POLISH COMPANIES BANKRUPTCY PREDICTION

BY

ARYAMAN KAKAD

FOR

IE 7300 - STATISTICAL LEARNING FOR ENGINEERS

WHAT IS BANKRUPTCY AND BANKRUPTCY PREDICTION

- Financial insolvency is a legal condition faced by individuals or entities unable to settle their debts with creditors.
- Typically, a court order, often initiated by the debtor, is how bankruptcy is declared in most jurisdictions.
- Distinguishing itself from insolvency, bankruptcy is one of the several legal states that an insolvent person or entity may find themselves in.
- Providing an opportunity for individuals or businesses to make a fresh start, bankruptcy involves forgiving debts that cannot be repaid. Creditors, in turn, have a chance to recoup some of their losses based on the assets available for liquidation.
- Upon the successful conclusion of bankruptcy proceedings, the debtor is released from the debt obligations incurred before filing for bankruptcy.
- The practice of forecasting bankruptcy and various indicators of financial distress in publicly traded companies is known as bankruptcy prediction.
- This field, which spans finance and accounting research, holds significance for creditors and investors assessing the likelihood of a company facing bankruptcy.
- The evolution of bankruptcy prediction involves the application of various statistical tools that gradually became accessible, along with a growing awareness of potential pitfalls in early analyses.
- The nature of this domain makes it well-suited for the testing of increasingly sophisticated and data-intensive forecasting approaches.

CHALLENGES IN BANKRUPTCY PREDICTION

- Collecting data on bankrupt and solvent companies proves challenging, often resulting in sparse datasets even in successful collection efforts.
- The extent of research conducted is contingent upon data availability. In the case of public firms, whether they faced bankruptcy or not, numerous accounting ratios indicating potential risk can be computed, along with several other potential explanatory variables.
- Due to varying operational timelines among companies, obtaining meaningful data on bankruptcies or other outcomes is complicated. While some companies may experience bankruptcy within the first year, others may thrive in the initial two years only to face bankruptcy in the third year.
- The factors influencing bankruptcy prediction are not as direct as the financial ratios evident on company balance sheets. The complexity lies in extracting features and generating synthetic features based on foundational attributes.

DATASET DESCRIPTION

- This dataset pertains to forecasting the likelihood of bankruptcy for Polish enterprises.
- Information for this dataset was sourced from the Emerging Markets Information Service (EMIS), a database that compiles data on emerging markets globally.
- The examination of insolvent companies covers the timeframe from 2000 to 2012, whereas the assessment of currently operational companies spans from 2007 to 2013.
- The data files encompass multiple variables and are designed for classification purposes.
- Numeric (Real) values represent all the features in the dataset, and it is worth noting that there are instances of missing values in the data.
- For our project, we will be using the 3rd year data encompassing financial rates from the 3rd year of the forecasting period and corresponding class labels indicating bankruptcy status three years later.
- There are a total of 10,503 instances, with 495 being classified as bankrupt and 10,008 as non-bankrupt. This multivariate dataset comprises 64 features.

