Company Bankruptcy Predictor

A machine learning model for predicting company bankruptcies based on financial data and risk factors

Teammates: Max Boggus, Danny Long, Blake Rodgers, John Wright Stanly, Drew Verzino

Final Report

Introduction / Background

Assessing the risk of bankruptcy is crucial for companies, impacting not only their financial stability but also their employees, stakeholders, and investors. Research has shown that ML can be a helpful tool in accurately determining risk, performing up to 35% better than traditional models in one study on North American firms from 1985 to 2013 [1]. Still, there are limiting factors with ML based predictions, like an overlapping feature space between financial variables of bankrupt companies [2]. In our project, we will use accounting data from American public companies listed on the NYSE and NASDAQ from 1999 to 2018. Some features, like EBITDA, Market Value, Total Assets and others will be used to make predictions of which companies went bankrupt during this period. DATASET LINK

Problem Definition

Unexpected corporate bankruptcies can be problematic for stakeholders, as they can lead to substantial financial losses and job insecurity. Despite advancements in predictive modeling, many existing methods lack accuracy and may fail to capture the complexities of what actually causes or indicates risk of bankruptcy. The inability to accurately predict bankruptcy leaves investors vulnerable to unforeseen market volatility and instability. Therefore, there is a pressing need for more effective solutions that leverage traditional and more cutting-edge machine learning techniques to enhance prediction accuracy. By improving bankruptcy prediction models, we can better inform stakeholders and at-risk companies, which contributes to a more resilient financial ecosystem.

Methods + Results and Discussion

Exploring the Dataset

We begin by loading the dataset into the environment and exploring its structure. A glimpse of the dataset is shown below:

	company_name	status_label	year	X1	X2	X3	X4	X5	X6	X7	...	X9	X10	X11	X12	X13	X14	X15
0	C_1	alive	1999	511.267	833.107	18.373	89.031	336.018	35.163	128.348	...	1024.333	740.998	180.447	70.658	191.226	163.816	201.026
1	C_1	alive	2000	485.856	713.811	18.577	64.367	320.590	18.531	115.187	...	874.255	701.854	179.987	45.790	160.444	125.392	204.065
2	C_1	alive	2001	436.656	526.477	22.496	27.207	286.588	-58.939	77.528	...	638.721	710.199	217.699	4.711	112.244	150.464	139.603
3	C_1	alive	2002	396.412	496.747	27.172	30.745	259.954	-12.410	66.322	...	606.337	686.621	164.658	3.573	109.590	203.575	124.106
4	C_1	alive	2003	432.204	523.302	26.680	47.491	247.245	3.504	104.661	...	651.958	709.292	248.666	20.811	128.656	131.261	131.884

We then explore the dataset structure, figuring out the size and composition of the dataset, helping to identify any issues and plan subsequent data pre-processing steps. We find that the dataset contains 78682 rows and 21 columns and is

about 12.6 MB in size. The columns consist of the company name (object), the status label (object), the year (int64) and 18 financial company features (float64). The features are described as:

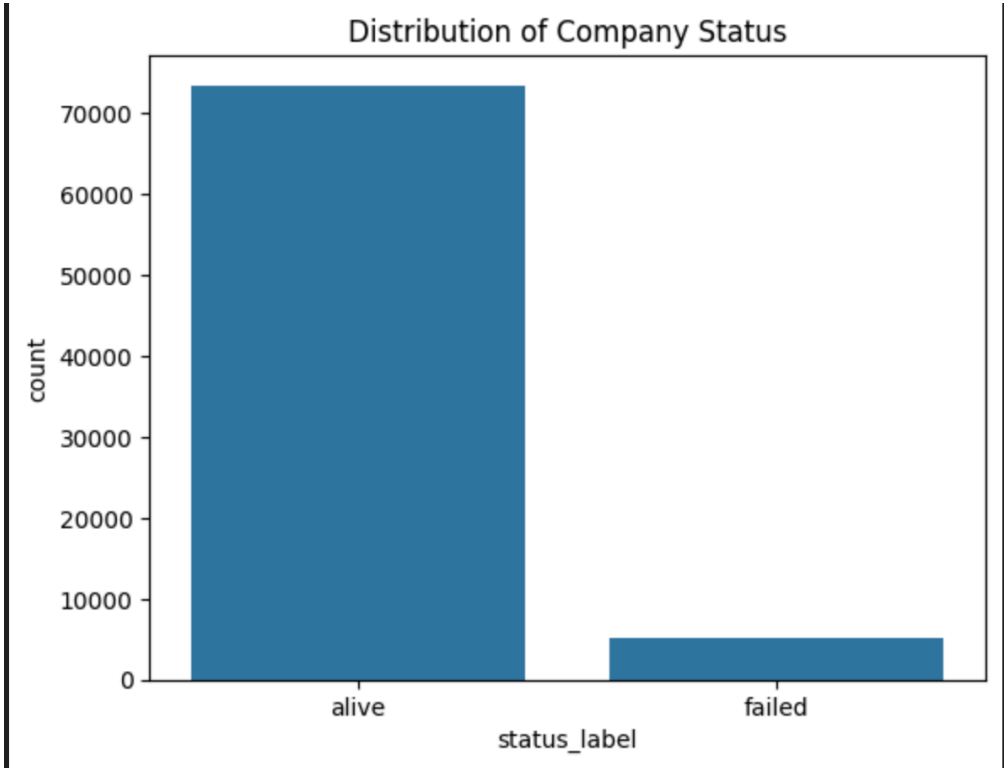
Variable	Description
X1	Current assets - All the assets of a company that are expected to be sold or used as a result of standard business operations over the next year.
X2	Cost of goods sold - The total amount a company paid as a cost directly related to the sale of products.
X3	Depreciation and amortization - Depreciation refers to the loss of value of a tangible fixed asset over time (such as property, machinery, buildings, and plant). Amortization refers to the loss of value of intangible assets over time.
X4	EBITDA - Earnings before interest, taxes, depreciation, and amortization. It is a measure of a company's overall financial performance, serving as an alternative to net income.
X5	Inventory - The accounting of items and raw materials that a company either uses in production or sells.
X6	Net Income - The overall profitability of a company after all expenses and costs have been deducted from total revenue.
X7	Total Receivables - The balance of money due to a firm for goods or services delivered or used but not yet paid for by customers.
X8	Market value - The price of an asset in a marketplace. In this dataset, it refers to the market capitalization since companies are publicly traded in the stock market.
X9	Net sales - The sum of a company's gross sales minus its returns, allowances, and discounts.
X10	Total assets - All the assets, or items of value, a business owns.

