# Predicting Bankruptcy Using Machine Learning

**Ong Jing Xuan U2122081B | Somesh Sahu U2122235C | Nathaniel Yew U2223621F**
**Lab Group: C126**

# The Problem: Bankruptcy can result in significant financial losses for entities

**Dataset**

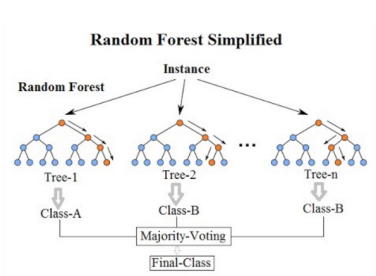
**Bankruptcy data from the Taiwan Economic Journal (1999–2009)**

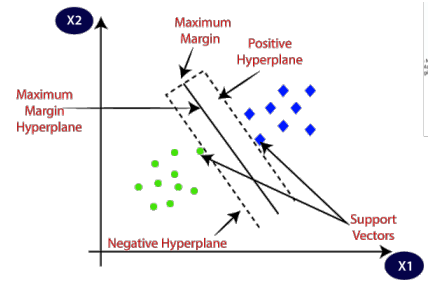
kaggle

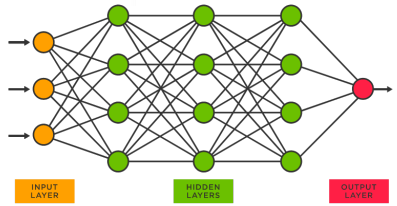
TEJ Ratings ◦ Risk ◦ Index ◦ Databank

**What is the situation?**

- **745,566 Businesses** have filed for Bankruptcy in the US since 2000[1]
- The **abundance of financial data** can pose a challenge in determining which factors to prioritize when forecasting the likelihood of a company's bankruptcy




**Approach**

- Which model is the **best** in accurately predicting bankruptcy?
- Should they focus on a **specific set of financial metrics**?

- Explore 3 Machine Learning Models
- Compare effect of number of variables

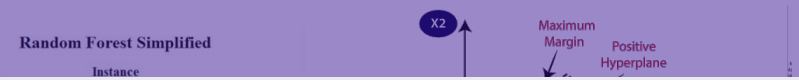
# The Problem: Bankruptcy can result in significant financial losses for entities

**Dataset**

**Bankruptcy data from the Taiwan Economic Journal (1999–2009)**

kaggle

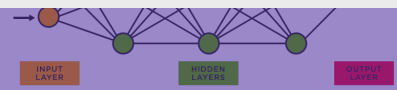
TEJ Ratings ◦ Risk ◦ Index ◦ Databank







**What is the optimum Machine Learning model to predict the likelihood of a company going bankrupt and are there specific variables that can better determine bankruptcy?**




**Approach**

- Which model is the **best** in accurately predicting bankruptcy?
- Should they focus on a **specific set of financial metrics**?

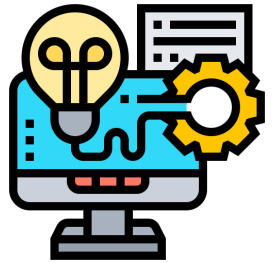
- Explore 3 Machine Learning Models
- Compare effect of number of variables

# **Data Set**: Bankruptcy data from the Taiwan Economic Journal (1999–2009) Used to Answer Our Questions

**Problem**

**Is there a best ML model we can use to predict bankruptcy based on financial data?**

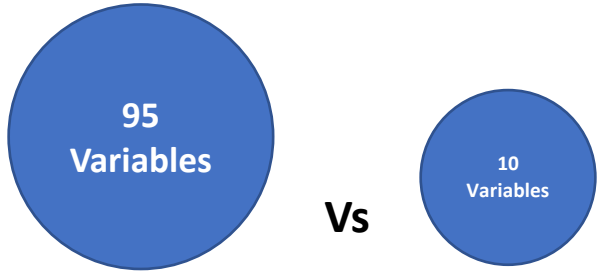
1. Support Vector Machine
2. Decision Tree
3. Neural Network



**Hypothesis: We can identify the best ML model using financial data from the dataset**

**Extending the problem**

**We will then see if it is better to narrow the number of metrics used to predict bankruptcy**



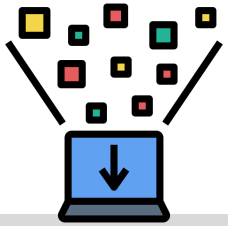
# Tech Stack: These are the main libraries and tools we deployed to conduct our analysis

Keras

seaborn

pandas

scikit learn



Data Collection

Data Cleaning

Data Analysis

# Data Collection: Used Kaggle to import the preliminary data to form our main Data Set


## Data Set

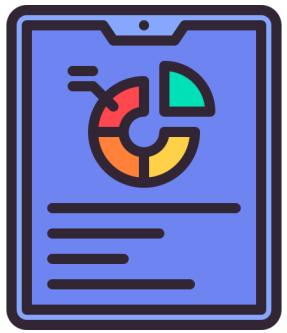
The dataset is **highly relevant**, **well-structured**, and **challenging**, making it a valuable resource for machine learning, data cleaning and analysis

Collected up to **95 Columns** of statistics to paint a comprehensive picture of the company

# Cleaning and Curation: Upon initial exploration, there was a need to further clean and select for relevant features to build our Data Set

Checking for missing or null values in a dataset is crucial as they can impact the **accuracy and reliability** of the analysis and modeling results, leading to biased or incomplete results if not handled appropriately

**Hence, we engaged in Data imputation**



**Check and remove any missing or null values in dataframe**

```
print("Null or missing values:",df.isnull().values.any())
```

**Null or missing values: False**

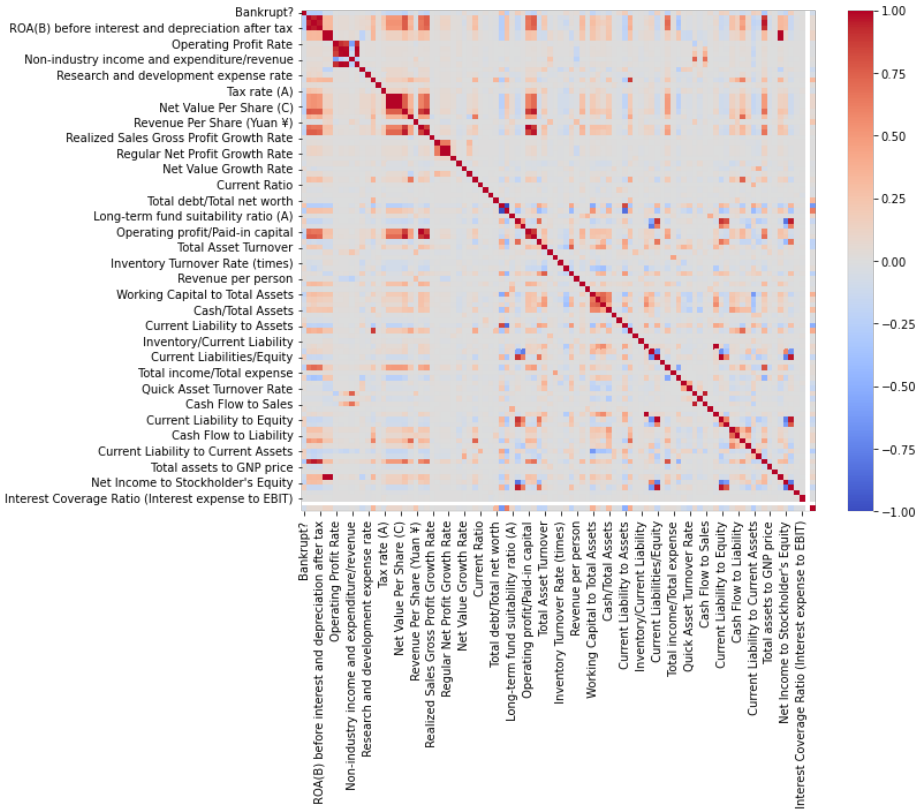
# Cleaning and Curation: Upon initial exploration, there was a need to further clean and select for relevant features to build our Data Set