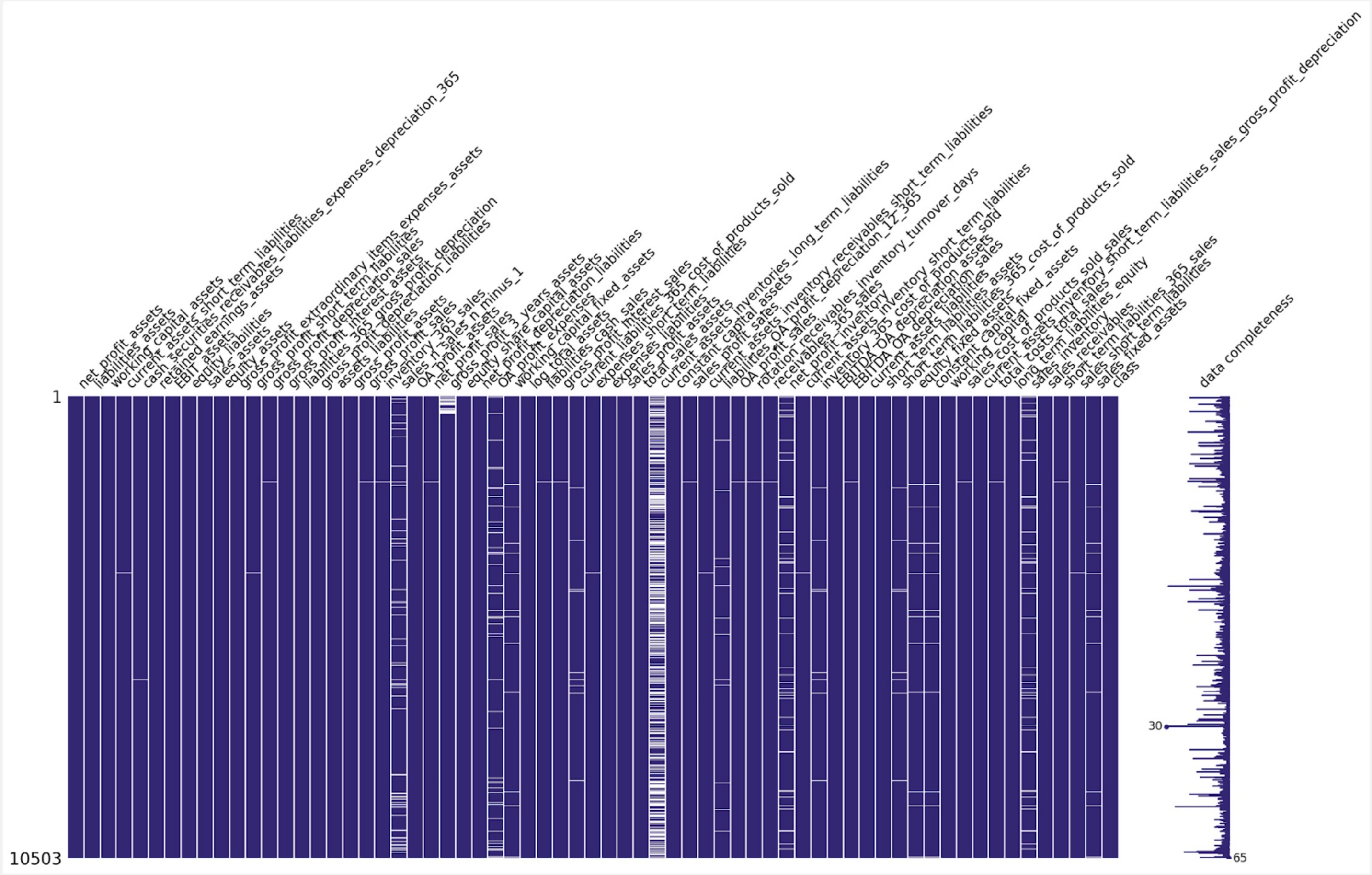
FEATURE ENGINEERING

- In our project, we investigated two imputation techniques, which will be discussed in the subsequent sections:

1. Imputation using the Mean
2. Imputation using Missing Forest

- Mean imputation involves replacing missing values in a dataset with the average of the corresponding variable. In our dataset, mean imputation was applied using the **SimpleImputer** class in scikit-learn, replacing missing values in each feature with the mean derived from available non-missing values for that feature. While mean imputation is beneficial for univariate analysis, it can introduce challenges for multivariate analysis as it diminishes correlations related to the imputed variable(s).
- Missing Forest Imputation is an advanced technique designed to address missing data in datasets. It involves creating a "forest" of decision trees, each trained on subsets with complete information to capture intricate dataset relationships. Collective Prediction: The forest collectively predicts and fills missing values, making it effective for non-linear and complex datasets, providing a robust imputation strategy.

