Predicting Bankruptcy with Machine Learning

UW FinTech – Nov 2022

Amrita Prithiani
Yu Takahashi
Jeremy Vargas

Bankruptcy

What is?

When a business is unable to repay its debts or obligations. It follows a legal process handled by the U.S. Federal Court and ruled by U.S. Bankruptcy Code.

How is it measured?

Financial ratios are used to measure a business financial fitness and can be used to provide some foresight on a business stability.

Ratios:

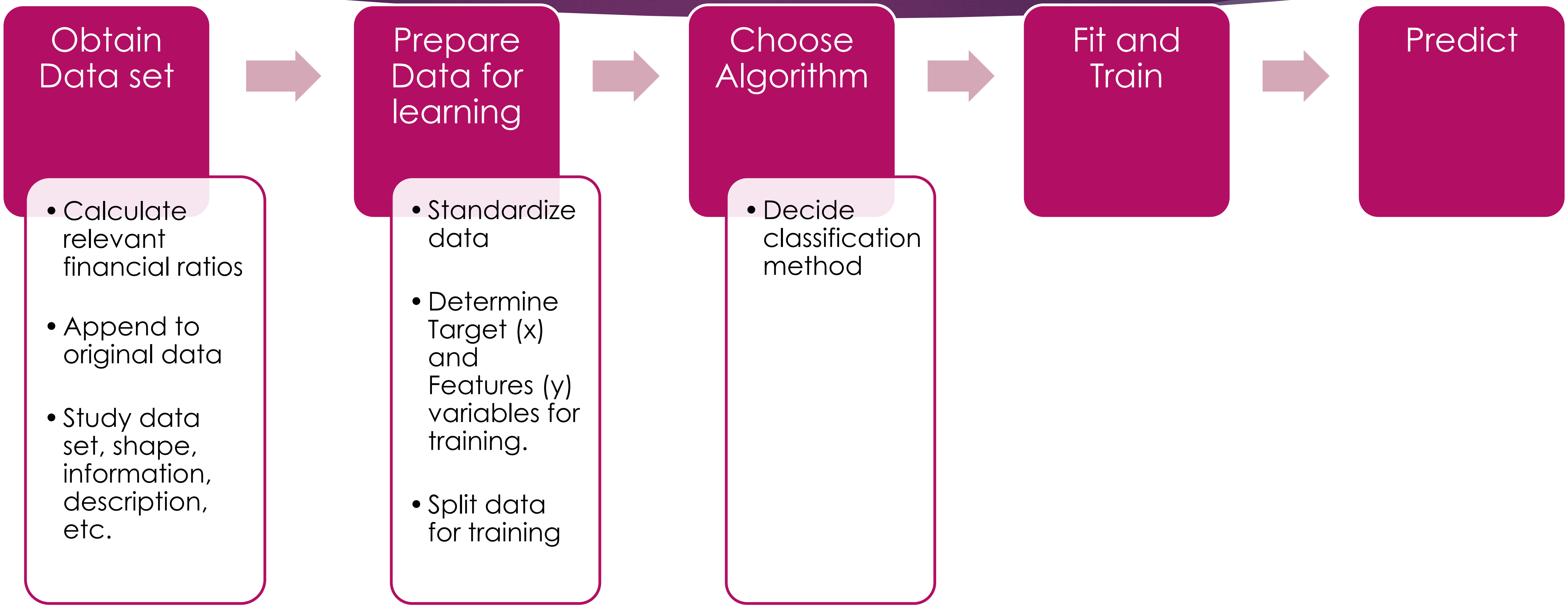
- **Current Ratio**
- Operating Cash Flow to Sales
- **Debt / Equity Ratio**
- Cash Flow to Debt Ratio

Project Scope

Our team presents a series of applications that aim to predict business bankruptcy by using data analysis techniques and machine learning algorithms.

These models and approaches are presented and differentiated as part of this effort.

Generalized approach



Obtain Data set

- Calculate relevant financial ratios
- Append to original data
- Study data set, shape, information, description, etc.

