Company Bankruptcy Prediction

A project that focuses in predicting company bankruptcy based on 95 different variables. Mostly financial metrics. The data were collected from the Taiwan Economic Journal for the years 1999 to 2009. Company bankruptcy was defined based on the business regulations of the Taiwan Stock Exchange. The kaggle source can be seen here: https://www.kaggle.com/datasets/fedesoriano/company-bankruptcy-prediction

Foto de Frolda en Unsplash

0.1 Import libraries and data

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams["figure.figsize"] = (20,10)

import seaborn as sns

from sklearn.model_selection import train_test_split
```

In [2]:
```
# Declare random seed.
seed = 42

# Import data and show first 5 rows.
file = 'data/data.csv'
df = pd.read_csv(file)
df.head()
```

Out[2]:

	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue	...	Net Income to Total Assets	Total assets to GNP price	No-credit Interval	Gross Profit to Sales	Net Income to Stockholder's Equity	Liability to Equity	Degree of Financial Leverage (DFL)	Interest Coverage Ratio (Interest expense to EBIT)	Net Income Flag	L
0	1	0.370594	0.424389	0.405750	0.601457	0.601457	0.998969	0.796887	0.808809	0.302646	...	0.716845	0.009219	0.622879	0.601453	0.827890	0.290202	0.026601	0.564050	1	0.0
1	1	0.464291	0.538214	0.516730	0.610235	0.610235	0.998946	0.797380	0.809301	0.303556	...	0.795297	0.008323	0.623652	0.610237	0.839969	0.283846	0.264577	0.570175	1	0.0
2	1	0.426071	0.499019	0.472295	0.601450	0.601364	0.998857	0.796403	0.808388	0.302035	...	0.774670	0.040003	0.623841	0.601449	0.836774	0.290189	0.026555	0.563706	1	0.0
3	1	0.399844	0.451265	0.457733	0.583541	0.583541	0.998700	0.796967	0.808966	0.303350	...	0.739555	0.003252	0.622929	0.583538	0.834697	0.281721	0.026697	0.564663	1	0.0
4	1	0.465022	0.538432	0.522298	0.598783	0.598783	0.998973	0.797366	0.809304	0.303475	...	0.795016	0.003878	0.623521	0.598782	0.839973	0.278514	0.024752	0.575617	1	0.0

5 rows × 96 columns

1.0 Exploratory Data Analysis

1.1 Exploring variables

We will display information like name, data type, null count, number of entries, memory usage and more. This will help us gain an insight of what kind of model or what transformations need to be made to the data in order to have optimal performance.

In [3]:
```
# Visualize variable information.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6819 entries, 0 to 6818
Data columns (total 96 columns):
 #   Column                                                    Non-Null Count  Dtype
---  ------                                                    --------------  -----
 0   Bankrupt?                                                 6819 non-null   int64
 1    ROA(C) before interest and depreciation before interest  6819 non-null   float64
 2    ROA(A) before interest and % after tax                   6819 non-null   float64
 3    ROA(B) before interest and depreciation after tax        6819 non-null   float64
 4    Operating Gross Margin                                   6819 non-null   float64
 5    Realized Sales Gross Margin                              6819 non-null   float64
 6    Operating Profit Rate                                    6819 non-null   float64
 7    Pre-tax net Interest Rate                                6819 non-null   float64
 8    After-tax net Interest Rate                              6819 non-null   float64
 9    Non-industry income and expenditure/revenue              6819 non-null   float64
 10   Continuous interest rate (after tax)                     6819 non-null   float64
 11   Operating Expense Rate                                   6819 non-null   float64
 12   Research and development expense rate                    6819 non-null   float64
 13   Cash flow rate                                           6819 non-null   float64
 14   Interest-bearing debt interest rate                      6819 non-null   float64
 15   Tax rate (A)                                             6819 non-null   float64
 16   Net Value Per Share (B)                                  6819 non-null   float64
 17   Net Value Per Share (A)                                  6819 non-null   float64
 18   Net Value Per Share (C)                                  6819 non-null   float64
 19   Persistent EPS in the Last Four Seasons                  6819 non-null   float64
 20   Cash Flow Per Share                                      6819 non-null   float64
 21   Revenue Per Share (Yuan ¥)                               6819 non-null   float64
 22   Operating Profit Per Share (Yuan ¥)                      6819 non-null   float64
 23   Per Share Net profit before tax (Yuan ¥)                 6819 non-null   float64
 24   Realized Sales Gross Profit Growth Rate                  6819 non-null   float64
 25   Operating Profit Growth Rate                             6819 non-null   float64
 26   After-tax Net Profit Growth Rate                         6819 non-null   float64
 27   Regular Net Profit Growth Rate                           6819 non-null   float64
 28   Continuous Net Profit Growth Rate                        6819 non-null   float64
