Business failure prediction based on 4 classification models with hyperparameter adjustment.

In the field of finance and investment, many theories deal with the indicators and ratios that are crucial for assessing a company's financial health. In-depth research has been carried out to help managers anticipate and avoid any delicate financial situation. Similarly, investors also seek to use these tools to minimise their investment risks.

The aim of this work is to develop classification models to predict the financial distress of a company based on accounting and macroeconomic data. In the field of machine learning, several models have been proposed based on statistical models or, more recently, artificial intelligence, in particular neural networks. I decided to work on 4 different models: Random Forest, Logistic Regression, Multilayer Perceptron Classifier and Neural Network in Keras.

For this project, I worked with a dataset found on Kaggle, collecting bankruptcy cases from the Taiwan Economic Journal for the period 1999 to 2009. The database contains 6819 companies and 96 columns, mainly consisting of public accounting data that all listed companies are required to publish. It also includes macroeconomic financial data such as interest rates, inflation, etc. The definition of company bankruptcy is based on the commercial regulations of the Taiwan Stock Exchange.

One might think that the greater the number of variables, the better our prediction models will perform. The first step is to analyse the quality of the independent variables. This involves visualising the data and analysing certain statistical characteristics so as to be able to deal with the outliers that distort the performance of future classification models. Moreover, multicollinearity often creates problems in the models by making the regression coefficients unstable. It is difficult to identify the individual impact of each variable and can make the model's predictions less reliable, especially for new data. To solve this problem, we can eliminate certain variables that are highly correlated, choose methods to select the important variables, or use regularisation. These approaches help to make the models more robust and better able to make accurate predictions. This stage gives you an idea of the data management technique. (Rosidi.N, 2023)

Secondly, if we leave so many explanatory variables, our model will be overtrained, which will create overfitting. In other words, during the training process, the model will also take into account the extreme values (noises). As a result, the model will overperform on the training data because it will have analysed the true trends but will be biased by the extreme values, which will completely prevent it from generalising correctly. As a result, the model's performance on new variables will be poor. (Yu, Ye, & Liu, 2013)

The real problem is to identify the subset of variables for which the relationship with the target is due to the true underlying law, and not to sample artefacts on marginal contributions to prediction. (Provost.F & Fawcett.T, 2013)

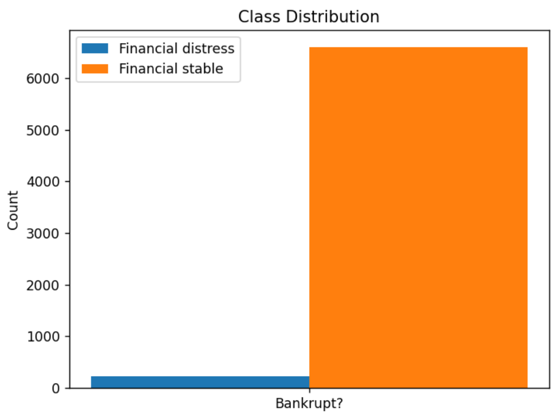
1. Data preprocessing

My first step is to analyse the cleanliness of my database. To do this, I check for missing data and whether all the variables have been correctly encoded. In my case, after a short analysis I realise that the database is complete and clean. My dependent variable is already binary, so I don't need to change anything.

```
any_missing_values = data.isnull().any().any()
print(f"Are there any missing values? {any_missing_values}")
```

```
Are there any missing values? False
```

However, I see that my database is not balanced at all. In fact, we can see that there are only 3% of bankrupt companies, which is very low.



To overcome this problem I mainly used 2 techniques:

- Synthetic Minority Oversampling Technique (SMOTE): Creation of new minority individuals who resemble the others, without being strictly identical. This makes it possible to make the population of minority individuals more homogeneous. A machine learning model is then trained on this transformed data, to which false synthetic individuals have been added. (Chawla, Bowyer, Hall, & Kegelmeyer, 2002)

- ADASYN: The distribution of synthetic examples is adapted according to local density, generating more synthetic examples for areas where the minority class is less dense. (Pilotti.G, 2020)

Finally, our database contains 94 variables. I decided to standardise the values for certain models. This gives an average of 0 and a deviation of 1 for all the data. (Jaadi.Z, 2023)

Classification models

1.1 Random Forest

The Random forest algorithm will create a number N of trees all containing the same number M of input data. This is based on a very simple principle: the power of the majority.