**Too many variables can negatively impact a model if they are irrelevant or partially relevant**

## Hence, we engaged in Dimension Reduction

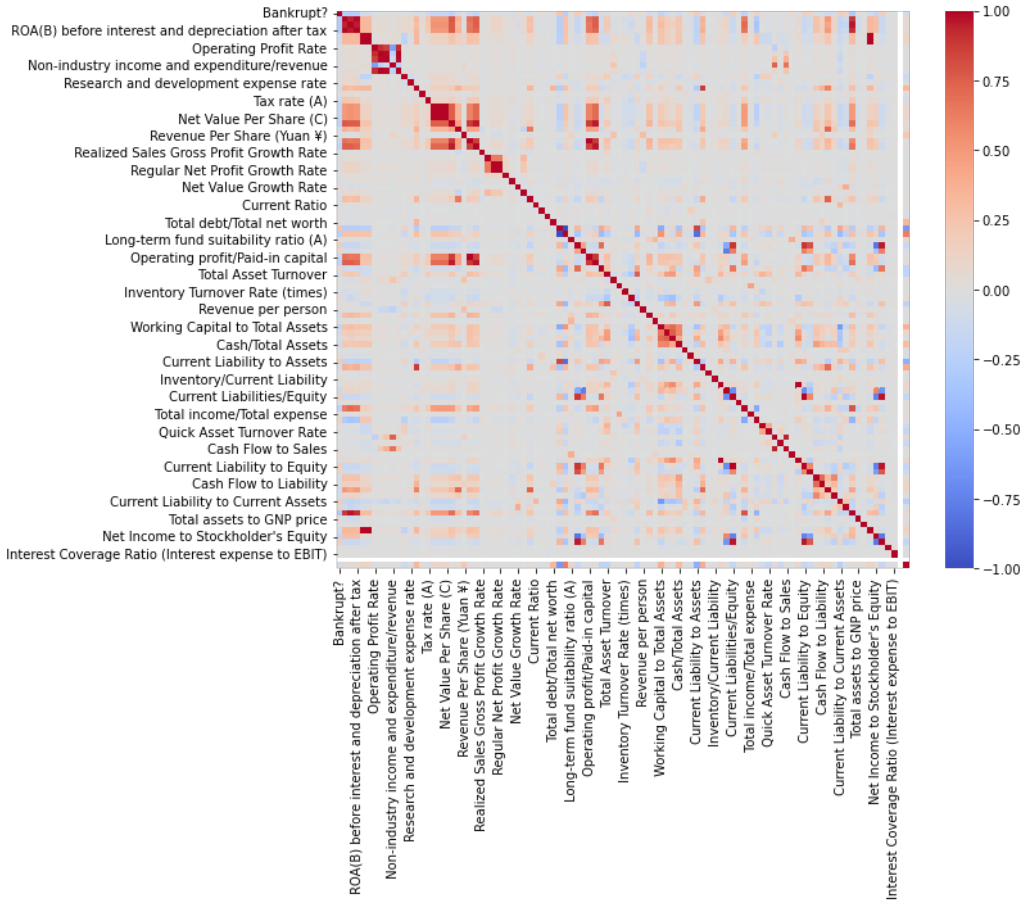
To eliminate irrelevant/partially relevant data, we considered the **Correlation** between various data points against **Bankrupt**

