WHO'S LOSING THE GAME

---POLISH COMPANY BANKRUPTCY PREDICTION

KEN BOXEN SAMMUEL DOROTHY

QTM 347 FINAL PROJECT

CONTENT

Intro & Data Set

Hypo & Method

Result

Discussion

INTRODUCTION

BANK RUPTCY

- **Background:** Research on predicting bankruptcy has been ongoing for nearly a century, highlighting its importance in economics.
- **Objective:** Develop a model to predict a company's financial health using various economic indicators.
- **Historical Approaches:** Initially focused on statistical hypothesis testing and modeling, with a shift towards artificial intelligence in recent years.
- **Project Focus:** Applying contemporary ML methods learned in this course to enhance prediction accuracy.

DATASET

Source

Polish Companies Bankruptcy

- **Source:** UCI Machine Learning Repository
- **Description:** This dataset focuses on the prediction of bankruptcy for Polish companies.
- **Time Frame:**
  - Bankrupt companies were analyzed from 2000 to 2012.
  - Operating companies were evaluated from 2007 to 2013.
- **Purpose:** To develop models that can accurately predict the financial health and potential bankruptcy of companies based on historical data.

Polish Companies Bankrupty

Additional Variable Information

X1	net profit / total assets
X2	total liabilities / total assets
X3	working capital / total assets
X4	current assets / short-term liabilities
X5	[(cash + short-term securities + receivables - short-term liabilities) / (operating e
X6	retained earnings / total assets
X7	EBIT / total assets
X8	book value of equity / total liabilities
X9	sales / total assets
X10	equity / total assets
X11	(gross profit + extraordinary items + financial expenses) / total assets
X12	gross profit / short-term liabilities
X13	(gross profit + depreciation) / sales
X14	(gross profit + interest) / total assets
X15	(total liabilities * 365) / (gross profit + depreciation)
X16	(gross profit + depreciation) / total liabilities
X17	total assets / total liabilities
X18	gross profit / total assets

.

.

...64 features in total

DATA SETUP

1. **Drop Missing Values**
2. **Split into Test & Train**
3. **Target Variable: Bankruptcy (Binomial)**
4. **Feature Variables A1-A64 (Numerical)**

HYPOTHESIS

Bankruptcy can be predicted by enterprise feature index

METHOD

Logistic Regression

Logistic Regression is a linear classification algorithm used for binary classification tasks. It models the probability that a given input belongs to a certain class by applying a logistic function to a linear combination of input features

Gradient Boost

Gradient Boosting is an ensemble learning technique that builds a sequence of weak learners, typically decision trees, to make accurate predictions. Each weak learner is trained to correct the errors of its predecessor, gradually improving the overall model's predictive performance.

METHOD

Logistic Regression

Strengths: computationally efficient and interpretable, It provides probabilistic predictions, allowing for intuitive interpretation of the model's confidence in its predictions. Additionally, logistic regression performs well with linearly separable data and is robust to noise

Weaknesses: limitation to linear decision boundaries, sensitivity to outliers, and inability to handle missing data without preprocessing