Prepare Data for learning

- Standardize data
- Determine Target (x) and Features (y) variables for training.
- Split data for training

Choose Algorithm

- Decide classification method

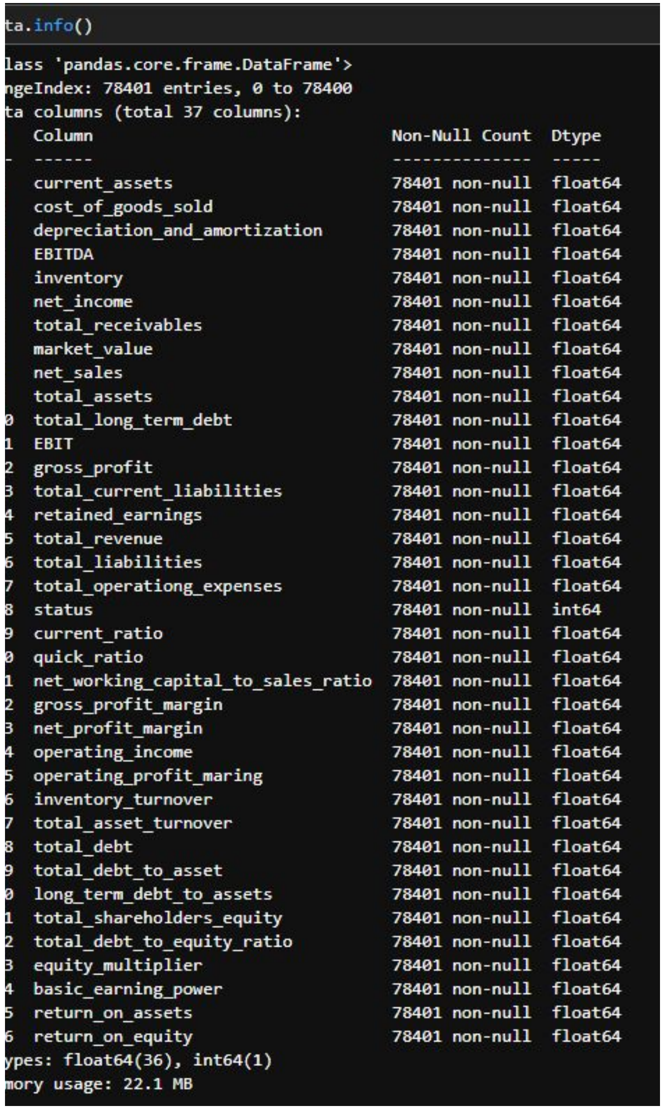
Fit and Train

Predict

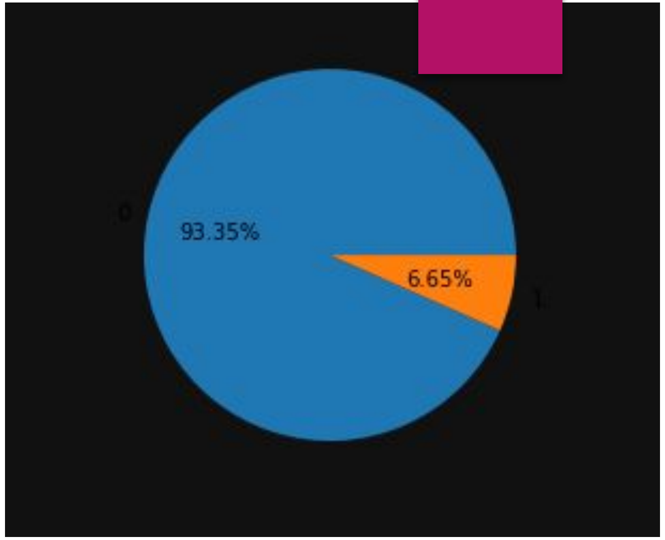
Original vs. Expanded with financial ratios.

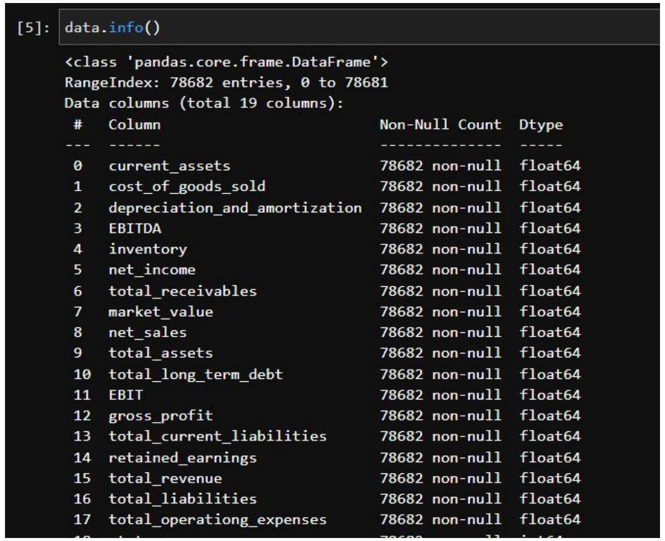
What we know:

