when armed with the information about potential bankruptcy of the companies they invent in. Career corporate finance professionals can better plan for financial liquidity and take possible risk mitigation and cost reduction activities in response to early detection of potential bankruptcy. Other sectors in the industry that can benefit from the tool include firms in Private Equity, Investment Banking, Mergers and Acquisitions advisors, and Credit rating agencies.

The concept of bankruptcy is quite old with early bankruptcy laws dating back to 1841. These laws govern corporations to ensure business continuity that result in well being of the society through supply of products and services. Because bankruptcy ensures orderly control of operations or liquidation of firms in extreme competitive situations, predicting bankruptcy early will cause less social and economic damage. There is considerable research both in academia and in industry to address the prediction of possible bankruptcy. The academic paper Bankruptcy Prediction with Financial Ratios, http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=3918017&fileOId=3918043 provides an illustration of the concept of using financial ratios for prediction. I will use financial ratios as well to predict bankruptcies in this capstone project.

I am personally motivated to implement this project because I have a very deep interest in the discipline of Finance. I pursued MBA degree with focus in Finance. I also successfully pursed Chartered Financial Analyst program (CFA program) and passed all three levels of the program.

A representative data set from UCI Machine Learning Repository, Polish Companies Bankruptcies Data Dataset, https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data is used to train and test the bankruptcy prediction model. The dataset has one-year financial ratios and corresponding classification label indicating bankruptcy status after five years. The file has financial data for 7027 companies. Among these companies, 6756 companies did not file for bankruptcy after five years and 271 companies were bankrupt. The raw data file has been processed to be completely compliant to be in tabular form and to populate the missing values as NaN. Although the data set is for companies in Poland, the underlying data can be changed to represent any set of companies with appropriate feature columns, primarily because the models can be agnostic to the names of the features as well as leverage dimensionality reduction using Principal Component Analysis (PCA).

There are 64 features in the data set. Below is a set of sample features. The full listing can be found at: https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data#

X1 net profit / total assets, X9 sales / total assets, X10 equity / total assets, X23 net profit / sales, X27 profit on operating activities / financial expenses, X28 working capital / fixed assets, X40 (current assets - inventory - receivables) / short-term liabilities, X48 EBITDA (profit on operating activities - depreciation) / total assets, X64 sales / fixed assets

For example, X1, net profit/total assets, is an indicating of how well a company is using its assets; higher the value for X1 less likely it is to get bankrupt. Similarly other features influence the prediction.

Problem Statement

All pubic companies have their financial information publicly available from their filings with the regulators in their respective countries. The data available can be leveraged to evaluate its current as well as future financial performance. I will use Machine Learning algorithms (SVM and DecisionTreeClassifier) to train the bankruptcy prediction algorithms to classify the data and then test the algorithms to predict bankruptcy prediction accuracy.

The project design follows the following steps:

- Pre process the input features to fill missing values with average of the feature values. Where the number of missing values are excessive(>1000), remove the features if appropriate
- Normalize the data using MinMaxScalar so that the transformed data can be used for PCA
- Iteratively select suitable number of components for PCA for dimensionality reduction
- Use the principal components as inputs to classification algorithms SVM and DecisionTreeClassifer
- Optimize the algorithm to using GridSearchCV method
- Capture model performance metrics defined by accuracy_score and fbeta_score for both SVM and DecisionTreeClassifer
- Capture the baseline model metrics, accuracy_score and fbeta_score and compare to that of the model
- Analyze and report the observations

Scaling the input data is necessary to have an accurate dimensionality reduction using PCA. PCA uses distance and different scales for different features can skew the components. An iterative approach will be used to select the number of PCA components so that 94 to 95% of the explained variance is captured in the components.

The PCA components will be used as inputs to the classification algorithms, SVM and DecisonTreeClassifer, identified earlier. The build models will then be optimized for best performance using combinations of hyperparameters. Then the models will be trained and tested for accuracy using the metrics accuracy_score and fbeta_score from sklearn. These results will then be compared with those of the baseline benchmark model. Conclusions will then be drawn from the observed results.

Metrics

The metric accuracy_score represents percentage of accuracy of predicted values relative to true value. Accuracy may not always be the accurate metric when very high percentage of positives or negatives represent the data. Say for example, 99% of the companies are not bankrupt, then if we predict that a company is not bankrupt, then the prediction is likely accurate 99% of the times.

fbeta_score represents the weighted average of precision and recall. We need to ensure that we capture all bankruptcies, so we need a high recall model. Precision is the percentage of positive predictions among all positive predictions (true positives and false positives). Recall is the percentage of true positive predictions among all positive samples (includes true positives and false positives).

F1 score the harmonic mean of precision and recall. F-Beta score ranges between zero and one, with zero representing pure precision for beta = zero and infinity represents pure recall for beta = infinity.

It is better to have as many bankruptcies as possible predicted correctly, so we need high recall. So, accordingly, a beta value greater then 1 will be used to skew towards recall model.

II. Analysis

Data Exploration

The cell below loads the inputs data file and captures the inputs into different pandas variables needed for further processing.

Below is a list of features and how their values are calculated from financial data.