Gradient Boost

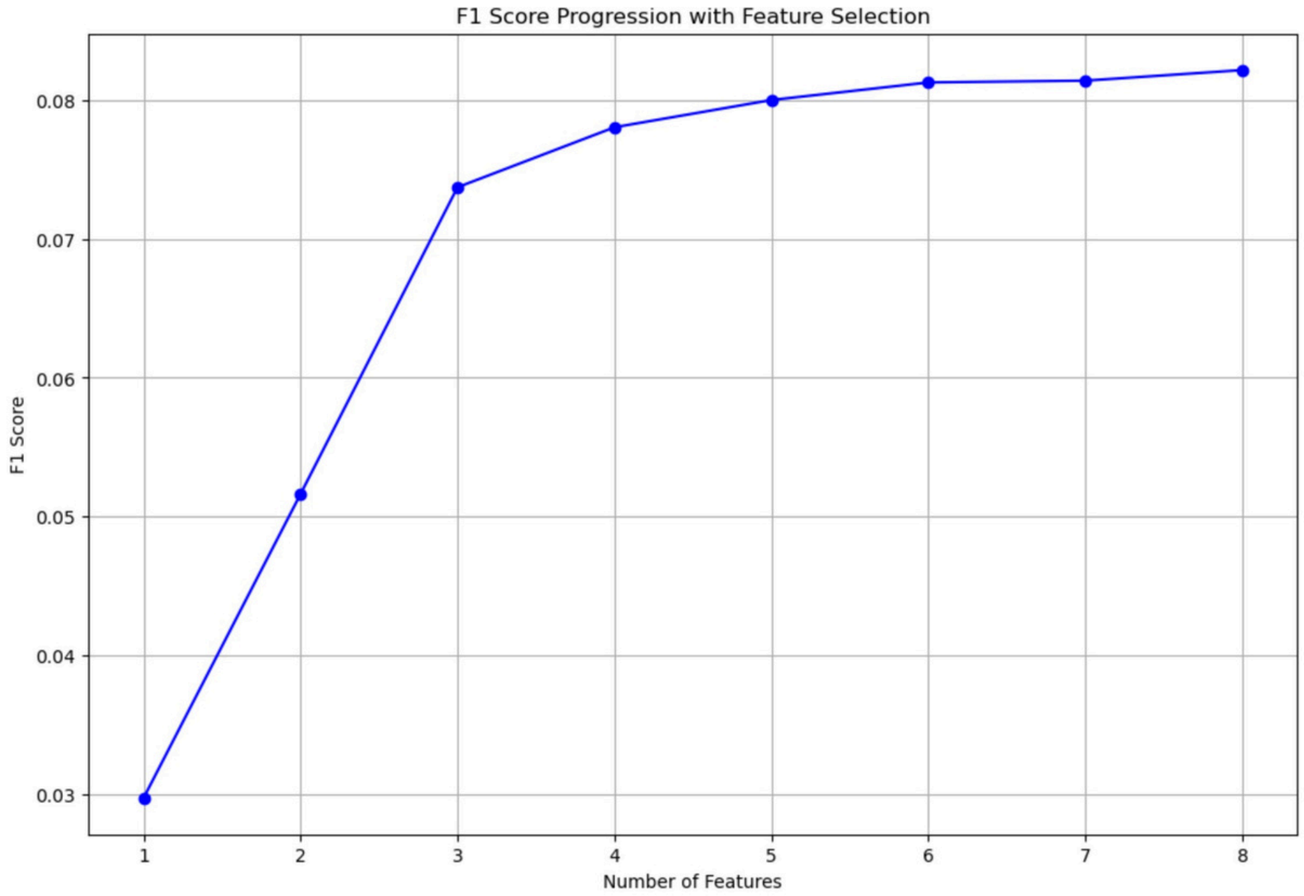
Strengths: ability to handle complex datasets and capture non-linear relationships by sequentially refining the model's predictions. It often achieves state-of-the-art performance in predictive tasks due to its adaptability and flexibility

Weaknesses: computationally intensive and sensitive to overfitting, requiring careful tuning of hyperparameters and regularization techniques to prevent model complexity from leading to reduced generalization performance.

LOGISTIC REGRESSION

```
processSubset(feature_set):
# Fit model on feature_set and calculate cross-validated F1-score
model = LogisticRegression()
f1 = cross_val_score(model, X_train[list(feature_set)], y_train, cv=5, scoring='f1').mean(
return {"model": model, "features": feature_set, "f1_score": f1}

forward(predictors):
remaining_predictors = [p for p in X_train.columns if p not in predictors]
tic = time.time()
results = []

for p in remaining_predictors:
    results.append(processSubset(predictors+[p]))

models = pd.DataFrame(results)
best_model = models.loc[models['f1_score'].idxmax()]

toc = time.time()
print("Processed ", models.shape[0], "models on", len(predictors)+1, "predictors in", (toc

return best_model

els_fwd = pd.DataFrame(columns=["f1_score", "model", "features"])
```

```
Processed  65 models on 1 predictors in 2.6306240558624268 seconds.
Processed  64 models on 2 predictors in 13.502313137054443 seconds.
Processed  63 models on 3 predictors in 16.61509394645691 seconds.
Processed  62 models on 4 predictors in 26.884103059768677 seconds.
Processed  61 models on 5 predictors in 21.413753986358643 seconds.
Processed  60 models on 6 predictors in 27.247490882873535 seconds.
Processed  59 models on 7 predictors in 32.587640047073364 seconds.
Processed  58 models on 8 predictors in 22.86913275718689 seconds.
Processed  57 models on 9 predictors in 30.244354248046875 seconds.
Total elapsed time: 194.01595497131348 seconds.
```



