ISyE 6740 – Summer 2023
Project Report

Team Member Names: Enhong Liu,Yong Yan (Karen) Zhu,Wen Yu Ho

Project Title: Predicting Company Bankruptcy through Machine Learning using Data from Taiwan Stock Exchange

Contents

Problem Statement

The occurrence of company bankruptcies poses a critical problem with far-reaching consequences, including job losses, economic instability, and financial setbacks for investors. Recent statistics indicate that over 6 million Americans between May 2020 and June 2021 were unable to work because employer closed or lost business, as reported by the U.S. Bureau of Labor Statistics. Additionally, research published by the University of Pennsylvania reveals that the bankruptcy of a company leads to a loss of approximately 20-45% for shareholders (Reindl et al., 2017). Given these detrimental effects, it is imperative to develop effective methods for predicting company bankruptcy, as it would aid in risk assessment, loss mitigation, and stabilization of employment and economics. Furthermore, the urgency to address this issue has been heightened in the wake of the COVID-19 pandemic. The global economy has experienced unprecedented disruption, with numerous industries facing challenges they have never encountered before. Reduced consumer spending, supply chain disruptions, and temporary closures have had a profound negative effect on the financial stability of businesses. As a result, a significant number of companies have faced financial difficulties, unable to sustain their operations and leading to a surge in bankruptcies or financial restructuring. Given the widespread impact of the pandemic on the global economy, predicting company bankruptcy has become even more critical. It is essential to develop robust and accurate models that can assess the financial health of companies in these unprecedented times.

Much effort has been put into developing prediction models; however, predicting company bankruptcy is still a complex and challenging task, due to the numerous factors and variables involved. Traditional financial analysis methods, such as financial ratio analysis, cash flow analysis, and qualitative assessment (e.g., multiple discriminant analysis), have been the primary approaches employed in the field. However, these methods have certain limitations and may not adequately capture the dynamic nature of financial data and the intricate interrelationships between variables. To address the limitations of traditional modeling methods, we propose utilizing advanced machine learning techniques and predictive modeling algorithms for predicting company bankruptcy. Recent advancements in machine learning algorithms and increased computing power have made it possible to model complex financial data more accurately and efficiently. Additionally, the availability of large-scale financial datasets and alternative data sources provides a rich resource for training and validating predictive models

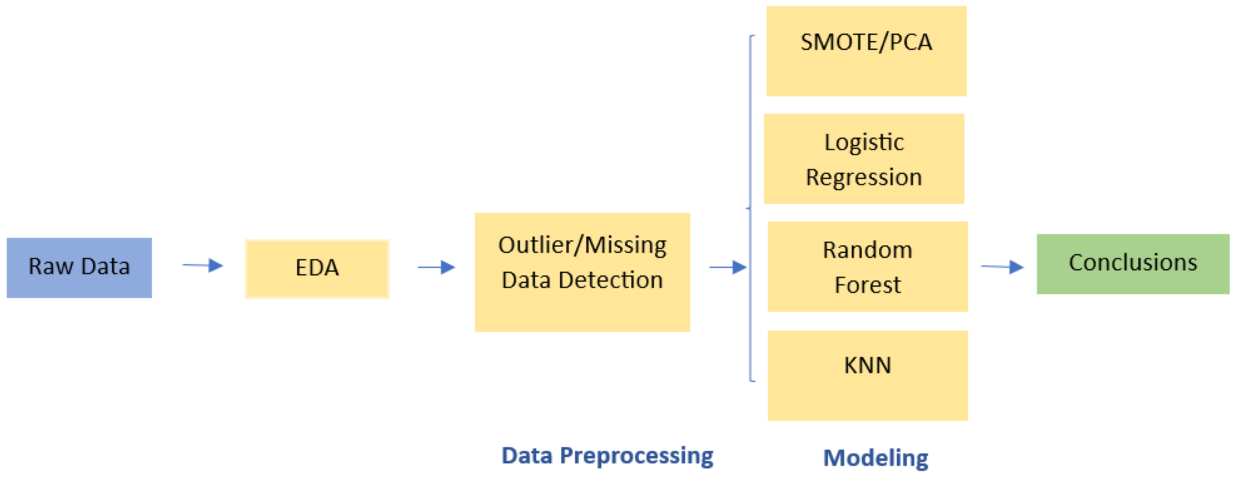
In a literature review on bankruptcy prediction (Narvekar and Guha, 2021), various machine learning models, such as logistic regression, random forest, k-nearest neighbor, gradient boosting, and neural networks, have been explored. However, none of these models were investigated in combination with PCA or feature selections, which has significantly impeded further performance improvements in these models. Many of the existing models assume linear relationships in the data and necessitate the construction of interaction variables with high polynomials, leading to extensive computation time when dealing with a substantial number of independent variables. To address these challenges, we propose deploying PCA and potential kernels to transform our linear models and capture non-linear patterns in the data, potentially alleviating the need for strict linear assumptions. Additionally, incorporating feature selection can also help reduce noise and enhance model performance. By synergizing these techniques – feature selection, PCA, and SMOTE – with the 3 machine learning models (logistic regression, random forest, and k-nearest neighbor), we aim to substantially improve the efficacy and efficiency of bankruptcy prediction, presenting a cutting-edge solution for this significant problem.

The overarching goal of this project is to develop and implement advanced machine learning techniques and predictive modeling algorithms to further improve the efficacy and efficiency of predicting company bankruptcy. Successful implementation of the proposed predictive models will have significant impacts across various sectors. It will enable businesses to proactively identify financial distress and take corrective actions to avoid bankruptcy, thus ensuring their survival and preserving jobs. Investors will be empowered to make more informed decisions, mitigating the risk of financial loss. Financial institutions can enhance their risk assessment processes and make sound lending decisions, which ultimately contribute to overall financial stability.

Overview of Data

Our dataset originates from Kaggle Company Bankruptcy Prediction, which the author further sources the data from UCI Machine Learning Repository. It was used in the study "Financial Ratios and Corporate Governance Indicators in Bankruptcy Prediction: A Comprehensive Study "published in the European Journal of Operational Research in 2016. The data contains financial information and bankruptcy status of ~7000 Taiwanese companies that were on the Taiwan Stock Exchange from 1999-2009. The companies belonged to various industries including manufacturing, electronics, service, and more. For each company, there are 97 input variables including total asset turnover, current liability to assets, equity to liability, and more (Referred in Appendix). Each of the variables has been normalized into the range from 0 to 1. Additionally, there is one dependent variable indicating the bankruptcy status, 0 viable and 1 bankrupt.

Methodology



Exploratory Data Analysis

Commencing our analysis, an initial step involved undertaking exploratory data analysis (EDA) to cultivate a comprehensive understanding of our datasets. Employing conventional tools such as correlation maps, label counts, and null value investigations, we sought to illuminate the inherent structure of the data. It is imperative to emphasize that EDA represents an iterative process, inextricably linked with the subsequent stages of data pre-processing and model construction.

Data Pre-processing

In this project, a rigorous set of methodologies was deployed to validate and prepare the dataset for subsequent analysis. Initially, a comprehensive assessment of missing data points (NAs) was conducted, and there were no missing data values. Following that, outliers were detected by isolating data points that fell outside the range of plus or minus 3 standard deviations from the mean. Despite the detection of outliers in our dataset, we made a cautious decision not to remove them. The reason behind this choice was the lack of concrete evidence to confirm that these data points were true outliers. Instead of hasty removal, we chose to retain the outliers during our analysis to avoid potential data loss or bias. By keeping these data points, we maintained the integrity of the dataset and ensured that our subsequent analyses and models would not be skewed by premature data manipulation. It also allowed us to consider the possibility that these seemingly unusual observations might hold valuable and legitimate insights into the underlying patterns or characteristics of the data.

Model Building and Optimization

Training/test data Splitting: In the pursuit of developing robust and reliable predictive models, we conducted a principled training/test data splitting procedure, adhering to the widely acknowledged ratio of 75/25. Through this approach, the dataset was partitioned into two distinct subsets, wherein 75% of the data was designated for model training, and the remaining 25% was reserved for rigorous evaluation. This division ensures that a substantial portion of the data is dedicated to model training, enabling the learning of intricate underlying patterns and relationships within the dataset. The test subset, consisting of unseen data, serves as an independent benchmark for assessing the model's generalization capacity.

SMOTE: During the model building phase, we addressed the issue of class imbalance. Only 2-3% of companies within the dataset being classified as bankrupt. Failure to address this concern prior to model development would inadvertently yield models with a seemingly impressive prediction accuracy of 96%. To rectify this discrepancy, we employed the Synthetic Minority Oversampling Technique (SMOTE), a resampling method that generates synthetic samples pertaining to the minority class (Chawla et al., 2002). It allows for a more equitable representation.

PCA and Kernel PCA: Another potential issue existing in this dataset was the presence of interdependencies among the data, particularly the financial information pertaining to companies, serving as the input variables for our analysis. Notably, various financial ratios exhibited entwined relationships, sharing common financial variables, or entailing reliance on specific market factors such as taxation and interest rates. The financial dynamics of businesses are intricately intertwined, whereby heightened revenues are oftentimes accompanied by correspondingly elevated expenses, including operational costs. As a result, the correlation among these variables often violates certain model assumptions and introduces unnecessary noise that hinders the interpretability of the resulting models. To redress the issue of correlation, we conducted variable selection or shrinkage techniques, ridge regularization, lasso regularization and elastic net regularization prior to model construction. Moreover, we also employed cross-validation, up-sampling or down-sampling procedures, feature standardization, and, in certain instances, the extraction of preeminent eigenvectors through Principal Component Analysis (PCA). PCA, widely recognized as a dimensionality reduction method, helps identify significant features, making it possible to extract relevant information more effectively and efficiently. This technique is especially valuable for addressing the issue of multicollinearity *and non-linear pattern* inherent in this dataset. For scenarios characterized by nonlinearity, kernel methods such as 'cosine', 'sigmoid', 'rbf', were implemented. These kernel tricks implicitly construct interaction variables in high dimensions to better capture complex non-linear relationships within our data.

Logistic Regression over Regularization: Based on our Exploratory Data Analysis and the objective of predicting bankruptcies, we initially employed logistic regression, one of the most common supervised classification models. However, due to the high number of predictors, we anticipate encountering correlated variables and a substantial amount of noise. These factors are likely to have a negative impact on our model's performance and may also violate the collinearity assumption of logistic regression. To address these challenges, we explored different approaches. First, we used PCA eigenvectors, which are orthogonal and can help alleviate the non-collinearity assumption. Additionally, we employed regularization techniques to aid in feature selection and shrinkage, particularly given the presence of noisy data. Lastly, we explored the use of kernel tricks, which have the potential to capture high-dimensional non-linear relationships with complex boundaries in the data. By combining these approaches, we enhanced the robustness and effectiveness of our bankrupt prediction model.