X1 net profit / total assets
X2 total liabilities / total assets
X3 working capital / total assets
X4 current assets / short-term liabilities

X5 [(cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] *365*
X6 retained earnings / total assets
X7 EBIT / total assets
X8 book value of equity / total liabilities
X9 sales / total assets
X10 equity / total assets
X11 (gross profit + extraordinary items + financial expenses) / total assets
X12 gross profit / short-term liabilities
X13 (gross profit + depreciation) / sales
X14 (gross profit + interest) / total assets
X15 (total liabilities 365) / (gross profit + depreciation)
X16 (gross profit + depreciation) / total liabilities
X17 total assets / total liabilities

X18 gross profit / total assets
X19 gross profit / sales
X20 (inventory *365) / sales*
X21 sales (n) / sales (n-1)
X22 profit on operating activities / total assets
X23 net profit / sales
X24 gross profit (in 3 years) / total assets
X25 (equity - share capital) / total assets
X26 (net profit + depreciation) / total liabilities
X27 profit on operating activities / financial expenses
X28 working capital / fixed assets
X29 logarithm of total assets
X30 (total liabilities - cash) / sales
X31 (gross profit + interest) / sales
X32 (current liabilities 365) / cost of products sold
X33 operating expenses / short-term liabilities
X34 operating expenses / total liabilities
X35 profit on sales / total assets
X36 total sales / total assets
X37 (current assets - inventories) / long-term liabilities
X38 constant capital / total assets
X39 profit on sales / sales
X40 (current assets - inventory - receivables) / short-term liabilities
X41 total liabilities / ((profit on operating activities + depreciation) *(12/365))*
X42 profit on operating activities / sales
X43 rotation receivables + inventory turnover in days
X44 (receivables 365) / sales
X45 net profit / inventory
X46 (current assets - inventory) / short-term liabilities
X47 (inventory *365) / cost of products sold*
X48 EBITDA (profit on operating activities - depreciation) / total assets
X49 EBITDA (profit on operating activities - depreciation) / sales
X50 current assets / total liabilities
X51 short-term liabilities / total assets
X52 (short-term liabilities 365) / cost of products sold)
X53 equity / fixed assets
X54 constant capital / fixed assets
X55 working capital
X56 (sales - cost of products sold) / sales
X57 (current assets - inventory - short-term liabilities) / (sales - gross profit - depreciation)
X58 total costs /total sales
X59 long-term liabilities / equity
X60 sales / inventory
X61 sales / receivables
X62 (short-term liabilities *365) / sales
X63 sales / short-term liabilities
X64 sales / fixed assets

In the following section of the code, the data file is uploaded in Pandas DataFrame followed by a display of first five rows of the data and descriptive analysis of the features in the data set. Because there are 64 features, the plots are getting too crowded in the graphics. So, only the first, last, and every eight features (a total of 9 features) are selected for representative plots. These features comprise X1, X8, X16, X24, X32, X40, X48, X56, and X64.

X1 net profit / total assets

X1, net profit / total assets, represents how the assets of a company are performing. The higher the ratio the better the financial health of a company and it is less likely to go bankrupt.

X8 book value of equity / total liabilities

X8, book value of equity/ total liabilities, represents the book value of equity as a multiple of total liabilities. It determines whether in an event of bankruptcy the equity holder will be disbursed any residual value of the company after credit holder are paid off with the cash obtained by liquidating assets. Lower the value indicates higher debt, which indicates higher need for cash flow to service the debt, I.e. to pay interest and principal that's outstanding. So, lower the ratio, ceteris paribus, higher the likelihood for a bankruptcy.

X16 (gross profit + depreciation) / total liabilities

X16, (gross profit + depreciation) / total liabilities, is an indication of what percentage of total liabilities can be met with current operations. Higher the ratio, the better a company can meet its obligations and less likely it is to go bankrupt.

X24 gross profit (in 3 years) / total assets

X24, gross profit (in 3 years) / total assets, is an indication of how stable the companies profits are over a period of three years. It allows for some fluctuation over the three years that can be attributed to market conditions; however, higher a three year ratio the better the financial health of a company and less likely that will go bankrupt. The ratio also indicates how well the company's assets are being leveraged to generate profits.

X32 (current liabilities * 365) / cost of products sold

X32, (current liabilities * 365) / cost of products sold, is an indication how well the cost of good sold (COGS) is being managed with short-term loans. Higher the value of this ratio, the weaker the financial health of the company. It implies that the company is borrowing more than it needs for the COGS over the year. Higher the ratio, more likely the company can fail to meet its payment obligations and more likely that it can go bankrupt.

X40 (current assets - inventory - receivables) / short-term liabilities

X40, (current assets - inventory - receivables) / short-term liabilities, is the ratio of liquid assets such as marketable securities and pre-paid expenses that can be leveraged to meet the short-term payment obligations. Higher the ratio, better the financial strength of a company and less likely that it will go bankrupt.

X48 EBITDA (profit on operating activities - depreciation) / total assets

X48 EBITDA, (profit on operating activities - depreciation) / total assets, is the ratio of earnings before interest, taxes, depreciation, and amortization to total assets. This ratio represents the true operational value that the company generates per unit asset that the company uses to generate the value. Higher the ratio, the better performance of the company, and less likely it is to go bankrupt.

X56 (sales - cost of products sold) / sales

X56, (sales - cost of products sold) / sales, is also referred to as gross margin, represents the parentage of sales that the company retains after meeting the COGS. The higher the gross margin, the better the company's performance and less likely it is to go bankrupt.

X64 sales / fixed assets

X64, sales / fixed assets, is an indication of sales per unit assets, meaning, it is a ratio that determines productivity and efficiency of assets of a company measured by the sales revenue generated by them. Higher the ratio, better the financial health of the company and less likely it is to get bankrupt.

