AI Bootcamp

Project Numero Dos

Company ~~$Bankruptcy$~~ Prediction

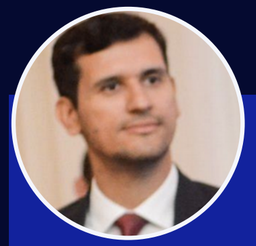
Team 7

Content

Our ~~$Bankrupt$~~ Team

For now

Andrew Carson



Manuel Parra

Alghalia Alsammak

Jie Zhu

Syed Shah

Our Goal

The goal is to develop a machine learning model that can predict whether a company is likely to go bankrupt taking into account various financial indicators, ~~so we can invest to eventually make millions and live on a 60' yacht.~~

Data Inputs / Cleaning

Sources

- Taiwan Economic Journal for the years 1999-2009 @Kaggle
- Bankruptcy was defined based on regulations of the Taiwan Stock Exchange.

Data Cleaning

- Data came already scaled
- Made a new dataframe with the 10 most important columns

Aggregation

- Compared the original 95 feature data frame vs. the 10 most important columns.

Solution

Address missing data

Outliers

Relevant features

DATA FORMATTING

Logistic Regression

K-Nearest Neighbors (KNN)

Random Forest

Decision Tree

CREATING AND TUNING THE MODELS

Accuracy

Precision

Recall

F1-score

TESTING & MODELS COMPARISON

Analysis

insights

Recommendations

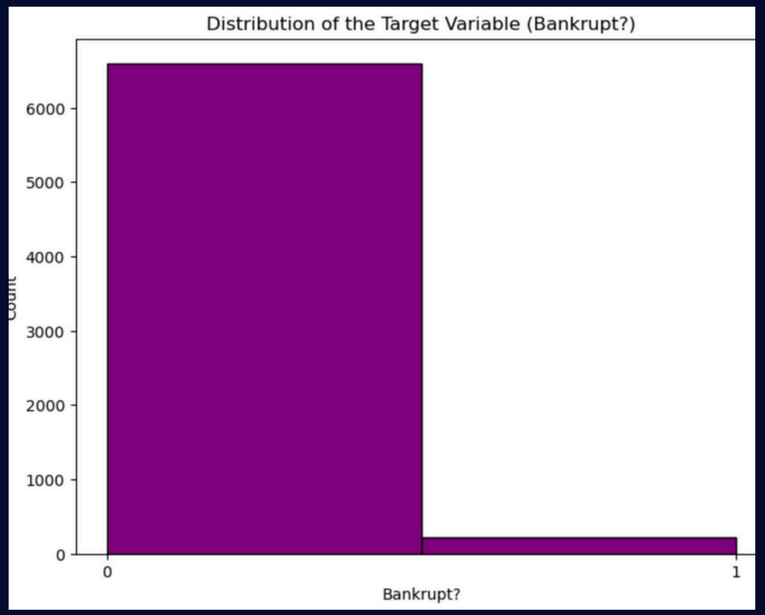
Decision-making

FINAL REPORT

Data - Summary

Number of Features and Description:

- Number of Features: 95
- Feature Types: Binary



Number of Instances: 6,819 companies

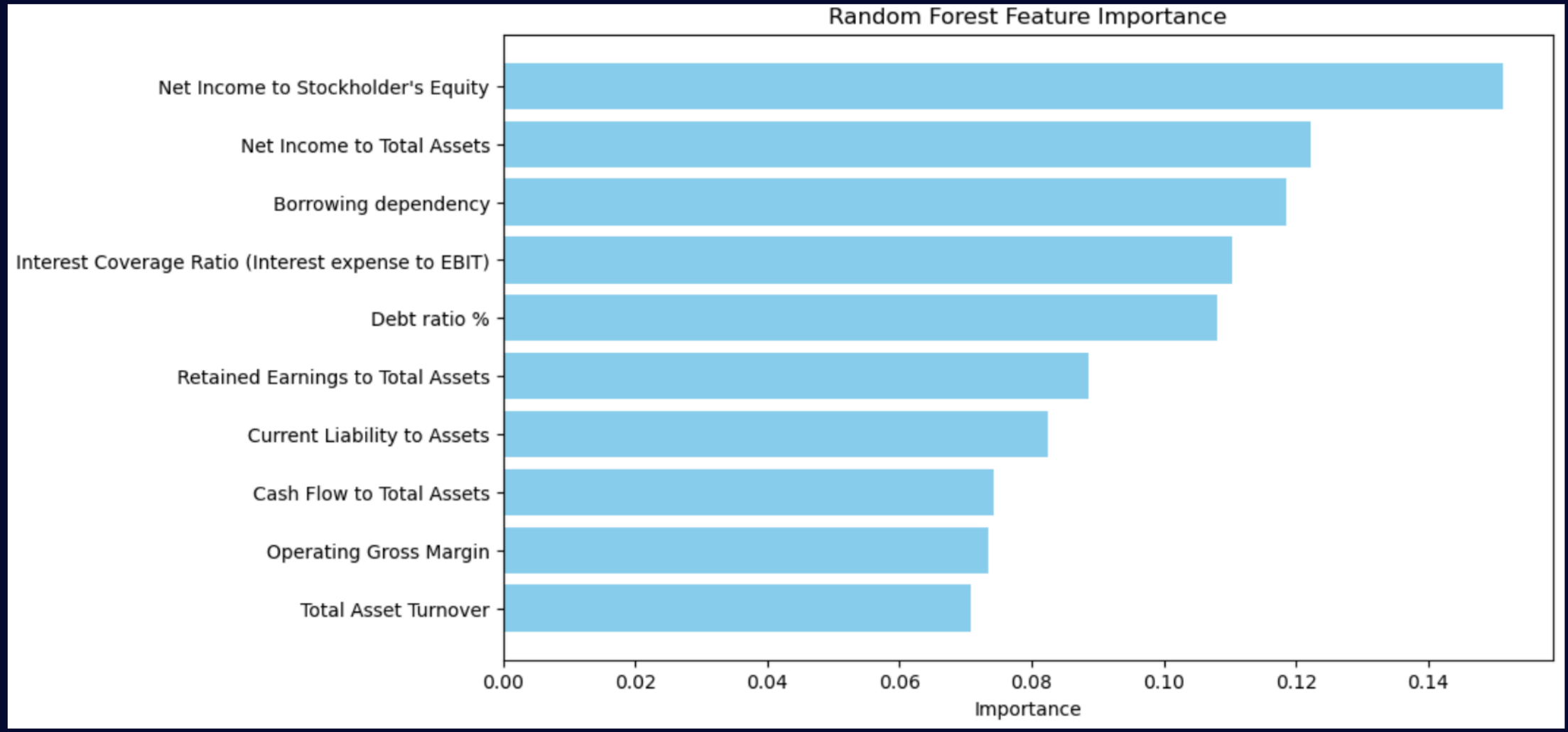
Non Bankrupt	0	6599
Bankrupt	1	220

Target Variable:

- Bankrupt? (Binary classification - 1 for bankrupt, 0 for not bankrupt)
- Missing Values: No missing values in the dataset
- Imbalanced Data: Only ~3.2% of companies in the dataset are bankrupt

Key Features for Bankruptcy Prediction:

- These 10 features were selected because they each provide important information regarding the financial health of a company and can be used to predict its likelihood of going bankrupt. We preformed analysis using the Random Forest model to predict bankruptcy.



Model Analysis Full Dataset vs Filtered Key Dataset

Model	Data Treatment	Accuracy	Balanced Accuracy	ROC-AUC	Precision (Class 0)	Recall (Class 0)	F1-score (Class 0)	Precision (Class 1)	Recall (Class 1)	F1-score (Class 1)
KNN	As Is	0.9677	0.5000	0.6362	0.97	1.00	0.98	0.00	0.00	0.00
KNN	Filtered	0.9660	0.5606	0.8466	0.97	0.99	0.98	0.41	0.13	0.19
Logistic Regression	As Is	0.9566	0.4942	0.5848	0.97	0.99	0.98	0.00	0.00	0.00
Logistic Regression	Filtered	0.9683	0.5179	0.9332	0.97	1.00	0.98	0.67	0.04	0.07
Random Forest	As Is	0.9689	0.5709	0.9255	0.97	1.00	0.98	0.57	0.15	0.23
Random Forest	Filtered	0.9695	0.5888	0.9008	0.97	1.00	0.98	0.59	0.18	0.28
Decision Tree	As Is	0.9584	0.6533	0.6533	0.98	0.98	0.98	0.35	0.33	0.34
Decision Tree	Filtered	0.9537	0.6333	0.6333	0.98	0.98	0.98	0.29	0.29	0.29

The Pipeline

Our first attempt at the pipeline.