Random Forest: Taking advantage of the unique benefits provided by the random forest paradigm, we proceeded to build models using both the unscaled full dataset and the principal components obtained through dimension reduction. Unlike the underlying assumption of linearity in logistic regression, random forest models overcome this constraint and demonstrate the ability to effectively capture any nonlinearity that may exist within the data.

K-Nearest Neighbors (KNN): The KNN algorithm was enlisted to delineate a dichotomous categorization of companies, effectively partitioning them into the classes of bankrupt and viable. Leveraging the notion of nearest neighbors, the KNN model classified new data points or companies based on the prevailing status of their proximate counterparts. Determination of the optimal number of neighbors (k), appropriate weights methods ('uniform' and 'distance'), and optimal algorithms among ['auto', 'ball_tree', 'kd_tree', 'brute'] was effectuated through the judicious application of cross-validation methodologies.

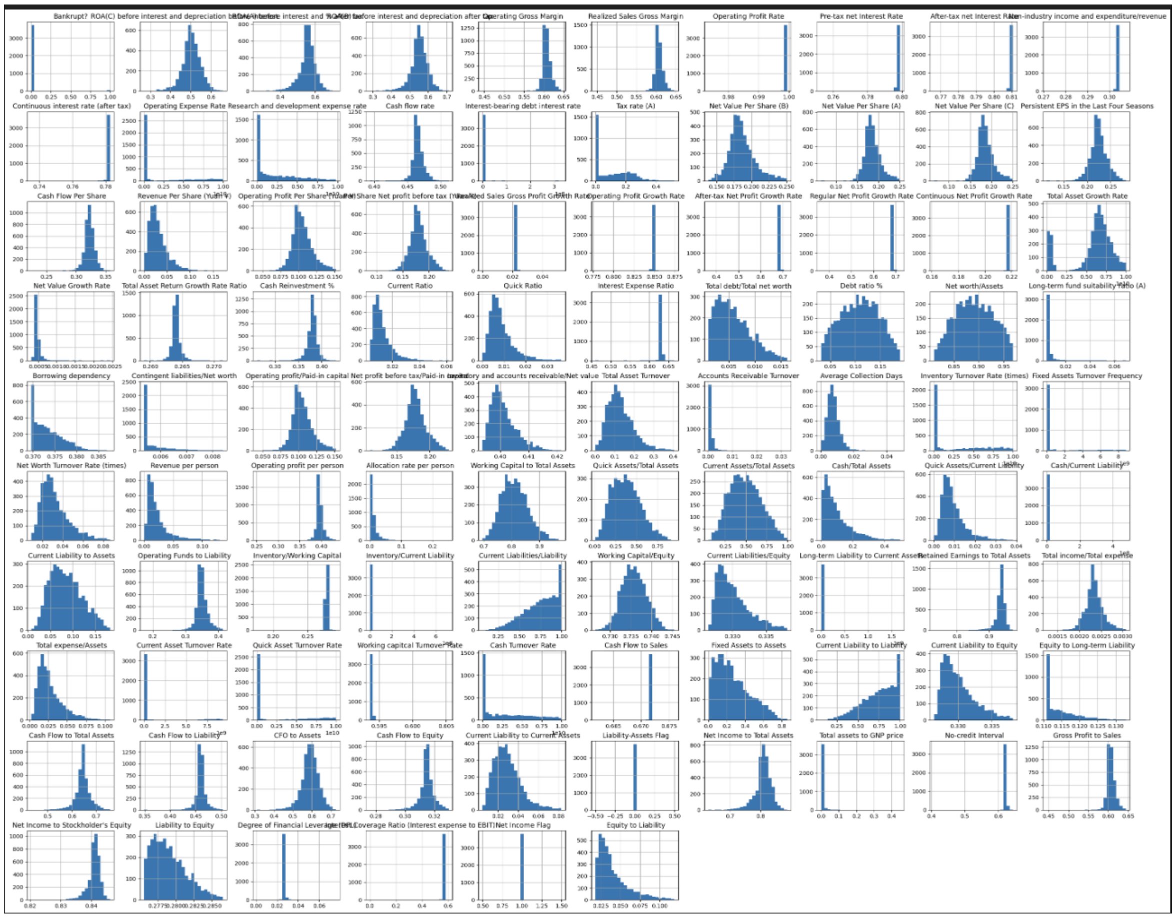
Model Performance Evaluation: In our efforts to improve bankruptcy detection, we made a thoughtful decision to prioritize the recall rate for several reasons. Firstly, in an imbalanced dataset like bankruptcy classification, the recall rate focuses on correctly identifying instances of the minority class, which is crucial for detecting company bankruptcies. Secondly, it helps minimize false negatives, which is vital in financial analysis, where missing critical bankruptcy cases can be costly. Moreover, it aids in identifying potential risks in screening and safety-critical systems, enhances early warning systems by swiftly detecting financial distress, and ensures accurate detection of rare events like bankruptcies in various industries. Overall, the recall rate plays a vital role in enhancing our company bankruptcy detection effectiveness and reducing the risks associated with false negatives. To holistically evaluate the performance of our bankruptcy detection model, we also utilized the confusion matrix as a comprehensive set of metrics. This allowed us to gain a thorough understanding of the model's predictive capacity and the robustness of its outcomes. For deeper insights into the model's discriminatory ability and its capability to distinguish between classes at different recall levels, we employed a Precision-Recall Curve (PR-AUC) instead of the traditional Receiver Operating Characteristic-Area Under the Curve (ROC-AUC) plot. This choice was made due to the imbalanced nature of the dataset and the focus on predicting bankruptcies, which represent only 2-3% of our dataset.

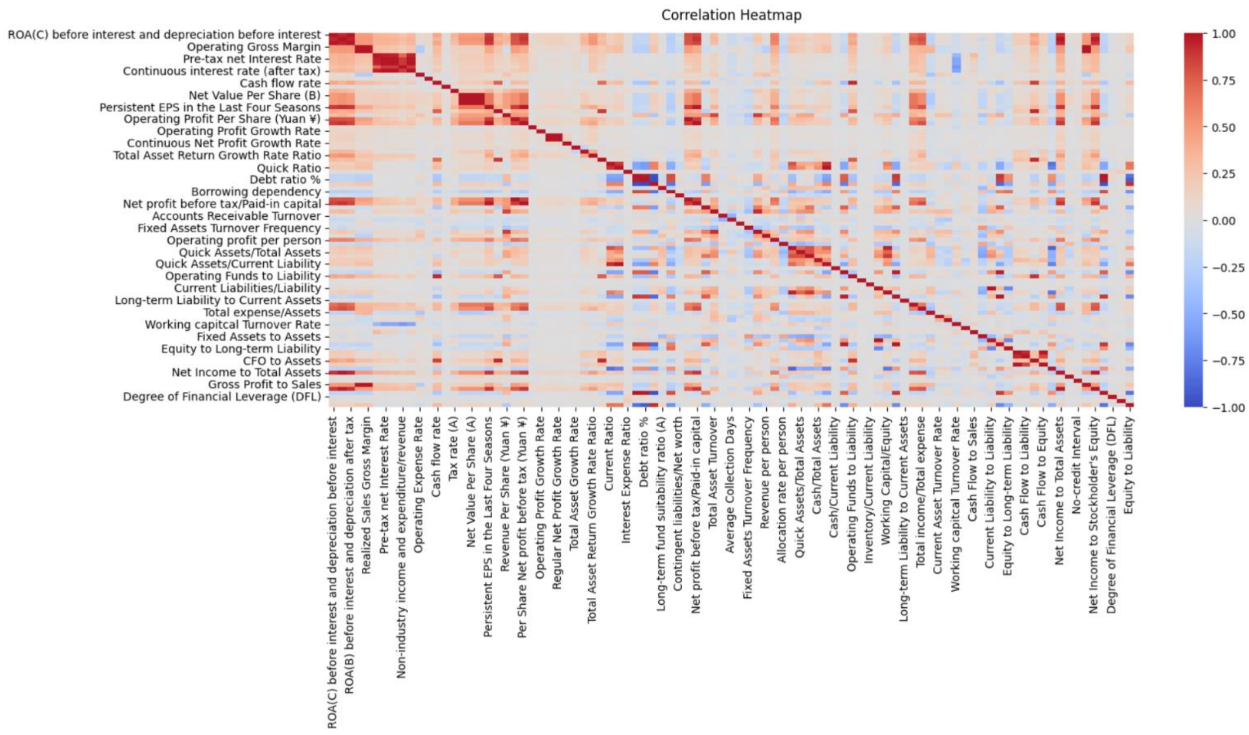
Results and Discussions

Exploratory Data Analysis

We investigated the types of data present in the dataset, identifying whether they were categorical, numerical, or other specific types. All variables were numerical except for 2 categorical variables, and we removed the categorical variables before further data analysis since they would not add any additional information to our models.

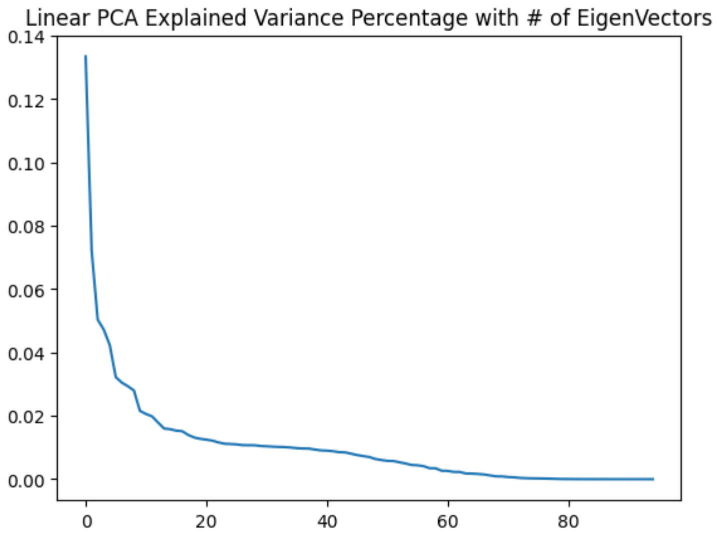
We also examined the distribution of each variable as shown in the histograms below, and the variables had various distributions. Furthermore, we leveraged correlation matrix heatmaps to explore the relationships between variables. Many variables were correlated, and the multicollinearity of variables could affect model performance. We built models in which the effect of multicollinearity variables would be mitigated. Overall, through this thorough examination of the data's type, distribution, and correlation, we laid the foundation for making informed decisions and constructing meaningful statistical models to extract valuable insights from the dataset.