In fact, a large number of relatively uncorrelated trees, operating like a committee that votes to determine each prediction, has greater decision-making power and will outperform any model made up of a single tree, taken in isolation. (Yiu.T, 2019) This is because the trees protect each other from their individual errors. As long as they are not correlated and do not all point in the same direction.

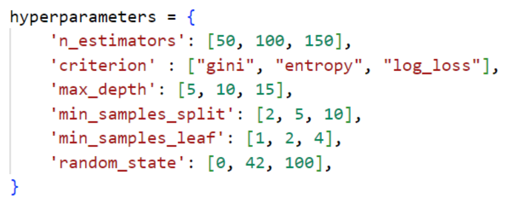
To avoid this problem of correlation between trees, two methods are taken into account by the Random Forest algorithm:

— Bagging: Since decision trees are sensitive to the data they are trained on, each decision tree will be presented with different data, taken from the same initial database comprising a number M of rows. Using a method known as bootstrapping, a sample of size M will be formed by randomly selecting rows with replacement. So, for a number N of trees, a number N of different samples of size M will be created by this bagging technique.

— Random variables: In the case of a simple decision tree, when it comes to making a division within a node, all the independent variables are considered and the division takes place on the one that causes the greatest separation between the data. In the Random Forest model, the trees can only choose from a limited number of variables (which vary from tree to tree). This implies greater variations between tree decisions. As a result, diversification is enhanced and the correlation between trees is greatly reduced.

1.1.2 Hyperparameter tunning

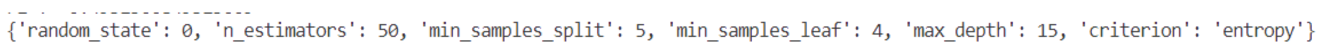
The random forest algorithm has a built-in method for choosing the best combination of parameters with the best performance. The best performance will be based on a penalty calculation during the selection process.



```
hyperparameters = {
    'n_estimators': [50, 100, 150],
    'criterion' : ["gini", "entropy", "log_loss"],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'random_state': [0, 42, 100],
}
```

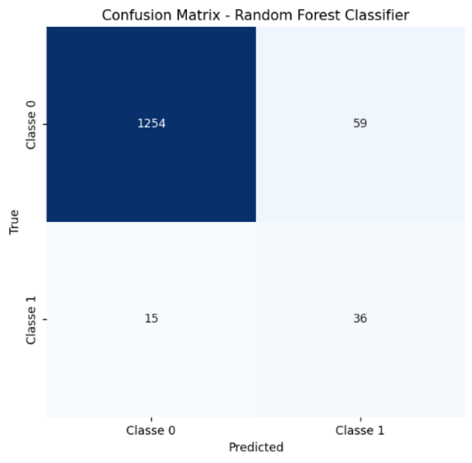
As an example, in the context of my work, I used the RandomizedSearchCV() method, which randomly groups the parameters together and, on the basis of the entropy penalty, keeps the best combination while minimising entropy.

After several attempts, the RandomizedSearchCV() method tells me that the optimal combination of parameters for my random forest classifier is as follows:



```
{'random_state': 0, 'n_estimators': 50, 'min_samples_split': 5, 'min_samples_leaf': 4, 'max_depth': 15, 'criterion': 'entropy'}
```

1.1.3 Results



```
Accuracy : 0.9457478005865103
Precision :  0.37894736842105264
Recall :  0.7058823529411765
Cohen's Kappa :  0.4672268730140299
F1 :  0.4931506849315069
```

By analysing the main performance criteria for classification models, we can see that at first sight the model has an accuracy of 94%, which seems to be magnificent. However, this is not really significant given that in our test sample the majority of companies are not bankrupt and are almost all well ranked. Hence our high accuracy. Another important criterion is recall. Among the bankrupt companies, 70% are well predicted to be bankrupt. The probability of detection is fairly high. Unfortunately, the model is not very accurate. It classifies more companies as bankrupt when they are not. The model has an accuracy of 38%.