## Heat Maps

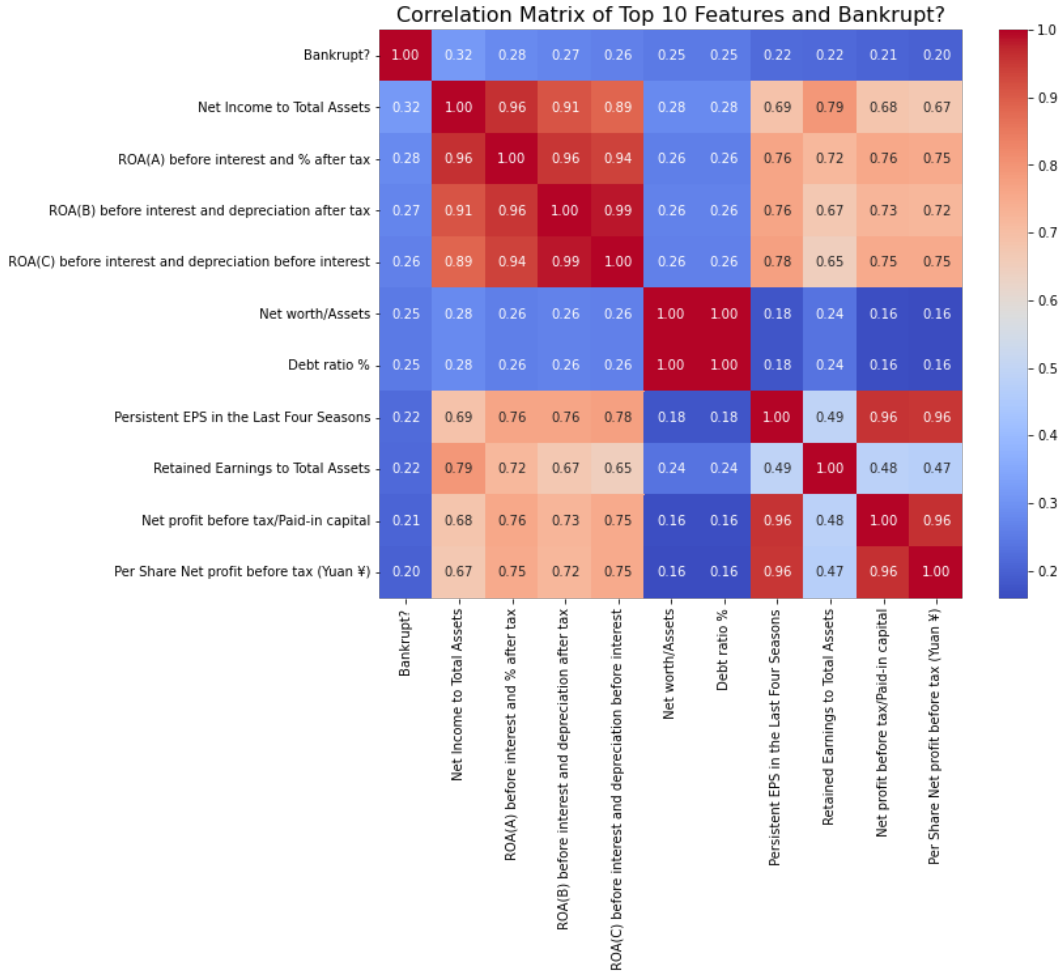


# Cleaning and Curation: Narrowed down the Top 10 variables which will form the new data frame

## Before

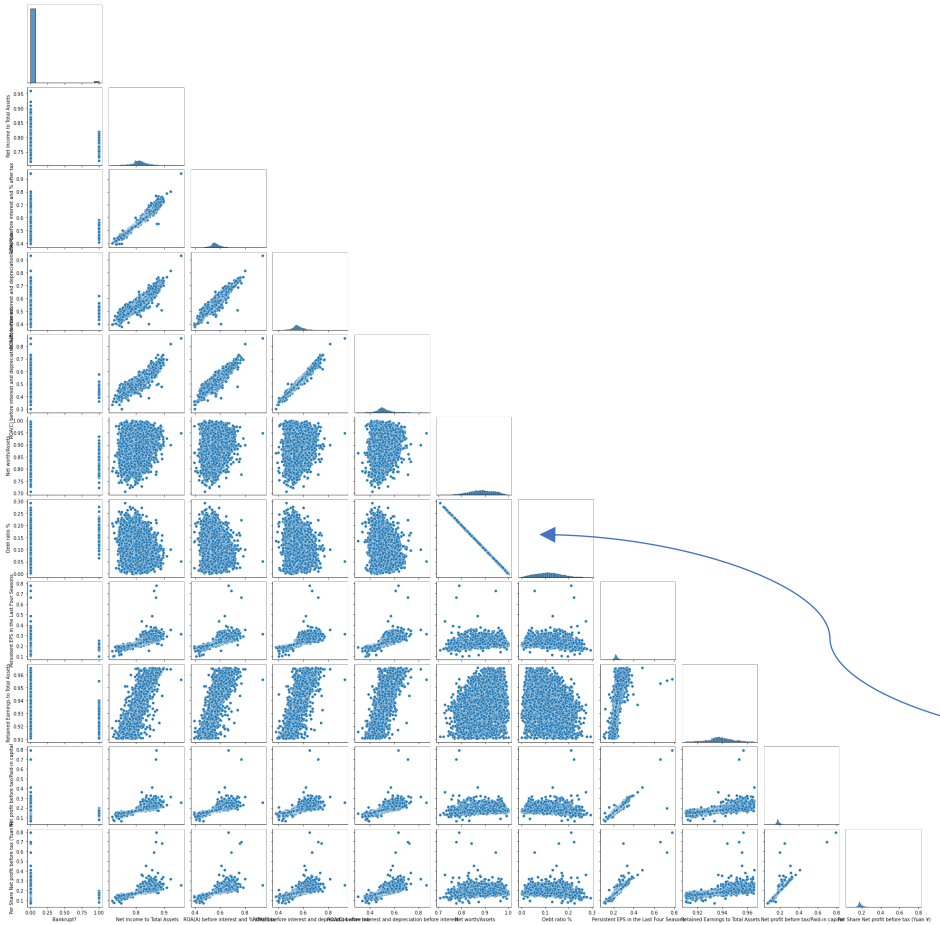



## After

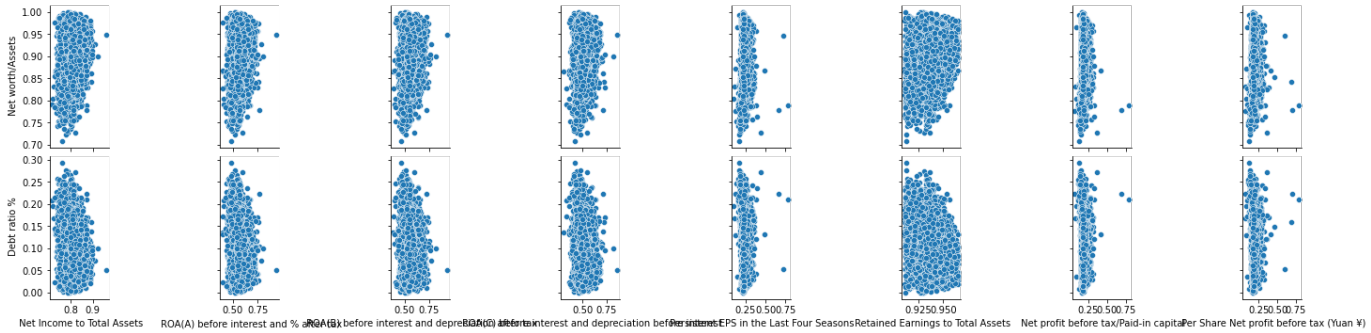


# Data analysis: Explore non-linear models, Scatter plots

## Top 10 variables



## "Net worth/Assets" and "debt ratio%"



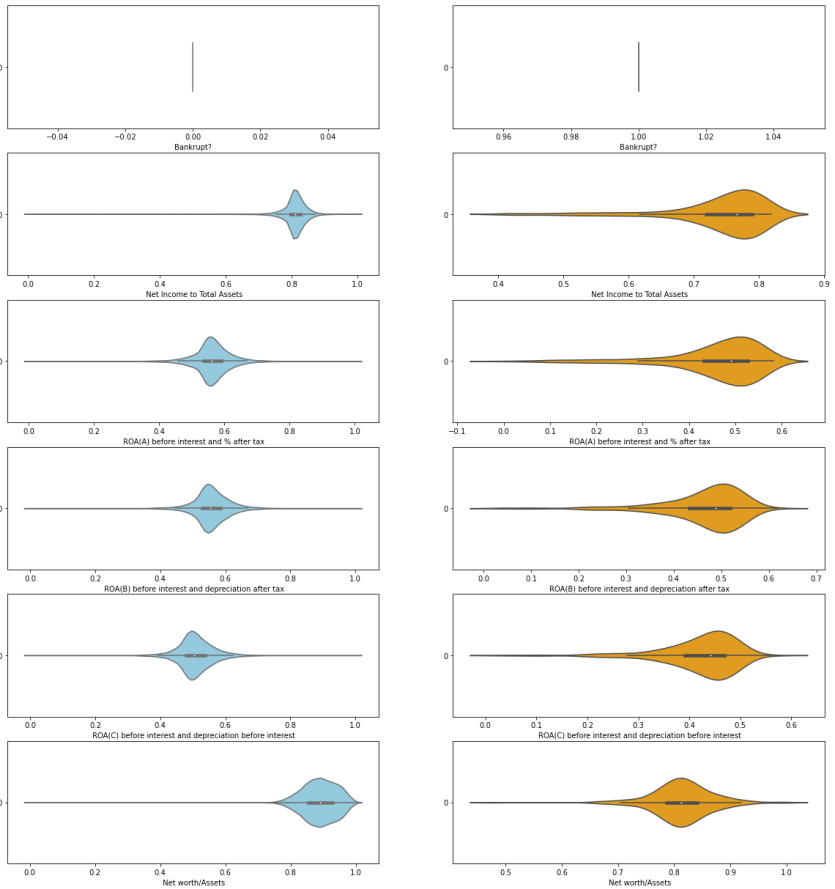

## Insight

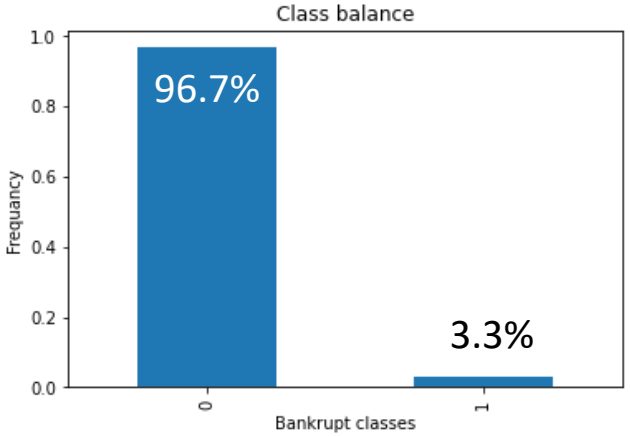
Strong **Negative correlations** between these 2 variables

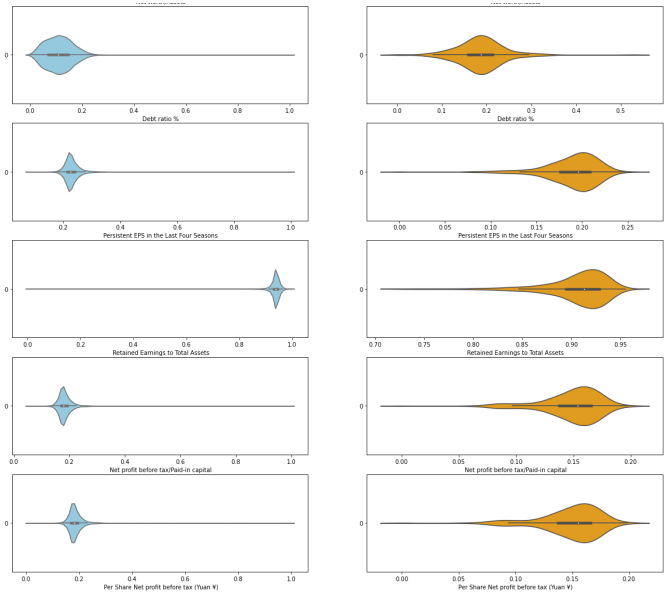
**Poor relationship** between these 2 variables against others

# Data analysis: Explore the relationship of non-bankrupt vs bankrupt

Non-Bankrupt vs. Bankrupt




## Insights

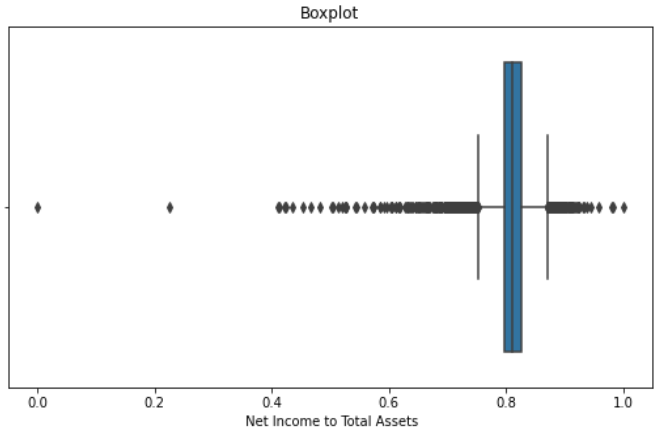
The violin plot helped us to visualize the density and shape of the distribution of variables within each group

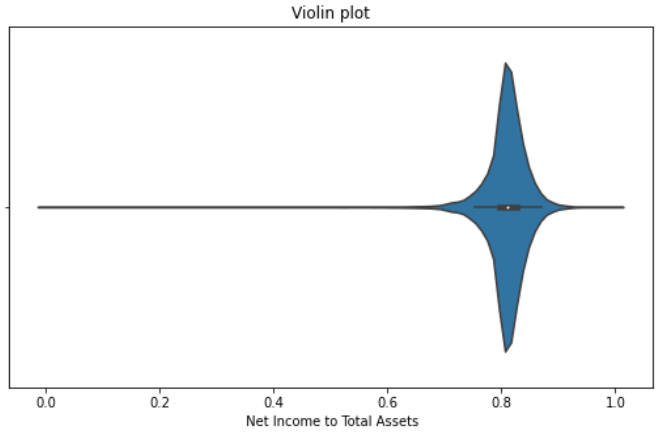
Both sets of data do not differ significantly from each other