Model Performance

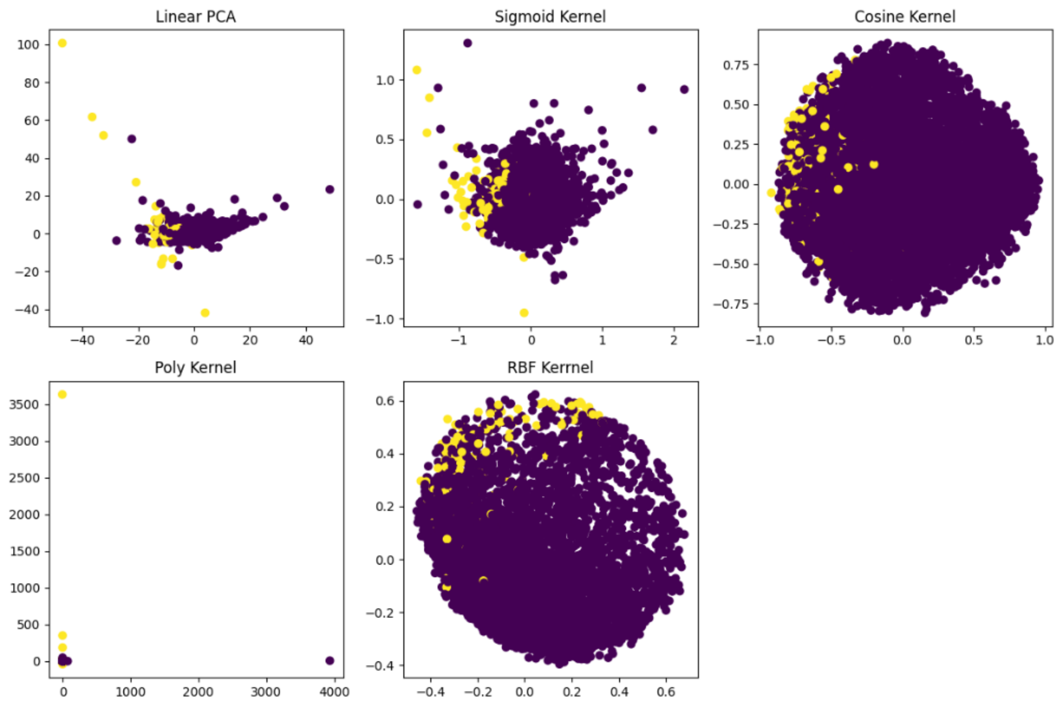
Before building our models with PCA, we examined if there was a noticeable eigengap in the explained variance ratio. An eigen gap is a significant drop in the variance explained by each principal component and identifying it helps us determine the optimal number of principal components to retain. As shown in the plot below, the explained variance ratio obtained from the PCA did not exhibit a distinct eigen gap, as we would typically observe in the elbow method visual. This lack of a clear eigen gap made it challenging to easily determine the optimal number of eigenvectors to include when building our models.



Nevertheless, to proceed with our model-building process, we have decided to select 20 as the initial number of eigenvectors. While this choice is not based on a clear turning point in the variance explained, it served as a reasonable starting point for our analysis and can be adjusted as we fine-tune our models.

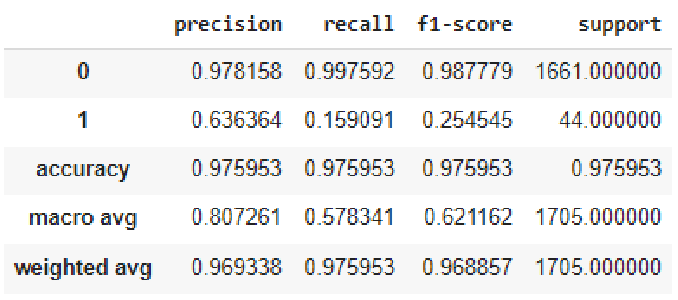
We conducted visualizations using different kernels to determine whether a linear model would suffice or if we should explore the incorporation of kernel tricks to enhance our models. The results of these plots suggested that employing kernel tricks could be beneficial, but the improvement might not be substantial. From the visuals, it appears that "Linear" PCA, "Sigmoid," "Cosine," and "RBF" kernels all seem reasonable choices, while the "Poly" kernel is expected to perform the worst.

As a result, our team decided to transform the scaled dataset using various kernels and then use the transformed data for model training. This approach will enable us to compare the performance of different models and assess the impact of kernel tricks on our overall predictive accuracy.

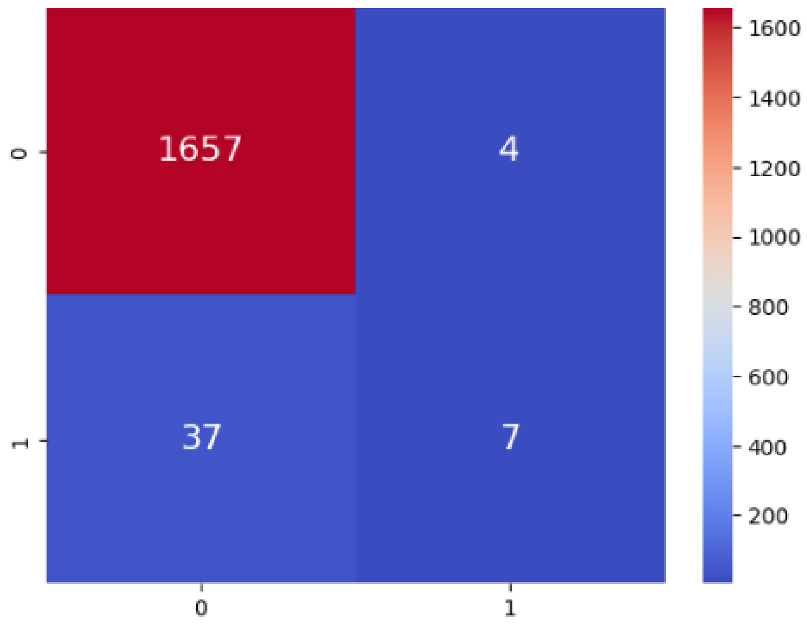


Logistic Regression over Regularization and Kernel PCA:

The initial model employed only data standardization with Logistic Regression and cross validation. Unfortunately, the model yielded poor results, especially a recall score of 0.16. It appears that the imbalanced data negatively impacted the model's performance, as most of the true bankrupt companies were misclassified as non-bankrupt in the confusion matrix.



	precision	recall	f1-score	support
0	0.978158	0.997592	0.987779	1661.000000
1	0.636364	0.159091	0.254545	44.000000
accuracy	0.975953	0.975953	0.975953	0.975953
macro avg	0.807261	0.578341	0.621162	1705.000000
weighted avg	0.969338	0.975953	0.968857	1705.000000



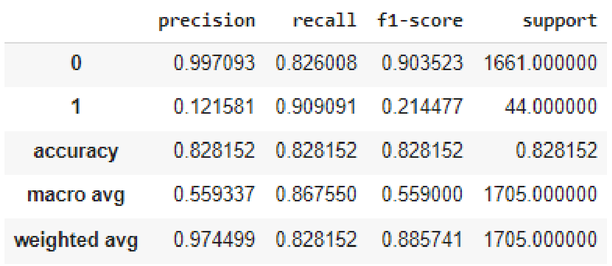
Based on our previous analysis and model output, we confirmed that using Synthetic Minority Over-sampling Technique (SMOTE) or "class_weight" parameter was a necessary step to ensure our model from overly focused on non-bankrupt (class label = 0) observations. As for the rest of the model tuning, we proceeded with SMOTE instead of class_weight parameter, as SMOTE showed slightly better overall performance when tuning other parameters.

Our main approach of tuning was through using LogisticRegressionCV with Ridge, Lasso or Elastic Net regularization. In our model comparison, Elastic Net and Lasso models overall performed slightly better than the Ridge model. A possible explanation is that some of the variables might have contributed too much noise and the high correlation of some variables could also have an impact.

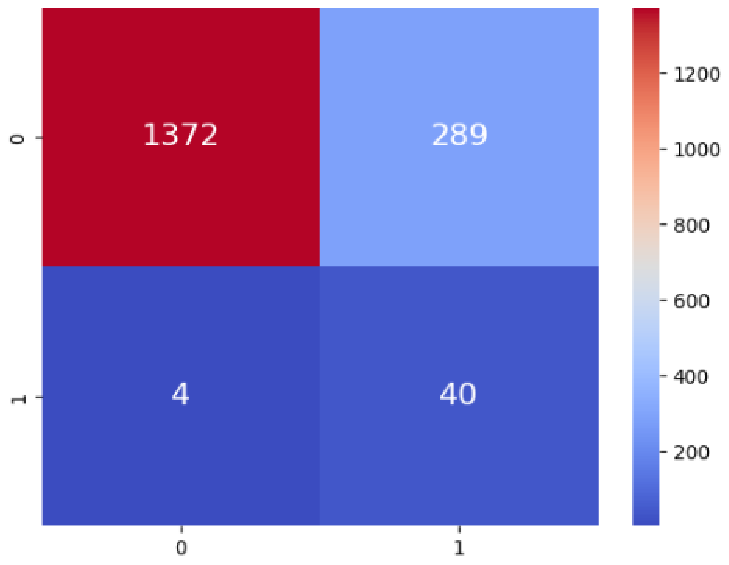
Some hyperparameters we explored included:1. Inverse regularization strength - We increased or decreased the strength manually, but the best model used a value automatically determined by function; 2. Cross validation – We used Cv = 5 for all Logistic Regression, Random Forest, and KNN models to identify the best set of hyperparameters for the training data set; 3. Solver: ["saga", "liblinear"] - We mainly used the "saga" solver as it was the only solver that could handle all three types of regularization.

In addition to tuning hyperparameters within LogisticRegressionCV, we explored different methods of pre-processing data for model training. Instead of solely using a scaled dataset, we used eigenvectors from various kernels, including "Linear," "Sigmoid," "Cosine," "Poly," and "RBF" as input data for model building. Eigenvectors offer the advantage of orthogonality, helping alleviate non-collinearity assumptions in logistic regression and stabilize model coefficients. After experimenting with eigenvector numbers ranging from 10 to 60, around 20 eigenvectors performed best, possibly due to excessive noise from large number of eigenvectors. While kernel models slightly outperformed linear PCA under high recall requirements, the difference was minor. For low recall requirements, logistic regression with linear PCA seemed more effective, suggesting a linear separable approach may suffice for future analysis.