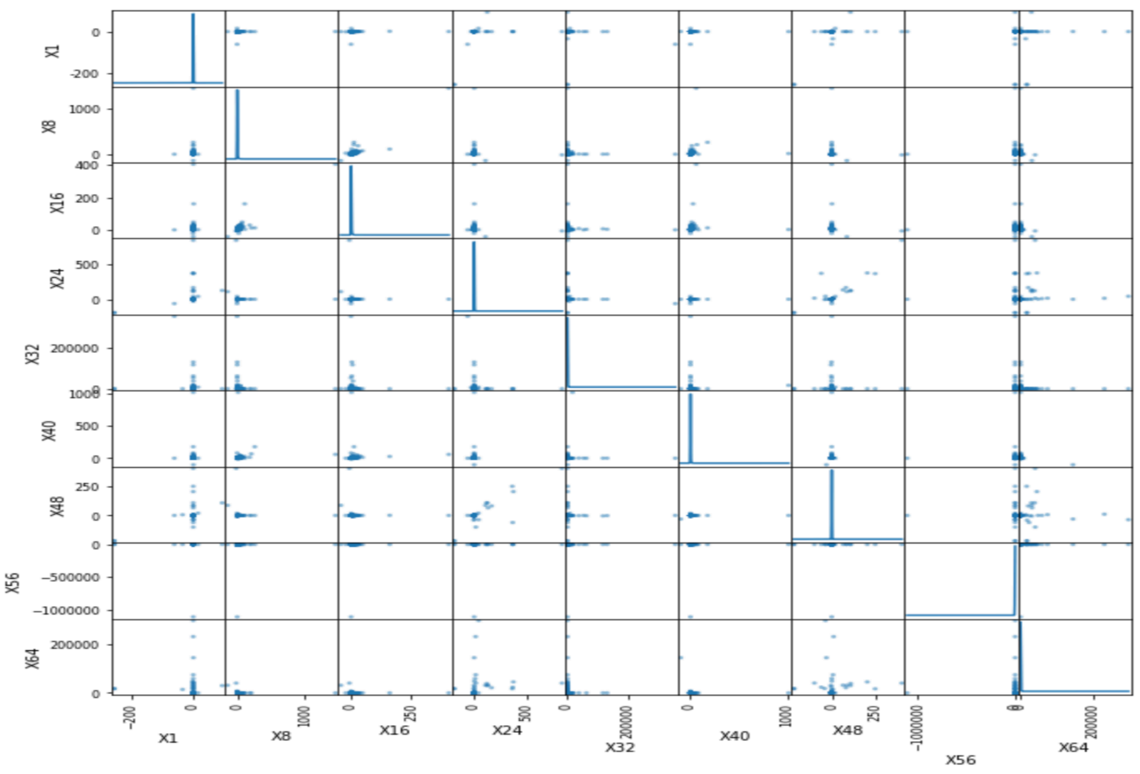
	X1	X8	X16	X24	X32	X40	X48	X56	X64
count	7024.000000	7002.000000	7002.000000	6903.000000	6989.000000	6997.000000	7024.000000	7.027000e+03	6993.000000
mean	0.034660	2.623996	0.870690	0.540231	236.571445	0.826132	0.148865	-1.577367e+02	208.731950
std	4.565504	18.708327	5.595342	13.501067	4841.844793	12.594535	7.793534	1.322125e+04	5140.708804
min	-256.890000	-141.410000	-42.322000	-189.560000	0.000000	-101.270000	-218.420000	-1.108300e+06	0.000010
25%	0.021182	0.445710	0.122810	0.049077	48.291000	0.052276	-0.016980	2.031450e-02	2.538600
50%	0.075802	1.015100	0.311115	0.164470	76.163000	0.156790	0.042192	6.338200e-02	4.637700
75%	0.160268	2.267675	0.737822	0.336050	119.300000	0.517260	0.142662	1.376950e-01	9.782200
max	94.280000	1452.200000	405.330000	831.660000	351630.000000	1014.600000	405.590000	1.000000e+00	294770.000000

The above description of the features data can has been performed only on the nine representative features. The features are ratios in values and are relative to the financial metric that represents the company. Relative to the mean of each of the features, the minimum and maximum values are quite far away, more than 6 times the standard deviation. So, we can state that there are outliers in several features. We can visually observe the outliers in the histogram for each of representative features. We can transform/scale the data using Box-Cox test or by applying natural logarithm. However, we use MinMaxScaler and apply PCA. So, the features are scaled to range of [0,1] as part of preprocessing.

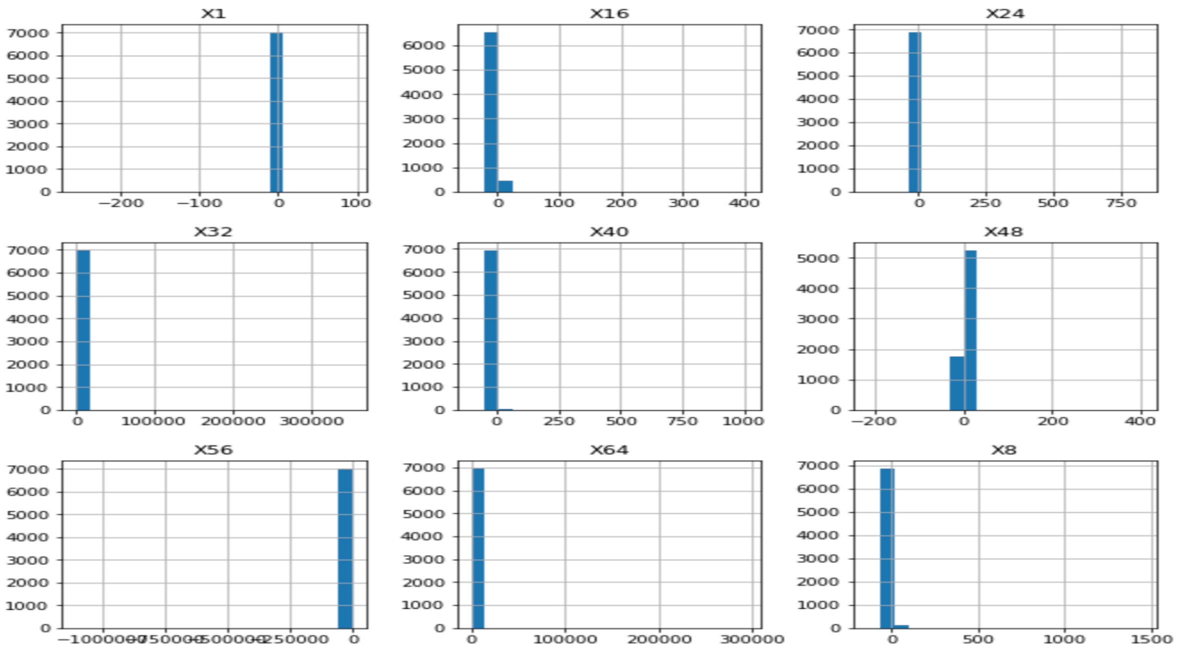
Exploratory Visualization

There is no significant correlation between the features. We can do not see much linear relationship between the selected features in the scatter plot below. There appears to some linearity between features (X24, X48).