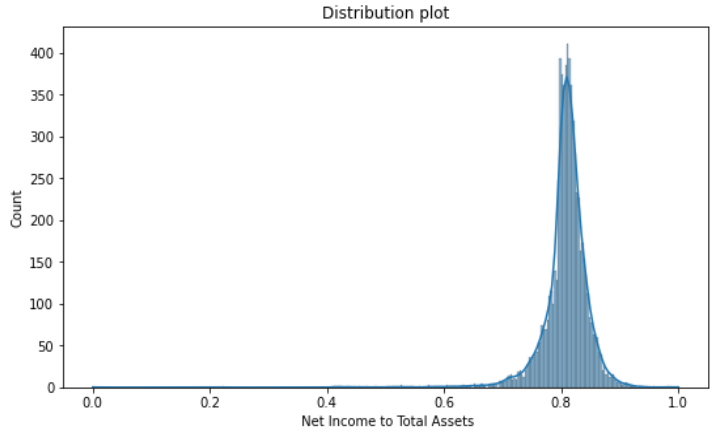
The dataset is highly imbalanced

# Data analysis: Explore best indicator

## ' Net Income to Total Assets '




**Outliers** congested outside 0.25 & 0.75 percentile, with some points further away

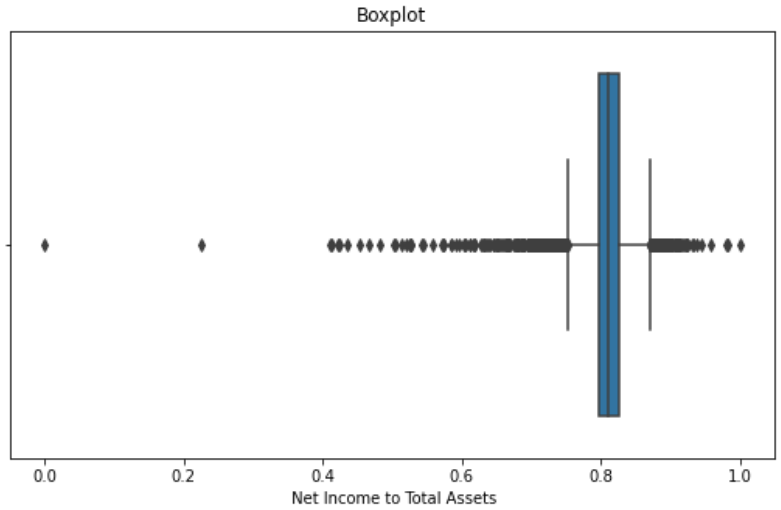
Highly **concentrated**

**Hence, we will engage in the Removal of outliers**

# Further exploration: Exploration of outliers

## How were the outliers obtained?

Outliers were identified using our best indicator **'Net Income to Total Assets'** as our datum point

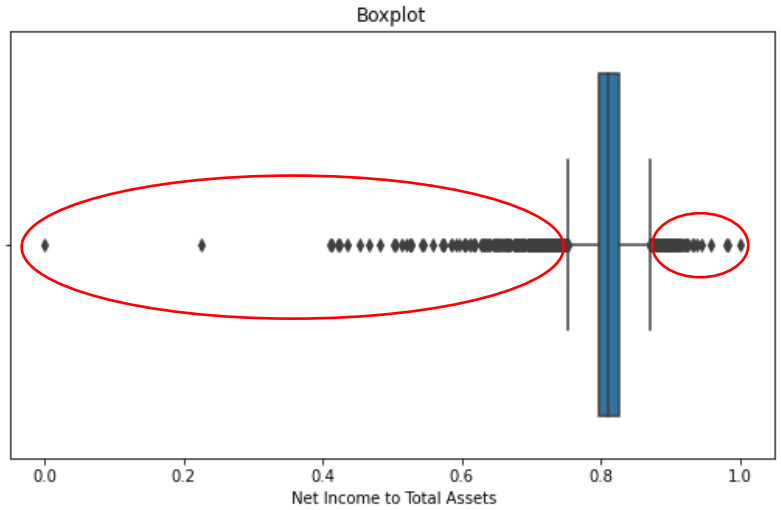



```
new_df_for_outliers = new_df_with_bankrupt.copy()
# Calculate the quartiles and interquartile range for the 'Net Income to Total Assets' column
Q1 = new_df_with_bankrupt[' Net Income to Total Assets'].quantile(0.25)
Q3 = new_df_with_bankrupt[' Net Income to Total Assets'].quantile(0.75)
IQR = Q3 - Q1

# Identify the indices of the outliers in the 'Net Income to Total Assets' column
outliers = np.where((new_df_with_bankrupt[' Net Income to Total Assets'] < (Q1 - 1.5 * IQR)) |
(new_df_with_bankrupt[' Net Income to Total Assets'] > (Q3 + 1.5 * IQR)))

# Create a new dataframe called new_df_no_outliers that does not contain the outliers
new_df_no_outliers =
new_df_with_bankrupt.drop(new_df_with_bankrupt.index[outliers]).reset_index(drop=True)

# Print information about the new_df_no_outliers DataFrame
new_df_no_outliers.info()
```

# Further exploration: Exploration of outliers

## How were the outliers obtained?

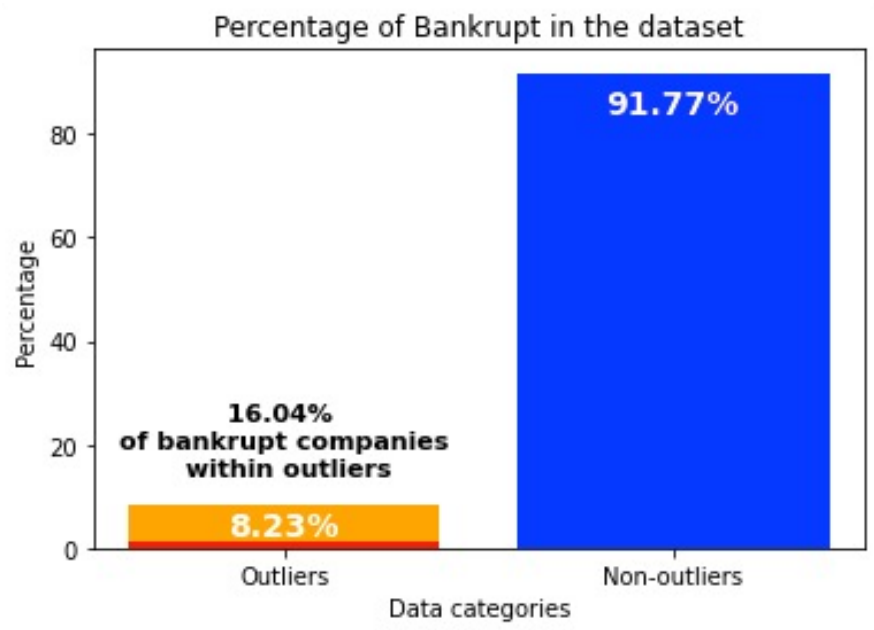
Outliers were identified using our best indicator **'Net Income to Total Assets'** as our datum point




```
new_df_for_outliers = new_df_with_bankrupt.copy()
# Calculate the quartiles and interquartile range for the 'Net Income to Total Assets' column
Q1 = new_df_with_bankrupt[' Net Income to Total Assets'].quantile(0.25)
Q3 = new_df_with_bankrupt[' Net Income to Total Assets'].quantile(0.75)
IQR = Q3 - Q1

# Identify the indices of the outliers in the 'Net Income to Total Assets' column
outliers = np.where((new_df_with_bankrupt[' Net Income to Total Assets'] < (Q1 - 1.5 * IQR)) |
(new_df_with_bankrupt[' Net Income to Total Assets'] > (Q3 + 1.5 * IQR)))

# Create a new dataframe called new_df_no_outliers that does not contain the outliers
new_df_no_outliers =
new_df_with_bankrupt.drop(new_df_with_bankrupt.index[outliers]).reset_index(drop=True)

# Print information about the new_df_no_outliers DataFrame
new_df_no_outliers.info()
```

# Further exploration: Exploration of outliers

## Analysis of outliers



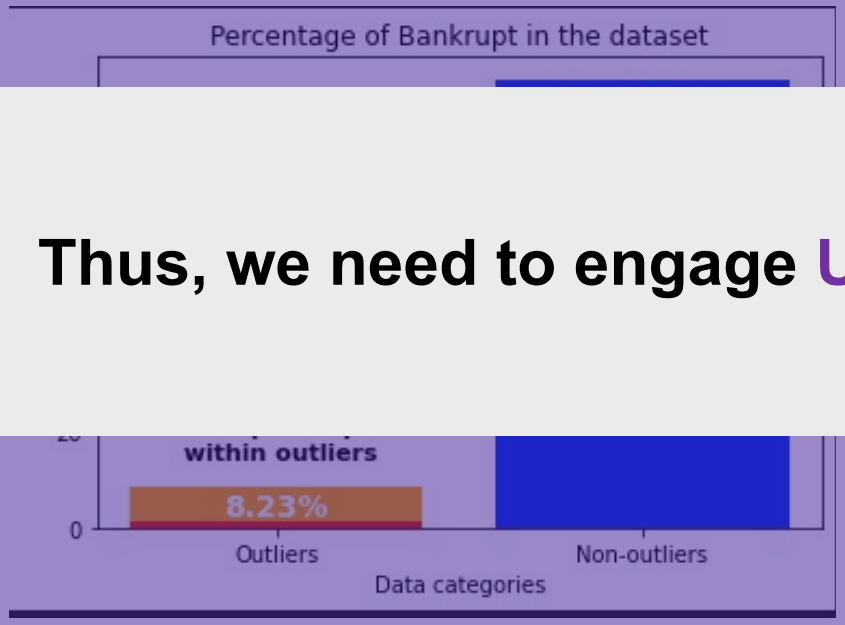

## Insight

Removing the outliers from the dataset may result in a **loss of information**, a **biased sample**, thus resulting in inaccurate conclusions