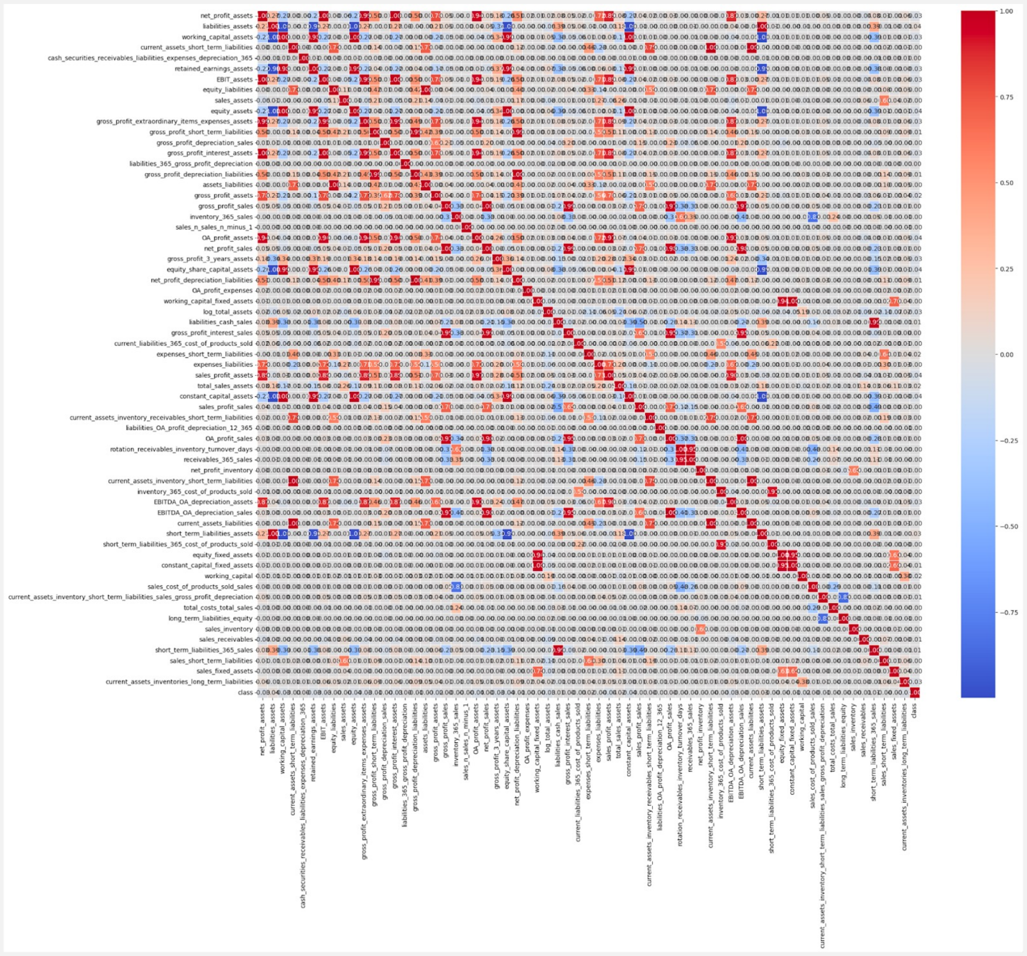
FEATURE ENGINEERING



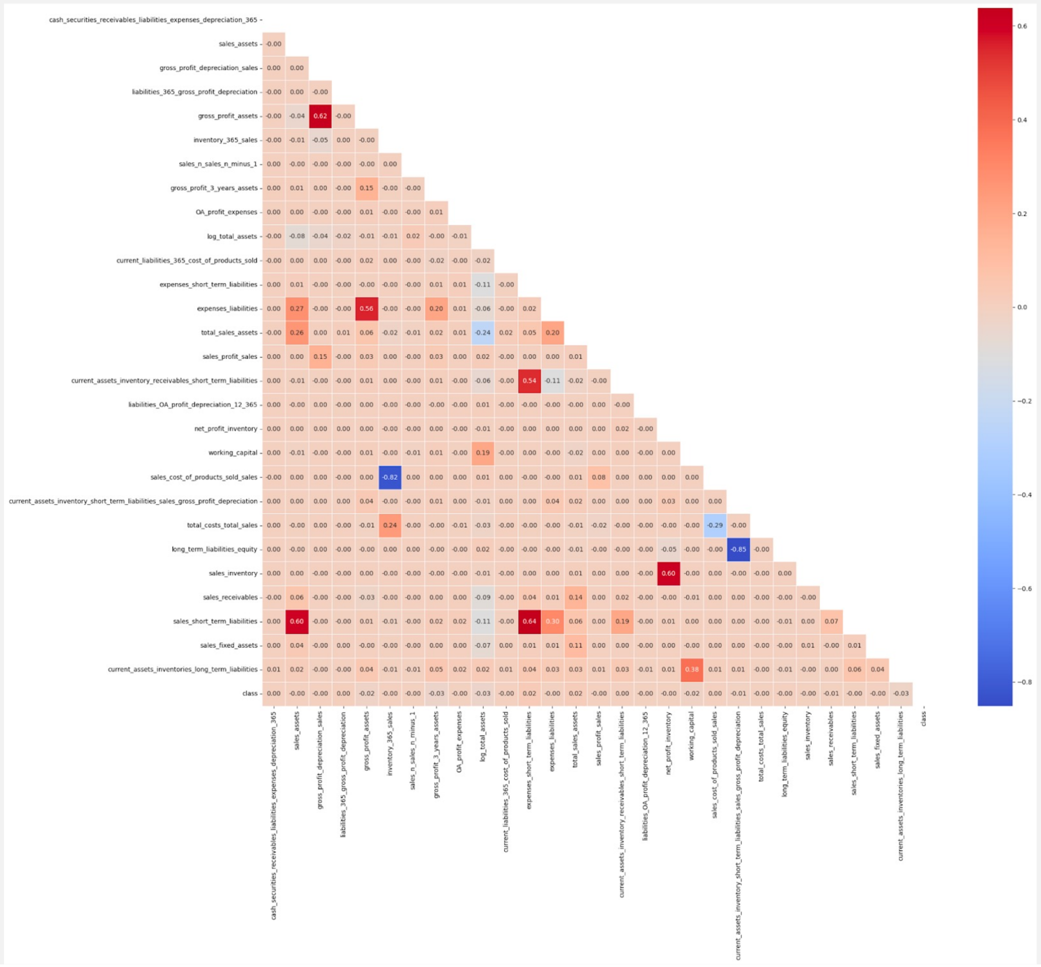
FEATURE SELECTION

- **Logistic Regression with Backward Elimination**: Systematically removes less significant variables to optimize the model's performance, reducing features from 64 to 60 in the dataset.
- **Logistic Regression with L1 Regularization**: Encourages sparsity in the feature set by promoting zero coefficients, aiding in identifying the most crucial attributes. The number of features remains at 60 after this extraction method.
- **Correlation Matrix Analysis:** Identifies associations and redundancies within features, enhancing model efficiency by eliminating highly correlated pairs. Reduces features from 60 to 29, maintaining diverse and informative attributes.
- **Distribution Analysis of Selected Features:** Examines feature distributions for bankruptcy and non-bankruptcy categories, preserving features that genuinely differentiate financial health conditions. Identifies left-skewed, right-skewed, and normal data distributions among features.
- **Outlier Detection with Box Plots:** Utilizes box plots to identify outliers impacting model performance, highlighting features with different scales and the presence of outliers. Suggests the need for data scaling and potential removal of extreme outliers to enhance model reliability.