- Labeled: Classified
- Bankruptcy “Status”
  - 0 = No
  - 1 = Yes
- Imbalanced:
  - 73,191 companies
  - 5210 bankrupt (6.65%)
- Non-linear
- Financial ratios give us more features learn from



```
ta.info()

lass 'pandas.core.frame.DataFrame'>
ngeIndex: 78401 entries, 0 to 78400
ta columns (total 37 columns):
   Column                             Non-Null Count  Dtype
-  ------                             --------------  -----
   current_assets                     78401 non-null  float64
   cost_of_goods_sold                 78401 non-null  float64
   depreciation_and_amortization      78401 non-null  float64
   EBITDA                             78401 non-null  float64
   inventory                          78401 non-null  float64
   net_income                         78401 non-null  float64
   total_receivables                  78401 non-null  float64
   market_value                       78401 non-null  float64
   net_sales                          78401 non-null  float64
   total_assets                       78401 non-null  float64
0  total_long_term_debt               78401 non-null  float64
1  EBIT                               78401 non-null  float64
2  gross_profit                       78401 non-null  float64
3  total_current_liabilities          78401 non-null  float64
4  retained_earnings                  78401 non-null  float64
5  total_revenue                      78401 non-null  float64
6  total_liabilities                  78401 non-null  float64
7  total_operationg_expenses          78401 non-null  float64
8  status                             78401 non-null  int64
9  current_ratio                      78401 non-null  float64
0  quick_ratio                        78401 non-null  float64
1  net_working_capital_to_sales_ratio 78401 non-null  float64
2  gross_profit_margin                78401 non-null  float64
3  net_profit_margin                  78401 non-null  float64
4  operating_income                   78401 non-null  float64
5  operating_profit_maring            78401 non-null  float64
6  inventory_turnover                 78401 non-null  float64
7  total_asset_turnover               78401 non-null  float64
8  total_debt                         78401 non-null  float64
9  total_debt_to_asset                78401 non-null  float64
0  long_term_debt_to_assets           78401 non-null  float64
1  total_shareholders_equity          78401 non-null  float64
2  total_debt_to_equity_ratio         78401 non-null  float64
3  equity_multiplier                  78401 non-null  float64
4  basic_earning_power                78401 non-null  float64
5  return_on_assets                   78401 non-null  float64
6  return_on_equity                   78401 non-null  float64
ypes: float64(36), int64(1)
mory usage: 22.1 MB
```

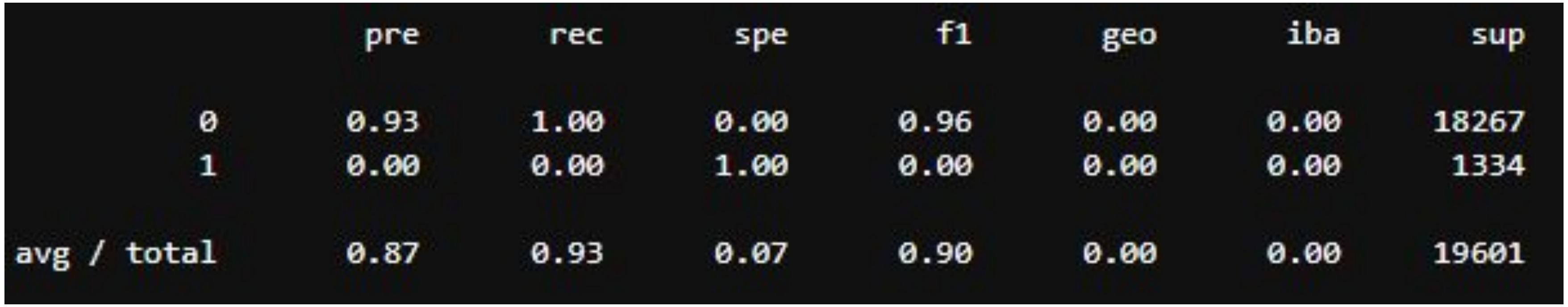




```
[5]: data.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 78682 entries, 0 to 78681
     Data columns (total 19 columns):
      #   Column                          Non-Null Count  Dtype
     ---  ------                          --------------  -----
      0   current_assets                  78682 non-null  float64
      1   cost_of_goods_sold              78682 non-null  float64
      2   depreciation_and_amortization   78682 non-null  float64
      3   EBITDA                          78682 non-null  float64
      4   inventory                       78682 non-null  float64
      5   net_income                      78682 non-null  float64
      6   total_receivables               78682 non-null  float64
      7   market_value                    78682 non-null  float64
      8   net_sales                       78682 non-null  float64
      9   total_assets                    78682 non-null  float64
      10  total_long_term_debt            78682 non-null  float64
      11  EBIT                            78682 non-null  float64
      12  gross_profit                    78682 non-null  float64
      13  total_current_liabilities       78682 non-null  float64
      14  retained_earnings               78682 non-null  float64
      15  total_revenue                   78682 non-null  float64
      16  total_liabilities               78682 non-null  float64
      17  total_operationg_expenses       78682 non-null  float64
```