Potentially **overlook important factors** that contributes to the bankruptcy of those companies

# Further exploration: Exploration of outliers

## Analysis of outliers



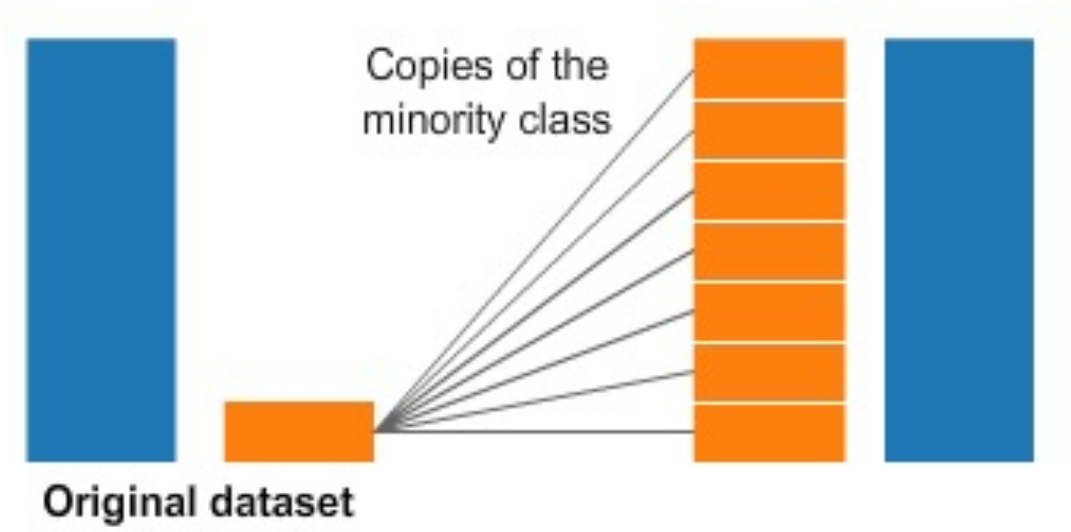

## Insight

Removing the outliers from the dataset potentially overlook important factors that contributes to the bankruptcy of those companies.

**Thus, we need to engage Up-Sampling to help balance the data**

# Further exploration: Up-Sampling of Data

## Random Up-Sampling




## Resample() method

```python
# Load the data from CSV file
df = pd.read_csv("bankruptcy.csv")
# df.head()

# Separate majority and minority classes
df_majority = df[df["Bankrupt?"] == 0]
df_minority = df[df["Bankrupt?"] == 1]

# Upsample minority class
df_minority_upsampled = resample(df_minority,
                                 replace=True,     # sample with replacement
                                 n_samples=len(df_majority),    # to match majority class
                                 random_state=42)  # reproducible results

# Combine majority class with upsampled minority class
df_upsampled = pd.concat([df_majority, df_minority_upsampled])

# Display new class counts
df_upsampled["Bankrupt?"].value_counts()
```

**new data frame with equal amount of bankrupt and non-bankrupt entries**

# Further exploration: Up-Sampling of Data

**Top 10 Variables from original dataset**

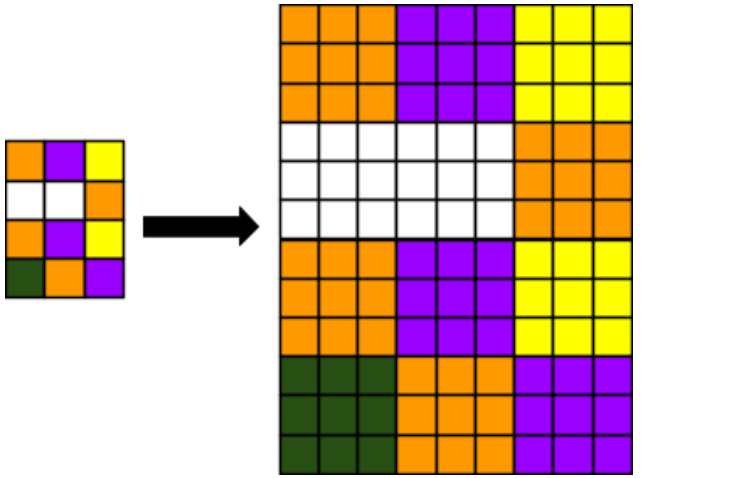
```
[' Net Income to Total Assets',
 ' ROA(A) before interest and % after tax',
 ' ROA(B) before interest and depreciation after tax',
 ' ROA(C) before interest and depreciation before interest',
 ' Net worth/Assets',
 ' Debt ratio %',
 ' Persistent EPS in the Last Four Seasons',
 ' Retained Earnings to Total Assets',
 ' Net profit before tax/Paid-in capital',
 ' Per Share Net profit before tax (Yuan ¥)']
```

**Top 10 Variables from up-Sampled dataset**

```
[' Debt ratio %',
 ' Net worth/Assets',
 ' Persistent EPS in the Last Four Seasons',
 ' ROA(C) before interest and depreciation before interest',
 ' Net profit before tax/Paid-in capital',
 ' Per Share Net profit before tax (Yuan ¥)',
 ' ROA(B) before interest and depreciation after tax',
 ' ROA(A) before interest and % after tax',
 ' Net Value Per Share (B)',
 ' Net Income to Total Assets']
```

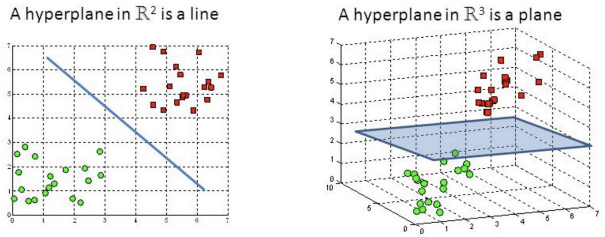
## Evaluation

**Good sign** that both datasets are similar, it suggests that no major changes to the underlying relationships between features and target variable



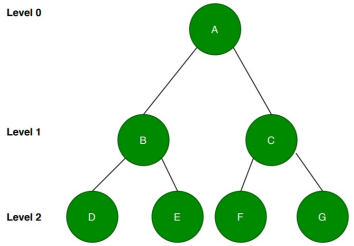
# Machine Learning: 3 Machine Learning Models Were Employed to Predict Bankruptcy

## 1 Support Vector Machine



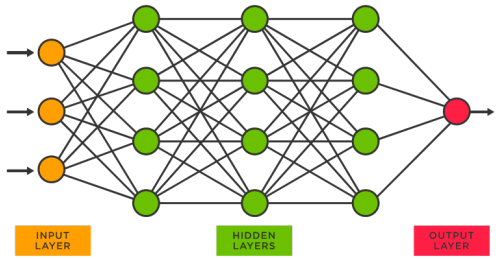

- Finds a **hyperplane in N-dimensional space** (N = Number of Features) that distinctly classifies the data points
- The hyperplane is chosen to **maximize the margin between the closest data points from each class**, also known as the support vectors

## 2 Decision Tree




- **Recursively partitions** the input data into subsets based on the values of one of the input features, until the subsets become as **homogeneous** as possible in terms of the target variable

## 3 Neural Network




- Consists of a large number of **interconnected processing nodes** that work together to process input data and make predictions or classifications
- Layers are **stacked together** and each layer is responsible for performing a specific transformation on its inputs

# Machine Learning: Support Vector Machine Implementation

## 1 Full Dataset

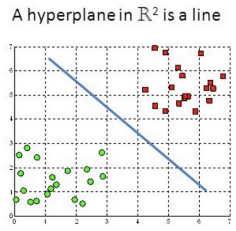
```
X = df.drop("Bankrupt?", axis=1)
y = df["Bankrupt?"]
X = pd.get_dummies(X, drop_first=True) # encode categorical variables
X = StandardScaler().fit_transform(X) # scale numerical variables

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train an SVM model with RBF kernel
svm_model = SVC(kernel='rbf', probability=True)
svm_model.fit(X_train, y_train)
```

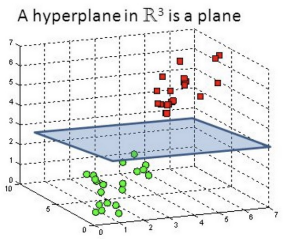
## 2 Top 10 Dataset

```
X_top10_upsample = df_top10_upsampled
y_top10_upsample = df["Bankrupt?"]

X2 = df_top10_upsampled
y2 = df["Bankrupt?"]
X2 = pd.get_dummies(X2, drop_first=True) # encode categorical variables
X2 = StandardScaler().fit_transform(X2) # scale numerical variables

# Split the dataset into training and testing sets
X_train2, X_test2, y_train2, y_test2 = train_test_split(X2, y2, test_size=0.2, random_state=42)

# Train an SVM model with RBF kernel
svm_model_top10 = SVC(kernel='rbf', probability=True)
svm_model_top10.fit(X_train2, y_train2)
```