1.2 Logistic regression

Instead of computing the regression of Y on X, we would like the regression to give us what is the probability of being positive given a certain vector space. Proba(1/x) (Mercckt.T). So, we would like to formulate the problem like this: compute the likelihood of:

$$P(1/x_i) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_k x_k \; (= \beta_0 + \textstyle\sum_{j=1}^{k} \beta_j \, x_{ij})$$

where k = nbre of features, $P(1/\ x_i)$ = the probability of positive given case x_i

However, using linear regression to calculate probabilities is not efficient. The linear probability may be greater than 1. In addition, the growth in probabilities is not uniform.

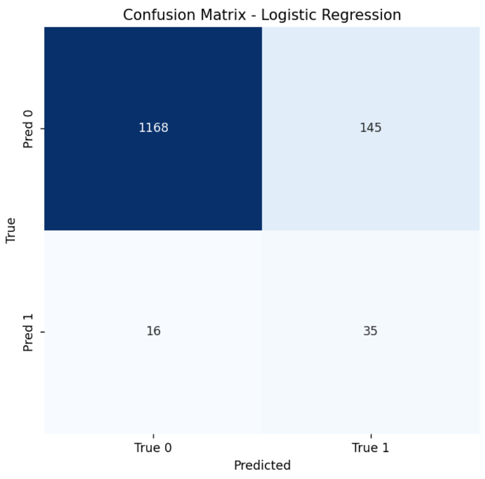
This is actually what we see when we do logistic regression: The probability to be positive given a certain vector is given by this expression.

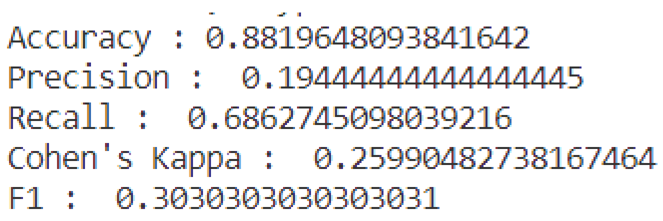
$$P(1|x_i) = \frac{1}{1+e^{-\sum_{j=1}^{k} \beta_j x_{ij}}}$$

1.2.1 Unsupervised Features Reductions

For my logistic regression model, I obtained the best results using Principal component analysis, which is an unsupervised feature reduction technique. Reduce the dimensionality of a data set by computing a new set of variables, smaller than the original size Retain most of the sample's information. By information we mean the variation present in the sample, measured by the correlations between the original variables. (Mercckt.T) The new variables, called principal components, are uncorrelated, and are ordered by the fraction of the total information they capture. A necessary condition for using the PCA is the standardisation of the data. After several tests I decided to keep the first 10 principal components. If I take the same number of PCAs as e variables, I don't make any feature selection. If I don't use enough I could miss out on important information contained in the variables.

1.2.2 Results



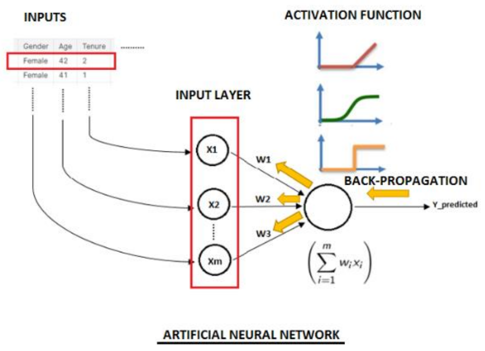


```
Accuracy : 0.8819648093841642
Precision :  0.19444444444444445
Recall :  0.6862745098039216
Cohen's Kappa :  0.25990482738167464
F1 :  0.30303030303030031
```

The results are very similar to those of the Random Forest Classifier. Once again we have an accuracy of 88% for the same reasons as above. Recall remains just as high, which means that among bankrupt companies it correctly classifies bankrupt companies at 70%. But accuracy remains mediocre at 20%. In other words, it predicts too many bankrupt companies when in reality they are not.

2. Neural network methods

The models that follow are called artificial neural networks. Let's start by looking at how they work in general terms.