Approach # 1 ADA BOOST

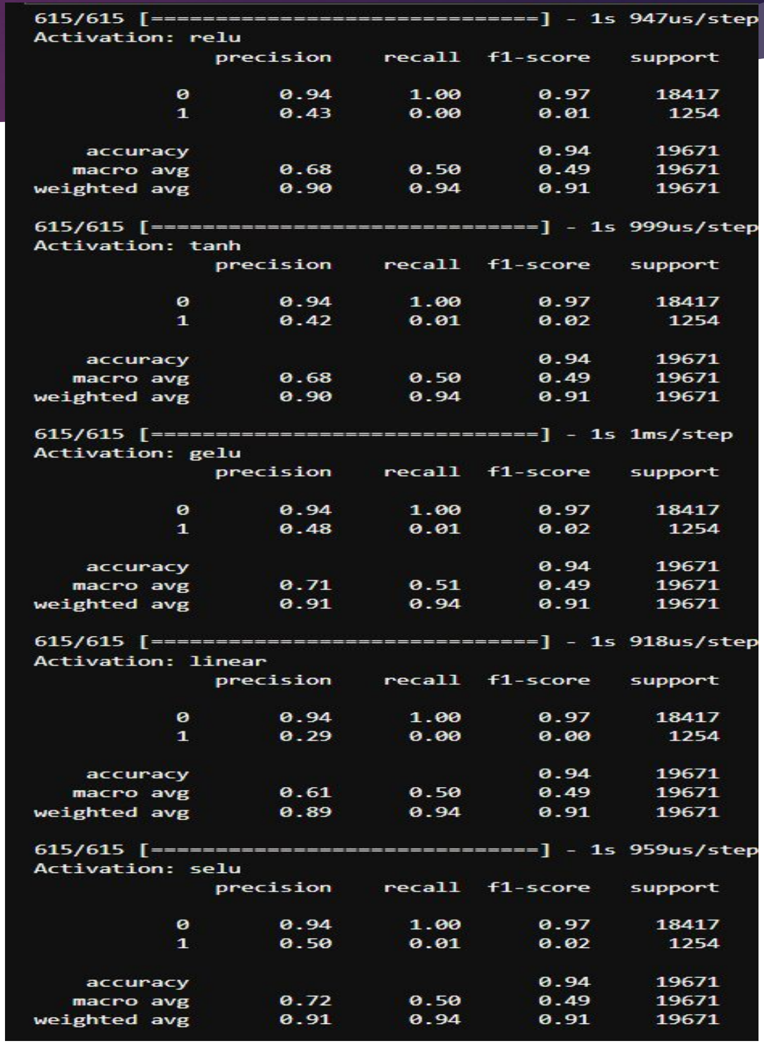
- Data set + Financial Ratios (18 vs. 38 features)
- ADABOOST requires the selection of a "stump tree" (estimator)
- Used ADABOOST Regression to obtain the optimum number of trees (estimator)
- Trained model with 1 and 20 trees (0.93 accuracy score - R score)
- GridSearchCV function to obtain ideal estimator
- Classification report: Imbalanced. ⇒ **Results: NOT IDEAL**



```
                   pre       rec       spe        f1       geo       iba       sup

          0       0.93      1.00      0.00      0.96      0.00      0.00     18267
          1       0.00      0.00      1.00      0.00      0.00      0.00      1334

avg / total       0.87      0.93      0.07      0.90      0.00      0.00     19601
```