## 3 Explanation of Model Training

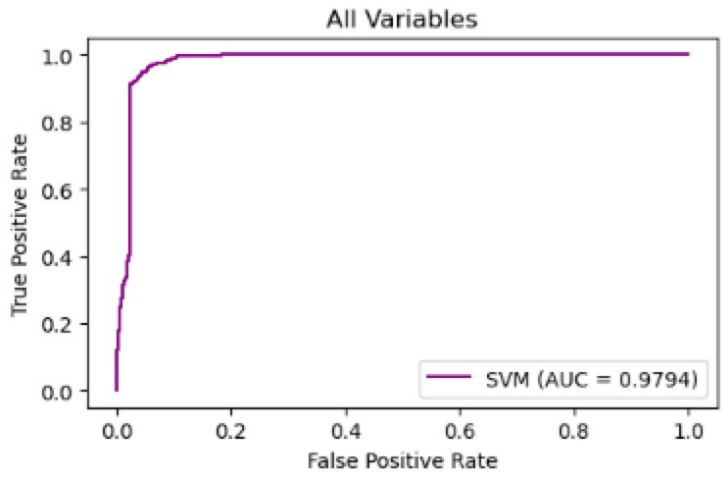
- Creates **dummy variables for categorical features** and prevents **Multicollinearity**
- Scales **numerical features** to have **zero mean and unit variance**, ensuring all features have a similar impact on the model



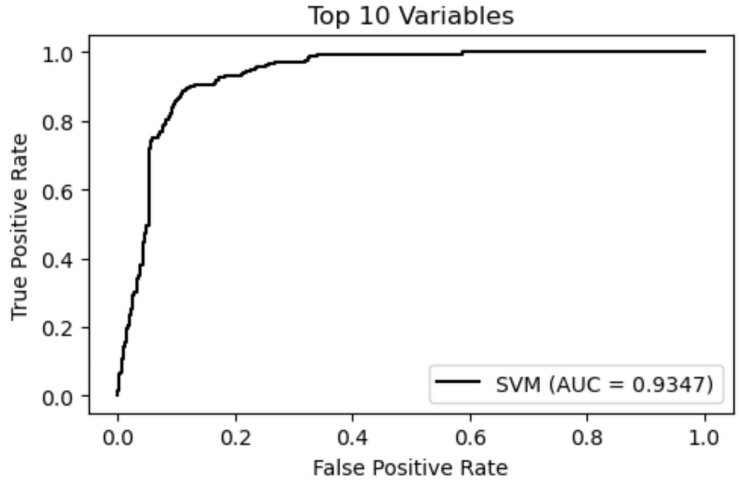
# Machine Learning: Support Vector Machine (SVM) to determine Bankruptcy

## 1 All Variables




| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 (Not Bankrupt) | 0.9767 | 0.9134 | 0.9440 | 1328 |
| 1 (Bankrupt) | 0.9177 | 0.9779 | 0.9569 | 1312 |
| | | | | |
| Accuracy | | | 0.9455 | 2640 |
| Macro Average | 0.9472 | 0.9456 | 0.9454 | 2640 |
| Weighted Average | 0.9474 | 0.9455 | 0.9454 | 2640 |

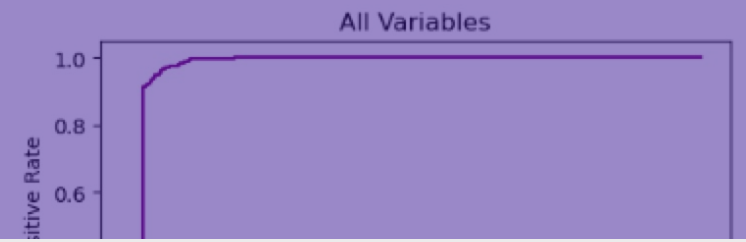
## 2 Top 10 Variables



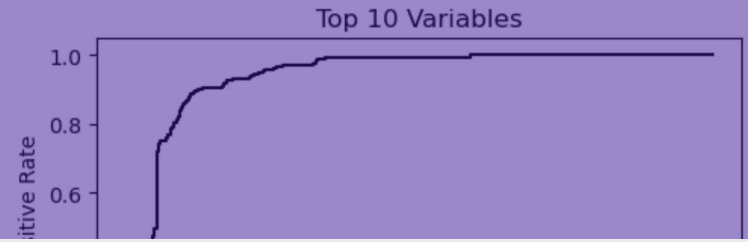

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 (Not Bankrupt) | 0.8996 | 0.8434 | 0.8706 | 1328 |
| 1 (Bankrupt) | 0.8509 | 0.9047 | 0.8770 | 1312 |
| | | | | |
| Accuracy | | | 0.8739 | 2640 |
| Macro Average | 0.8752 | 0.8740 | 0.8738 | 2640 |
| Weighted Average | 0.8754 | 0.8739 | 0.8738 | 2640 |

# Machine Learning: Support Vector Machine (SVM) to determine Bankruptcy

1 All Variables




2 Top 10 Variables




**SVM Model** trained on the **full dataset (Model 1)** is **more accurate (94.55%)** compared to the model trained on the **top 10 variables (Model 2) (87.39%)**

High AUC suggests that SVM Model is able to **distinguish positive and negative cases very well**

| | | | | |
|---|---|---|---|---|
| Bankrupt) | | | | |
| 1 (Bankrupt) | 0.9177 | 0.9779 | 0.9569 | 1312 |
| | | | | |
| Accuracy | | | 0.9455 | 2640 |
| Macro Average | 0.9472 | 0.9456 | 0.9454 | 2640 |
| Weighted Average | 0.9474 | 0.9455 | 0.9454 | 2640 |

| | | | | |
|---|---|---|---|---|
| Bankrupt) | | | | |
| 1 (Bankrupt) | 0.8509 | 0.9047 | 0.8770 | 1312 |
| | | | | |
| Accuracy | | | 0.8739 | 2640 |
| Macro Average | 0.8752 | 0.8740 | 0.8738 | 2640 |
| Weighted Average | 0.8754 | 0.8739 | 0.8738 | 2640 |

# Machine Learning: Decision Tree Implementation

## 1 Full Dataset and Top 10 Dataset

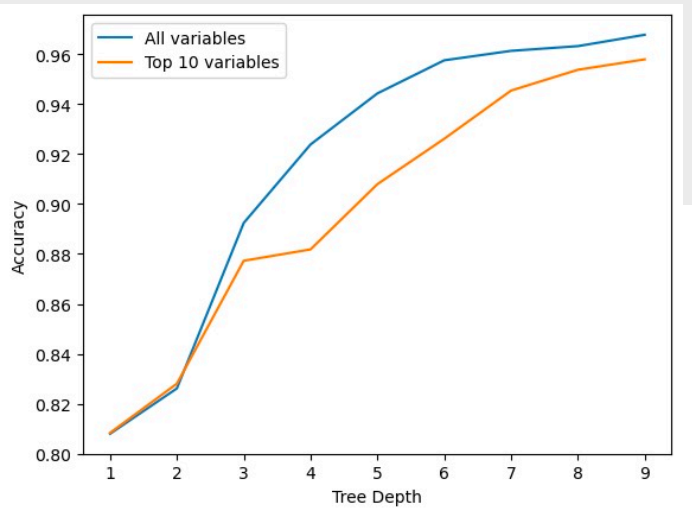
```python
# For top 95 upsampled
for i in range(1,10):
    dectree = DecisionTreeClassifier(max_depth=i)
    dectree.fit(X_train_upsample, y_train_upsample)
    y_pred_upsample = dectree.predict(X_test_upsample)
    accuracy = accuracy_score(y_test_upsample, y_pred_upsample)
    print(f"Accuracy of the model for all variables for tree of depth {i}: {accuracy*100:.2f}%")
print("----------")


# For top 10 upsampled
for i in range(1,10):
    dectree = DecisionTreeClassifier(max_depth=i)
    dectree.fit(X_train_top10_upsample, y_train_top10_upsample)
    y_pred_upsample_top10 = dectree.predict(X_test_top10_upsample)
    accuracy_top10 = accuracy_score(y_test_top10_upsample, y_pred_upsample_top10)
    print(f"Accuracy of the model for top 10 variables for tree of depth {i}: {accuracy_top10*100:.2f}%")
print("----------")
```

## 2 Tree Pruning

```python
# Reduce the depth of the tree until the performance on the validation set starts to decrease
while True:
    depth = dectree.get_depth()
    if depth == 1:
        break
    dectree = DecisionTreeClassifier(max_depth=depth-1)
    dectree.fit(X_train, y_train)
    y_pred_val = dectree.predict(X_val)
    accuracy_val_new = accuracy_score(y_val, y_pred_val)
    if accuracy_val_new < accuracy_val:
        break
    accuracy_val = accuracy_val_new
```