Variable	Description
X11	Total Long-term debt - A company's loans and other liabilities that will not become due within one year of the balance sheet date.
X12	EBIT - Earnings before interest and taxes.
X13	Gross Profit - The profit a business makes after subtracting all the costs that are related to manufacturing and selling its products or services.
X14	Total Current Liabilities - The sum of accounts payable, accrued liabilities, and taxes such as Bonds payable at the end of the year, salaries, and commissions remaining.
X15	Retained Earnings - The amount of profit a company has left over after paying all its direct costs, indirect costs, income taxes, and its dividends to shareholders.
X16	Total Revenue - The amount of income that a business has made from all sales before subtracting expenses. It may include interest and dividends from investments.
X17	Total Liabilities - The combined debts and obligations that the company owes to outside parties.
X18	Total Operating Expenses - The expenses a business incurs through its normal business operations.

We then generate descriptive statistics for the dataset, including count, mean, standard deviation, minimum, maximum, and quartile values for numerical columns, as well as counts and unique values for categorical columns. We also check for missing values in each column and duplicate rows and find that no columns contain missing values, and there are no duplicate rows.

Understanding the distribution of the target variable is essential in any machine learning project. This graphic visualizes the distribution of the `status_label` to assess class balance and gain insights into the dataset's composition. Visualizing

the target variable aids in identifying potential issues such as class imbalance, which can significantly impact model performance and evaluation.



Data Preprocessing

Preparing the dataset for machine learning involves several crucial preprocessing steps to ensure that the data is clean, consistent, and in a suitable format for modeling. This section outlines the steps taken to preprocess the dataset, including dropping unnecessary columns, handling data types, and encoding categorical variables.

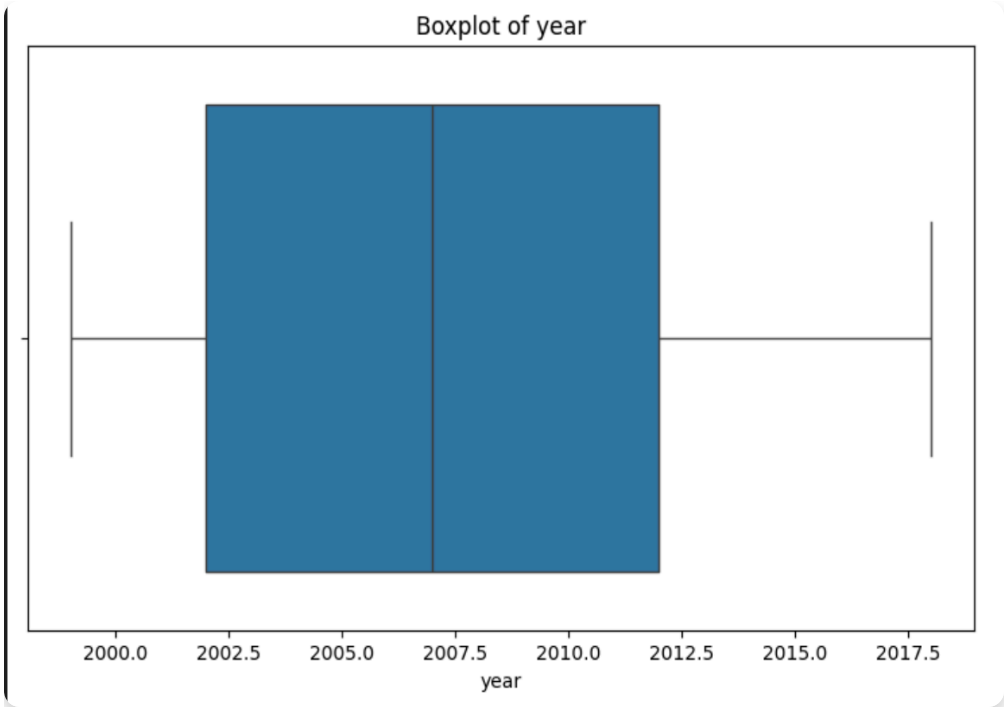
We first drop any unnecessary columns, ensure the 'year' is numeric and encode the 'status_label' if needed.

Ensuring data quality is a fundamental step in the data preprocessing pipeline. Negative values in certain columns may indicate data entry errors, invalid measurements, or specific characteristics of the data that need to be addressed. We