 29   Total Asset Growth Rate                                  6819 non-null   float64
 30   Net Value Growth Rate                                    6819 non-null   float64
 31   Total Asset Return Growth Rate Ratio                     6819 non-null   float64
 32   Cash Reinvestment %                                      6819 non-null   float64
 33   Current Ratio                                            6819 non-null   float64
 34   Quick Ratio                                              6819 non-null   float64
 35   Interest Expense Ratio                                   6819 non-null   float64
 36   Total debt/Total net worth                               6819 non-null   float64
 37   Debt ratio %                                             6819 non-null   float64
 38   Net worth/Assets                                         6819 non-null   float64
 39   Long-term fund suitability ratio (A)                     6819 non-null   float64
 40   Borrowing dependency                                     6819 non-null   float64
 41   Contingent liabilities/Net worth                         6819 non-null   float64
 42   Operating profit/Paid-in capital                         6819 non-null   float64
 43   Net profit before tax/Paid-in capital                    6819 non-null   float64
 44   Inventory and accounts receivable/Net value              6819 non-null   float64
 45   Total Asset Turnover                                     6819 non-null   float64
 46   Accounts Receivable Turnover                             6819 non-null   float64
 47   Average Collection Days                                  6819 non-null   float64
 48   Inventory Turnover Rate (times)                          6819 non-null   float64
 49   Fixed Assets Turnover Frequency                          6819 non-null   float64
 50   Net Worth Turnover Rate (times)                          6819 non-null   float64
 51   Revenue per person                                       6819 non-null   float64
 52   Operating profit per person                              6819 non-null   float64
 53   Allocation rate per person                               6819 non-null   float64
 54   Working Capital to Total Assets                          6819 non-null   float64
 55   Quick Assets/Total Assets                                6819 non-null   float64
 56   Current Assets/Total Assets                              6819 non-null   float64
 57   Cash/Total Assets                                        6819 non-null   float64
 58   Quick Assets/Current Liability                           6819 non-null   float64
 59   Cash/Current Liability                                   6819 non-null   float64
 60   Current Liability to Assets                              6819 non-null   float64
 61   Operating Funds to Liability                             6819 non-null   float64
 62   Inventory/Working Capital                                6819 non-null   float64
 63   Inventory/Current Liability                              6819 non-null   float64
 64   Current Liabilities/Liability                            6819 non-null   float64
 65   Working Capital/Equity                                   6819 non-null   float64
 66   Current Liabilities/Equity                               6819 non-null   float64
 67   Long-term Liability to Current Assets                    6819 non-null   float64
 68   Retained Earnings to Total Assets                        6819 non-null   float64
 69   Total income/Total expense                               6819 non-null   float64
 70   Total expense/Assets                                     6819 non-null   float64
 71   Current Asset Turnover Rate                              6819 non-null   float64
 72   Quick Asset Turnover Rate                                6819 non-null   float64
 73   Working capitcal Turnover Rate                           6819 non-null   float64
 74   Cash Turnover Rate                                       6819 non-null   float64
 75   Cash Flow to Sales                                       6819 non-null   float64
 76   Fixed Assets to Assets                                   6819 non-null   float64
 77   Current Liability to Liability                           6819 non-null   float64
 78   Current Liability to Equity                              6819 non-null   float64
 79   Equity to Long-term Liability                            6819 non-null   float64
 80   Cash Flow to Total Assets                                6819 non-null   float64
 81   Cash Flow to Liability                                   6819 non-null   float64
 82   CFO to Assets                                            6819 non-null   float64
 83   Cash Flow to Equity                                      6819 non-null   float64
 84   Current Liability to Current Assets                      6819 non-null   float64
 85   Liability-Assets Flag                                    6819 non-null   int64
 86   Net Income to Total Assets                               6819 non-null   float64
 87   Total assets to GNP price                                6819 non-null   float64
 88   No-credit Interval                                       6819 non-null   float64