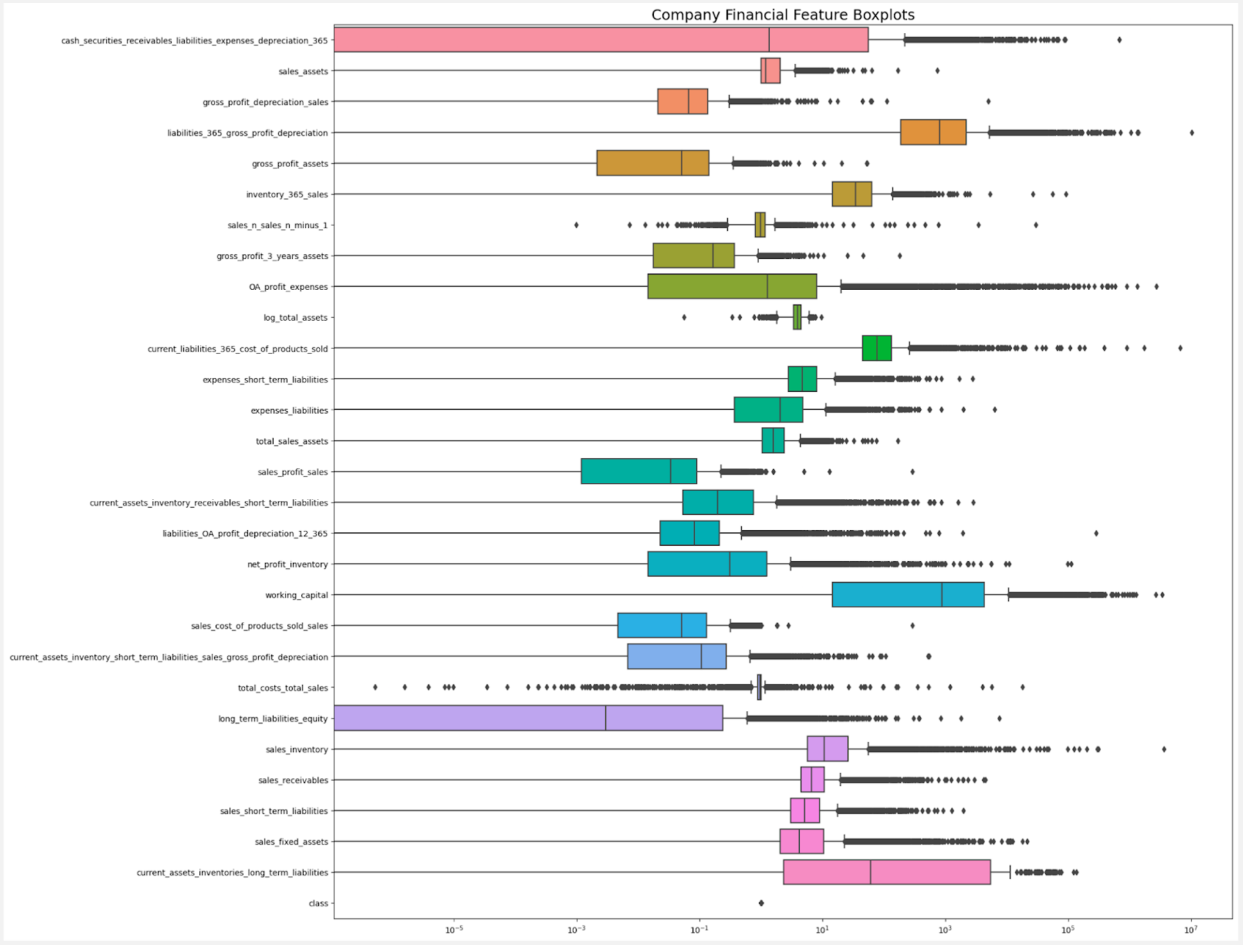
CORRELATION MATRIX OF 64 FEATURES



CORRELATION MATRIX OF SELECTED 29 FEATURES



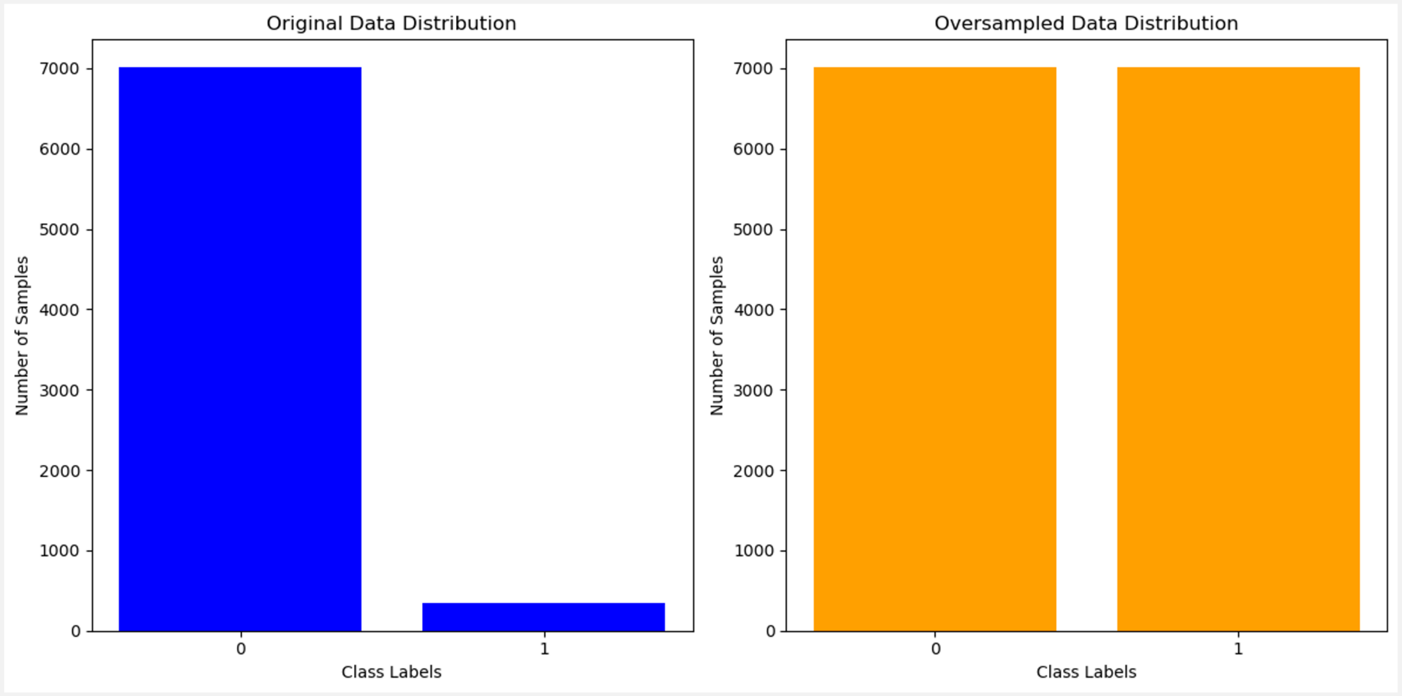
BOX PLOT BEFORE SCALING



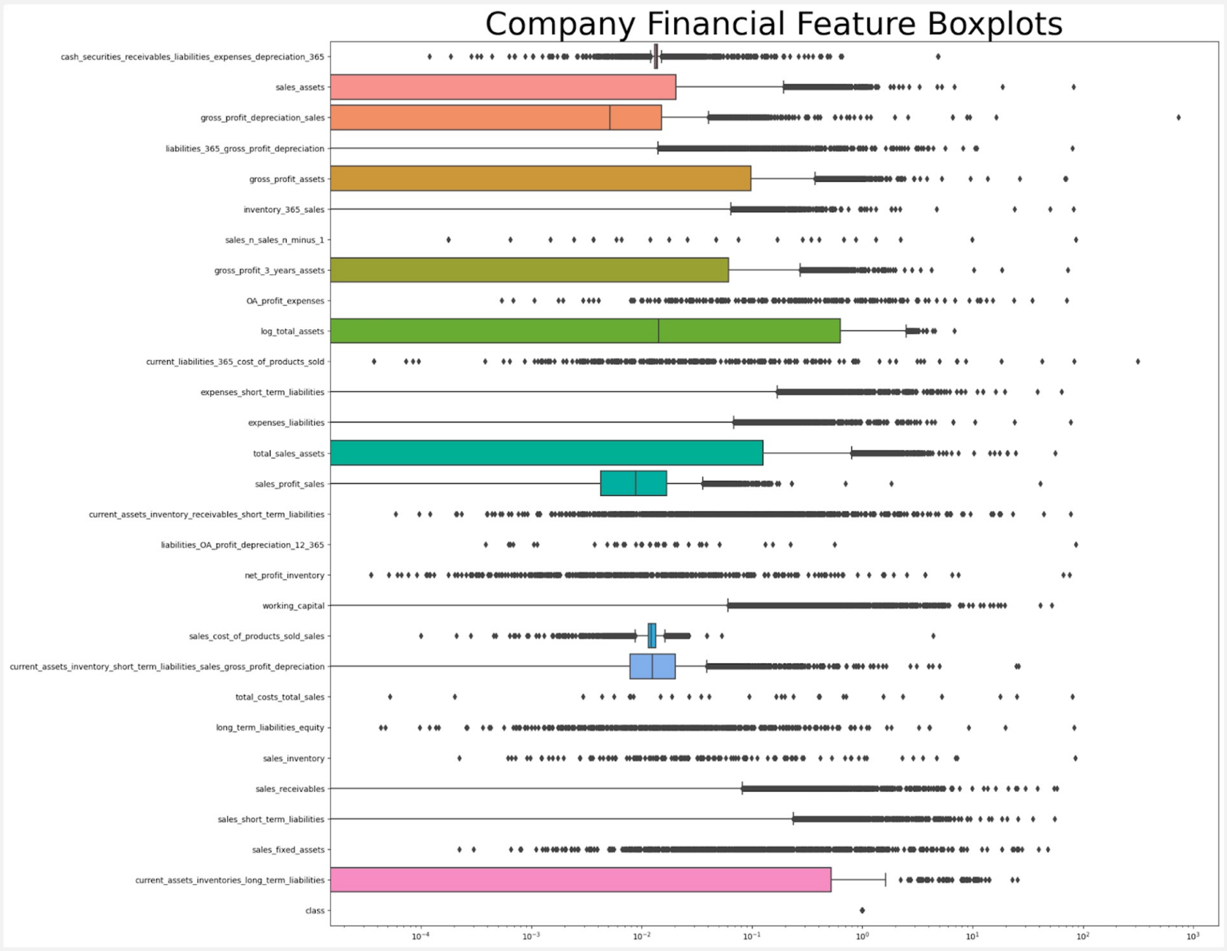
DATA PRE-PROCESSING

- **Train/Test Split Importance:** Separates dataset into training and testing sets for model evaluation, ensuring assessment on unseen data and gauging generalization ability.
- **Normalization for Diverse Scales:** Crucial in financial predictions like bankruptcy, where varied scales exist; normalization techniques like Z-score ensure proportional feature contribution, fostering model stability and convergence.
- **Addressing Class Imbalance:** In cases where one class dominates, Synthetic Minority Over-sampling Technique (SMOTE) is employed, generating synthetic instances of the minority class. This prevents biased learning and enhances the model's exposure to underrepresented cases.
- **SMOTE Procedure:** Involves selecting a sample and considering its k nearest neighbors in feature space. A synthetic data point is then created by multiplying the vector between a neighbor and the current data point by a random number (0 to 1) and adding it to the current data point.
- **Enhancing Exposure to Underrepresented Cases:** SMOTE implementation from the imbalanced-learn library improves the model's ability to identify subtle patterns associated with potential bankruptcies, preventing biased learning.
- **Overall Objective:** These steps collectively aim to ensure robust model training, mitigate the impact of diverse scales, and address class imbalance, enhancing the model's predictive capacity for bankruptcy scenarios.

CLASS DISTRIBUTION BEFORE/AFTER SMOTE