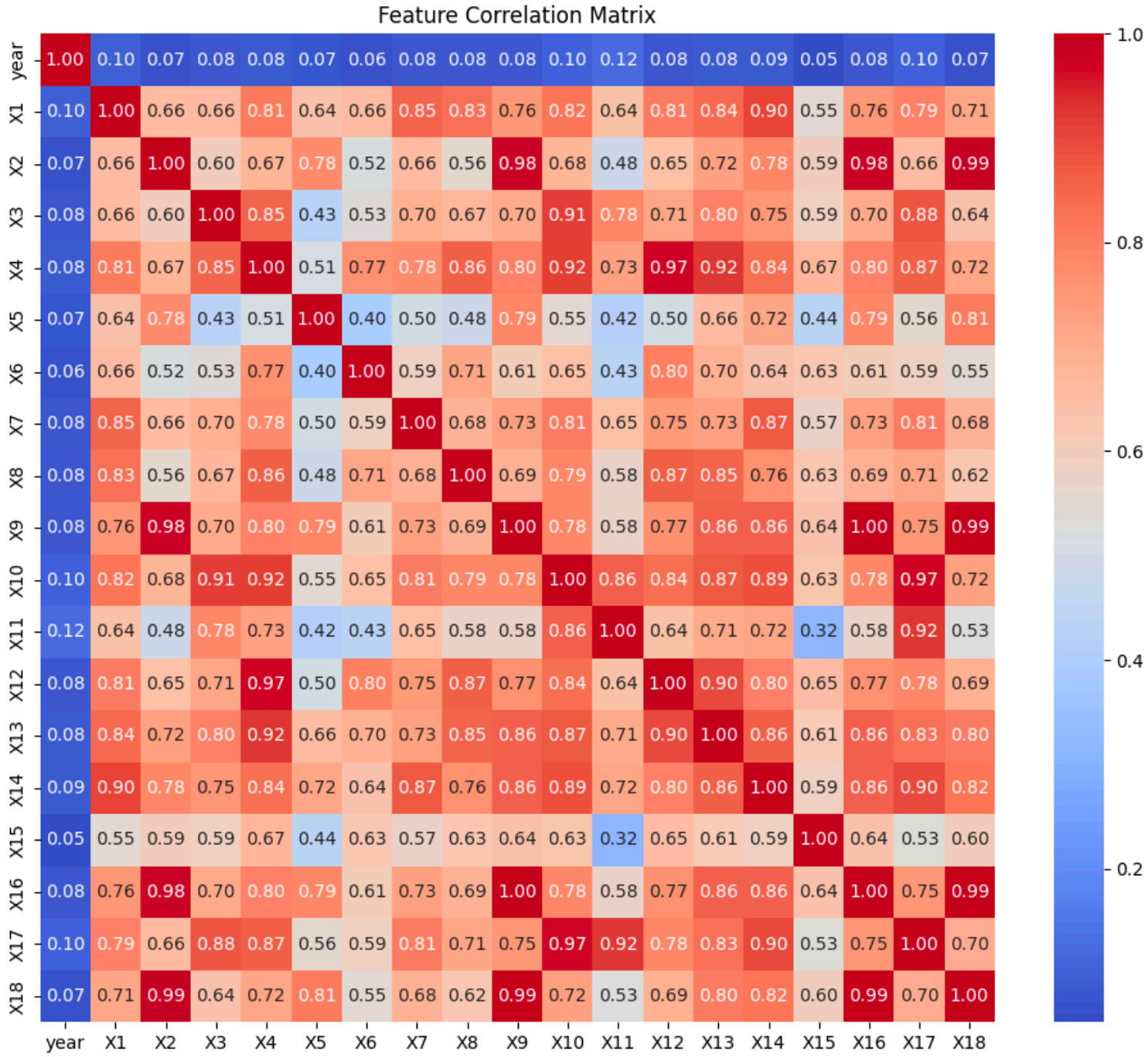
identify that columns X1, X2, X4, X6, X7 , X9, X11, X12, X13, X15, X16 and X18 contain negative values, allowing for informed decisions on how to handle them in subsequent analysis and modeling steps.

We then visualize the numerical features with boxplots. Boxplots are an effective way to visualize the distribution of numerical features in a dataset. They help in identifying the presence of outliers, understanding the spread of the data, and comparing distributions across different variables. We generated boxplots for all numerical features (excluding the target variable) in the dataset, providing a comprehensive overview of their distributions.

Feature Boxplots



We then generated a correlation matrix for all the features in order to understand which features were minimally/maximally correlated.


Feature Correlation Matrix
year 1.00 0.10 0.07 0.08 0.08 0.07 0.06 0.08 0.08 0.08 0.10 0.12 0.08 0.08 0.09 0.05 0.08 0.10 0.07
X1 0.10 1.00 0.66 0.66 0.81 0.64 0.66 0.85 0.83 0.76 0.82 0.64 0.81 0.84 0.90 0.55 0.76 0.79 0.71
X2 0.07 0.66 1.00 0.60 0.67 0.78 0.52 0.66 0.56 0.98 0.68 0.48 0.65 0.72 0.78 0.59 0.98 0.66 0.99
X3 0.08 0.66 0.60 1.00 0.85 0.43 0.53 0.70 0.67 0.70 0.91 0.78 0.71 0.80 0.75 0.59 0.70 0.88 0.64
X4 0.08 0.81 0.67 0.85 1.00 0.51 0.77 0.78 0.86 0.80 0.92 0.73 0.97 0.92 0.84 0.67 0.80 0.87 0.72
X5 0.07 0.64 0.78 0.43 0.51 1.00 0.40 0.50 0.48 0.79 0.55 0.42 0.50 0.66 0.72 0.44 0.79 0.56 0.81
X6 0.06 0.66 0.52 0.53 0.77 0.40 1.00 0.59 0.71 0.61 0.65 0.43 0.80 0.70 0.64 0.63 0.61 0.59 0.55
X7 0.08 0.85 0.66 0.70 0.78 0.50 0.59 1.00 0.68 0.73 0.81 0.65 0.75 0.73 0.87 0.57 0.73 0.81 0.68
X8 0.08 0.83 0.56 0.67 0.86 0.48 0.71 0.68 1.00 0.69 0.79 0.58 0.87 0.85 0.76 0.63 0.69 0.71 0.62
X9 0.08 0.76 0.98 0.70 0.80 0.79 0.61 0.73 0.69 1.00 0.78 0.58 0.77 0.86 0.86 0.64 1.00 0.75 0.99
X10 0.10 0.82 0.68 0.91 0.92 0.55 0.65 0.81 0.79 0.78 1.00 0.86 0.84 0.87 0.89 0.63 0.78 0.97 0.72
X11 0.12 0.64 0.48 0.78 0.73 0.42 0.43 0.65 0.58 0.58 0.86 1.00 0.64 0.71 0.72 0.32 0.58 0.92 0.53
X12 0.08 0.81 0.65 0.71 0.97 0.50 0.80 0.75 0.87 0.77 0.84 0.64 1.00 0.90 0.80 0.65 0.77 0.78 0.69
X13 0.08 0.84 0.72 0.80 0.92 0.66 0.70 0.73 0.85 0.86 0.87 0.71 0.90 1.00 0.86 0.61 0.86 0.83 0.80
X14 0.09 0.90 0.78 0.75 0.84 0.72 0.64 0.87 0.76 0.86 0.89 0.72 0.80 0.86 1.00 0.59 0.86 0.90 0.82
X15 0.05 0.55 0.59 0.59 0.67 0.44 0.63 0.57 0.63 0.64 0.63 0.32 0.65 0.61 0.59 1.00 0.64 0.53 0.60
X16 0.08 0.76 0.98 0.70 0.80 0.79 0.61 0.73 0.69 1.00 0.78 0.58 0.77 0.86 0.86 0.64 1.00 0.75 0.99
X17 0.10 0.79 0.66 0.88 0.87 0.56 0.59 0.81 0.71 0.75 0.97 0.92 0.78 0.83 0.90 0.53 0.75 1.00 0.70
X18 0.07 0.71 0.99 0.64 0.72 0.81 0.55 0.68 0.62 0.99 0.72 0.53 0.69 0.80 0.82 0.60 0.99 0.70 1.00
year X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X16 X17 X18
1.0
0.8
0.6
0.4
0.2