 89   Gross Profit to Sales                                    6819 non-null   float64
 90   Net Income to Stockholder's Equity                       6819 non-null   float64
 91   Liability to Equity                                      6819 non-null   float64
 92   Degree of Financial Leverage (DFL)                       6819 non-null   float64
 93   Interest Coverage Ratio (Interest expense to EBIT)       6819 non-null   float64
 94   Net Income Flag                                          6819 non-null   int64
 95   Equity to Liability                                      6819 non-null   float64
dtypes: float64(93), int64(3)
memory usage: 5.0 MB
```

We can see that most variables don't have null values and all of them are of a numeric type so no encoding needs to be made. It is important to check as well that there are no variables with the same value.

In [4]:
```
# Find variables with the same value for all data points.
df.nunique()[df.nunique()==1]
```

Out[4]:
```
 Net Income Flag    1
dtype: int64
```

In [5]:
```
# See the name for each variable.
df.columns
```

Out[5]:
```
Index(['Bankrupt?', ' ROA(C) before interest and depreciation before interest',
       ' ROA(A) before interest and % after tax',
       ' ROA(B) before interest and depreciation after tax',
       ' Operating Gross Margin', ' Realized Sales Gross Margin',
       ' Operating Profit Rate', ' Pre-tax net Interest Rate',
       ' After-tax net Interest Rate',
       ' Non-industry income and expenditure/revenue',
       ' Continuous interest rate (after tax)', ' Operating Expense Rate',
       ' Research and development expense rate', ' Cash flow rate',
       ' Interest-bearing debt interest rate', ' Tax rate (A)',
       ' Net Value Per Share (B)', ' Net Value Per Share (A)',
       ' Net Value Per Share (C)', ' Persistent EPS in the Last Four Seasons',
       ' Cash Flow Per Share', ' Revenue Per Share (Yuan ¥)',
       ' Operating Profit Per Share (Yuan ¥)',
       ' Per Share Net profit before tax (Yuan ¥)',
       ' Realized Sales Gross Profit Growth Rate',
       ' Operating Profit Growth Rate', ' After-tax Net Profit Growth Rate',
       ' Regular Net Profit Growth Rate', ' Continuous Net Profit Growth Rate',
       ' Total Asset Growth Rate', ' Net Value Growth Rate',
       ' Total Asset Return Growth Rate Ratio', ' Cash Reinvestment %',
       ' Current Ratio', ' Quick Ratio', ' Interest Expense Ratio',
       ' Total debt/Total net worth', ' Debt ratio %', ' Net worth/Assets',
       ' Long-term fund suitability ratio (A)', ' Borrowing dependency',
       ' Contingent liabilities/Net worth',
       ' Operating profit/Paid-in capital',
       ' Net profit before tax/Paid-in capital',
       ' Inventory and accounts receivable/Net value', ' Total Asset Turnover',
       ' Accounts Receivable Turnover', ' Average Collection Days',
       ' Inventory Turnover Rate (times)', ' Fixed Assets Turnover Frequency',
       ' Net Worth Turnover Rate (times)', ' Revenue per person',
       ' Operating profit per person', ' Allocation rate per person',
       ' Working Capital to Total Assets', ' Quick Assets/Total Assets',
       ' Current Assets/Total Assets', ' Cash/Total Assets',
       ' Quick Assets/Current Liability', ' Cash/Current Liability',
       ' Current Liability to Assets', ' Operating Funds to Liability',
       ' Inventory/Working Capital', ' Inventory/Current Liability',
       ' Current Liabilities/Liability', ' Working Capital/Equity',
       ' Current Liabilities/Equity', ' Long-term Liability to Current Assets',
       ' Retained Earnings to Total Assets', ' Total income/Total expense',
       ' Total expense/Assets', ' Current Asset Turnover Rate',
       ' Quick Asset Turnover Rate', ' Working capitcal Turnover Rate',
       ' Cash Turnover Rate', ' Cash Flow to Sales', ' Fixed Assets to Assets',
       ' Current Liability to Liability', ' Current Liability to Equity',
       ' Equity to Long-term Liability', ' Cash Flow to Total Assets',
       ' Cash Flow to Liability', ' CFO to Assets', ' Cash Flow to Equity',
       ' Current Liability to Current Assets', ' Liability-Assets Flag',
       ' Net Income to Total Assets', ' Total assets to GNP price',
       ' No-credit Interval', ' Gross Profit to Sales',
       ' Net Income to Stockholder's Equity', ' Liability to Equity',
       ' Degree of Financial Leverage (DFL)',
       ' Interest Coverage Ratio (Interest expense to EBIT)',
       ' Net Income Flag', ' Equity to Liability'],
      dtype='object')
```

In [6]:
```
# Drop variables which do not contribute towards prediction.
new_df = df.drop(' Net Income Flag',axis=1)
new_df.nunique()[df.nunique()==1]
```

Out[6]:
```
Series([], dtype: int64)
```

Finally double check to make sure that there are no missing values.

In [7]:
```
# Display all variables that have null values.
new_df.isnull().sum()[df.isnull().sum()>0]
```

Out[7]:
```
Series([], dtype: int64)
```

1.2 Check for Multicollinearity

In [8]:
```
# Calculate correlation array between variables and sort it.
correlation_array = new_df.corr().unstack().drop_duplicates().sort_values(kind='Quicksort')

