Company Bankruptcy Prediction

This document outlines the development of a machine learning model designed to predict company bankruptcy based on various financial indicators. It covers the entire process from exploratory data analysis and preprocessing to model architecture and evaluation, highlighting key steps like feature extraction, oversampling, and ensemble modeling.

Outline of the Prediction Model Development

1 Exploratory Data Analysis

Initial examination of data patterns, null values, and class distribution.

2 Data Preprocessing

Includes feature extraction, handling redundant data, and oversampling techniques.

3 Model Architecture

Details the design of the Deep Neural Network and Gaussian Naive Bayes models.

4 Evaluation

Assessment of model performance using Accuracy, Precision, Recall, and F1 Score.

Exploratory Data Analysis

The initial phase involved exploring patterns and characteristics within the dataset. We identified no null values or duplicate rows in the dataset, which comprised 7027 rows and 66 columns (65 features and 1 target variable). A significant class imbalance was noted, with 271 bankrupted companies (class 1) and 6756 non-bankrupted companies (class 0).

Handling Redundant Data and Errors

Out of 65 features, 63 were numerical, and 2 were categorical ("Liability-Assets Flag" and "Net Income Flag"). The "Net Income Flag" was dropped as it had a constant value (1) across all data points. We also addressed data errors where some features had extremely high mean values due to false data. Specifically, 16 features with over 800 errors were removed, and 19 features with fewer than 200 errors were corrected using median replacement, resulting in 50 retained features.

Feature Correlation

Pairwise correlation was calculated for all feature pairs. If the absolute correlation between any pair exceeded a threshold of 0.90, one feature was removed. For correlated pairs, the feature with a higher absolute correlation to the target variable ("class") was retained, and the other was dropped. This process effectively reduced the number of features from 63 to 50, streamlining the dataset for modeling.

Data Preprocessing: Feature Extraction Using ANOVA

Analysis of Variance (ANOVA) was employed to determine the significance of each feature for model training. ANOVA calculates F-scores for each feature; a high F-score indicates that feature values vary significantly between target variable classes and less within a single class, making the feature highly significant.

$$S\_B^2 = sum\_i = 1^k N\_i(mu\_i - mu)^2$$

Here, k=2 (number of classes), N_i is the number of data points in class i, μ is the overall mean of the feature, and μ_i is the mean of that feature within class i. The variance within groups is calculated by summing the weighted variances of each class. Comparing between-group variance to within-group variance provides a measure of feature significance.

```
def select_features_anova(X_data, y_data, top_n=30):
    selector = SelectKBest(score_func=f_classif, k=top_n)
    selector.fit(X_data, y_data)
    mask = selector.get_support()
    selected_features = X_data.columns[mask].tolist()
    return selected_features
```

The `SelectKBest` function selects the top 30 features based on their F-scores, and the `mask` extracts only these selected features.

Data Preprocessing: Oversampling using SMOTE

To address the class imbalance in the dataset, where models tend to be biased towards the majority class, we utilized SMOTE (Synthetic Minority Over-sampling Technique). Initially, our dataset had 271 samples of class '1' (bankrupted companies) and 5756 samples of class '0' (non-bankrupted companies).

After an 80-20 train-test split, our training data contained 5397 samples of class '0' and 224 samples of class '1'. SMOTE works by randomly selecting a minority class sample, finding its k nearest neighbors, and then generating new synthetic samples between the original sample and one of its neighbors using linear interpolation. This process artificially increases the number of minority class samples.

Following the application of SMOTE, our training dataset achieved a balanced distribution with 5397 samples for each class. It is crucial to note that SMOTE is applied only to the training data and not to the test data, ensuring an unbiased evaluation of the model's performance on unseen data.

Data Preprocessing: Standardisation

Standardization is a critical step in data preprocessing to ensure that all features contribute equally to the model training process. We employ the `StandardScaler` to transform our feature values. This scaler adjusts the data such that all features have a mean of 0 and a unit variance.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_sm)
X_test_scaled = scaler.transform(X_test)
```

The primary reason for this transformation is to prevent features with larger numerical scales from disproportionately influencing the model's learning process. Without standardization, features with larger values might be assigned unnecessarily large weights, leading to a biased model. By scaling the data, we ensure that the model considers the relative importance of each feature rather than its absolute magnitude.

The `fit_transform` method is applied to the training data (`X_train_sm`) to learn the scaling parameters (mean and standard deviation) and then apply the transformation. For the test data (`X_test`), only the `transform` method is used, applying the scaling parameters learned from the training data. This prevents data leakage from the test set into the training process.

Model Architecture

Our prediction system employs an ensemble approach combining a Deep Neural Network (DNN) and a Gaussian Naive Bayes (GNB) model, utilizing soft voting for final probability prediction.

Deep Neural Network (DNN)

The DNN is structured with an input layer, three hidden layers, and an output layer. The input layer takes the processed features. The first hidden layer has 256 neurons, followed by a second with 128 neurons, and a third with 64 neurons. All hidden layers use the ReLU activation function, Batch Normalization, and Dropout (0.5 for the first two, 0.4 for the third) to prevent overfitting. The output layer consists of a single neuron with a sigmoid activation function for binary classification. The model is optimized using Adam (learning rate = 0.0005) and uses Binary Crossentropy as its loss function. It is trained for 200 epochs with a batch size of 64 and a 20% validation split.

Gaussian Naive Bayes (GNB)

The GNB model applies Bayes' theorem, assuming feature independence given the class label. It calculates bankruptcy probability by multiplying Gaussian likelihoods of each feature and normalizing with prior probabilities.

Ensemble Approach

We combine the predicted probabilities from both models by averaging them. The final prediction of bankruptcy is determined by tuning a threshold value, which is optimized by calculating the F1-score for various threshold values.

Evaluation

Using our "Train.csv" dataset, we evaluated the model's performance on a 20% test data split. The results for various metrics are as follows:

Precision	0.0128
Recall	0.6667
F1-Score	0.0252
Accuracy	0.9723
Best threshold	0.4

The model achieved a high accuracy of 0.9723, indicating its ability to correctly classify a large proportion of companies. However, the low precision (0.0128) suggests a high rate of false positives, meaning many companies predicted to go bankrupt did not. The recall of 0.6667 indicates that the model successfully identified two-thirds of the actual bankrupt companies. The F1-Score of 0.0252, which is the harmonic mean of precision and recall, reflects the overall balance between these two metrics, highlighting the challenge of predicting the minority class (bankruptcy) in an imbalanced dataset. The optimal threshold for predicting bankruptcy was determined to be 0.4.