We continue to detecting outliers and removing them in the majority class. Outliers can significantly skew the results of data analysis and machine learning models. This section focuses on identifying and removing outliers from the majority class in the dataset using the Interquartile Range (IQR) method. By targeting the majority class, we aim to refine the dataset without compromising the integrity of the minority class, which is crucial for maintaining class balance.

```
Original majority class shape: (73462, 20)
Original minority class shape: (5220, 20)
Majority class shape after outlier removal: (27785, 20)
Class distribution after outlier removal:
status_label
0    27785
1     5220
Name: count, dtype: int64
```

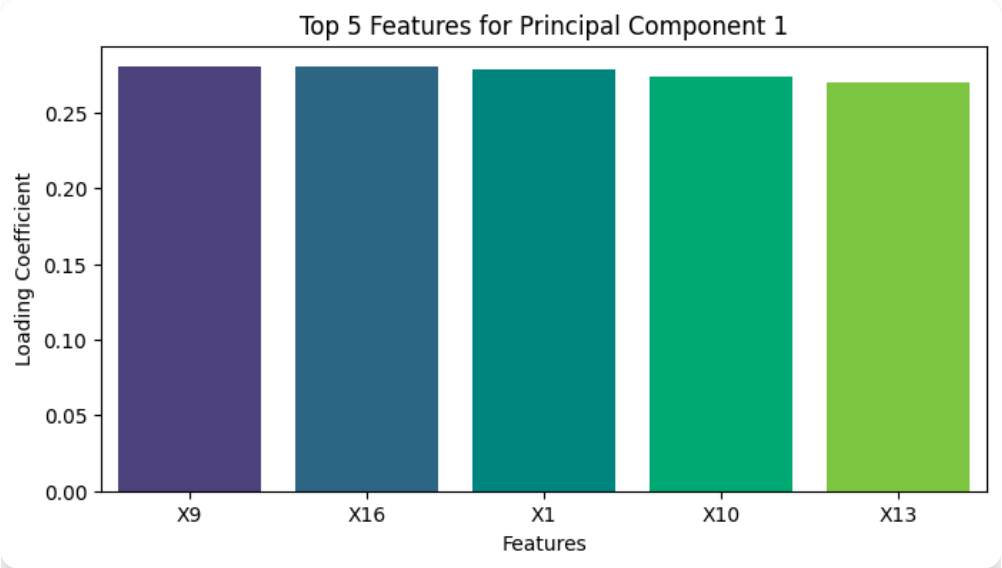
We then split the dataset into training and testing sets. After preprocessing the data and ensuring its quality, the next crucial step is to prepare it for machine learning model training and evaluation. This involves separating the dataset into features and the target variable, followed by splitting the data into training and testing subsets. Proper splitting ensures that the model can generalize well to unseen data and provides an unbiased evaluation of its performance. We used a 'test_size' of 0.2 and calculated that the 'status_label' in the training set had 22228 instances of 0 and 4176 instances of 1 - keeping the distribution balanced among training and testing sets.

We then applied Feature Scaling with StandardScalar. Scaling features is a critical preprocessing step in machine learning, especially for algorithms that are sensitive to the scale of input data (e.g., Support Vector Machines, Logistic Regression, and Neural Networks). We applied feature scaling to the dataset using `StandardScaler` from Scikit-learn. The scaler is fitted on the training data to prevent data leakage and then applied to both the training and testing sets to ensure consistency.

Next, we perform Dimensionality Reduction with Principal Component Analysis. Dimensionality reduction is a crucial step in the machine learning pipeline, especially when dealing with high-dimensional data. It helps in simplifying models, reducing computational costs, and mitigating the risk of overfitting. We applied Principal Component Analysis (PCA) to the

scaled features to reduce the number of dimensions while retaining 90% of the variance in the data. We reduced the number of features in the original dataset from 19 to 6 after PCA.

Principal Component Analysis



We then hnadled class imbalance with SMOTE (Synthetic Minority Over-sampling Technique). Class imbalance is a common issue in machine learning, particularly in classification tasks where one class significantly outnumbers another. Imbalanced datasets can lead to biased models that favor the majority class, resulting in poor predictive performance for the minority class. To address this, SMOTE (Synthetic Minority Over-sampling Technique) is employed to create synthetic examples of the minority class, thereby balancing the class distribution and enhancing the model's ability to generalize effectively. By performing SMOTE, we produced the following results.

```
Class distribution in y_train before SMOTE:
status_label
0    22228
1     4176
Name: count, dtype: int64
Class distribution in y_train after SMOTE:
status_label
0    22228
1    22228
Name: count, dtype: int64
```

With the dataset now preprocessed, scaled, and balanced, the next step is to build and train models to predict the target variable. We used three different models: Logistic Regression, Support Vector Machine, and Random Forest. But before diving into each of the models, it's important we first established evaluation metrics used for all models.

Evaluation Metrics

To evaluate the performance of our models, we used several standard classification metrics:

Metric	Formula	Description
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	The proportion of correct predictions (both true positives and true negatives) among all predictions. While useful, accuracy alone can be misleading with imbalanced datasets.
Precision	$\frac{TP}{TP + FP}$	The proportion of true positive predictions compared to all positive predictions. This tells us how many of the companies our model predicted would go bankrupt actually did go bankrupt.

Metric	Formula	Description
Recall	$\frac{\text{TP}}{\text{TP} + \text{FN}}$	The proportion of actual positive cases that were correctly identified. This indicates how many of the actually bankrupt companies our model successfully identified.
F1-Score	$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$	The harmonic mean of precision and recall, providing a single score that balances both metrics. This is particularly useful when dealing with imbalanced classes.