# Get all entries of array that are less than -0.5
top_negative_corrs = correlation_array[correlation_array<-0.5]
print(top_negative_corrs)
```

```
 Debt ratio %                        Net worth/Assets                               -1.000000
 Net worth/Assets                    Current Liability to Assets                    -0.842583
 Borrowing dependency                Net Income to Stockholder's Equity             -0.806478
 Net Income to Stockholder's Equity  Liability to Equity                            -0.791836
 Contingent liabilities/Net worth    Working Capital/Equity                         -0.767778
 Current Liabilities/Equity          Net Income to Stockholder's Equity             -0.749621
 Working Capital/Equity              Current Liabilities/Equity                     -0.692675
                                     Liability to Equity                            -0.650474
 Debt ratio %                        Equity to Liability                            -0.625879
 Working Capital to Total Assets     Current Liability to Current Assets            -0.625560
 Equity to Long-term Liability       Net Income to Stockholder's Equity             -0.615905
 Operating Profit Rate               Non-industry income and expenditure/revenue   -0.592006
 Retained Earnings to Total Assets   Total expense/Assets                           -0.541559
 Borrowing dependency                Working Capital/Equity                         -0.535714
 Debt ratio %                        Working Capital to Total Assets                -0.528797
 Current Liability to Assets         Equity to Liability                            -0.506360
dtype: float64
```

In [9]:
```
# Inverse order of array and plot the variables that have a correlation higher than 0.75.
top_positive_corrs = correlation_array[correlation_array>0.75][::-1]
print(top_positive_corrs)
```