Approach # 2 Neural Network

- Data set
  - Original Dataset
- Model
  - SVC kernel = rbf
  - MLP Classifier
  - IMBLearn models
  - Tensorflow Keras
    - Nodes: 8 nodes to 120 nodes **[96]**
    - activation_list = ['relu', 'tanh', 'gelu', 'linear', **'selu'**]
    - output_list = ['sigmoid', 'softmax', **'softplus'**, 'softsign', 'swish']
    - optimizer_list = ['adadelta', 'adagrad', 'adam', 'adamax', **'nadam'**, 'ftrl', 'rmsprop', 'sgd']

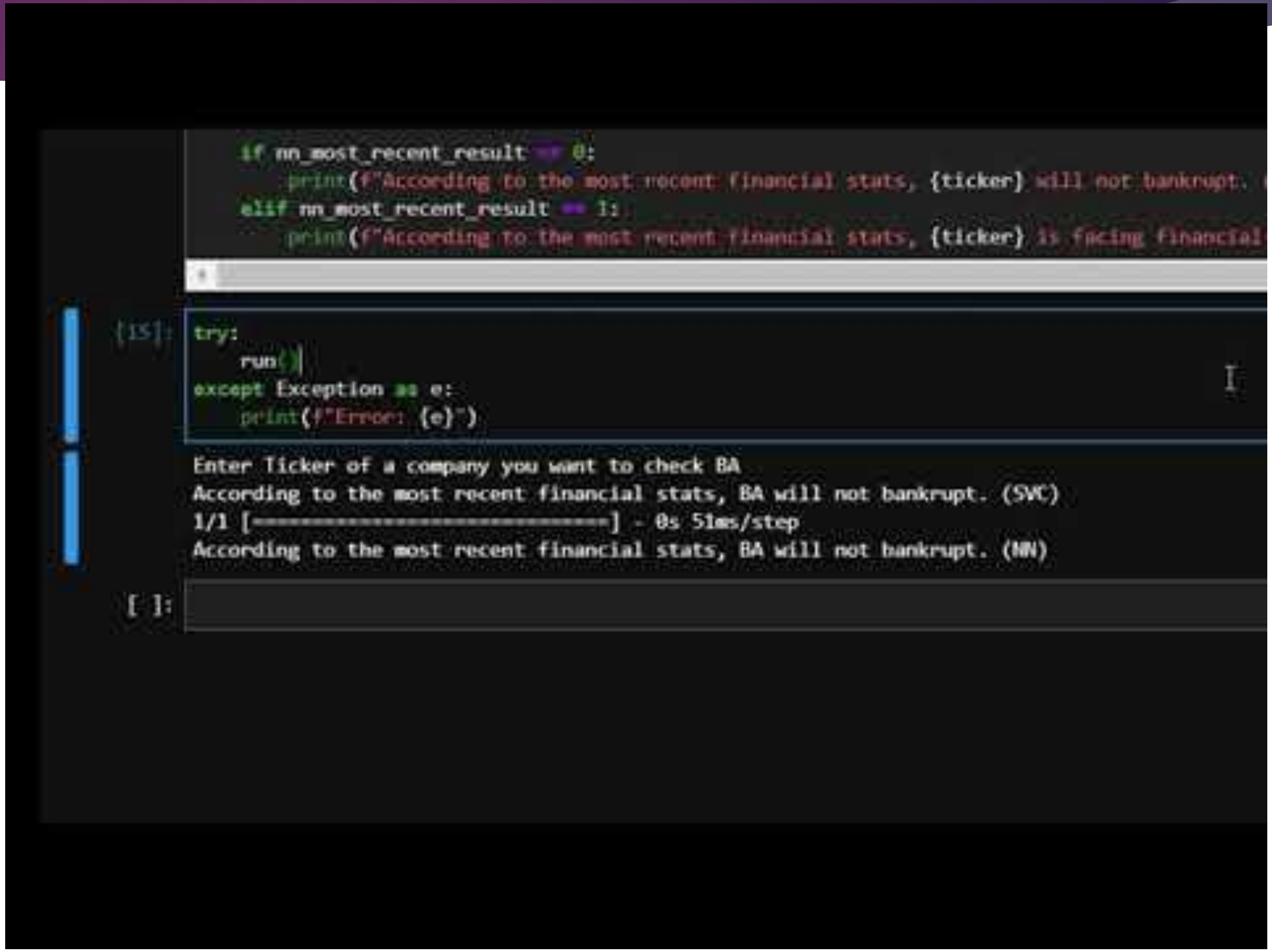


```
615/615 [==============================] - 1s 947us/step
Activation: relu
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     18417
           1       0.43      0.00      0.01      1254

    accuracy                           0.94     19671
   macro avg       0.68      0.50      0.49     19671
weighted avg       0.90      0.94      0.91     19671

615/615 [==============================] - 1s 999us/step
Activation: tanh
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     18417
           1       0.42      0.01      0.02      1254

    accuracy                           0.94     19671
   macro avg       0.68      0.50      0.49     19671
weighted avg       0.90      0.94      0.91     19671

615/615 [==============================] - 1s 1ms/step
Activation: gelu
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     18417
           1       0.48      0.01      0.02      1254

    accuracy                           0.94     19671
   macro avg       0.71      0.51      0.49     19671
weighted avg       0.91      0.94      0.91     19671

615/615 [==============================] - 1s 918us/step
Activation: linear
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     18417
           1       0.29      0.00      0.00      1254

    accuracy                           0.94     19671
   macro avg       0.61      0.50      0.49     19671
weighted avg       0.89      0.94      0.91     19671

615/615 [==============================] - 1s 959us/step
Activation: selu
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     18417
           1       0.50      0.01      0.02      1254

    accuracy                           0.94     19671
   macro avg       0.72      0.50      0.49     19671
weighted avg       0.91      0.94      0.91     19671
```