- **Subset Selection**
- **Cross-validation**

```
Accuracy: 0.9574397888485648
Precision: 0.09090909090909091
Recall: 0.029411764705882353
F1 Score: 0.04444444444444444
```

A15 (TOTAL LIABILITIES * 365) / (GROSS PROFIT + DEPRECIATION)
A5 [(CASH + SHORT-TERM SECURITIES + RECEIVABLES - SHORT-TERM LIABILITIES) / (OPERATING EXPENSES - DEPRECIATION)] * 365
A28 WORKING CAPITAL / FIXED ASSETS
A56 (SALES - COST OF PRODUCTS SOLD) / SALES
A13 (GROSS PROFIT + DEPRECIATION) / SALES
A12 GROSS PROFIT / SHORT-TERM LIABILITIES
A49 EBITDA (PROFIT ON OPERATING ACTIVITIES - DEPRECIATION) / SALES
A57 (CURRENT ASSETS - INVENTORY - SHORT-TERM LIABILITIES) / (SALES - GROSS PROFIT - DEPRECIATION)

```python
bag_polish = RF(max_features=X_train.shape[1],
                random_state=0)
bag_polish.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(max_features=65, random_state=0)
```

```python
feature_names = X_train.columns
feature_imp = pd.DataFrame({'importance':bag_polish.feature_importances_}, index=feature_names)
feature_imp.sort_values(by='importance', ascending=False).head(8)
```

[56] ✓ 0.0s

	importance
A24	0.047636
A46	0.046624
A58	0.041193
A5	0.040817
A34	0.040738
A39	0.034354
A56	0.029402
A26	0.026688

PRESENTATION

GRAIDENT BOOSTING

```python
best_clf = grid.best_estimator_
test_f1= f1_score(y_test, best_clf.predict(X_test))

print("Best parameters:", best_clf)
print("Test F1 Score with best parameters:", test_f1
```