```
Bankrupt?                                                 Bankrupt?                                            1.000000
 Operating Gross Margin                                   Gross Profit to Sales                                1.000000
 Net Value Per Share (A)                                  Net Value Per Share (C)                              0.999837
 Operating Gross Margin                                   Realized Sales Gross Margin                          0.999518
 Realized Sales Gross Margin                              Gross Profit to Sales                                0.999518
 Net Value Per Share (B)                                  Net Value Per Share (A)                              0.999342
                                                          Net Value Per Share (C)                              0.999179
 Operating Profit Per Share (Yuan ¥)                      Operating profit/Paid-in capital                     0.998696
 After-tax Net Profit Growth Rate                         Regular Net Profit Growth Rate                       0.996186
 Pre-tax net Interest Rate                                Continuous interest rate (after tax)                 0.993617
 ROA(C) before interest and depreciation before interest  ROA(B) before interest and depreciation after tax    0.986849
 Pre-tax net Interest Rate                                After-tax net Interest Rate                          0.986379
 After-tax net Interest Rate                              Continuous interest rate (after tax)                 0.984452
 Current Liabilities/Equity                               Liability to Equity                                  0.963908
 Per Share Net profit before tax (Yuan ¥)                 Net profit before tax/Paid-in capital                0.962723
 ROA(A) before interest and % after tax                   Net Income to Total Assets                           0.961552
 Persistent EPS in the Last Four Seasons                  Net profit before tax/Paid-in capital                0.959461
 Borrowing dependency                                     Liability to Equity                                  0.955857
 ROA(A) before interest and % after tax                   ROA(B) before interest and depreciation after tax    0.955741
 Persistent EPS in the Last Four Seasons                  Per Share Net profit before tax (Yuan ¥)             0.955591
 Working capitcal Turnover Rate                           Cash Flow to Sales                                   0.948194
 ROA(C) before interest and depreciation before interest  ROA(A) before interest and % after tax               0.940124
 Operating Profit Rate                                    Pre-tax net Interest Rate                            0.916448
                                                          Continuous interest rate (after tax)                 0.915544
 ROA(B) before interest and depreciation after tax        Net Income to Total Assets                           0.912040
 Borrowing dependency                                     Current Liabilities/Equity                           0.892772
 ROA(C) before interest and depreciation before interest  Net Income to Total Assets                           0.887670
 Operating profit/Paid-in capital                         Net profit before tax/Paid-in capital                0.887370
 Operating Profit Per Share (Yuan ¥)                      Net profit before tax/Paid-in capital                0.886157
 Cash flow rate                                           Operating Funds to Liability                         0.880562
 Persistent EPS in the Last Four Seasons                  Operating Profit Per Share (Yuan ¥)                  0.876769
                                                          Operating profit/Paid-in capital                     0.873641
 Operating Profit Rate                                    After-tax net Interest Rate                          0.862191
 Operating Profit Per Share (Yuan ¥)                      Per Share Net profit before tax (Yuan ¥)             0.861813
 Per Share Net profit before tax (Yuan ¥)                 Operating profit/Paid-in capital                     0.858310
 Debt ratio %                                             Current Liability to Assets                          0.842583
 Borrowing dependency                                     Equity to Long-term Liability                        0.806889
 Retained Earnings to Total Assets                        Net Income to Total Assets                           0.794189
 Equity to Long-term Liability                            Liability to Equity                                  0.778135
 ROA(C) before interest and depreciation before interest  Persistent EPS in the Last Four Seasons              0.775006
 ROA(A) before interest and % after tax                   Persistent EPS in the Last Four Seasons              0.764828
 ROA(B) before interest and depreciation after tax        Persistent EPS in the Last Four Seasons              0.764597
 ROA(A) before interest and % after tax                   Net profit before tax/Paid-in capital                0.758234
 Total Asset Turnover                                     Net Worth Turnover Rate (times)                      0.757414
 Net Value Per Share (B)                                  Persistent EPS in the Last Four Seasons              0.755568
 Quick Assets/Total Assets                                Current Assets/Total Assets                          0.755453
 Net Value Per Share (A)                                  Persistent EPS in the Last Four Seasons              0.755409
 Net Value Per Share (C)                                  Persistent EPS in the Last Four Seasons              0.755217
 ROA(C) before interest and depreciation before interest  Net profit before tax/Paid-in capital                0.753339
 ROA(A) before interest and % after tax                   Per Share Net profit before tax (Yuan ¥)             0.752578
 ROA(C) before interest and depreciation before interest  Per Share Net profit before tax (Yuan ¥)             0.750564
dtype: float64
```

We can see that there are many variables that do possess a high correlation which needs to be addressed. One way to do this is by either eliminating some of the variables or by applying a dimensionality reduction technique like PCA. Another approach is to use a model that is more robust to these cases like tree-based algorithms.

1.3 Check data for imbalance & split into train/test sets

In [10]:
```
# Count the number of cases for each class
new_df['Bankrupt?'].value_counts()
```

Out[10]:
```
0    6599
1     220
Name: Bankrupt?, dtype: int64
```

We can see how companies with bankruptcy represent only around 3% of all the cases. This will be an issue because the model could be biased to always predict the class with the bigger number of examples. This will need to be addressed down the road.

In order to prevent data leakage from our training data into our test data, we will split our data before doing any transformations to keep an accurate representation of real test cases.

In [11]:
```
# Separate dependent variable and split data before transformation.
X, y = new_df.loc[:,new_df.columns != 'Bankrupt?'], new_df['Bankrupt?']
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=seed)
X_train.head()
```

Out[11]:

	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue	Continuous interest rate (after tax)	...	Liability-Assets Flag	Net Income to Total Assets	Total assets to GNP price	No-credit Interval	Gross Profit to Sales	Net Income to Stockholder's Equity	Liability to Equity	Degree of Financial Leverage (DFL)	Interes Coverag Rati (Interes expens to EBI
2825	0.492663	0.550207	0.539751	0.646168	0.646168	0.999330	0.797829	0.809604	0.303537	0.781916	...	0	0.807538	0.018425	0.624224	0.646167	0.842213	0.295068	0.026791	0.56515
251	0.451567	0.494003	0.502757	0.594740	0.594791	0.998916	0.797137	0.809078	0.303194	0.781333	...	0	0.769345	0.014932	0.624274	0.594740	0.837952	0.279378	0.026568	0.56380
1209	0.514698	0.571195	0.564002	0.600528	0.600528	0.999054	0.797446	0.809353	0.303446	0.781614	...	0	0.807833	0.000979	0.690700	0.600523	0.841299	0.283328	0.027549	0.56726
4989	0.512163	0.599270	0.566733	0.599021	0.599021	0.999020	0.797484	0.809413	0.303584	0.781627	...	0	0.820229	0.009966	0.628624	0.599020	0.842259	0.281235	0.027002	0.56599
5396	0.598011	0.643262	0.643825	0.617233	0.617233	0.999229	0.797643	0.809517	0.303424	0.781790	...	0	0.857429	0.000398	0.624207	0.617234	0.844172	0.277753	0.026791	0.56516

5 rows × 94 columns

2.0 Model building & evaluation

There are two main issues in our data that need to be taken into consideration when choosing the appropriate statistical method. The first is the data imbalance and the second is the correlation between the variables. We will use tree-based models from the imblern.ensemble library to address both.

2.1 Model validation & selection.

In [12]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold

from imblearn.ensemble import BalancedRandomForestClassifier
from imblearn.ensemble import EasyEnsembleClassifier
from imblearn.ensemble import BalancedBaggingClassifier

from sklearn.model_selection import RandomizedSearchCV
```

In [13]:
```
from helpers import metrics
```

In [14]:
```
# Initiate classification models for imbalanced data.
rf_model = BalancedRandomForestClassifier(
    n_jobs=-1,
    random_state=seed)

easy_model = EasyEnsembleClassifier(
    n_jobs=-1,
    random_state=seed)

bagging_model = BalancedBaggingClassifier(
    n_jobs=-1,
    random_state=seed)
```

We will perform cross-validation to choose between three different models to perform hyperparameter tuning. We will use the AUC score to measure each model in order to account for the imbalance in the data.