User Interface (see video demo)

Save and load fitted models

- **Joblib**
  - Fitted scaler model
  - Fitted ML model
- **Tensorflow**
  - Fitted Neural Network



```
    if nn_most_recent_result == 0:
        print(f"According to the most recent financial stats, {ticker} will not bankrupt.
    elif nn_most_recent_result == 1:
        print(f"According to the most recent financial stats, {ticker} is facing financial
```

```
[15]: try:
          run()
      except Exception as e:
          print(f"Error: {e}")
```

```
Enter Ticker of a company you want to check BA
According to the most recent financial stats, BA will not bankrupt. (SVC)
1/1 [==============================] - 0s 51ms/step
According to the most recent financial stats, BA will not bankrupt. (NN)
```

```
[ ]:
```

Approach # 3 - Iterating Models

- **Bankruptcy Dataset** -
  - Financial statement data of American companies in the stock market (1999 -2018) - Github
  - 8262 companies
  - Did not use Financial Ratios

- **Preparing Process**
  - Cleaned and prepared data.
  - Standard Scaler to scale variances
  - Created X and Y split for training.
  - Applied Smote Oversampling for Imbalanced data
  - Used KFOLD variation for splitting and testing data and training.

- **Training Process:**
  - K-Folds cross-validator: Provides train/test indices to split data in train/test sets.
  - Split dataset into k consecutive folds (without shuffling by default).
  - Each fold is then used once as a validation while the k - 1 remaining folds form the training set. Definition by scikit-learn.org

Approach # 3 - Iterating Models (cont.)