67%

[0] ✓ 23m 55.1s Python

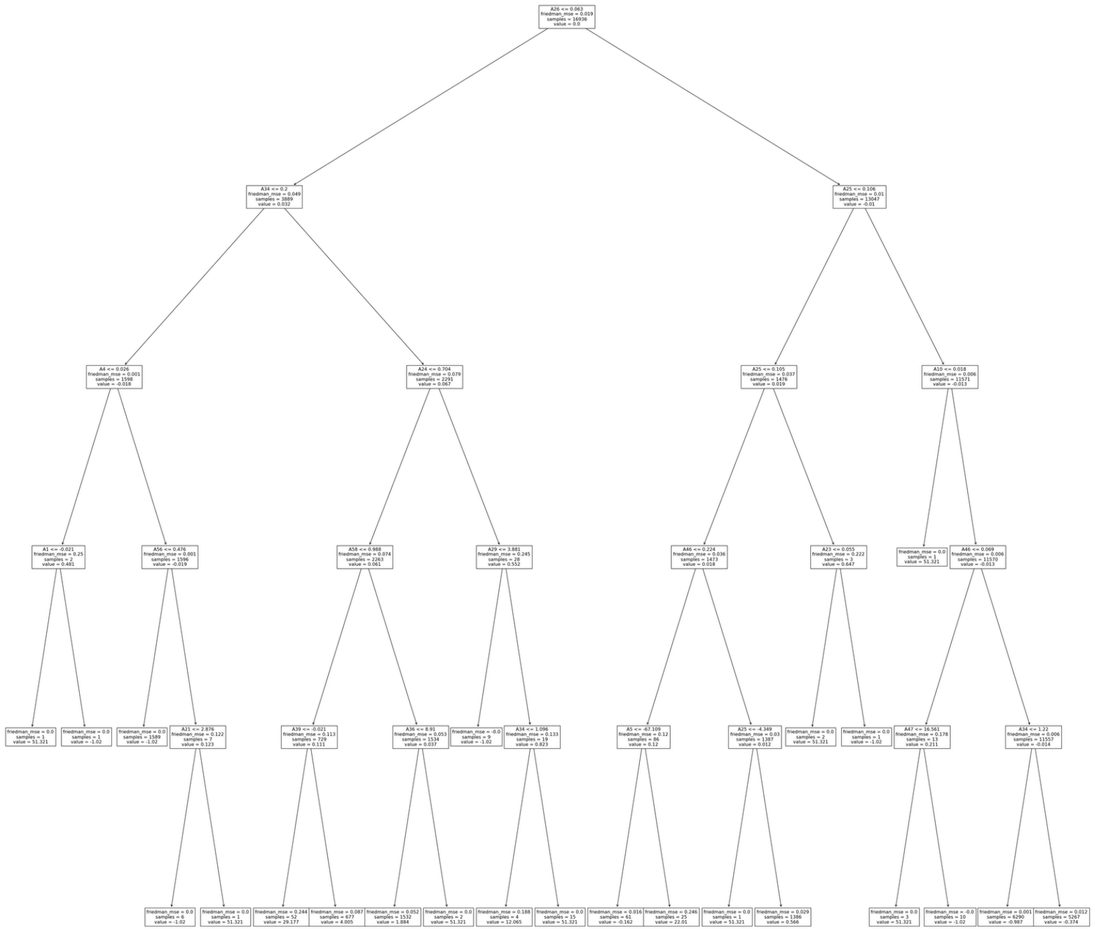
```
Best parameters: GradientBoostingClassifier(learning_rate=0.08999999999999998, max_depth=5,
                                            random_state=0)
Test F1 Score with best parameters: 0.4172661870503597
```

```python
from sklearn.metrics import recall_score
from sklearn.metrics import accuracy_score

y_pred = best_clf.predict(X_test)
recall = recall_score(y_test, y_pred)
print("Recall rate:", recall)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

[1] ✓ 0.0s Python

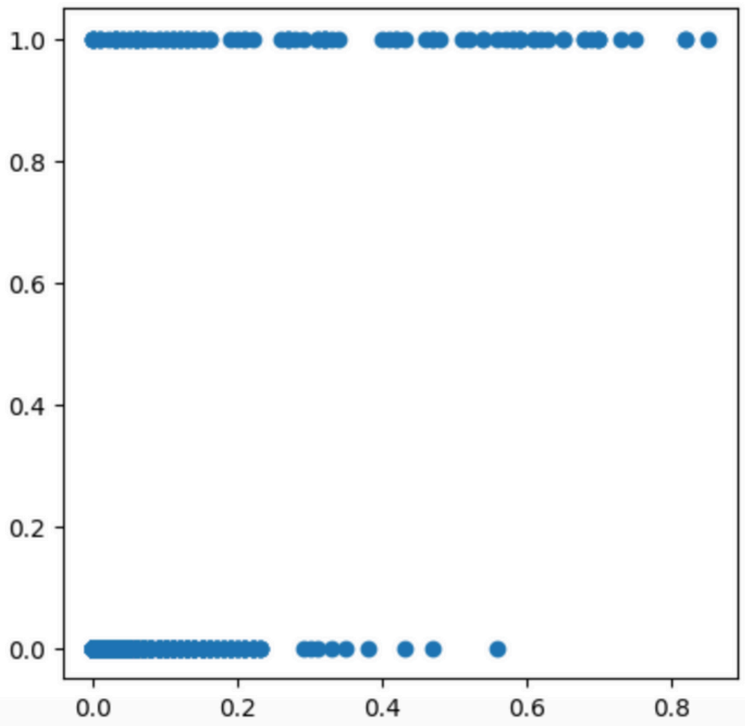
```
Recall rate: 0.28431372549019607
Accuracy: 0.9732761464863081
```

TREES AND FIGURES



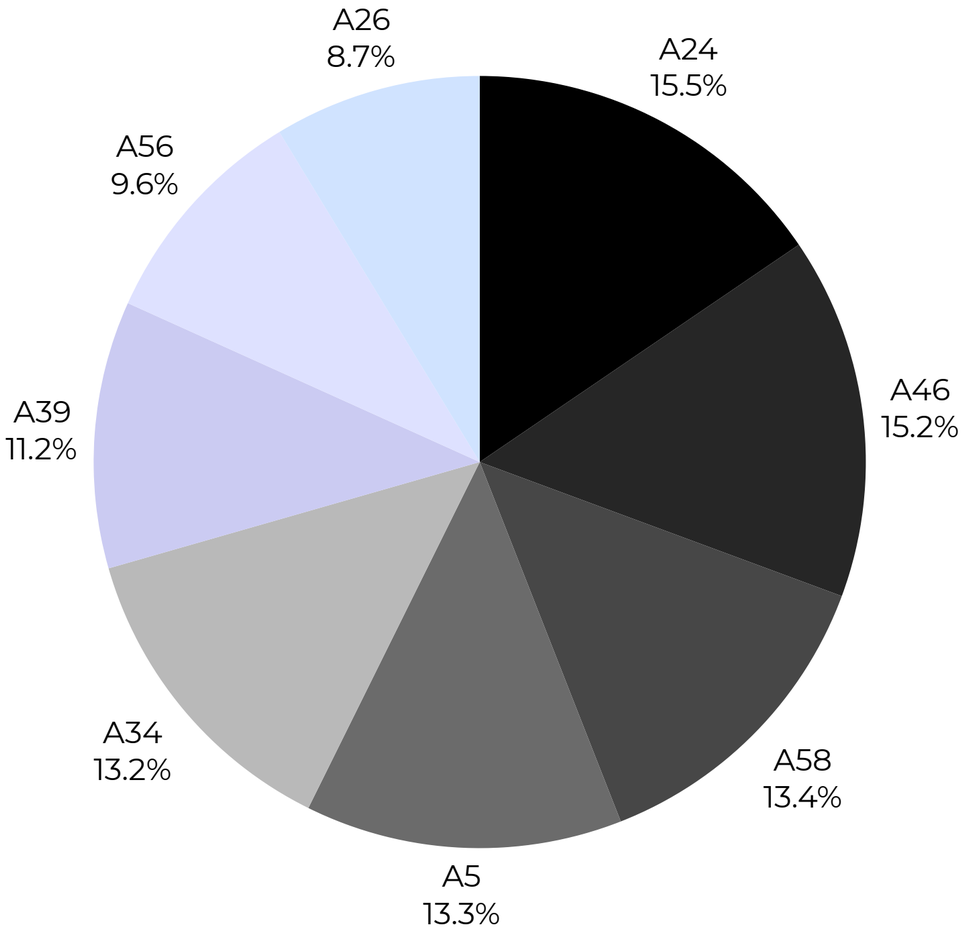
```
ax = subplots(figsize=(5,5))[1]
y_hat_bag = bag_polish.predict(X_test)
ax.scatter(y_hat_bag, y_test)
np.mean((y_test - y_hat_bag)**2)
```

```
0.020342923127680636
```



PRESENTATION

FEATURE IMPORTANCE



We can implement this method and help banking industry reduce risks

	importance
A24	0.047636
A46	0.046624
A58	0.041193
A5	0.040817
A34	0.040738
A39	0.034354
A56	0.029402
A26	0.026688

A24 gross profit (in 3 years) / total assets
A58 total costs /total sales
A5 [(cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] * 365
A34 operating expenses / total liabilities
A39 profit on sales / sales
A56 (sales - cost of products sold) / sales
A26 (net profit + depreciation) / total liabilities

DISCUSSION

01

The predictors selected by the 2 model are not overlapped, so we are not sure about which variables are the most important in predicting bankruptcy.

02

Although the model accuracy is high, the **Recall Rate** is low, suggesting that the model is failing to identify the majority of positive cases (bankruptcy).

03

Besides, there may exist correlations between variables.
All the 64 features are transformations of basic indexes like gross profit, total liability, book value, and so on. So by treating them as individual predictors, we might encounter severe multicollinearity. It might be better to do some calculations, maybe try other methods.

Q&A

PRESENTATION

THANK YOU