BOX PLOT AFTER SCALING



- The visual representation on the left clearly illustrates the effects of scaling applied to all features, influencing both extreme outliers and the central tendency of the data.

- Initially, the dataset harbored a notable presence of 5-6 extreme outliers, discernible prior to scaling. However, post-scaling, this number has dwindled to a more manageable 2 outliers.

- It's important to note that a complete elimination of all outliers is not a feasible option. Such an approach would substantially diminish the quantity of available data records, thereby affecting the integrity and comprehensiveness of the dataset. Striking a balance between outlier management and preserving dataset size is crucial for maintaining a meaningful and representative dataset.

IMPLEMENTED MODELS

- In our project, we followed a systematic approach to implement classification models, including Logistic Regression, Gaussian Naive Bayes Classifier, Decision Tree Classifier, and Support Vector Machines (SVM).
- The primary workflow involved creating models from scratch and conducting hyperparameter tuning for selected models.
- For other models, we explored the impact of dataset size variation instead of hyperparameter tuning.
- Additionally, we assessed the models on both class-balanced and class-imbalanced data, incorporating Synthetic Minority Over-sampling Technique (SMOTE) to visualize its effects.
- The Bias-Variance trade-off analysis was conducted for all four models.

LOGISTIC REGRESSION

- We applied Logistic Regression as a baseline model after addressing assumptions such as multicollinearity and outlier handling during feature selection and dataset scaling. Logistic Regression functions as a linear classification model, employing the sigmoid function for probability modeling.

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

- The gradient descent formula was used for weight updates

$$\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

NAÏVE BAYES CLASSIFIER

- Assumptions for Gaussian Naive Bayes, including independent features and varied data distribution, were considered. The classifier, based on Bayes' theorem, assumes independence between features.

$$P(y \mid x_1, \ \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

- The Gaussian Naive Bayes application utilizes the Gaussian Naive Bayes algorithm for classification purposes. The concept of Maximum Likelihood Estimation is used to find the values of and uy (mean) and σy2 (variance). The underlying assumption in this approach is that the probability distribution of the f

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

DECISION TREE AND SUPPORT VECTOR MACHINES

- **Decision Tree Classifier:** Assumptions of independent features and balanced data were managed in feature selection and preprocessing. Decision Trees provide a non-parametric approach, constructing a model based on simple decision rules.
- The ‘Entropy' index was used for evaluating the quality of each split.

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

- **Support Vector Machines (SVM):** Assumptions regarding outliers and data imbalance were handled in feature engineering and preprocessing. SVMs, a powerful tool in machine learning, aim to find a hyperplane that effectively separates classes while maximizing the margin between them.
- The decision function is expressed as f(x) = sign(w·x + b).