The histogram for each of the selected features identified that most of the data for each of the features is strongly clustered around zero. This observation is not surprising considering that the data represents ratios and are relative to larger values such as fixed assets, etc.



X1
X8
X16
X24
X32
X40
X48
X56
X64



X1
X16
X24
X32
X40
X48
X56
X64
X8

Algorithms and Techniques

On the basis of some of the published research in the field of bankruptcy prediction using financial data of companies, Support Vector Machines (SVM) and Decision Tree Classifier will be used to define, implement, test, and refine the prediction model.

Ref: https://dspace.library.uu.nl/bitstream/handle/1874/351884/Thesis-Frank_Wagenmans-3870154.pdf?sequence=1&isAllowed=y

Support Vector Machines

SVM is a classification algorithm that separates different classes of data in such a manner that the closest points of each of the classes are farthest from the boundary of separation. There are other possible boundaries that can perfectly separate the training data but, the boundary that separates with the maximum distance is the best possible classification and least likely to produce errors during testing.l

The underlying logic behind SVM is to minimize two forms of errors obtained from applying SVM algorithm to classify data -- classification error and the margin error. We need to minimize these errors so that the boundary and margin has minimum number of data points that are incorrectly classified and minimum number of data points within the margin defined by the C parameter. The larger the C parameter, the narrower the margin and higher the margin error. The C parameter offers a tradeoff between correctly classifying the training data at the expense of not having a wider margin. The wider the margin, the fewer the testing errors will be and more accurate the prediction. The goal is to maximize the margin while simultaneously minimizing the margin error. The choice of C parameters depends on the problem being solved. Smaller C parameters apply better to applications that have tolerance for error such as making movie recommendations. Where as medical applications have low tolerance to errors and need larger C parameters.

The other parameter that defines SVM is the type of kernel used to model the classifier. Linear (linear), Polynomial (poly), or Radial Basis Function (rbf) kernels are used to define the SVM. Kernel and C parameter formulate the hyperparameters needed to define SVM. Other hyperparametrs such as degree and gamma are needed for Polynomial and rbf respectively.

Decision Tree Classifier

Decision Tree Classifier is simple and intuitive technique used for classification. It evaluates a series of questions about the features and after receiving a response it follows up with more questions until a classification of the target is established for the data record. The technique allows to group different ranges of values of categories for feature(s). This process of making inquiries (asking questions) is done at nodes (from which questions on features are asked) until the label (target) category is derived.

One of the advantages of using Decision Tree Classifier is that the raw features data need not be transformed or normalized for scale comparability across different features. The algorithm is based on conditions that are evaluated on the content of individual features to maximize information gain but not as a distance among different data points.

If the number of nodes is high and the depth of the tree hierarchy is high, then the model tends to memorize the outcomes and end up with over fit. So there has to be an optimal depth for the nodes hierarchies. It can also avoid to over fit scenarios by mandating a minimum number of samples needed to split at an internal node and work with specifying maximum number of features that should be used in determining a split at a node. There are other parameters (hyperparameters) that we can use to not over fit the model and obtain an optimal model. Decision Tree algorithm is applicable to solve both classification and regression problem.

Some of the hyperparameters that the DecsionTreeClassifier uses include max_depth, min_samples_split, and mimimum_samples_leaf among others.

- max_depth: The maximum depth of the tree.
- min_samples_split: The minimum number of samples that evaluate to warrant a split when on a node.
- minimum_samples_leaf: This is specified as an absolute number if the data is integer or as a percentage of the number of samples if the data is a float. It specifies the minimum number of samples that should resolve to form a node.

Hyperparameters tuning helps keep the model generic and avoids misfit (overfit or underfit) the training data.

Benchmark

The benchmark model uses Decision Tree Classifier on raw data using the five features that are picked to represent the features identified in the research paper:

http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=3918017&fileOId=3918043

The Altman's Z-Score model uses the features listed below:

- (Current assets – current liabilities)/total assets
- Retained earnings/total assets
- EBIT/average total assets
- MV of equity/BV of liabilities
- Sales/average total asset

The corresponding similar features from the data set are identified below:

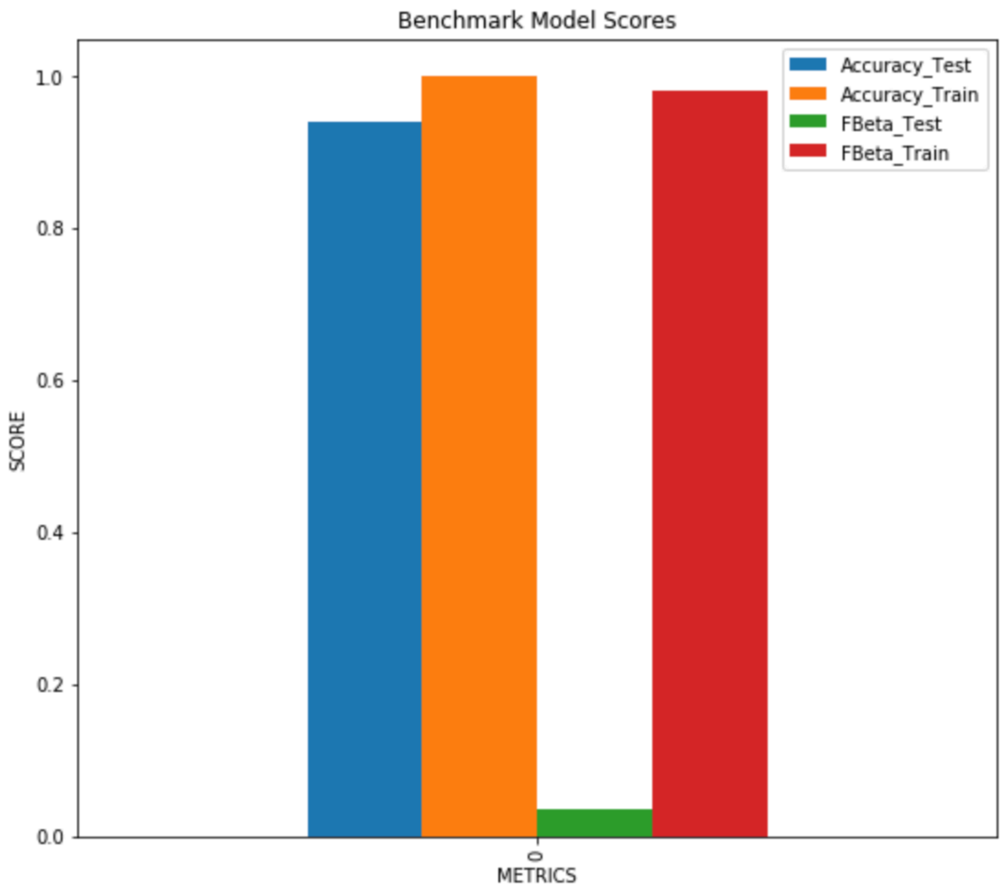
- X3 working capital / total assets
- X6 retained earnings / total assets
- X7 EBIT / total assets
- X8 book value of equity / total liabilities
- X9 sales / total assets

A comparable set of features from the input data set will be used to train, test, and measure Decision Tree Classifier and will be used as a benchmark model for the bankruptcy prediction algorithm.

The DecsionTreeClassifier will be trained and tested with 75% and 25% of the data for training and testing purposes. Because this is a benchmark model, default hyperpparameters will be applied for the algorithm as defined for DecsionTreeClassifier in sklearn package.

Metrics accuracy_score and fbeta_score will be evaluated for the classifier and compared with the performance of SVM and DecsionTreeClassifier models that are tuned and refined with different combinations of the hyperparameters for each of the algorithm. Below are the results from applying the benchmark features to DecsionTreeClassifier algorithm with default parameters.

- Accuracy_Train : 0.9991
- Accuracy_Test : 0.94024
- FBeta_Train : 0.9811
- FBeta_Test: 0.03533

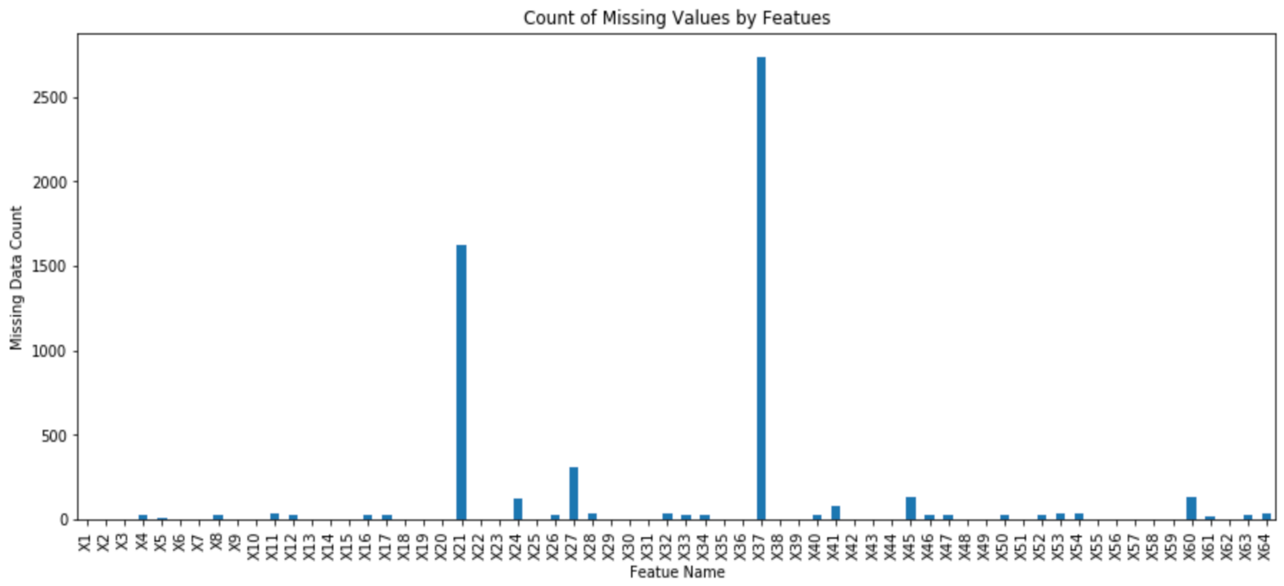


III. Methodology

Data Preprocessing

We can observe that there are several missing data elements among several features. Analysis has been performed to identify the number of missing data for each of the features. The histogram below plots the analysis.