In [15]:
```
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=seed)


score_rf = cross_val_score(rf_model, X_train, y_train, scoring='roc_auc', cv=cv, n_jobs=-1)
score_easy = cross_val_score(easy_model, X_train, y_train, scoring='roc_auc', cv=cv, n_jobs=-1)
score_bagging = cross_val_score(bagging_model, X_train, y_train, scoring='roc_auc', cv=cv, n_jobs=-1)

# summarize performance
print('The mean AUC score for each model is:\nBalanced Random Forest: {:.3f}\nEasy Ensemble: {:.3f}\nBalanced Bagging: {:.3f}'.format(
      np.mean(score_rf),
    np.mean(score_easy),
    np.mean(score_bagging))
     )
```

```
The mean AUC score for each model is:
Balanced Random Forest: 0.935
Easy Ensemble: 0.934
Balanced Bagging: 0.917
```

2.2 Model hyperparameter tuning

We can see that the balanced Random Forest has the best performance, so let's apply some hyperparameter tuning to get the best possible result.

In [16]:
```
rf_parameters = {'n_estimators':[50,100,200,300,400,500,600,700,900, 1000,1200,1400,1500,1700,2000],
                 'max_depth':[int(x) for x in np.linspace(10, 200, num = 20)],
                 'min_samples_leaf':[1, 2, 3, 4, 5, 10],
                 'min_samples_split': [2, 5, 10, 15],
                 'bootstrap':[True,False]}

#easy_parameters = {'n_estimators':[50,100,200,300,400,500,600,700,900, 1000]}
#bagging_parameters = {'n_estimators':[50,100,200,300,400,500,600,700,900, 1000]}

rf_hyper_tuner = RandomizedSearchCV(rf_model, rf_parameters,scoring= 'roc_auc', random_state=seed).fit(X_train,y_train)
#easy_hyper_tuner = RandomizedSearchCV(easy_model, easy_parameters,scoring= 'roc_auc', random_state=seed).fit(X_train,y_train)
#bagging_hyper_tuner = RandomizedSearchCV(bagging_model, bagging_parameters,scoring= 'roc_auc', random_state=seed).fit(X_train,y_train)
```

In [17]:
```
print('The AUC score for our model is: {:.3f}\nBest Parameters: {}'.format(
      rf_hyper_tuner.best_score_, rf_hyper_tuner.best_params_)
     )
```

```
The AUC score for our model is: 0.940
Best Parameters: {'n_estimators': 1200, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_depth': 10, 'bootstrap': False}
```

Using the best parameter let's build our model and evaluate our results.

3.0 Results

3.1 Optimal model building & fitting

Let's use the optimal variables from our cross-validation to build our random forest and fit it with our test data.

In [18]:
```
# Build optimal model.
clf = BalancedRandomForestClassifier(
    n_estimators = 1200,
    min_samples_split = 2,
    min_samples_leaf= 2,
    max_depth = 10,
    n_jobs=-1,
    random_state=seed)

# Fit model and predict on test data
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
y_prob = clf.predict_proba(X_test)
```

3.2 Model result metrics

We will fit our predicted values as well as the actual class of the data points to our metrics object to get all the results.

In [19]:
```
# Fit metrics object with probabilities and actual predicted class.
results = metrics(y_test,y_pred,y_prob)
```