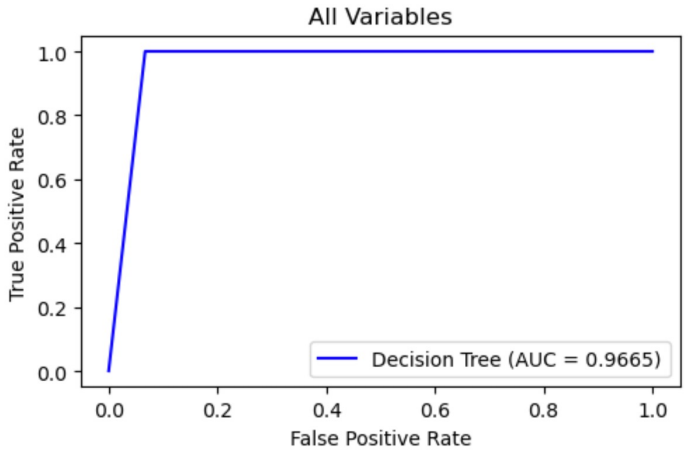
## 3 Explanation of Model Training

- Optimized accuracy of model by engaging in **Pruning**, whereby overfitting is prevented by **removing unnecessary branches and leaves** from the tree



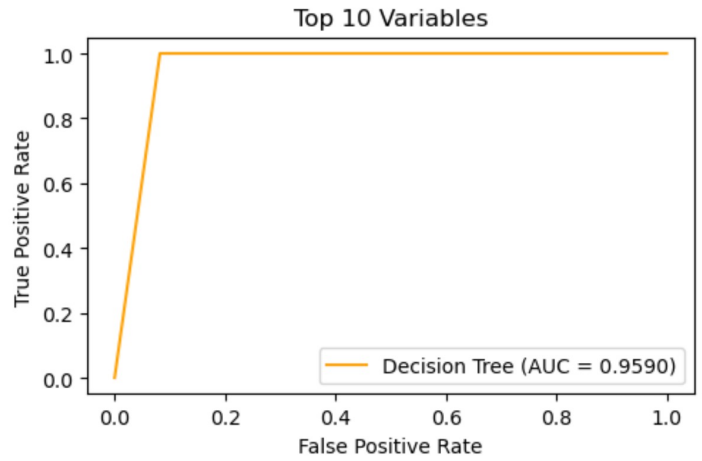

# Machine Learning: Decision Tree to determine Bankruptcy

## 1 All Variables




| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 (Not Bankrupt) | 1.000 | 0.9330 | 0.9653 | 1328 |
| 1 (Bankrupt) | 0.9365 | 1.000 | 0.9672 | 1312 |
| | | | | |
| Accuracy | | | 0.9663 | 2640 |
| Macro Average | 0.9682 | 0.9665 | 0.9663 | 2640 |
| Weighted Average | 0.9684 | 0.9663 | 0.9663 | 2640 |

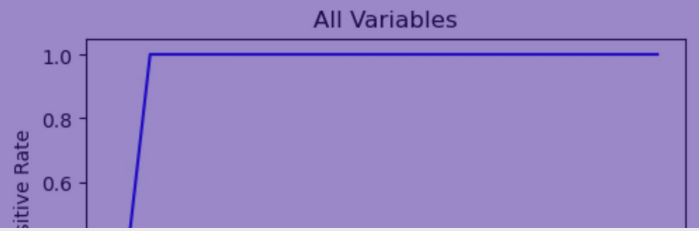
## 2 Top 10 Variables




| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 (Not Bankrupt) | 1.000 | 0.9179 | 0.9572 | 1328 |
| 1 (Bankrupt) | 0.9233 | 1.000 | 0.9601 | 1312 |
| | | | | |
| Accuracy | | | 0.9587 | 2640 |
| Macro Average | 0.9616 | 0.9590 | 0.9587 | 2640 |
| Weighted Average | 0.9619 | 0.9587 | 0.9587 | 2640 |

# Machine Learning: Decision Tree to determine Bankruptcy

1 All Variables




2 Top 10 Variables


**Decision Tree Model** trained on the **full dataset (Model 1)** is **more accurate (96.63%)** compared to the model trained on the **top 10 variables (Model 2) (95.87%)**

High AUC suggests that Decision Tree Model is able to **distinguish positive and negative cases very well**

| | | | | |
|---|---|---|---|---|
| 1 (Bankrupt) | 0.9365 | 1.000 | 0.9672 | 1312 |
| | | | | |
| Accuracy | | | 0.9663 | 2640 |
| Macro Average | 0.9682 | 0.9665 | 0.9663 | 2640 |
| Weighted Average | 0.9684 | 0.9663 | 0.9663 | 2640 |

| | | | | |
|---|---|---|---|---|
| 1 (Bankrupt) | 0.9233 | 1.000 | 0.9601 | 1312 |
| | | | | |
| Accuracy | | | 0.9587 | 2640 |
| Macro Average | 0.9616 | 0.9590 | 0.9587 | 2640 |
| Weighted Average | 0.9619 | 0.9587 | 0.9587 | 2640 |

# Machine Learning: Neural Network Implementation

## 1 Full Dataset

```
# Build the neural network model
# model for upsampled data (All)
model_upsample_all = Sequential()
model_upsample_all.add(Dense(128, input_dim=X_train_upsample.shape[1], activation='relu'))
model_upsample_all.add(BatchNormalization())
model_upsample_all.add(Dense(64, activation='relu'))
model_upsample_all.add(Dropout(0.2))
model_upsample_all.add(Dense(32, activation='relu'))
model_upsample_all.add(Dense(1, activation='sigmoid'))
model_upsample_all.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

## 2 Top 10 Dataset

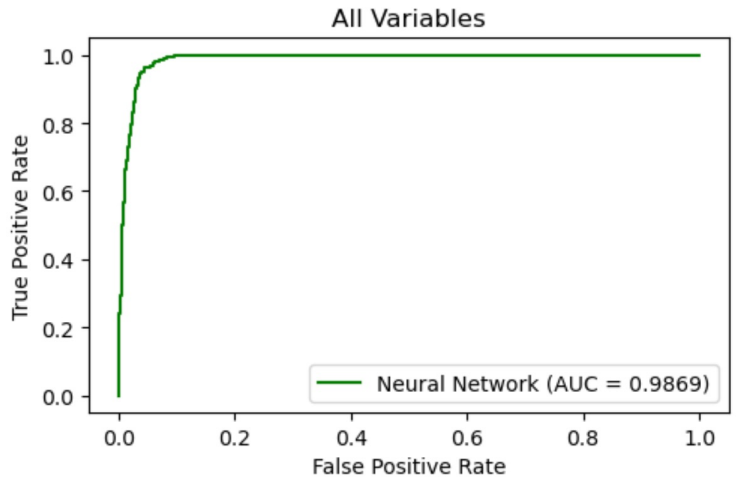
```
# model for upsampled data (Top 10)
model_upsample_top10 = Sequential()
model_upsample_top10.add(Dense(128, input_dim=X_train_top10_upsample.shape[1], activation='relu'))
model_upsample_top10.add(BatchNormalization())
model_upsample_top10.add(Dense(64, activation='relu'))
model_upsample_top10.add(Dropout(0.2))
model_upsample_top10.add(Dense(32, activation='relu'))
model_upsample_top10.add(Dense(1, activation='sigmoid'))
model_upsample_top10.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

## 3 Explanation of Model Training

- The **1st hidden layer has 128 neurons, the 2nd has 64 neurons, and the 3rd has 32 neurons**
- Output layer has a single neuron with a sigmoid activation function, used for **binary classification problems**
- **Batch Normalization** and **Dropout** used to prevent **overfitting**
- **Binary cross-entropy loss function** and the **Adam optimizer** used to optimize the model

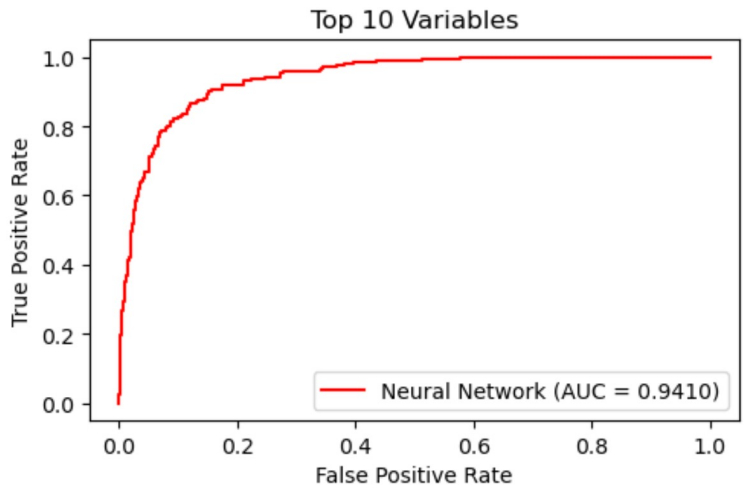
# Machine Learning: Neural Network to determine Bankruptcy