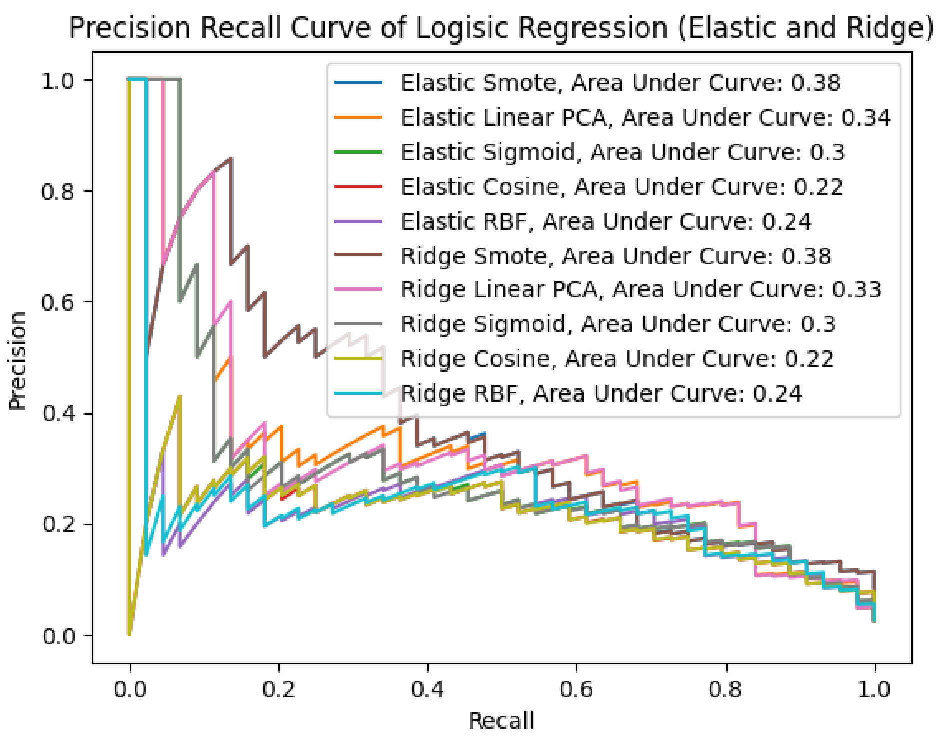
The best model (with a common 50% classification threshold) is a Logistic Regression with Elastic Net regularization (l1_ratios=0.5) that was trained with top 20 eigenvectors from the "RBF kernel" principal components. The data was transformed with SMOTE. This model has a 0.91 recall score, which outperforms other logistic regression models with recall scores hovering around 0.15 to 0.89. The next best model also used "RBF kernel" but only use Lasso regularization.



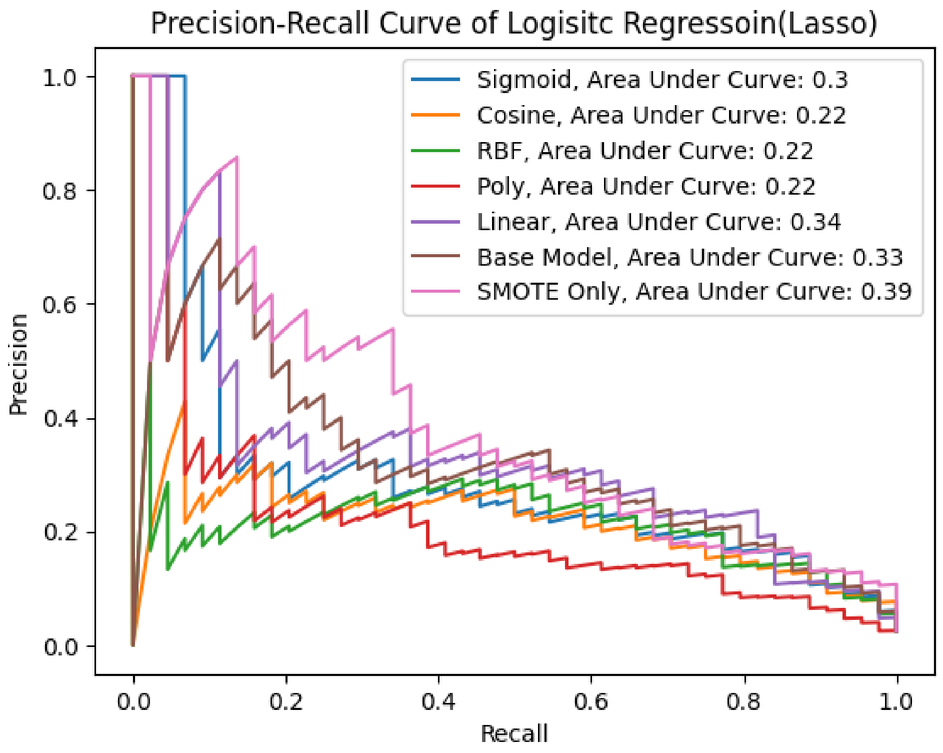
	precision	recall	f1-score	support
0	0.997093	0.826008	0.903523	1661.000000
1	0.121581	0.909091	0.214477	44.000000
accuracy	0.828152	0.828152	0.828152	0.828152
macro avg	0.559337	0.867550	0.559000	1705.000000
weighted avg	0.974499	0.828152	0.885741	1705.000000



Although we have selected which model performed best, there are some important caveats to consider. While our best model performs well using the common classification threshold of p=0.5, it has comparable performance to other models in certain scenarios. Specifically, when we demand a higher recall requirement and decrease the classification threshold to below 0.5, its performance becomes almost indistinguishable from the other models. Hence, if potential stakeholders demand a higher or lower recall score, we should evaluate our models based on the Precision-Recall (PR AUC) plot instead of relying solely on recall score. In the PR AUC plot, we plot precision on y-axis and recall on x-axis, then we calculate the area under curve for each model. We will notice that most models are indistinguishable when we have high recall requirements. When we have low recall requirements, however, linear models (without any kernel tricks) start to perform well as shown in the visual below. Therefore, we can still opt for linear model with original predicting variables instead of eigenvectors, where we can present the model in a straightforward and interpretable format "y=mx+b" without losing much on model performance.



Precision Recall Curve of Logisic Regression (Elastic and Ridge)
Elastic Smote, Area Under Curve: 0.38
Elastic Linear PCA, Area Under Curve: 0.34
Elastic Sigmoid, Area Under Curve: 0.3
Elastic Cosine, Area Under Curve: 0.22
Elastic RBF, Area Under Curve: 0.24
Ridge Smote, Area Under Curve: 0.38
Ridge Linear PCA, Area Under Curve: 0.33
Ridge Sigmoid, Area Under Curve: 0.3
Ridge Cosine, Area Under Curve: 0.22
Ridge RBF, Area Under Curve: 0.24
Precision
Recall



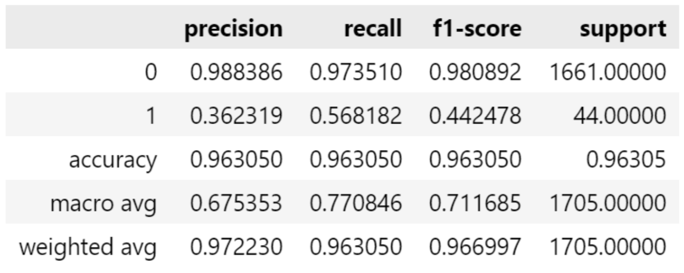
Precision-Recall Curve of Logisitc Regressoin(Lasso)
Sigmoid, Area Under Curve: 0.3
Cosine, Area Under Curve: 0.22
RBF, Area Under Curve: 0.22
Poly, Area Under Curve: 0.22
Linear, Area Under Curve: 0.34
Base Model, Area Under Curve: 0.33
SMOTE Only, Area Under Curve: 0.39
Precision
Recall

Random Forest:

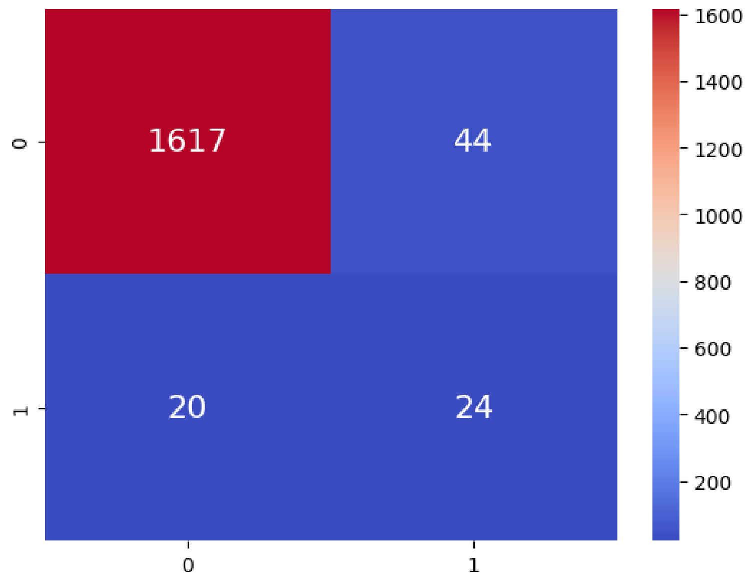
We built two random forest models, a base model where all default parameters were used and a tuned model. For the tuned model, we varied the following parameters, ‘n_estimators’, ‘min_samples_split’, ‘min_samples_leaf’, and ‘max_depth’, ‘n_estimators’ is the number of trees used to build the random forest model. ‘min_samples_leaf’ is the minimum number of samples required to be a at a leaf node. ‘max_depth’ defines the maximum depth of the decision trees in the random forest. Optimizing these parameters can improve model performance, particularly recall, and avoid underfitting/overfitting. We focused on recall since we think it is more important to correctly classify bankrupt companies than viable companies. We used RandomSearchCV to randomly sample the hyperparameters and narrow down the range. Then, we performed an exhaustive search to find the best parameters with GridSearchCV. The optimal hyperparameters are {'max_depth': 110, 'min_samples_leaf': 1, 'min_samples_split': 6, 'n_estimators': 100}. Both models performed very similarly in terms of precision and recall. For viable companies, the precision is ~0.98 and recall is 1.00. For bankrupt companies, the precision is ~0.53 and recall is ~0.16.