We can observer from the display of the representative features that there are missing, NaN, values in the data set. A detailed analysis has been done to identify the number of missing values in each of the features. Among them, features X21 and X37 have significant missing values at 1622 and 2740 each respectively. We can also observe from the histogram, Count of Missing Values by Features, that the rest of the features have fewer than 500 missing values. Because the number of features is large, I considered removing features X21 and X37.



As noted earlier, because the number of features is high, a total of 62 features after removing X21 and X37, Principal Component Analysis is leveraged for dimensionality reduction, and because PCA is sensitive to the relative scales of each of the features, MinMaxScaler class from sklearn.preprocessing is used to transform/normalize the features to [0,1] scale. The transformation also helps with handling the outliers to some degree.

Implementation

A brief outline of the methodology followed was presented in the Problem Statement section of the report. The discussion below elaborates on the implementation of the methodology.

The input dataset consists of 64 features. After observing the data for missing values, features with excessive missing values are removed and the other missing values are filled with the mean value for the feature. Impute class from sklearn is used to fill the missing values. In order to reduce the number of features, PCA is used to reduce the dimensionality. Sixteen components are extracted to achieve at least 95% of explained variance ratio. Below is cumulative explained variance ratio from one to 16 components.

explained_variance_ratio: [0.28310318, 0.428273, 0.54391193, 0.62690537, 0.69777124, 0.7526 3854, 0.79977479, 0.83168391, 0.85719397, 0.87620507, 0.89323453, 0.90797933, 0.92238729, 0.93434241, 0.94618251, 0.95579776]

Both DecsionTreeClassifier and SVC classed from sklearn are leveraged to build the base model with out much tuning for hyperparameters. The intent is to measure the base performance on the new dimensions resulting from PCA dimensionality reduction.

After obtaining the accuracy and fbeta_score metrics for the base DecsionTreeClassifier and SVC models, GridSearchCV is leveraged to perform hyperparameters tuning for both SVC and Decsion TreeClassifier. The results from both the base models and tuned models will be discussed in the R esults section of the report.

A special function train_predict() has been created to generically use any classifier and to optimize the model with best fit relevant hyperparameters. The function takes the classifier, hyperparameters, training, and testing values needed to train, test, and measure the metrics. The function is defined as: train_predict(learner, param_grid, X_train, y_train, X_test, y_test)

Refinement

Base Models:

A base SVM and Decision Tree Classifier models have been trained, tested, and evaluated for accuracy and fbeta_score. The algorithms use 16 dimensions extracted from PCA dimensionality reduction and use default hyperparameters. Below is a summary of metrics for SVM and DecsionTreeClassifier respectively.

Metric	**SVM**	**DecisionTreeClassifer**
• Accuracy_Train :	0.4599	1.0
• Accuracy_Test :	0.4689	0.9294
• FBeta_Train :	0.2353	1.0
• FBeta_Test:	0.1841	0.2827

Tuned Models:

After capturing the metrics from the base model, the methodology continues to tune the algorithm using GridSearchCV to iterate for the best-fit model over different ranges on select hyperparameters.

The tuned GridSearchCV for SVM uses 'kernel': ['poly', 'rbf'], 'C': [1, 10, 20, 30, 40, 50], 'gamma': [0.001, 0.005, 0.01, 0.1] for hyperparameters which resulted in the best estimator:

SVC(**C=40**, cache_size=200, class_weight='balanced', coef0=0.0, decision_function_shape='ovr', degree=3, **gamma=0.01**, **kernel='poly'**, max_iter=-1, probability=False, random_state=None, sh rinking=True, tol=0.001, verbose=False)

The DecisionTreeClassifer uses 'max_depth' : [3, 4, 5, 6, 7, 8], 'max_leaf_nodes' : [4, 5, 6], and 'min_samples_leaf' : [2,5,8,11,14] for hyperparameters which resulted in the best estimator:

DecisionTreeClassifier(class_weight=None, criterion='gini', **max_depth=6**, max_features=None**,** **max_leaf_nodes=6**, min_impurity_decrease=0.0, min_impurity_split=None, **min_samples_leaf=5**, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=None, splitter='best')

Below is a summary of metrics for best estimators for SVM and DecisonTreeClassifer respectively.

Metric	**SVM**	**DecisionTreeClassifer**
• Accuracy_Train :	0.948	0.9637
• Accuracy_Test :	0.9505	0.9623
• FBeta_Train :	0.0953	0.1789
• FBeta_Test:	0.1098	0.1639

IV. Results

Model Evaluation and Validation

The table below summarizes the metrics from all the different models that have been trained and tested.

Benchmark: The benchmark model, DecsionTreeClassifier, used raw data on selected features with default hyperparameters

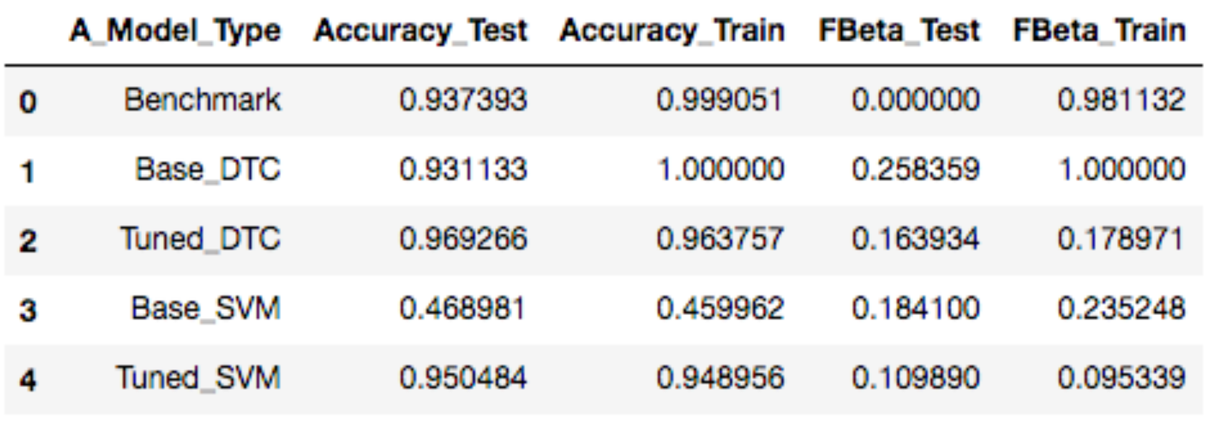
Base_DTC: The base DecsionTreeClassifier model that uses PCA components as inputs and default hyperparametes