Focused on models optimized for Imbalanced Data

➢ **Random Forest Classifier**

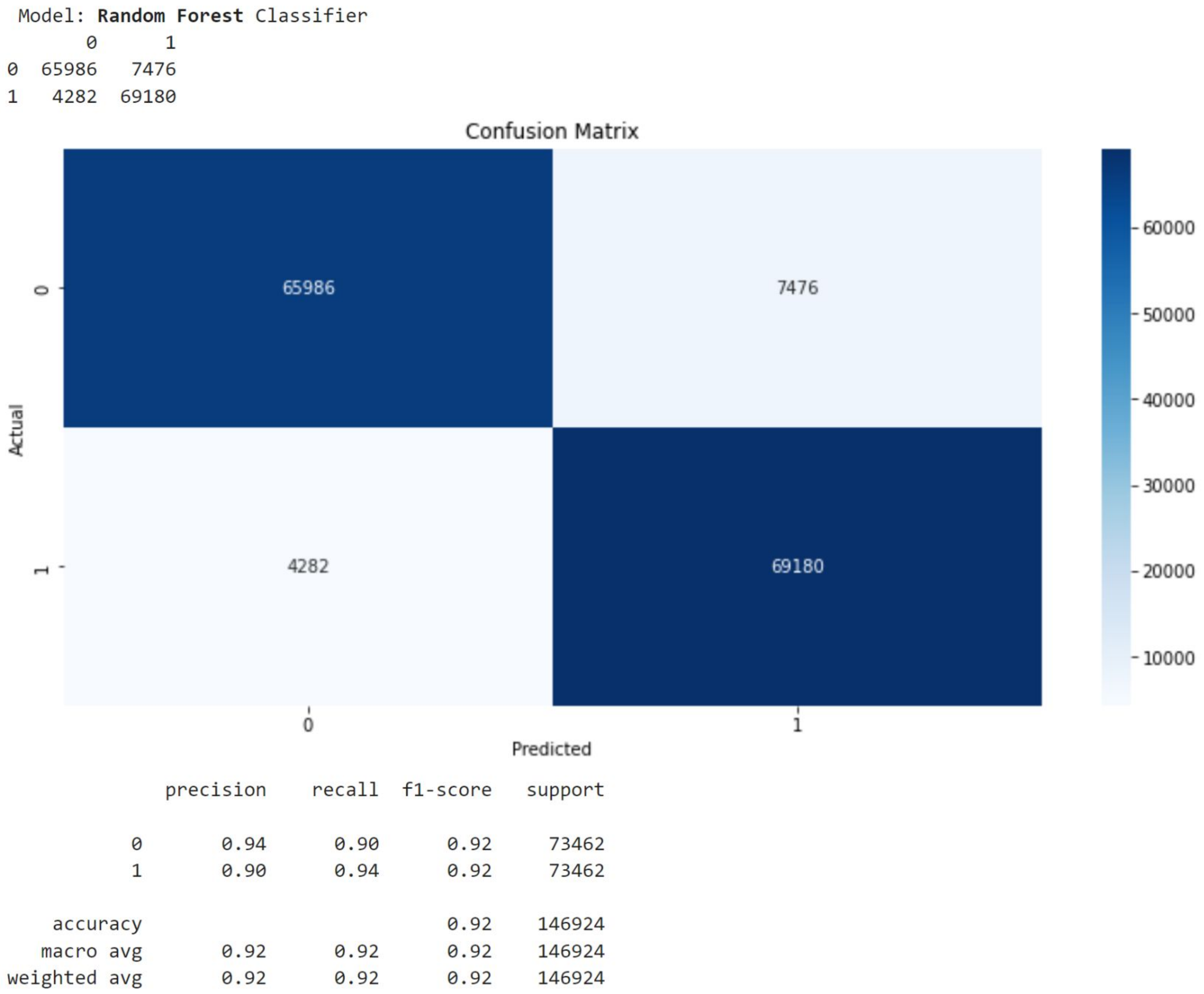
Set of decision trees (DT) from a randomly selected subset of the training set and then It collects the votes from different decision trees to decide the final prediction.

➢ **XgBoost classifier**

In XGBoost, weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. Individual classifiers/predictors adjusted by the weights give a strong and more precise model.

➢ **Decision tree classifier**

Follows a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



```
 Model: Random Forest Classifier
       0      1
0  65986   7476
1   4282  69180
```

```
              precision    recall  f1-score   support

           0       0.94      0.90      0.92     73462
           1       0.90      0.94      0.92     73462

    accuracy                           0.92    146924
   macro avg       0.92      0.92      0.92    146924
weighted avg       0.92      0.92      0.92    146924
```

Next steps

- Test the model on more recent data
  - Ideally Post pandemic.
- Include visual comparisons for all models applied.
- Continue exploring other techniques to classify unbalanced data

Resources

- Machine Learning for Bankruptcy Prediction in the American Stock Market: Dataset and Benchmarks
- Bankruptcy prediction dataset for american companies in the stock market
- Bankruptcy Prediction
- How to Plot a Confusion Matrix from a K-Fold Cross-Validation

Q&A