In [20]:
```
# Classification report metrics
results.report()
```

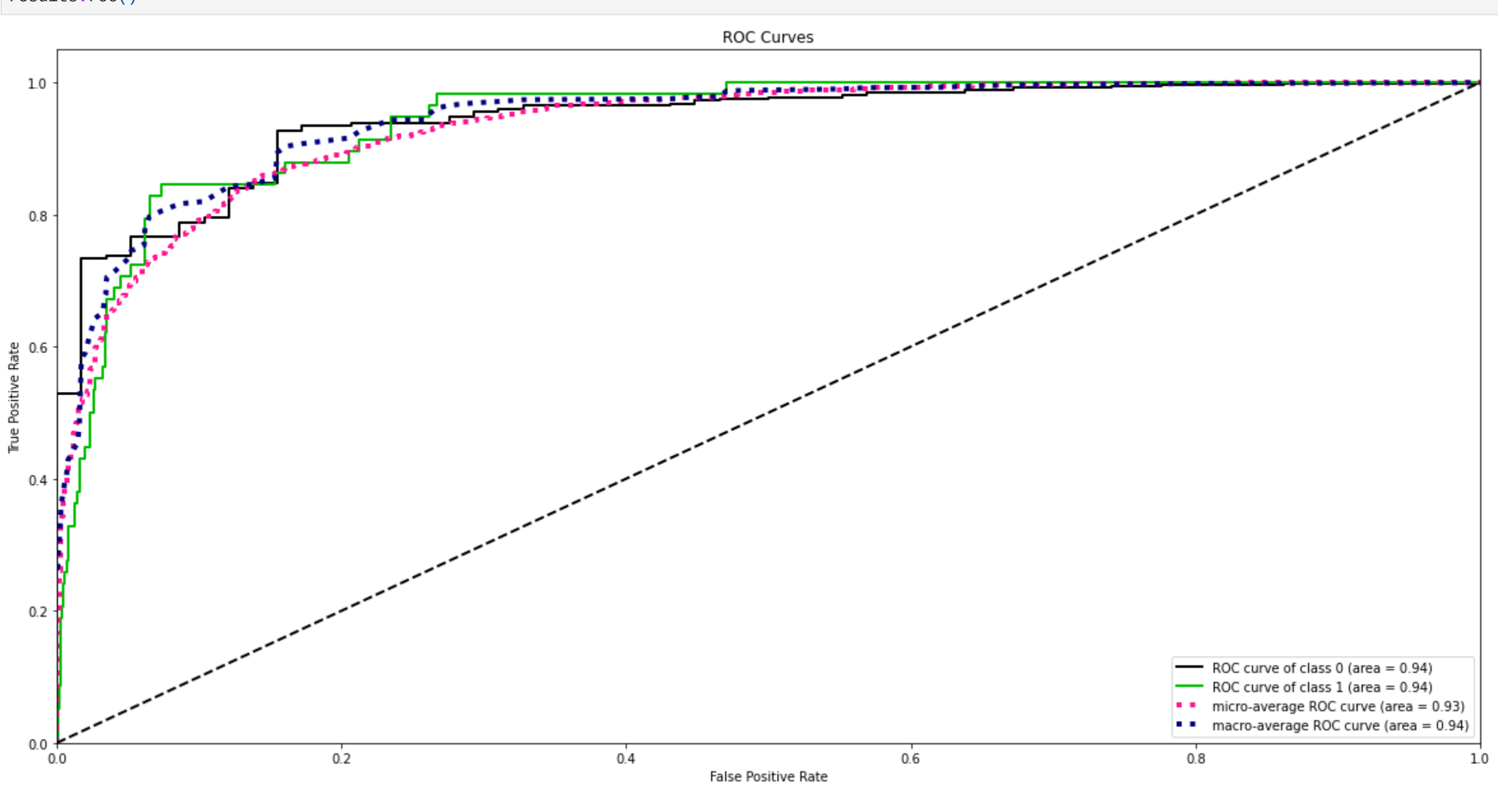
```
              precision    recall  f1-score   support

           0       0.99      0.86      0.92      1647
           1       0.17      0.84      0.29        58

    accuracy                           0.86      1705
   macro avg       0.58      0.85      0.60      1705
weighted avg       0.97      0.86      0.90      1705
```

The classification report explains classification metrics for the two classes. Seeing as the problem is to predict company bankruptcy, recall serves as a more appropriate metric when evaluating our model. This is due to not predicting bankruptcy when it is in fact true would have a lot more serious consequences than predicting bankruptcy and it not being the case. In other words, false negatives carry a higher cost than false positives do.

When analyzing our result, we can see that even though the precission value is low; we managed to achieve an 84% recall value as well as an 86% accuracy on our test data. Overall, a very good performance.