BIAS VARIANCE TRADE-OFF

- The discussion delved into the intricate relationship between bias and variance, impacting Logistic Regression, Gaussian Naive Bayes, Decision Tree Classifier, and SVM.
- Each model has its trade-off considerations, with Logistic Regression having reduced variance but potential bias, Gaussian Naive Bayes showing minimal variance with introduced bias, Decision Tree Classifier balancing interpretability and overfitting, and SVM addressing both linear and non-linear patterns.
- The formula that we used in our project to calculate Bias-Variance Trade off is:

$$\sigma^2 + \underbrace{\left(\mathbb{E}[\hat{f}(x_0)] - f(x_0)\right)^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}\left[\left(\hat{f}(x_0) - \mathbb{E}[\hat{f}(x_0)]\right)^2\right]}_{\text{Variance}}$$

PRINCIPAL COMPONENT ANALYSIS

- PCA, a potent statistical method, reduces dataset dimensionality while retaining vital information.
- Eigenvalues and eigenvectors play a key role in this process, with PCA establishing a fresh basis for data by organizing eigenvectors based on descending eigenvalues. This method aids in simplifying complex datasets, optimizing computational efficiency, and enhancing interpretability.
- The covariance matrix C is calculated as:

$$C = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})^T$$

where Xi represents the data points and X(bar) is the mean vector.

- The eigenvalue equation is given by: $\mathbf{C}\boldsymbol{v} = \boldsymbol{\lambda}\boldsymbol{v}$

where λ is the eigenvalue and v is the corresponding eigenvector.

RESULTS – MODEL EVALUATION METRICS

		Training			
Model	**Dataset**	**F1-Score**	**Precision**	**Recall**	**Accuracy**
Logistic Regression	Class Imbalanced	0.011236	0.200000	0.005780	0.952122
	Class Balanced	0.681886	0.648917	0.718384	0.664859
Support Vector Machines	Full Data - Class Imbalance	0.944	0.4	0.074	0.125
	Full Data - Class Balance	0.718	0.7167	0.7835	0.7486

		Testing			
Model	**Dataset**	**F1-Score**	**Precision**	**Recall**	**Accuracy**
Logistic Regression	Class Imbalanced	Nan	0.000000	0.000000	0.951761
	Class Balanced	0.157133	0.087558	0.765101	0.611869
Naive Bayes	Class Imbalanced	0.0925	0.048	0.973	0.098
	Class Balanced	0.092377	0.048524	0.959732	0.108220
Support Vector Machines	Full Data - Class Imbalance	0.943	0.03	0.0067	0.011
	Full Data - Class Balance	0.5982	0.090	0.8322	0.1669
Decision Tree Classifier	500 Sample - Class Imbalance	0.386	0.689	0.2684	0.959
	500 Sample - Class Balance	0.2571	0.175	0.4832	0.8679

RESULTS

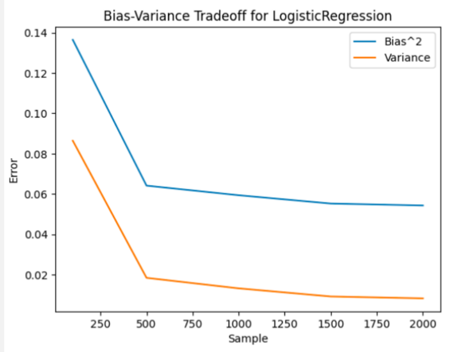
Model	Dataset	Time Taken to tune the model	Time Taken to fit the best model
Logistic Regression	Class Imbalanced	10.2 s	615 ms
	Class Balanced	42.5 s	757 ms
Naive Bayes	Class Imbalanced	NaN	16 s
	Class Balanced	NaN	15.9 s
Support Vector Machines	500 Sample - Class Imbalanced	11.9 µs	849 ms
	500 Sample - Class Balanced	16.607 s	2.5 s
Decision Tree Classifier	500 Sample - Class Imbalanced	208.14 s	41.7 s
	500 Sample - Class Balanced	3min 25s	42.7 s

Bias Variance Trade Off Analysis on Class Imbalanced Data - 12 min 59s

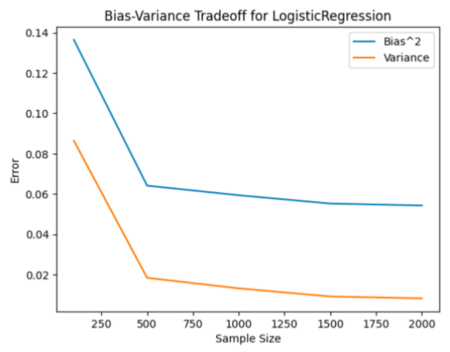
Bias Variance Trade Off Analysis on Class Balanced Data - 13 min 15s

BIAS VARIANCE TRADE OFF ANALYSIS

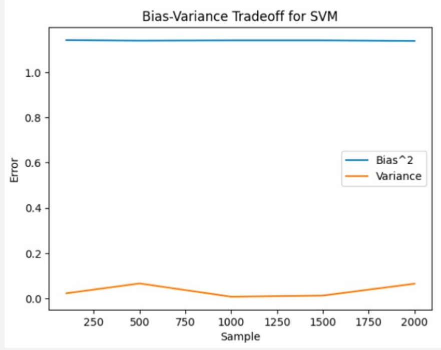
Logistic Regression with Class Imbalance