In [1]:

```
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import pandas as pd

# Load the dataset
file_path = '/Users/macbook/Desktop/project2/BankruptcyData.csv'  # Update with your actual path
data = pd.read_csv(file_path)

# Select the relevant features and target
features = data.drop(columns=['Bankrupt?'])  # All columns except the target variable
target = data['Bankrupt?']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.3, random_state=42, stratify=ta

# Define oversampling and undersampling strategy
over = SMOTE(sampling_strategy=0.5)  # Oversample the minority class to 50% of the majority class
under = RandomUnderSampler(sampling_strategy=0.7)  # Undersample the majority class to 70% of its original size

# Create a pipeline for sampling and training
pipeline = Pipeline(steps=[
    ('over', over),                 # Step 1: Oversample the minority class using SMOTE
    ('under', under),               # Step 2: Undersample the majority class using RandomUnderSampler
    ('model', RandomForestClassifier(n_estimators=100, random_state=42))  # Step 3: Train RandomForestClassifier
])

# Train the model using the pipeline
pipeline.fit(X_train, y_train)

# Make predictions on the test set
y_pred = pipeline.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Display the results
print(f"Accuracy: {accuracy:.4f}")
print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Classification Report:\n{class_report}")
```

```
Accuracy: 0.9526
Confusion Matrix:
[[1912   68]
 [  29   37]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1980
           1       0.35      0.56      0.43        66
```

Full Models Comparison

Model	Data Treatment	Accuracy	Balanced Accuracy	ROC-AUC	Precision Class 0	Recall Class 0	F1-score Class 0	Precision Class 1	Recall Class 1	F1-score Class 1
KNN	As Is	0.9660	0.5606	0.8466	0.97	0.99	0.98	0.41	0.13	0.19
KNN	Undersampled	0.8381	0.8461	0.9106	0.99	0.84	0.91	0.15	0.85	0.25
KNN	Oversampled	0.9097	0.7864	0.8402	0.99	0.92	0.95	0.21	0.65	0.32
Logistic Regression	As Is	0.9683	0.5179	0.9332	0.97	1.00	0.98	0.67	0.04	0.07
Logistic Regression	Undersampled	0.8575	0.8473	0.9296	0.99	0.86	0.92	0.16	0.84	0.27
Logistic Regression	Oversampled	0.8592	0.8482	0.9333	0.99	0.86	0.92	0.17	0.84	0.28
Random Forest	As Is	0.9695	0.5888	0.9008	0.97	1.00	0.98	0.59	0.18	0.28
Random Forest	Undersampled	0.8545	0.8282	0.9183	0.99	0.86	0.92	0.16	0.80	0.26
Random Forest	Oversampled	0.9689	0.6324	0.8979	0.98	0.99	0.98	0.54	0.27	0.36
Decision Tree	As Is	0.9537	0.6333	0.6333	0.98	0.98	0.98	0.29	0.29	0.29
Decision Tree	Undersampled	0.7871	0.7933	0.7933	0.99	0.79	0.88	0.11	0.80	0.20
Decision Tree	Oversampled	0.9554	0.6079	0.6079	0.97	0.98	0.98	0.28	0.24	0.25

Our Choice: Random Forest Oversampled

Model	Data Treatment	Accuracy	Balanced Accuracy	ROC-AUC	Precision Class 0	Recall Class 0	F1-score Class 0	Precision Class 1	Recall Class 1	F1-score Class 1
Random Forest	As Is	0.9695	0.5888	0.9008	0.97	1.00	0.98	0.59	0.18	0.28
Random Forest	Undersampled	0.8545	0.8282	0.9183	0.99	0.86	0.92	0.16	0.80	0.26
Random Forest	Oversampled	0.9689	0.6324	0.8979	0.98	0.99	0.98	0.54	0.27	0.36

Analysis: Why Random Forest Oversampled?

- Among all the metrics Random Forest had the highest metrics.
- Getting it Right (Precision Class 1 or Predicting Bankrupt =1) was the main target of the model, hence this was the heaviest metric to consider.
- Even though Random Forest “As Is” has a Higher Precision Class 1, it had a a lower Recall Class 1, hence we took a trade off when choosing Random Forest “Oversampled”, as the improvement in Class 1 Recall is higher than the decrease in Precision class 1

Analysis Report

Correlations:

- Most features have weak to moderate correlations with the target variable (Bankrupt?).
- Debt Ratio % and Current Liability to Assets have positive correlations, suggesting that higher values are linked to increased bankruptcy risk.
- Net Income to Total Assets and ROA variants show negative correlations, indicating that higher profitability reduces bankruptcy risk.

Modeling Recommendation:

- Due to the weak correlation of individual features, complex models like Random Forest, Gradient Boosting, are recommended.

Lessons Learned & Next Steps

Challenges faced

- The challenge we faced was that the data was very unbalanced. With this in mind, we got the balanced accuracy score for each of the models. This gave us a more holistic view on the dataset.

Future Improvements

- The analysis suggests that predicting bankruptcy is a complex task involving multiple factors. No single financial metric is a strong predictor; thus, utilizing a combination of features with advanced machine learning models is necessary to build a reliable prediction model.

~~Thank You~~