Tuned_DTC: The tuned DecsionTreeClassifier model that uses PCA components for input and GridSearchCV technique to iterate over a set of hyperparameters for best estimator

Base_SVM: The base SVM model that uses PCA components as inputs and default hyperparametes

Tuned_SVM: The tuned SVM model that uses PCA components for input and GridSearchCV technique to iterate over a set of hyperparameters for best estimator

The measured metrics Accuracy_Test and Accuracy_Train represent the corresponding sklearn accuracy measurements for test and training data for the respective models. The FBeta_Test and FBeta_Train represent the fbeta_score measurements for test and training data for their respective models.



	A_Model_Type	Accuracy_Test	Accuracy_Train	FBeta_Test	FBeta_Train
0	Benchmark	0.937393	0.999051	0.000000	0.981132
1	Base_DTC	0.931133	1.000000	0.258359	1.000000
2	Tuned_DTC	0.969266	0.963757	0.163934	0.178971
3	Base_SVM	0.468981	0.459962	0.184100	0.235248
4	Tuned_SVM	0.950484	0.948956	0.109890	0.095339

The accuracy metric for training on base models is 1.0 and 0.459962 for DecsionTreeClassifier and SVM models respectively. This is possibly due to reusing the data for measuring accuracy and the DecsionTreeClassifier is more predictable in than the SVM model. But when we compare the testing accuracy, the accuracy measures 0.931133 and 0.468981 respectively for DecsionTreeClassifier and SVM models. So, on the base model, we have better performance from DecsionTreeClassifier.

The accuracy metric for training on tuned models is 0.963757 and 0.948956 for DecsionTreeClassifier and SVM models respectively. This is training accuracy and it is not surprising that the accuracy is high because the predictions are obtained from the training data. DecsionTreeClassifier performed better on the training accuracy. The testing accuracy measures 0.969266 and 0.950484 for DecsionTreeClassifier and SVM models respectively. The DecsionTreeClassifier edged over SVM model for the tuned models as well.

The tuned DecsionTreeClassifier is the best model.

The fbeta_score fared consistently better with training data and predictions compared with testing data and predictions. The bankruptcy predictor prefers higher recall because when a prediction is positive, the cost of remediation can be quite expensive and effects of a bankruptcy can percolate through the broader economic stability. So, we should have fewer false negatives (high recall) and lower precision (high false positives) can be wasteful when remediation actions are not required.

A further evaluation of the model is done on the data set all all the rows with any missing values removed from data set. This resulted in 6,318 rows, reduced from the original count of 7,027 rows.

Metric	DecisionTreeClassifier
• Accuracy_Train :	1.0
• Accuracy_Test :	0.9366
• FBeta_Train :	1.0
• FBeta_Test:	0. 2795

The final model with modified data set fared slightly lower in testing accuracy compared with the accuracy when missing NaN values are populated with the mean value of the respective features. The tuned DecisionTreeClassifer model fares better and is reliable.

Justification

The Benchmark model evaluates select features on DecsionTreeClassifier with default hyperparameters. The metrics for the benchmark model are shown in the table above with train and test accuracy at 0.999051 and 0.937393. This compares with train and test accuracy of 0.963757 and 0.969266 for the tuned DecsionTreeClassifier. The benchmark shows signs of over fitting in the training accuracy. However, the tuned DecsionTreeClassifier fared better in testing and can be affirmed to be a better predictor than the default model.

The tuned DecsionTreeClassifier, Tuned_DTC, with hyperparameters max_depth=6, max_leaf_nodes=6, and min_samples_leaf=5 is the best model for predicting potential bankruptcies for the companies.

The accuracy of this model has surpassed all the other models tested in the process including after altering the data set to exclude features will high missing values and all the rows with any missing values for the rest of the features.

V. Conclusion

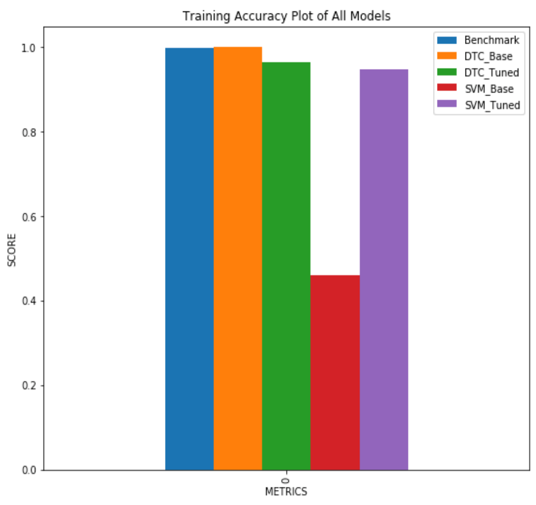
Free-Form Visualization

A few free form visualizations were provided in the earlier sections including a scatter matrix depicting visual correlation for the selected features, a histogram for the selected features, a histogram for the count of missing values for each of the features, and the accuracy plot for each of the models. The counts and visuals have helped refine the data by identifying the features (X21 and X37) with significant missing values.