We also utilized confusion matrices to visualize the models' performance across all prediction categories (true positives, true negatives, false positives, and false negatives). These metrics together provide a comprehensive view of each model's predictive capabilities.

Additionally, we generated Receiver Operating Characteristic (ROC) curves to help understand the models' performance across different classification thresholds. ROC curves plot the True Positive Rate (TPR) against the False Positive Rate (FPR), providing insight into the model's ability to distinguish between classes. The area under the ROC curve (ROC-AUC) serves as a single score that indicates the model's overall discriminative ability - a score of 1.0 represents perfect classification, while 0.5 suggests random guessing. This visualization is particularly valuable for comparing different models and selecting optimal probability thresholds for classification.

Logistic Regression (Method + Analysis)

Our first attempt at this binary classification problem was Logistic Regression. Logistic Regression is a statistical method that estimates the probability of an outcome from applying a logistic function to a linear combination of inputs and transforming those features into probability score ranging between 0 and 1. For our company bankruptcy situation, it can model the relationship between various financial indicators while still providing digestible results via its coefficients.

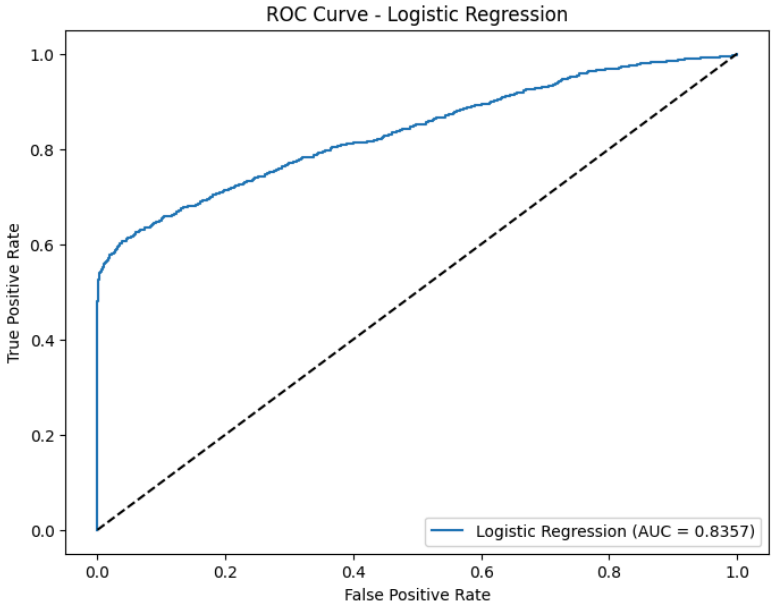
We used scikit-learn's LogisticRegression class with the following parameters:

```
log_reg = LogisticRegression(max_iter=1000, random_state=42, class_weight='balanced')
```

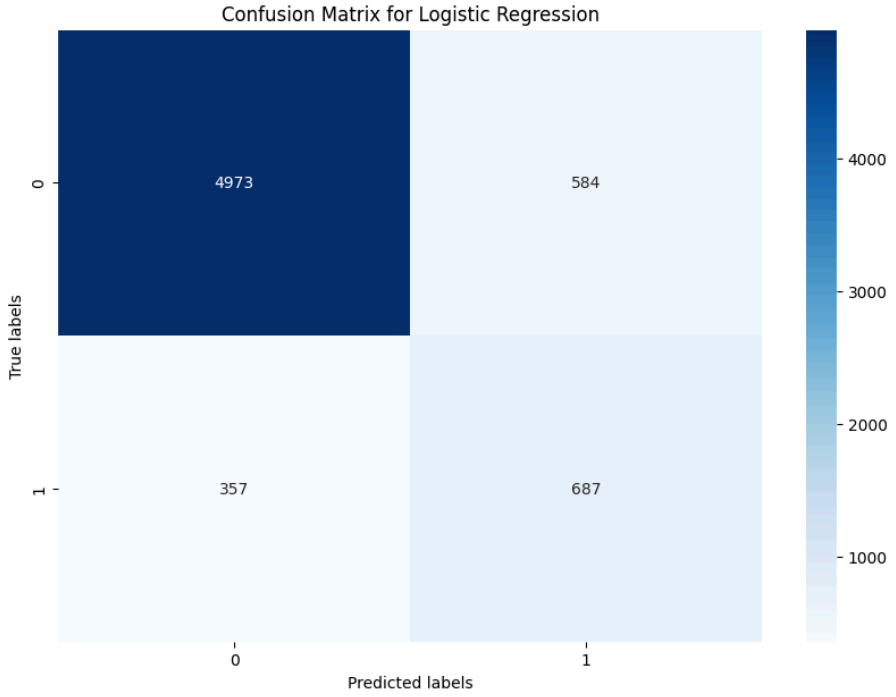
The logistic regression model achieved the following performance metrics:

Accuracy	Class	Precision	Recall	F1-Score
0.8574	Non-Bankrupt (0)	0.9330	0.8949	0.9136
	Bankrupt (1)	0.5405	0.6580	0.5935

Logistic Regression ROC Curve



Logistic Regression Confusion Matrix



Support Vector Machine (Method + Analysis)

We then tried using a Support Vector Machine (SVM), a more powerful machine learning algorithm useful in classification situations. For our case with company bankruptcies, we used SVM with a nonlinear kernel (config below) to find an

optimal hyperplane that separates bankrupt from non-bankrupt companies while still maximizing the distance between classes. Especially compared to logistic regression, SVMs are helpful with high-dimensional data where decision boundaries are potentially more nonlinear.