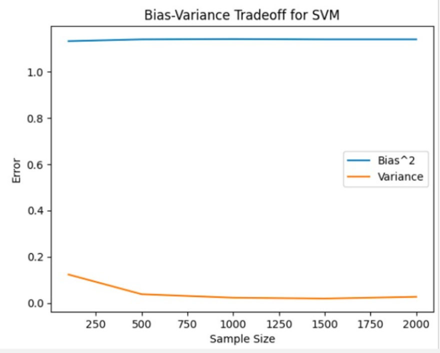
Logistic Regression with Class Balance



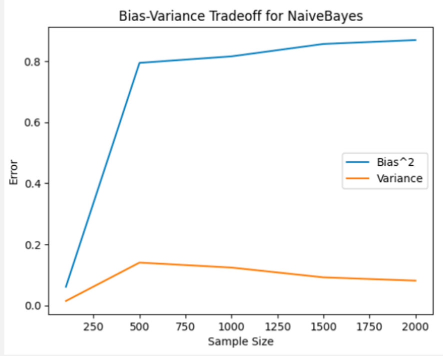
Support Vector Machine with Class Imbalance



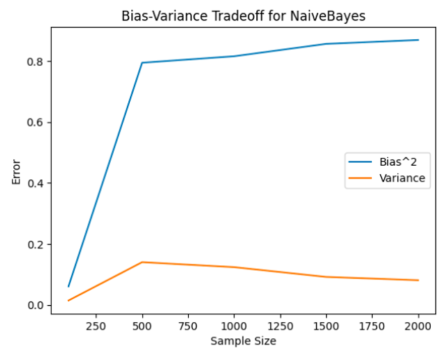
Support Vector Machine with Class Balance



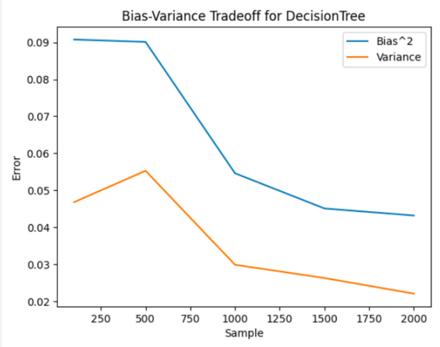
Naive Bayes with Class Imbalance



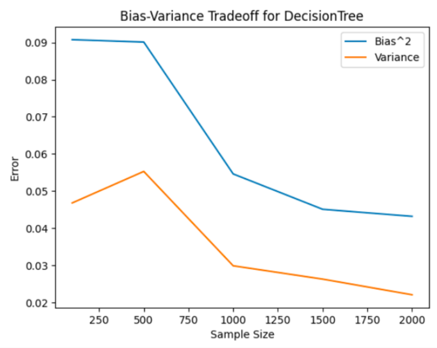
Naive Bayes with Class Balance



Decision Tree with Class Imbalance



Decision Tree with Class Balance



DISCUSSIONS AND OBSERVATIONS

- The logistic regression model encounters challenges in accurately identifying the minority class ('1') in an imbalanced dataset, resulting in "NaN" values in specific metrics.
- Its high accuracy is primarily attributed to effectively categorizing the majority class ('0') due to its prevalent presence in the dataset.
- Utilizing the model on a well-balanced dataset with fine-tuned hyperparameters demonstrates strong effectiveness in achieving a synergistic balance between precision and recall.
- SVM exhibits commendable F1-Score despite class imbalance during training, with challenges in precision and recall due to uneven class distribution.
- In the testing phase on the imbalanced dataset, SVM maintains a high F1-Score but struggles in predicting the minority class, as indicated by low precision and recall.
- Transitioning to a balanced dataset for training compromises the F1-Score but achieves better balance between precision and recall, improving performance across both classes.

DISCUSSIONS AND OBSERVATIONS

- Naive Bayes excels in identifying fewer common instances of '1' but struggles in achieving overall accuracy across both types of cases.
- Performance is influenced by the dataset's structure and the imbalance in occurrence frequencies.
- Gaussian Naive Bayes faces challenges in achieving accuracy and precision for both positive and negative classifications, even with a balanced dataset.
- In a class-imbalanced dataset during training, the Decision Tree Classifier exhibits moderate F1-Score, high Precision, and Recall for the majority class.
- When applied to a balanced dataset, the model shows a reduced F1-Score but achieves a better trade-off between Precision and Recall, still facing challenges in capturing minority instances.
- The Balanced Accuracy reflects a compromise for enhanced balance in predictions in the balanced dataset.

CONCLUSION

- Decision Tree Classifier on imbalanced dataset achieves highest accuracy (95.9%).
- High accuracy attributed to dominance of majority class in the imbalanced dataset.
- Accuracy metric can be misleading in imbalanced data scenarios.
- Model may achieve high accuracy by predominantly predicting the majority class.
- Despite high overall accuracy, model struggles with moderate F1-Score and imbalanced precision/recall.
- Decision Tree Classifier on balanced dataset shows reduced accuracy compared to imbalanced dataset.
- Reduced accuracy on balanced dataset attributed to trade-off for better balance between classes.
- Model faces challenges in accurately classifying both majority and minority classes in balanced dataset.
- Transition to balanced dataset requires model to adapt predictions for both classes, leading to more errors.
- Decision tree struggles to generalize well to balanced dataset, navigating between characteristics of both classes.
- Reduced accuracy in balanced scenario reflects model's effort to achieve better equilibrium between both classes, contrasting with imbalanced setting.

THANK YOU