## 1 All Variables




| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 (Not Bankrupt) | 0.9951 | 0.9104 | 0.9508 | 1328 |
| 1 (Bankrupt) | 0.9165 | 0.9954 | 0.9543 | 1312 |
| | | | | |
| Accuracy | | | 0.9527 | 2640 |
| Macro Average | 0.9558 | 0.9529 | 0.9526 | 2640 |
| Weighted Average | 0.9560 | 0.9527 | 0.9526 | 2640 |

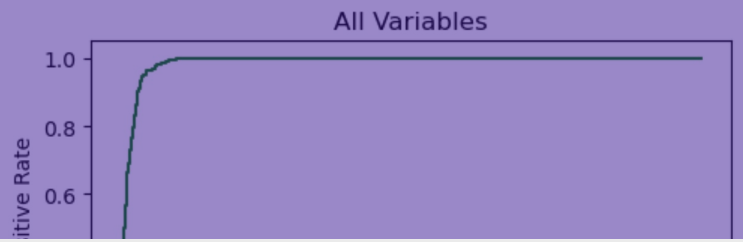
## 2 Top 10 Variables



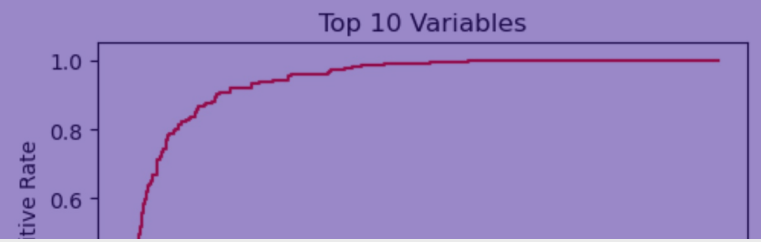

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 (Not Bankrupt) | 0.8140 | 0.9217 | 0.8647 | 1328 |
| 1 (Bankrupt) | 0.9085 | 0.7873 | 0.8436 | 1312 |
| | | | | |
| Accuracy | | | 0.8549 | 2640 |
| Macro Average | 0.8615 | 0.8545 | 0.8542 | 2640 |
| Weighted Average | 0.8612 | 0.8549 | 0.8542 | 2640 |

# Machine Learning: Neural Network to determine Bankruptcy

1 All Variables




2 Top 10 Variables




**Neural Network Model** trained on the **full dataset (Model 1)** is **more accurate (95.27%)** compared to the model trained on the **top 10 variables (Model 2) (85.49%)**

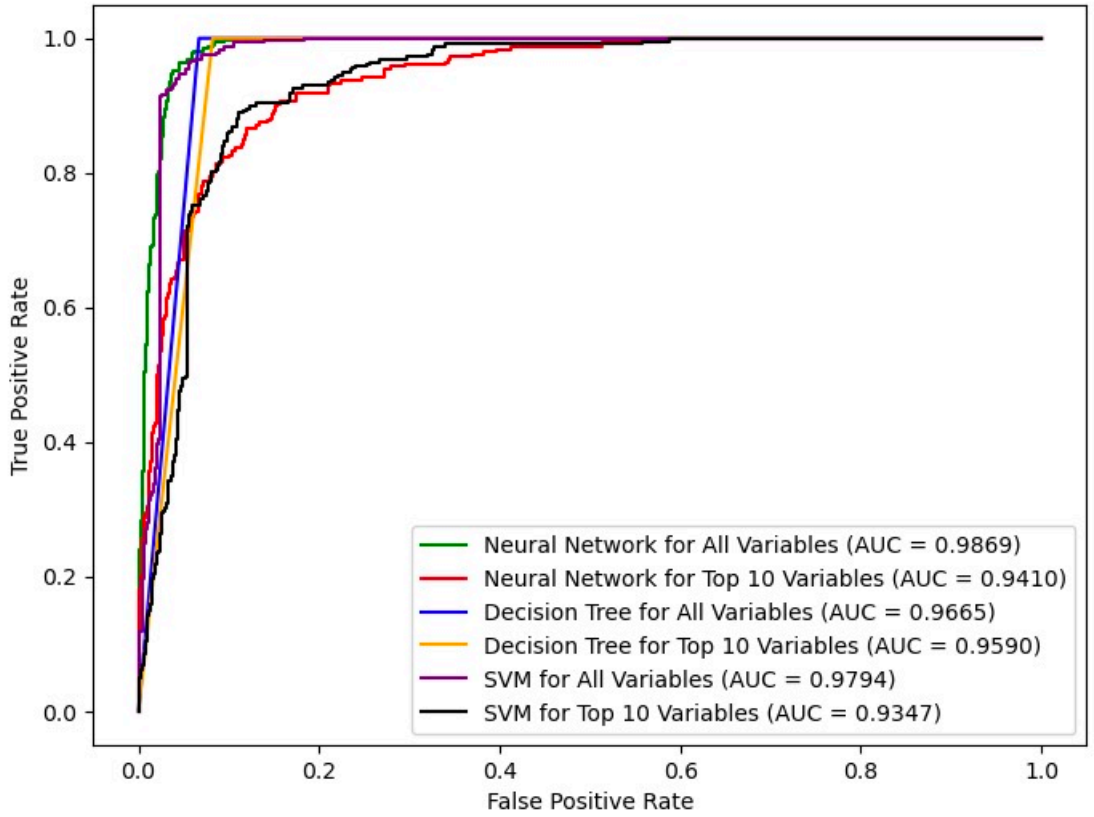
High AUC suggests that Neural Network Model is able to **distinguish positive and negative cases very well**

| | | | | |
|---|---|---|---|---|
| 1 (Bankrupt) | 0.9165 | 0.9954 | 0.9543 | 1312 |
| | | | | |
| Accuracy | | | 0.9527 | 2640 |
| Macro Average | 0.9558 | 0.9529 | 0.9526 | 2640 |
| Weighted Average | 0.9560 | 0.9527 | 0.9526 | 2640 |

| | | | | |
|---|---|---|---|---|
| 1 (Bankrupt) | 0.9085 | 0.7873 | 0.8436 | 1312 |
| | | | | |
| Accuracy | | | 0.8549 | 2640 |
| Macro Average | 0.8615 | 0.8545 | 0.8542 | 2640 |
| Weighted Average | 0.8612 | 0.8549 | 0.8542 | 2640 |

# Machine Learning: Comparison of all Machine Learning Models

## 1 Comparison




## 2 Insights

| Model | Dataset | AUC | Accuracy |
|---|---|---|---|
| SVM | All Variables | 0.9794 | 94.55% |
| SVM | Top 10 | 0.9347 | 87.39% |
| Decision Tree | All Variables | 0.9665 | 96.63% |
| Decision Tree | Top 10 | 0.9590 | 95.87% |
| Neural Network | All Variables | 0.9869 | 95.27% |
| Neural Network | Top 10 | 0.9410 | 85.49% |

- Top 2 models are **Neural Network Model and Decision Tree Model**
- Neural Network Model with **highest AUC**
- Decision Tree Model with **highest Accuracy**

# Final Takeaways: Conclusion and Final Deduction

**1. Factors affecting bankruptcy are not mainly limited to the top 10 correlated variables**

1 Decision Tree

- **Has a higher Accuracy than the Neural Network Model**
- **Can correctly classify maximum number of firms**

2 Neural Network

- **Lower Accuracy but higher AUC implies better ability to distinguish between bankrupt and non-bankrupt firms**

**2. Neural Network is the best choice**

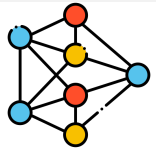
- Cost of false negatives is much higher than cost of false positives
- A model with higher sensitivity and AUC would therefore be preferred as entities tend to be more risk-averse

# Evaluation & Final Thoughts

## Learning Something New

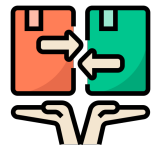
1. **Upsampling** of Data

2. Using **Neural Network Models** for Data Classification and Prediction



3. Using **Support Vector Machines** for Data Classification and Prediction

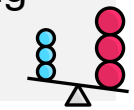
4. Utilizing **Receiver Operation Characteristic and Area Under Curve** to compare and evaluate ML models



# Evaluation & Final Thoughts

## Key Learning Points

1. Using a raw dataset can lead to **skewed results**, hence the need for appropriate sampling techniques



2. Accuracy is **not the only metric** one should use to determine the success of a model

3. **Context and intent** is critical when arriving at a conclusion to any data science related problem

## Future Exploration

1. Expand out of Taiwan to **validate** model

Nasdaq

2. Include relevant **sector-specific data** for each company

# Thank You!

**Ong Jing Xuan U2122081B | Somesh Sahu U2122235C | Nathaniel Yew U2223621F**

# References

1.US Courts. (January 1, 2023). Annual number of business bankruptcy cases filed in the United States from 2000 to 2022 [Graph]. In Statista. Retrieved April 21, 2023, from https://www.statista.com/statistics/817918/number-of-business-bankruptcies-in-the-united-states/

2.Bhandari, A. (2023). Guide to AUC ROC Curve in Machine Learning : What Is Specificity? Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/#What_is_the_AUC-ROC_Curve?

3.Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, *5*(4), 221–232. https://doi.org/10.1007/s13748-016-0094-0