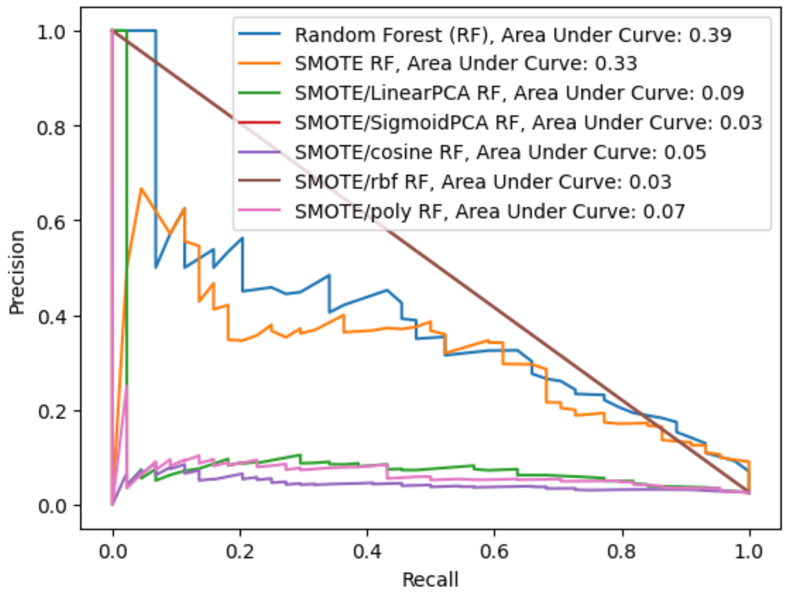
We hypothesized that the initial models predicted bankrupt companies poorly since there is class imbalance in the datasets. We performed SMOTE on the dataset to generate samples for the minority class (e.g., bankrupt). After performing SMOTE, we explored the following models: 1) random forest; 2) random forest with linear PCA, which sought to reduce feature dimensionality through Principal Component Analysis; and 3) random forest with kernel PCA, encompassing cosine, sigmoid, and radial basis function (rbf) kernels to improve data separability. The precision and recall were similar when comparing the base model to its tuned models. The same hyperparameters were used to optimize the models as mentioned above. The detailed outcomes of these variant models are presented in the Appendix for reference. Since both the base and tuned models performed similarly, only the base models are included in the Appendix.

The best model with the greatest recall value combines SMOTE with random forest. The recall score is ~0.57. The precision recall curve shows this model has an area under the curve of 0.33. The precision recall curve also shows the other model variants, and the area under the curve is <0.1 for most models.



	precision	recall	f1-score	support
0	0.988386	0.973510	0.980892	1661.00000
1	0.362319	0.568182	0.442478	44.00000
accuracy	0.963050	0.963050	0.963050	0.96305
macro avg	0.675353	0.770846	0.711685	1705.00000
weighted avg	0.972230	0.963050	0.966997	1705.00000



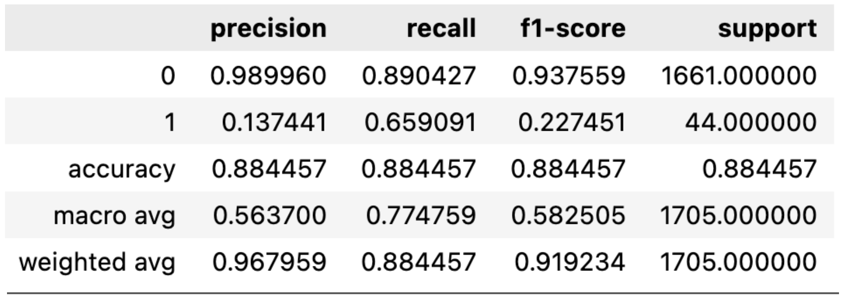


K-nearest neighbors (KNN):

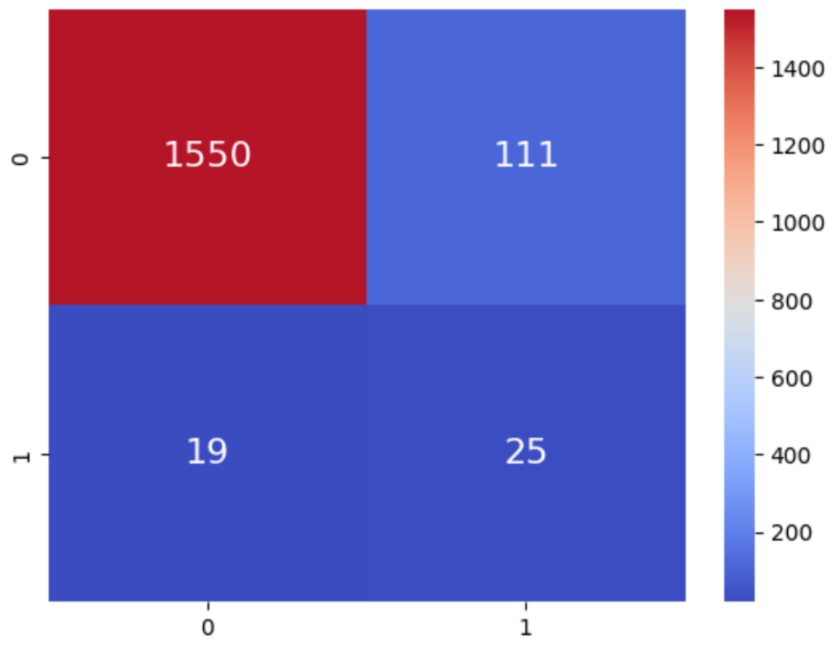
A comprehensive set of 95 features was incorporated. Employing the GridSearchCV technique, the optimal hyperparameters were determined to be {'algorithm': 'auto', 'n_neighbors': 11, 'weights': 'distance'}. Under these conditions, the model demonstrated satisfactory performance, with an overall accuracy of 0.82, overall recall rate of 0.82, and overall precision rate of 0.95. However, when specifically examining the bankruptcy class, the model exhibited a lower recall rate of 0.47 and precision rate of 0.07. Further details of the result can be found in the Appendix.

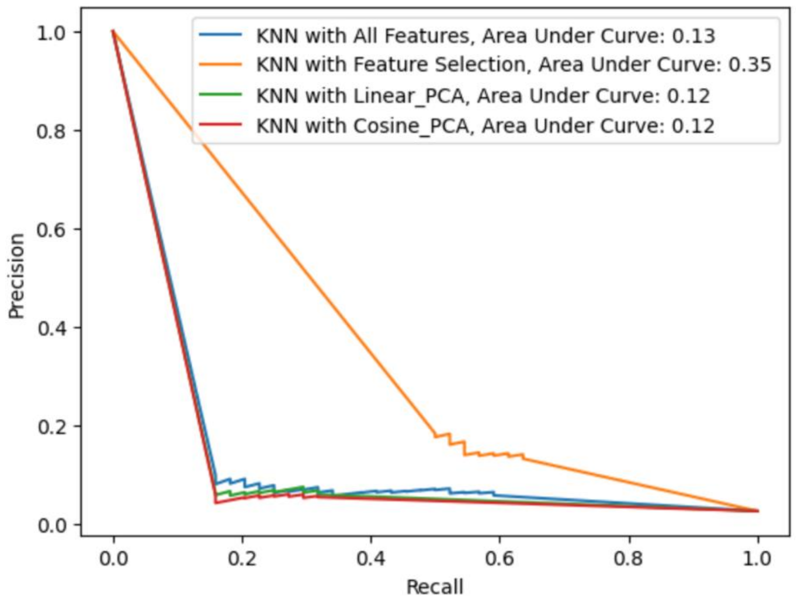
To enhance the predictive capabilities of the KNN model, various variants were explored. These included: 1) KNN with top 20-30-50 selected features, where the selection was based on the variation of class explained by each feature; 2) KNN with linear PCA, which sought to reduce feature dimensionality through Principal Component Analysis; and 3) KNN with kernel PCA, encompassing cosine, sigmoid, and radial basis function (rbf) kernels to improve data separability. By employing GridSearchCV, the optimal hyperparameters for each variant were identified, resulting in the attainment of the best performance under each specific circumstance. The detailed outcomes of these variant models are presented in the Appendix for reference.

The culmination of this investigation revealed that the most effective KNN model was the one utilizing the top 30 selected features in conjunction with the Synthetic Minority Over-sampling Technique (SMOTE). The tuned hyperparameters for this model were identified as {'algorithm': 'auto', 'n_neighbors': 5, 'weights': 'distance'}. The performance of this optimized KNN model exhibited substantial improvement, boasting an overall accuracy of 0.88, overall recall rate of 0.88, and overall precision rate of 0.97. Specifically, in the context of bankruptcy prediction, the model achieved a recall rate of 0.66 and a precision rate of 0.14. Further details regarding these outcomes can be found below, followed by the confusion matrix. Additionally, the precision- call plot demonstrated the same results, where KNN with top 30 features selected performed the best, compared to other KNN variants models.



	precision	recall	f1-score	support
0	0.989960	0.890427	0.937559	1661.000000
1	0.137441	0.659091	0.227451	44.000000
accuracy	0.884457	0.884457	0.884457	0.884457
macro avg	0.563700	0.774759	0.582505	1705.000000
weighted avg	0.967959	0.884457	0.919234	1705.000000





Model comparison

The analysis of our results reveals that kernel PCA did not yield superior performance compared to linear PCA transformation in this case, which also suggested that our data is likely to exhibit linear separability in the high-dimensional space, elucidating why the logistic regression model combined with PCA transformation emerged as the best performing approach.

In this context, the advantage of K-Nearest Neighbors (KNN) and Random Forest models over logistic regression in separating non-linearly separable classes is outweighed by their evident disadvantages: 1. KNN heavily relies on the local distribution of data points, rendering it sensitive to outliers and noise, and as a consequence, in datasets containing noisy points or regions with varying class densities, which is highly likely the case in this financial dataset, KNN may produce suboptimal results; 2. On the other hand, although Random Forest models are potent and robust in classifying non-linearly separatable data, random forest may necessitate a larger volume of data to fully exploit their predictive potential. Consequently, in our specific case, the performance of both KNN and Random Forest models was inferior to that of logistic regression combined with linear PCA transformation.

Limitation and Further Optimization

The analysis of the provided dataset revealed several critical pieces of information that warrant attention for future improvements in our further endeavors. First, the dataset lacks industry labels, such as agriculture, food, health, technology, and others, which could serve as crucial contextual information for enhancing the accuracy and applicability of our models for specific industries. Secondly, the absence of a clear time period (e.g., 1999-2005, 2005-2009) hinders the ability to study temporal trends and could potentially limit the temporal generalizability of our findings. Furthermore, the dataset's size appears to be insufficient, containing only 7,000 records, which might impact the robustness and statistical power of our analyses. Moreover, the dataset's limited scope, restricted to a specific region (e.g., only companies in Taiwan Stock Exchange), might constrain the generalizability of our results to other contexts and geographic locations.