3.21 ROC curve

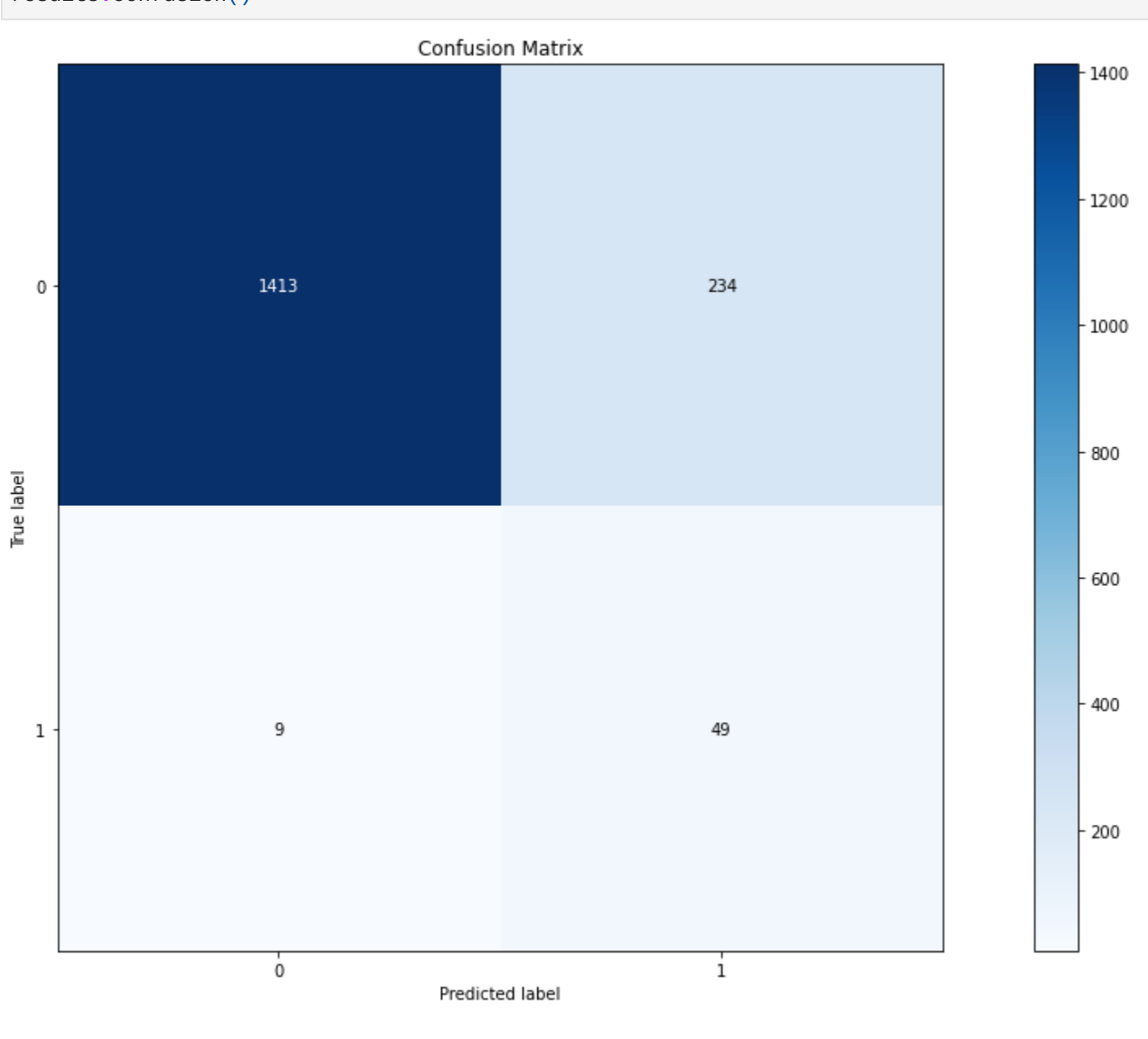
In [21]:
```
# Plot ROC curve
results.roc()
```



Furthermore, by plotting our ROC curve we see that a 0.94 AUC score is achieved for both classes. This shows our model is performing well.

3.22 Confusion matrix

In [22]:
```
#plot confusion matrix
results.confusion()
```



Finally, we are able to visualize our confusion matrix. Showing the overral performance on our test data.

4.0 Conclusions

In conclusion, we were able to build an accurate analytical model to in highly imbalanced data to be able to accurately predict company bankruptcy with an accuracy of 86% and a recall of 84% on our positive values. This was achieved using tree-based models in order to account for multicollinearity issues as well as a measure robust to outliers and the distribution of our dependent variable. Other techniques to increase our performance can be explored like using dimensionality reduction or data augmentation to solve the aforementioned issues. Overall, the project presents satisfactory results.

4.1 Variable importance

For analytical purposes we can also see the main variables that represent a high percentage of influence in predicting our result. This would allow companies to be able to focus on these factors more in the future in order to prevent bankruptcy. Although it is important to note that these results might not be entirely accurate due to the multicollinearity found in the exploratory stage of our project. Nonetheless it serves as a good insight to focus on.

The following graph shows the 20 most important variables that account for 58% of influence when classifying our dependent variable:

In [23]:
```
feature_importance = clf.feature_importances_
names = clf.feature_names_in_

feature_importance, names = zip(*sorted(zip(feature_importance, names),reverse=True))
feature_importance, names = feature_importance[:20], names[:20]

fig, ax = plt.subplots()
ax.barh(range(len(names)), feature_importance)
ax.set_yticks(range(len(names)), labels=names)

ax.invert_yaxis()  # labels read top-to-bottom
ax.set_xlabel('Feature Percentage')
ax.set_title('Feature Explanation of Potability')

print('Features cumulative importance: {:.2f}'.format(np.sum(feature_importance)))
```

```
Features cumulative importance: 0.58
```