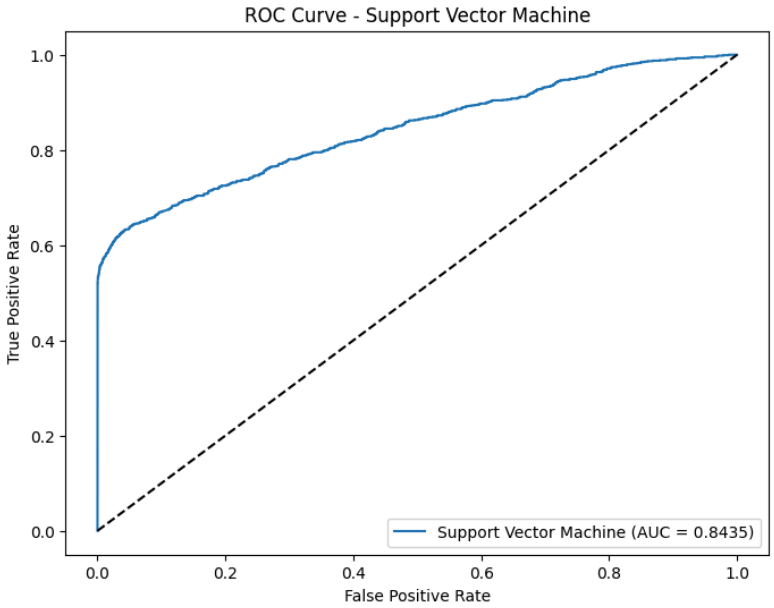
We implemented SVM using scikit-learn's SVC (Support Vector Classification) class with the following parameters:

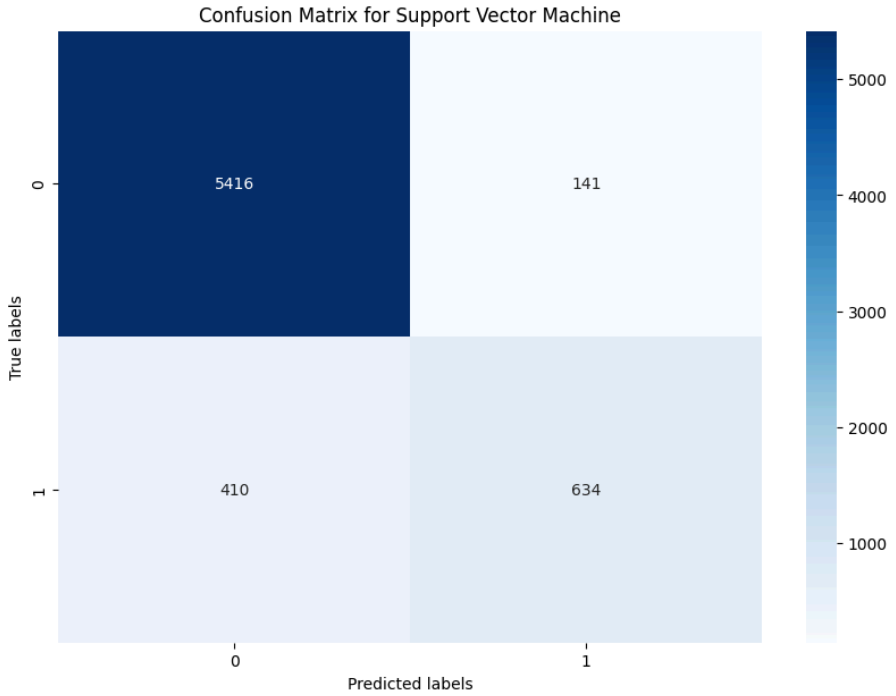
```
svc = SVC(probability=True, class_weight='balanced', random_state=42)
```

The SVM model achieved the following performance metrics:

Accuracy	Class	Precision	Recall	F1-Score
0.9165	Non-Bankrupt (0)	0.9296	0.9746	0.9516
	Bankrupt (1)	0.8181	0.6073	0.6971

SVM ROC Curve **SVM Confusion Matrix**





Random Forest (Method + Analysis)

We also implemented a Random Forest classifier, an ensemble learning method that constructs multiple decision trees and combines their predictions. Random Forests are particularly effective for complex classification tasks as they reduce overfitting through averaging multiple decision trees and handle non-linear relationships well. For our case with company bankruptcies, Random Forests can capture intricate patterns in financial data while still being robust to outliers or noise found in the data.

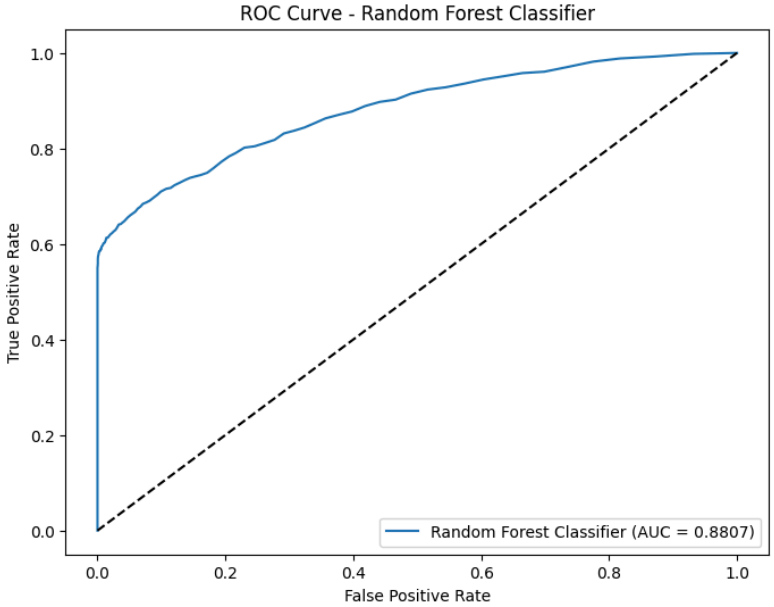
We implemented Random Forest using scikit-learn's RandomForestClassifier with the following parameters:

```
rf = RandomForestClassifier(n_estimators=100, class_weight='balanced', random_state=42)
```