Considering these limitations, it becomes evident that substantial improvements are necessary for our prediction models. Research in this domain has demonstrated the importance of regular model updates, typically recommended every six years, to account for changing patterns, technological advancements, and shifts in industry dynamics. Looking ahead, it is imperative to acknowledge the evolving trends in this field. With the advent of new technologies, the integration of big data, and the emergence of cutting-edge analytical techniques, future research may focus on incorporating new elements to enhance the accuracy and predictive capabilities of our models. Moreover, interdisciplinary collaborations and global datasets could offer valuable insights, enabling a more comprehensive understanding of the dynamics influencing businesses across diverse industries and regions. By addressing these aspects, we can strive to elevate the quality and relevance of the outcomes in this ever-evolving landscape.

There are several aspects we aim to explore and further improve in our model:

1. Outlier Removal: In our analysis, we refrained from removing outliers as it resulted in a significant loss of data. However, we observed that using too many eigenvectors (40 to 60) or variables in our models led to worse performance than when using around 20 eigenvectors. This suggests the presence of substantial noise in our dataset. To address this, we can try other outlier removal techniques that preserve more data while reducing the impact of noise.
2. Predictor Reduction: We would like to reduce the number of predictors, especially in logistic regression. Attempting to convert the logistic regression model for statistical inference becomes challenging due to the high number of variables and noise in the data. To tackle this, we can use Random Forest's variable ranking to select the most important variables and rebuild our models accordingly.
3. Cost Tradeoff and Threshold Selection: Although not explicitly mentioned earlier, we can explore sample scenarios to determine the ideal threshold for recall score. This will allow us to make recommendations based on our Precision-Recall plot for selecting the most suitable model.
4. Unsupervised Models: We can explore unsupervised models such as clustering and one-class SVM to potentially uncover surprising classification results.
5. Addressing Imbalanced Data: The imbalanced dataset significantly affects model performance. While we implemented SMOTE (oversampling) and shifted our evaluation metrics towards recall score and Precision-Recall AUC plot, we believe further improvement is possible. We can investigate models that are adept at handling imbalanced datasets without the need for oversampling, or we might consider a combination of under-sampling and oversampling techniques. Additionally, we can also explore the effectiveness of stratified k-fold cross-validation, known to perform well for imbalanced datasets, though computational constraints may limit our options.

By exploring these areas, we can further refine our models and enhance the overall performance of our bankruptcy prediction system.

Conclusion

Our analysis and modeling efforts have provided valuable insights into predicting bankruptcies in companies. By leveraging advanced techniques such as Logistic Regression, Random Forest, and KNN, we determined that logistic regression best captured complexities in the data and predicted bankruptcy more accurately. The focus on recall rate as a critical criterion when building models ensures the accurate identification of bankruptcies, vital for financial analysis. Moving forward, we recommend incorporating industry labels and temporal context to enhance model accuracy and generalizability. Moreover, further improvements, including outlier removal techniques, predictor reduction, and exploration of unsupervised models, will refine our predictions. Addressing imbalanced data and fine-tuning model selection through sample scenarios will also contribute to more robust and relevant bankruptcy predictions, empowering better decision-making in this dynamic landscape.

Collaboration

All the three team members contributed equally to this project as of ideas and workloads

- Data sourcing/preprocessing: Enhong Liu, Yong Yan Zhu, Wen Yu Ho
- Proposal writing: Enhong Liu, Yong Yan Zhu, Wen Yu Ho
- Model building and optimization
    - Logistic regression: Wen Yu Ho
    - Random Forest: Yong Yan Zhu
    - KNN: Enhong Liu
- Final Report writing: Enhong Liu, Yong Yan Zhu, Wen Yu Ho

Reference

1. Bureau of Labor Statistics, U.S. Department of Labor, *The Economics Daily*, 6.2 million unable to work because employer closed or lost business due to the pandemic, June 2021 at https://www.bls.gov/opub/ted/2021/6-2-million-unable-to-work-because-employer-closed-or-lost-business-due-to-the-pandemic-june-2021.htm (visited *June 19, 2023*).
2. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357.
3. Liang, D., Lu, C. C., Tsai, C. F., & Shih, G. A. (2016). Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study. *European journal of operational research*, *252*(2), 561-572.
4. Narvekar, A., & Guha, D. (2021). Bankruptcy prediction using machine learning and an application to the case of the COVID-19 recession. *Data Science in Finance and Economics*, *1*(2), 180-195.
5. Reindl, J., Stoughton, N., & Zechner, J. (2017). Market implied costs of bankruptcy. *Available at SSRN 2324097*.

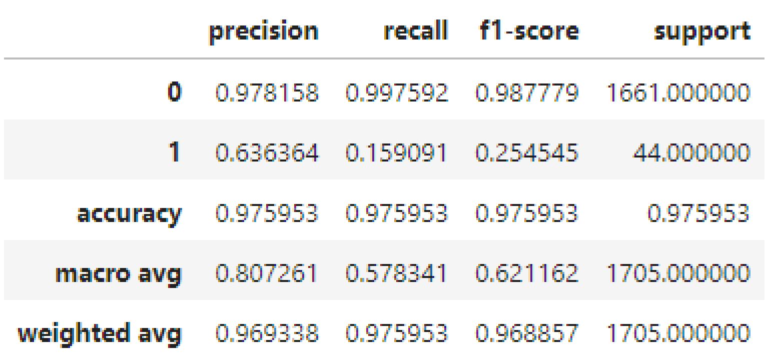
Appendix

Data variables:

X1	Cost of interest-bearing debt	X51	Return on total assets (C)
X2	Cash reinvestment ratio	X52	Return on total assets (A)
X3	current ratio	X53	Return on total assets (B)
X4	Acid test	X54	Gross profit/net sales
X5	Interest expenses/total revenue	X55	Realized gross profit/net sales
X6	Total liability/equity ratio	X56	Operating income/net
X7	Liability/total assets	X57	Pre-tax income/net sales
X8	Interest-bearing debt/equity	X58	Net income/net sales
X9	Contingent liability/equity	X59	Net non-operating income ratio
X10	Operating income/capital	X60	Net income-exclude disposal gain or loss/net sales
X11	Pretax income/capital	X61	EPS-net income
X12	Working capital to total assets	X62	Pretax income per share
X13	Quick assets/total assets	X63	Retained earnings to total assets
X14	Current assets/total assets	X64	Total income to total expenses
X15	Cash/total assets	X65	Total expenses to assets
X16	Quick assets/current liability	X66	Net income to total assets
X17	Cash/current liability	X67	Gross profit to sales
X18	Current liability to assets	X68	Net income to stockholder’s equity
X19	Operating funds to liability	X69	One if net income is negative for the last two years; zero otherwise
X20	Inventory/working capital	X70	(inventory + accounts receivables)/equity
X21	Inventory/current liability	X71	Total asset turnover
X22	Current liabilities/liability	X72	Accounts receivable turnover
X23	Working capital/equity	X73	Days receivable outstanding
X24	Current Liabilities/Equity	X74	Inventory turnover
X25	Long-term liability to current assets	X75	Fixed asset turnover
X26	Current liability to current assets	X76	Equity turnover
X27	One if total liability exceeds total assets; else, zero	X77	Current assets to sales
X28	Equity to liability	X78	Quick assets to sales
X29	Equity/total assets	X79	Working capital to sales
X30	(Long-term liability + equity)/fixed assets	X80	Cash to sales
X31	Fixed assets to assets	X81	Cash flow to sales
X32	Current liability to liability	X82	No-credit interval
X33	Current liability to equity	X83	Cash flow from operating/current liabilities
X34	Equity to long-term liability	X84	Cash flow to total assets
X35	Liability to equity	X85	Cash flow to liability
X36	Degree of financial leverage	X86	CFO to assets
X37	Interest coverage ratio	X87	Cash flow to equity
X38	Operating expenses/net sales	X88	Realized gross profit gross rate
X39	(Research and development expenses)/net sales	X89	Operating income growth
X40	Effective tax rate	X90	Net income growth
X41	Book value per share (B)	X91	Continuing operating income after tax growth
X42	Book value per share (A)	X92	Net income-excluding disposal gain or loss growth
X43	Book value per share (C)	X93	Total asset growth
X44	Cash flow per share	X94	Total equity growth
X45	Sales per share	X95	Return on total asset growth
X46	Operating income per share		
X47	Sales per employee		
X48	Operation income per employee		
X49	Fixed assets per employee		
X50	Total assets to G price		

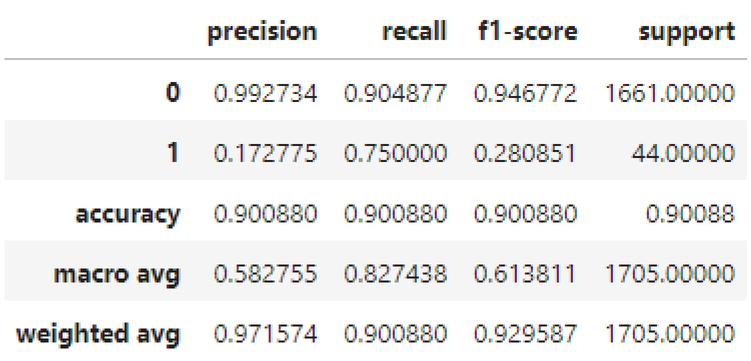
Logistic Regression Models:

- Initial Logistic Regression with original scale data:



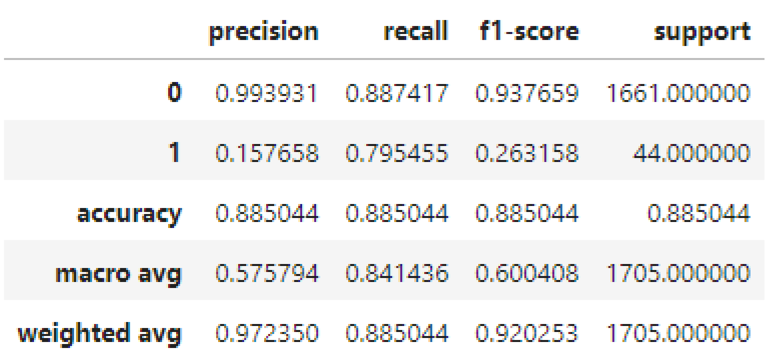
	precision	recall	f1-score	support
0	0.978158	0.997592	0.987779	1661.000000
1	0.636364	0.159091	0.254545	44.000000
accuracy	0.975953	0.975953	0.975953	0.975953
macro avg	0.807261	0.578341	0.621162	1705.000000
weighted avg	0.969338	0.975953	0.968857	1705.000000