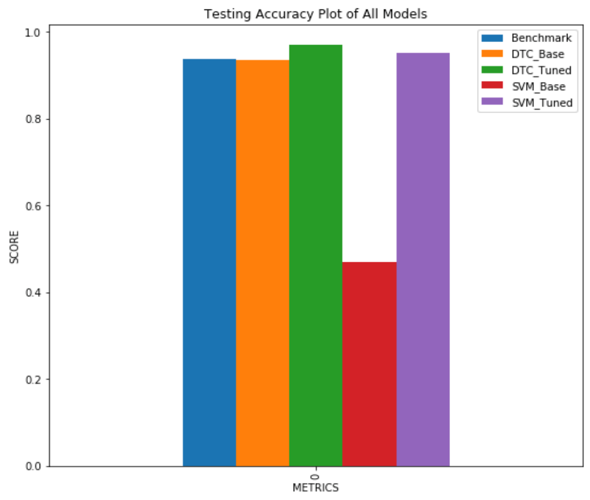
The plots below provide a side-by-side comparison of measures of accuracy for the models. These graphs provide a visual basis for selecting the best model.

The plots below provide the bar charts for each of metrics (Training Accuracy, Testing Accuracy, FBeta Score for Training, and FBeta Score for Testing) for comparing the metric across all the models (Benchmark, Base DecsionTreeClassifier, Tuned DecsionTreeClassifier, Base SVM, and Tuned SVM). We can readily see that the Tuned DecsionTreeClassifier (green bar in top right grid) has the highest accuracy for testing and is the winning model measured by accuracy.

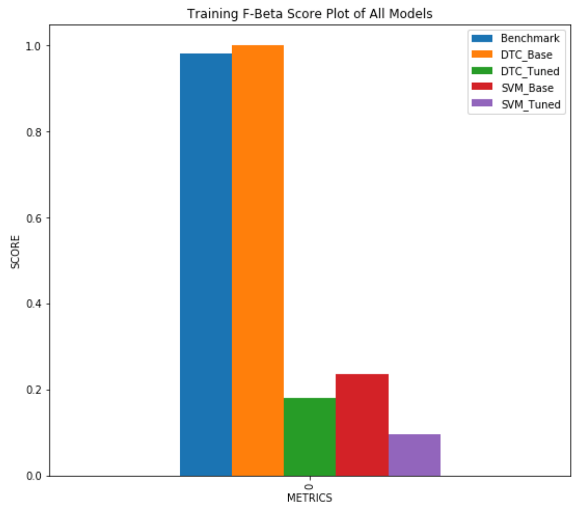
The FBeta scores are generally high for the benchmark models and the base models compared with tuned models and are higher for training data set compared with testing dataset.



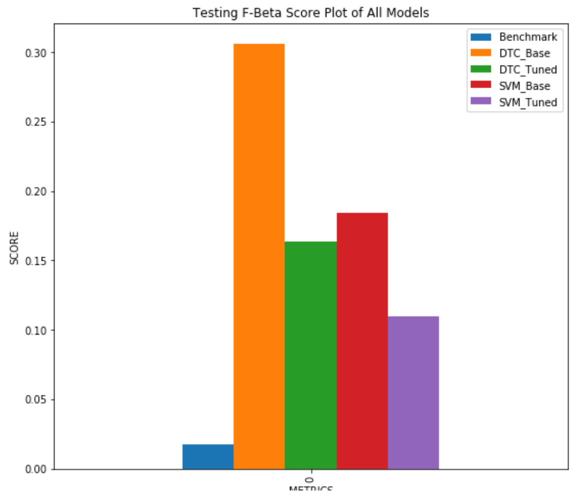
Training Accuracy for All Models



Testing Accuracy for All Models



Training FBeta Score



Testing FBeta Score

Reflection

The process of identifying the data set for the problem at hand was quite challenging until the discovery of the data set used in the model. After formatting the data into the desired CSV format, it is pre-processed to remove features with significant missing values and the other missing values are populated with average for each of the features.

This raw data of 62 features is further transformed into 16 PCA components that explain about 95% of the variance. These reduced dimensions are used to train and test base SVM and DecsionTreeClassifier models. The same dimensions are used to define tuned SVM and DecsionTreeClassifier. Measurement metrics accuracy and fbeta_socre are calculated for all the models.

A benchmark model defined by DecsionTreeClassifier and select features is evaluated for accuracy and fbeta_score. An evaluation model, DecsionTreeClassifier is created with tuned parameters and evaluated for metrics using data set with all the NaN data values removed from the data set, resulting in reduced number of companies used for testing and training the model.

The model with best accuracy is selected as the best bankruptcy predictor. The Tuned DecsionTreeClassifier is the best predictor.

The idea of reducing the number of dimensions using PCA while still capturing the essence of the dataset was quite interesting. After the scaling has been performed, the operating data size is reduced considerably and intuitively the response time for training and testing is relatively smaller.

The model has been quite satisfactory and has emerged as the best even after comparing with the evaluation model and can be great tool among the tool kit of practicing finance professional across different sectors of the industry.

Improvement

The model has been quite satisfactory and has emerged as the best even after comparing with the evaluation

While the model corroborates that the bankruptcy can be predicted with high accuracy, there are more improvements that can be done to the model.

The model can be enhanced to iterate over multiple values of Beta, capture the metrics for each of the defined model alternatives, and plot them for identifying the best performing model. The other improvement would be clearly identify all the outliers, remove them, and compare the results of the model with the best model selected. Although the data has been scaled using MinMaxScaler to [0,1] it is still possible to have outliers relative to other scaled values in each of the features.

Other classification algorithms defined in Ensemble Methods, clustering techniques, or neural networks can be used to experiment and classify companies with potential bankruptcy.

When used as a benchmark, the model can be compared with those models trained using neural networks. The resulting model can potentially yield better performance.

Ref: http://koasas.kaist.ac.kr/bitstream/10203/3705/1/1996-082.pdf