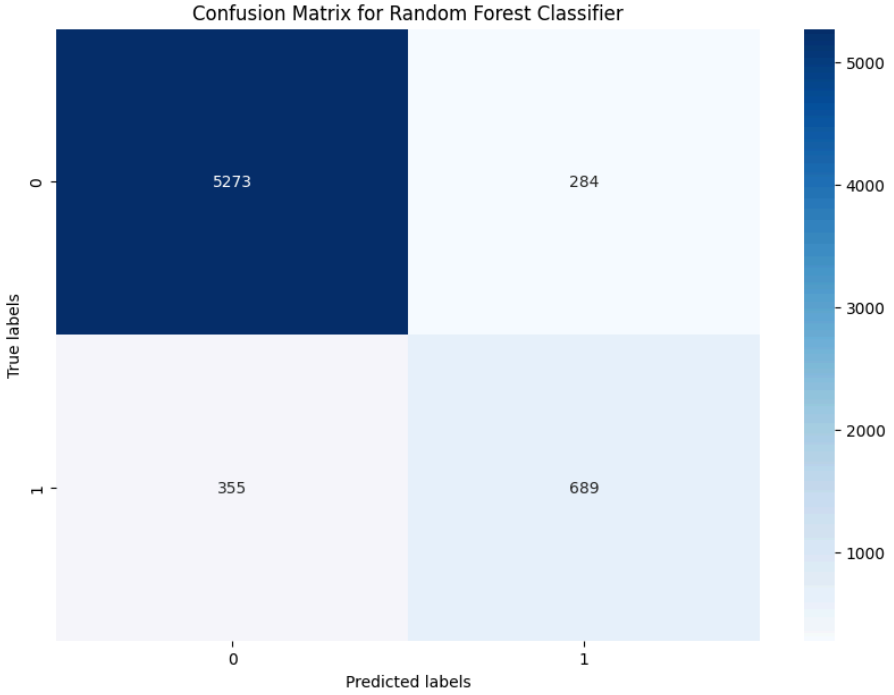
The Random Forest model achieved the following performance metrics:

Accuracy	Class	Precision	Recall	F1-Score
0.9032	Non-Bankrupt (0)	0.9369	0.9489	0.9429
	Bankrupt (1)	0.7081	0.6600	0.6832

Random Forest ROC Curve



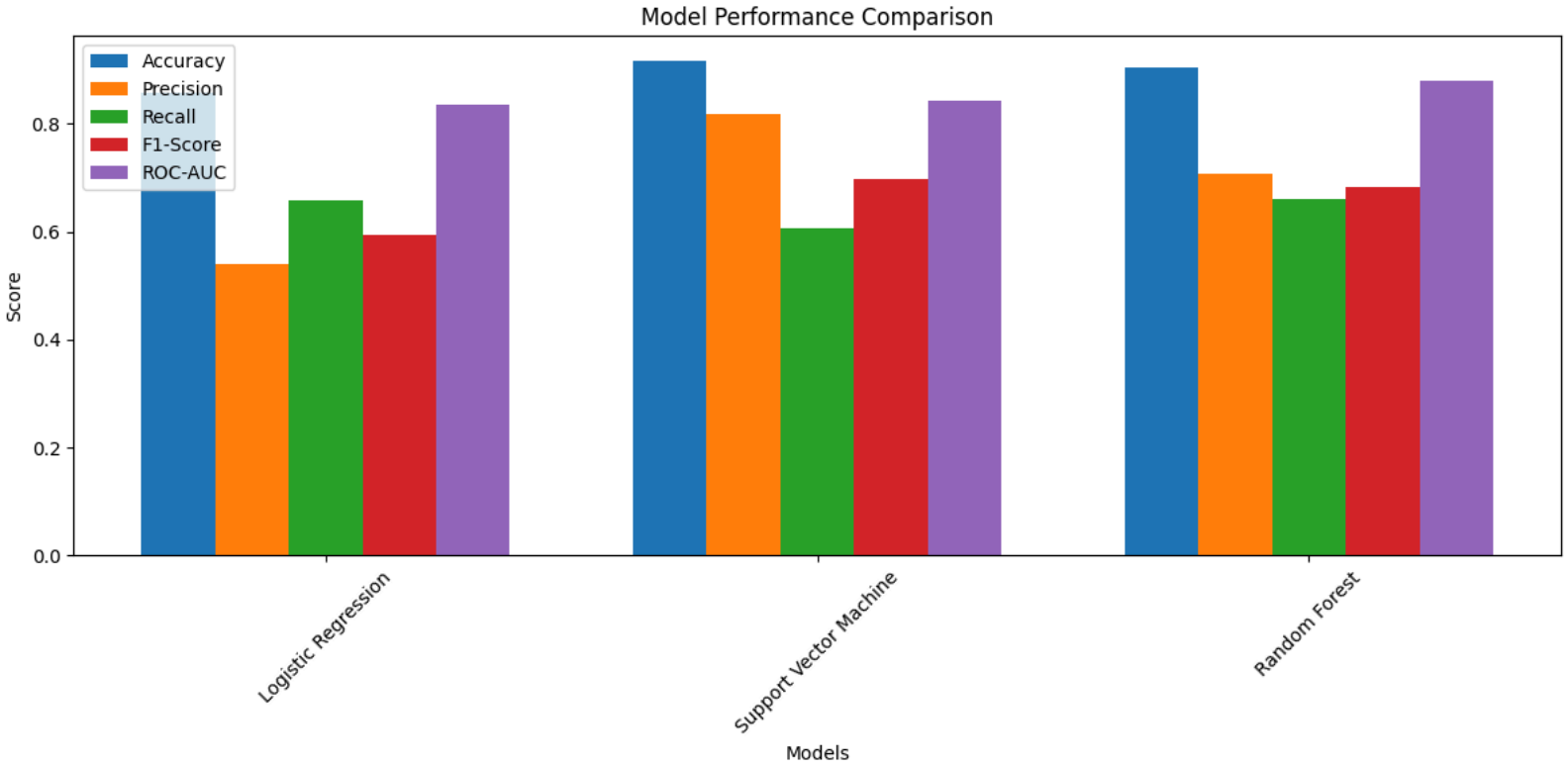
Random Forest Confusion Matrix



Comparison of Logistic Regression vs. Support Vector Machine vs. Random Forest

The three methods used result in the following performance metrics:

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Logistic Regression	0.8574	0.5405	0.6580	0.5935	0.8357
Support Vector Machine	0.9165	0.8181	0.6073	0.6971	0.8435
Random Forest	0.9032	0.7081	0.6600	0.6832	0.8807



Each model demonstrated their own strengths in identifying potential bankruptcies. The following table is an overview summary:

Logistic Regression	Support Vector Machine (SVM)	Random Forest
• Lowest overall accuracy (85.74%) • Lowest precision (54.05%) • Highest recall (65.80%)	• Highest overall accuracy (91.65%) • Best precision (81.81%) for bankruptcy predictions • Lower recall (60.73%)	• Strong overall accuracy (90.32%) • Good precision (70.81%) and recall (66.00%) • Highest ROC-AUC score (0.8807)

Also the training times for each model varied significantly (training times from a 2020 MacBook Air M1 below). Observed training times between Logistic Regression and SVM differed more than 500x:

Model	Training Time (seconds)
Logistic Regression	0.3
Support Vector Machine	157.8
Random Forest	8.1

While SVM offers high precision, its longer training time may make it less suitable for scenarios needing rapid model updates. These differences in training time could impact model selection, particularly if there were use cases demanding frequent retraining or real-time updates. However, we predict the domain of bankruptcy prediction will not require such frequent retraining, as this company dataset is unlikely to change significantly over time, nor are the models are not being used to make predictions in real-time.