- Logistic Regression with class_weight="balanced":



	precision	recall	f1-score	support
0	0.992734	0.904877	0.946772	1661.00000
1	0.172775	0.750000	0.280851	44.00000
accuracy	0.900880	0.900880	0.900880	0.90088
macro avg	0.582755	0.827438	0.613811	1705.00000
weighted avg	0.971574	0.900880	0.929587	1705.00000

- Logistic Regression with SMOTE data:



	precision	recall	f1-score	support
0	0.993931	0.887417	0.937659	1661.000000
1	0.157658	0.795455	0.263158	44.000000
accuracy	0.885044	0.885044	0.885044	0.885044
macro avg	0.575794	0.841436	0.600408	1705.000000
weighted avg	0.972350	0.885044	0.920253	1705.000000

- Logistic Regression with Linear PCA(n=20), Lasso:

	precision	recall	f1-score	support
0	0.977541	0.995786	0.986579	1661.000000
1	0.461538	0.136364	0.210526	44.000000
accuracy	0.973607	0.973607	0.973607	0.973607
macro avg	0.719540	0.566075	0.598553	1705.000000
weighted avg	0.964225	0.973607	0.966552	1705.000000

- Logistic Regression with Sigmoid PCA(n=20), Lasso:

	precision	recall	f1-score	support
0	0.996518	0.861529	0.924120	1661.00000
1	0.144981	0.886364	0.249201	44.00000
accuracy	0.862170	0.862170	0.862170	0.86217
macro avg	0.570750	0.873946	0.586661	1705.00000
weighted avg	0.974543	0.862170	0.906703	1705.00000

- Logistic Regression with Cosine PCA(n=20), Lasso:

	precision	recall	f1-score	support
0	0.995804	0.857315	0.921385	1661.000000
1	0.138182	0.863636	0.238245	44.000000
accuracy	0.857478	0.857478	0.857478	0.857478
macro avg	0.566993	0.860476	0.579815	1705.000000
weighted avg	0.973672	0.857478	0.903755	1705.000000

- Logistic Regression with Poly PCA(n=20), Lasso:

	precision	recall	f1-score	support
0	0.975323	0.999398	0.987214	1661.00000
1	0.666667	0.045455	0.085106	44.00000
accuracy	0.974780	0.974780	0.974780	0.97478
macro avg	0.820995	0.522426	0.536160	1705.00000
weighted avg	0.967358	0.974780	0.963934	1705.00000

- Logistic Regression with RBF PCA(n=20), Lasso:

	precision	recall	f1-score	support
0	0.997126	0.835641	0.909270	1661.000000
1	0.127796	0.909091	0.224090	44.000000
accuracy	0.837537	0.837537	0.837537	0.837537
macro avg	0.562461	0.872366	0.566680	1705.000000
weighted avg	0.974692	0.837537	0.891588	1705.000000

- Logistic Regression with SMOTE, Elastic Net:

	precision	recall	f1-score	support
0	0.993956	0.891030	0.939683	1661.000000
1	0.162037	0.795455	0.269231	44.000000
accuracy	0.888563	0.888563	0.888563	0.888563
macro avg	0.577996	0.843242	0.604457	1705.000000
weighted avg	0.972487	0.888563	0.922381	1705.000000

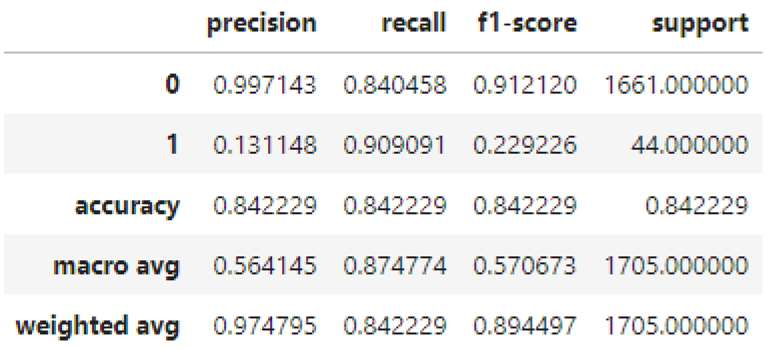
- Logistic Regression with Linear PCA(n=20), Elastic Net:

	precision	recall	f1-score	support
0	0.995149	0.864539	0.925258	1661.00000
1	0.141221	0.840909	0.241830	44.00000
accuracy	0.863930	0.863930	0.863930	0.86393
macro avg	0.568185	0.852724	0.583544	1705.00000
weighted avg	0.973112	0.863930	0.907621	1705.00000

- Logistic Regression with Sigmoid PCA(n=20), Elastic Net:

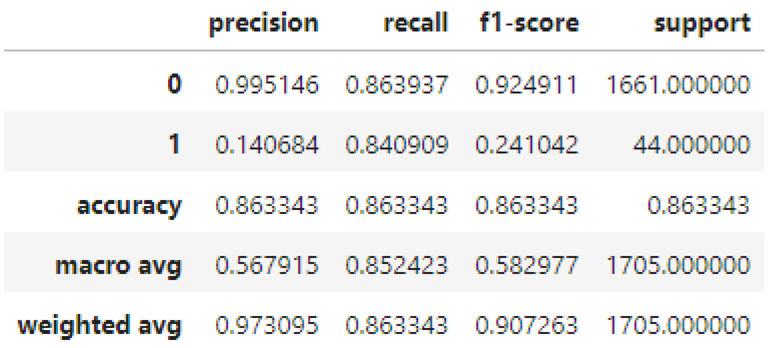
	precision	recall	f1-score	support
0	0.996518	0.861529	0.924120	1661.00000
1	0.144981	0.886364	0.249201	44.00000
accuracy	0.862170	0.862170	0.862170	0.86217
macro avg	0.570750	0.873946	0.586661	1705.00000
weighted avg	0.974543	0.862170	0.906703	1705.00000

- Logistic Regression with RBF PCA(n=20), Elastic Net:



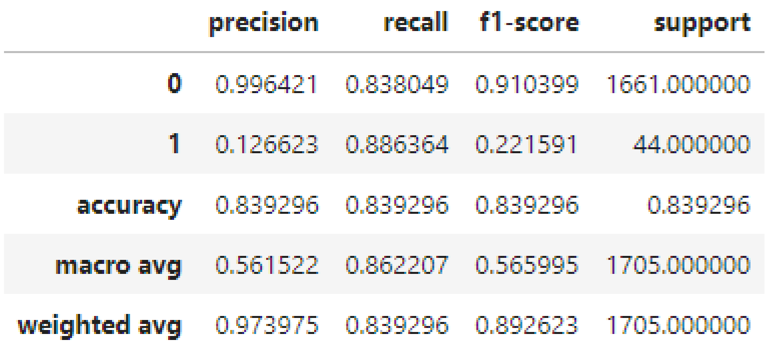
	precision	recall	f1-score	support
0	0.997143	0.840458	0.912120	1661.000000
1	0.131148	0.909091	0.229226	44.000000
accuracy	0.842229	0.842229	0.842229	0.842229
macro avg	0.564145	0.874774	0.570673	1705.000000
weighted avg	0.974795	0.842229	0.894497	1705.000000

- Logistic Regression with Linear PCA(n=20), Ridge:



	precision	recall	f1-score	support
0	0.995146	0.863937	0.924911	1661.000000
1	0.140684	0.840909	0.241042	44.000000
accuracy	0.863343	0.863343	0.863343	0.863343
macro avg	0.567915	0.852423	0.582977	1705.000000
weighted avg	0.973095	0.863343	0.907263	1705.000000

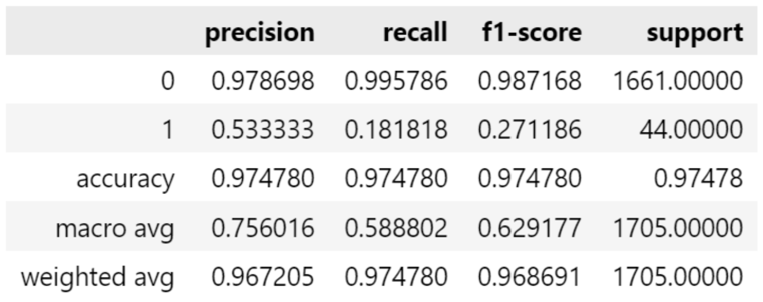
- Logistic Regression with RBF PCA(n=20), Ridge:



	precision	recall	f1-score	support
0	0.996421	0.838049	0.910399	1661.000000
1	0.126623	0.886364	0.221591	44.000000
accuracy	0.839296	0.839296	0.839296	0.839296
macro avg	0.561522	0.862207	0.565995	1705.000000
weighted avg	0.973975	0.839296	0.892623	1705.000000

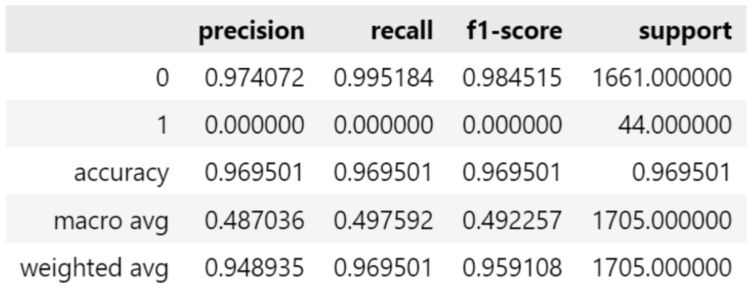
Random Forest Models:

- Random Forest



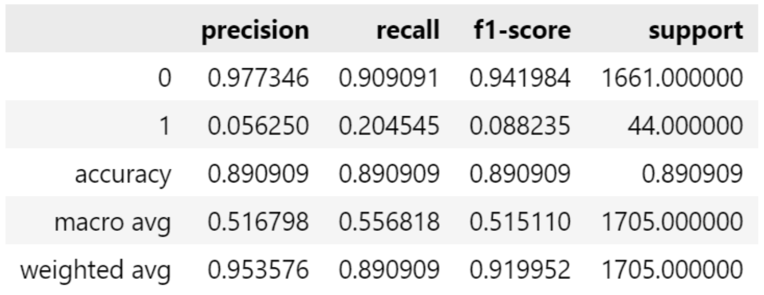
	precision	recall	f1-score	support
0	0.978698	0.995786	0.987168	1661.00000
1	0.533333	0.181818	0.271186	44.00000
accuracy	0.974780	0.974780	0.974780	0.97478
macro avg	0.756016	0.588802	0.629177	1705.00000
weighted avg	0.967205	0.974780	0.968691	1705.00000