If one model must be chosen, we'd recommend Random Forest as the primary model because of its well-rounded overall performance with an ROC-AUC of 0.8807 and balanced precision vs recall. However, model selection should be tailored to the specific use case. If decision making is more high stakes and false positives are costly, SVM is recommended for

its high precision. If initial screening is needed to catch potential bankruptcies, Logistic Regression may be more suitable due to its high recall rate.

The table below summarizes our recommendations for the best model to use in different scenarios when financial analyzing a company for bankruptcy risk:

Approach	Model	Key Characteristics
Conservative	SVM	• Use when false positives are costly (i.e. when missing a bankruptcy is more costly than false warnings) • Best for situations requiring high confidence in bankruptcy predictions • Suitable for high-stakes investment decisions
Balanced	Random Forest	• Best all-around performer • Good balance between identifying bankruptcies and avoiding false alarms • Recommended for general risk assessment
Cautious	Logistic Regression	• Catches more potential bankruptcies but with more false alarms • Suitable when missing a bankruptcy is more costly than false warnings • Good for initial screening

A multi-stage process could also be introduced-- we could start with Logistic Regression for broad screening, followed by SVM verification for precision or Random Forest for balanced validation. Note that no single model captures all bankruptcy cases perfectly. The final model selection should carefully consider business-specific contexts, including the relative costs of false positives versus false negatives.

Next Steps

Given the domain of bankruptcy prediction, there are some unique opportunities to improve our models' performance. Unlike many machine learning applications that require real-time predictions or frequent retraining, bankruptcy prediction would likely happen at much lower scale, typically requiring updates only when new financial statements are released (quarterly or annually). Inference time would also not likely be a concern. Additionally, there is likely a high cost with prediction errors. Therefore, all signs suggest we should prioritize model performance over computational efficiency.

For improving our models, extensive hyperparameter tuning could be conducted for each model type. For example with our training data and SMOTE, we could experiment with different sampling ratios or different types of SMOTE variants. Or for Random Forests, we could explore different numbers of trees, tree depths, or splitting criteria. For SVM, we could experiment with various kernel functions (RBF, polynomial, sigmoid) or their associated parameters, as well as different regularization strengths.

Ensemble methods present another promising avenue for improvement. We could implement a voting system that combines predictions from our best-performing models (SVM, Random Forest, and Logistic Regression), potentially weighted based on each model's historical performance.

In addition to optimizing our current models, we could also explore new model architectures entirely. We could try using gradient boosted models such as scikit-learn's GradientBoostingClassifier instead of Random Forests, assuming we'd be willing to accept the higher research costs needed for hyperparameter tuning gradient boosted models. We could also explore deep learning models like neural networks with multiple hidden layers, which could learn more complex non-linear relationships in our company data traditional models might miss.

These efforts might take days or even weeks of computation time; however, research costs are likely orders of magnitude smaller than the financial decisions downstream from these models' decisions. Therefore, high research costs are justified. Even a marginal improvement in model performance could translate to millions of dollars in prevented losses.

References

1. F. Barboza, H. Kimura, and E. Altman, "Machine learning models and bankruptcy prediction," Expert Systems with Applications, vol. 83, pp. 405-417, 2017.
2. S. Shetty, M. Musa, and X. Brédart, "Bankruptcy prediction using machine learning techniques," Journal of Risk and Financial Management, vol. 15, no. 1, p. 35, 2022. doi: 10.3390/jrfm15010035.
3. N. Wang, "Bankruptcy prediction using machine learning," Journal of Mathematical Finance, vol. 7, pp. 908-918, 2017. doi: 10.4236/jmf.2017.74049.

Gantt Chart

GanttChart Company Bankruptcy Predictor : Fall

GANTT CHART

PROJECT TITLE	Company Bankruptcy Predictor

TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PHASE ONE									
					Sep 30									
					M	T	W	R	F	S	U	M	T	W
Project Proposal														
Github Pages Setup	John Wright	9/29/2024	10/1/2024	2										
Proposal Write Up	Danny, Drew	9/29/2024	10/1/2024	2										
Gantt Chart and Contribution Table	John Wright	10/1/2024	10/4/2024	3										
Google Slides	Max, Blake	10/1/2024	10/4/2024	3										
Final Video	Max, Blake	10/1/2024	10/4/2024	3										
Model 1														
Data Sourcing and Cleaning	Blake	10/14/2024	10/18/2024	4										
Model Selection	Drew	10/21/2024	10/23/2024	2										
Data Pre-Processing	Blake	10/24/2024	10/25/2024	1										
Model Coding	Drew	10/28/2024	11/1/2024	3										
Results Evaluation and Analysis	Blake	11/4/2024	11/6/2024	2										
Midterm Report	All	11/7/2024	11/8/2024	1										
Model 2														
Data Sourcing and Cleaning	Max	10/14/2024	10/18/2024	4										
Model Selection	Danny	10/21/2024	10/23/2024	2										
Data Pre-Processing	Max	10/24/2024	10/25/2024	1										
Model Coding	Danny	10/28/2024	11/1/2024	3										

Fall

Source Google Sheets Gantt chart here.

Contribution Table

Task	Owner
Github Pages Setup	John Wright Stanly
Proposal Write Up	Drew Verzino, Danny Long
Gantt Chart and Contribution Table	John Wright Stanly
Google Slides	Blake Rodgers
Final Video	Max Boggus
Model Implementation	Drew Verzino
Model Analysis	Max Boggus
Midterm Setup	Danny Long, Blake Rodgers
Report	Max Boggus, Danny Long, Blake Rodgers, John Wright Stanly, Drew Verzino