- Random Forest with linear PCA and SMOTE



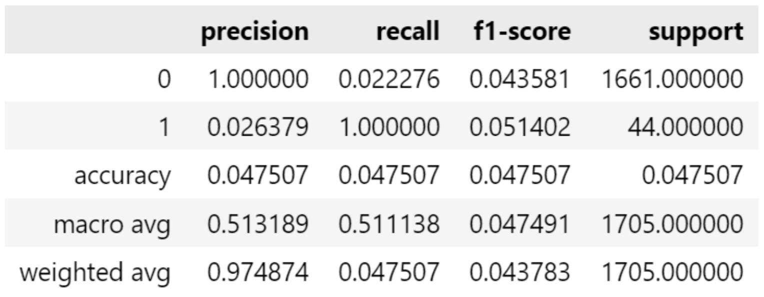
	precision	recall	f1-score	support
0	0.974072	0.995184	0.984515	1661.000000
1	0.000000	0.000000	0.000000	44.000000
accuracy	0.969501	0.969501	0.969501	0.969501
macro avg	0.487036	0.497592	0.492257	1705.000000
weighted avg	0.948935	0.969501	0.959108	1705.000000

- Random Forest with cosine PCA and SMOTE



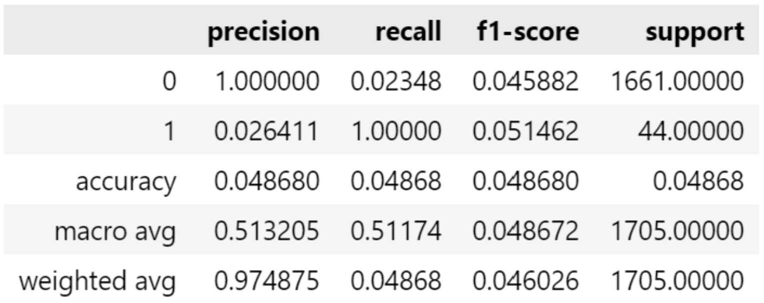
	precision	recall	f1-score	support
0	0.977346	0.909091	0.941984	1661.000000
1	0.056250	0.204545	0.088235	44.000000
accuracy	0.890909	0.890909	0.890909	0.890909
macro avg	0.516798	0.556818	0.515110	1705.000000
weighted avg	0.953576	0.890909	0.919952	1705.000000

- Random Forest with sigmoid PCA and SMOTE



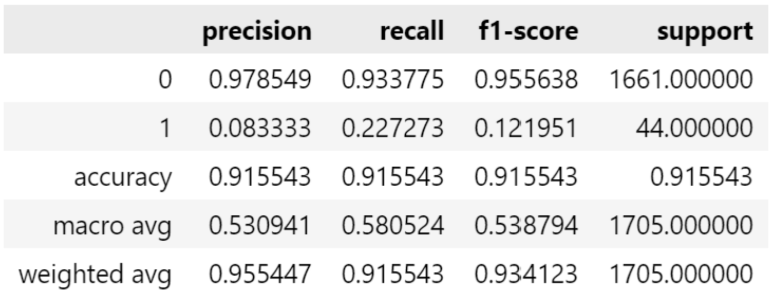
	precision	recall	f1-score	support
0	1.000000	0.022276	0.043581	1661.000000
1	0.026379	1.000000	0.051402	44.000000
accuracy	0.047507	0.047507	0.047507	0.047507
macro avg	0.513189	0.511138	0.047491	1705.000000
weighted avg	0.974874	0.047507	0.043783	1705.000000

- Random Forest with rbf PCA and SMOTE



	precision	recall	f1-score	support
0	1.000000	0.02348	0.045882	1661.00000
1	0.026411	1.00000	0.051462	44.00000
accuracy	0.048680	0.04868	0.048680	0.04868
macro avg	0.513205	0.51174	0.048672	1705.00000
weighted avg	0.974875	0.04868	0.046026	1705.00000

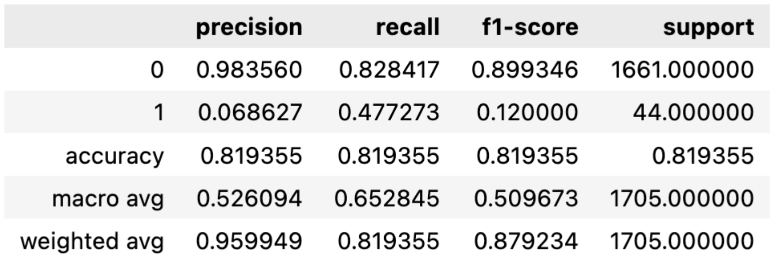
- Random Forest with poly PCA and SMOTE



	precision	recall	f1-score	support
0	0.978549	0.933775	0.955638	1661.000000
1	0.083333	0.227273	0.121951	44.000000
accuracy	0.915543	0.915543	0.915543	0.915543
macro avg	0.530941	0.580524	0.538794	1705.000000
weighted avg	0.955447	0.915543	0.934123	1705.000000

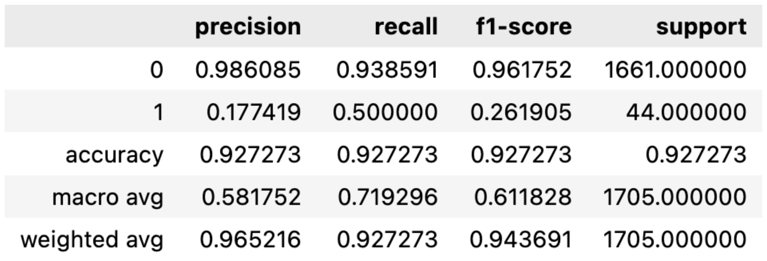
KNN models:

- KNN with all features and SMOTE



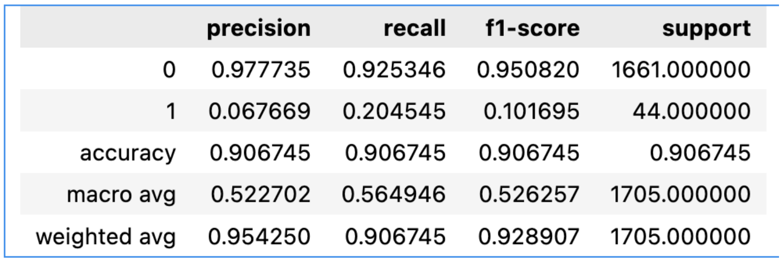
	precision	recall	f1-score	support
0	0.983560	0.828417	0.899346	1661.000000
1	0.068627	0.477273	0.120000	44.000000
accuracy	0.819355	0.819355	0.819355	0.819355
macro avg	0.526094	0.652845	0.509673	1705.000000
weighted avg	0.959949	0.819355	0.879234	1705.000000

- KNN with feature selection (top 20 features) and SMOTE



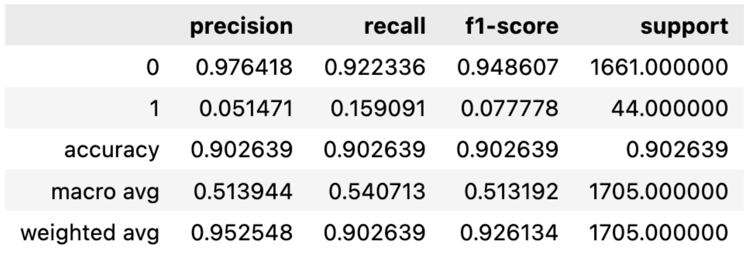
	precision	recall	f1-score	support
0	0.986085	0.938591	0.961752	1661.000000
1	0.177419	0.500000	0.261905	44.000000
accuracy	0.927273	0.927273	0.927273	0.927273
macro avg	0.581752	0.719296	0.611828	1705.000000
weighted avg	0.965216	0.927273	0.943691	1705.000000

- KNN with linear PCA and SMOTE



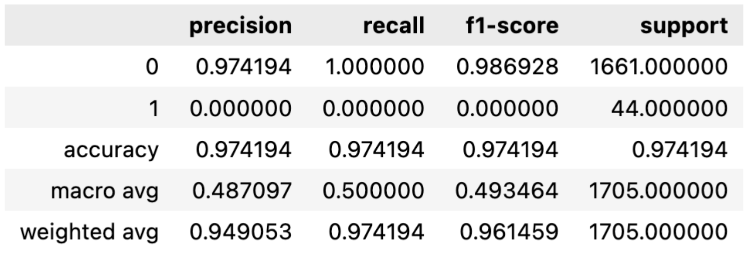
	precision	recall	f1-score	support
0	0.977735	0.925346	0.950820	1661.000000
1	0.067669	0.204545	0.101695	44.000000
accuracy	0.906745	0.906745	0.906745	0.906745
macro avg	0.522702	0.564946	0.526257	1705.000000
weighted avg	0.954250	0.906745	0.928907	1705.000000

- KNN with cosine PCA and SMOTE



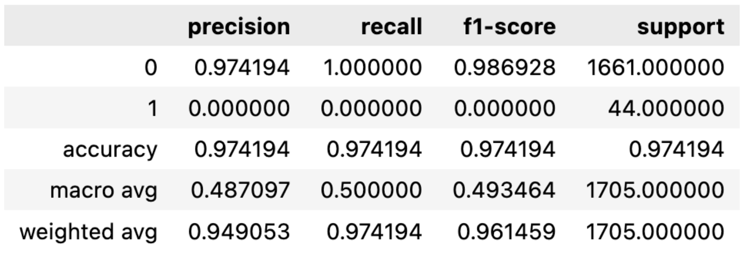
	precision	recall	f1-score	support
0	0.976418	0.922336	0.948607	1661.000000
1	0.051471	0.159091	0.077778	44.000000
accuracy	0.902639	0.902639	0.902639	0.902639
macro avg	0.513944	0.540713	0.513192	1705.000000
weighted avg	0.952548	0.902639	0.926134	1705.000000

- KNN with sigmoid PCA and SMOTE



	precision	recall	f1-score	support
0	0.974194	1.000000	0.986928	1661.000000
1	0.000000	0.000000	0.000000	44.000000
accuracy	0.974194	0.974194	0.974194	0.974194
macro avg	0.487097	0.500000	0.493464	1705.000000
weighted avg	0.949053	0.974194	0.961459	1705.000000

- KNN with rbf PCA and SMOTE



	precision	recall	f1-score	support
0	0.974194	1.000000	0.986928	1661.000000
1	0.000000	0.000000	0.000000	44.000000
accuracy	0.974194	0.974194	0.974194	0.974194
macro avg	0.487097	0.500000	0.493464	1705.000000
weighted avg	0.949053	0.974194	0.961459	1705.000000