(Chauhan.A, 2021)

The following diagram shows a typical artificial neural network. First of all, our model will collect data via its input layer. It is responsible for receiving the input data and transmitting this information to the next layers of the network. Each node (or neuron) in the input layer represents a characteristic or dimension of the input data. We will not typically use an activation function in our input layer. (Chauhan.A, 2021)

$$\hat{y} = \sigma(w^T x + b)$$

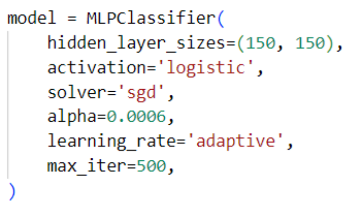
(Yatnish.V, 2022)

During the training process, the model initially performs forward propagation. In other words, it receives the input data and, through the various layers, adjusts its prediction until it obtains a final output prediction. Then, depending on whether or not the output is correct, the neural network will perform backward propagation in order to adjust its weights in order to improve predictions. The error between the predicted output and the actual output is calculated, and the weights and biases of each neuron are adjusted to reduce the error. The neural network updates its hyperparameters, the weights, wT, and biases, b, to satisfy the equation above. (Yatnish.V, 2022)

A loss function is a function that compares the target and predicted output values; measures how well the neural network models the training data. When training, we aim to minimize this loss between the predicted and target outputs. (Yatnish.V, 2022) The hyperparameters are adjusted to minimize the average loss — we find the weights, wT, and biases, b, that minimize the value of J (average loss).

2.1 Multilayer Perceptron Classifier

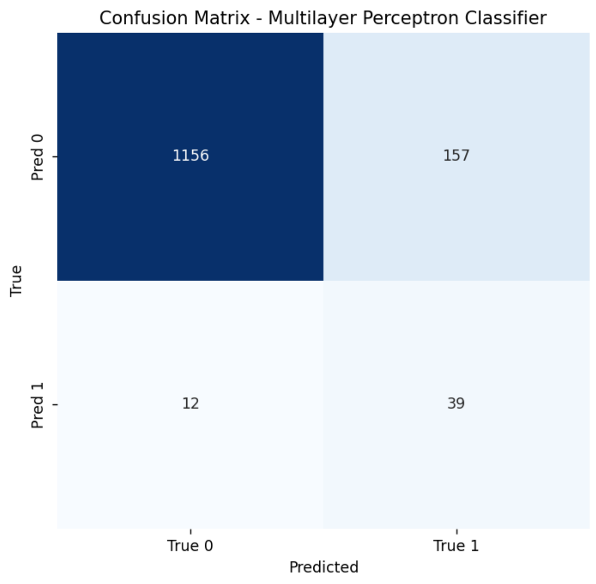
During the construction of my multilayer perceptron classifier I realised that I had obtained better results using an ADASYN balancing technique. As explained above, this technique will try to add points in areas where the minority class is not really represented.



```
model = MLPClassifier(
    hidden_layer_sizes=(150, 150),
    activation='logistic',
    solver='sgd',
    alpha=0.0006,
    learning_rate='adaptive',
    max_iter=500,
)
```

My neuronal model here is fairly simple. I have 2 hidden layers of 150 neurons each. After each training phase, the weights are updated using the 'sgd' (Stochastic Gradient Descent) solver. I also introduced a slight L2 penalty where the coefficient of certain variables is set to zero. Finally, the learning rate strategy is adaptive, which means that during the training process the model will update the weights while trying to minimise the error.

2.1.2 Results



```
Accuracy : 0.876099706744868
Precision :  0.1989795918367347
Recall :  0.7647058823529411
Cohen's Kappa :  0.27262744702066155
F1 :  0.3157894736842105
```

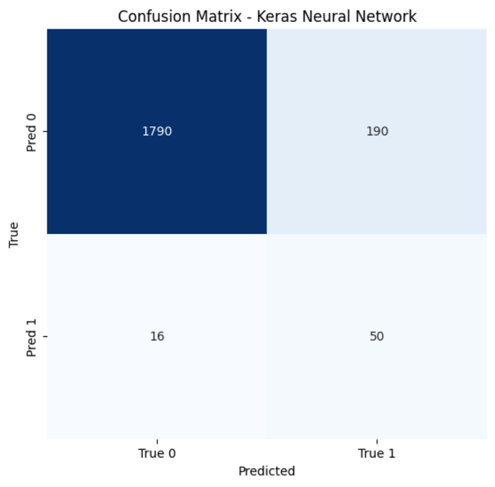
Once again, the results are very similar to the 2 previous models. I've tried several combinations of different parameters and it's this model that gives me the best results. There's still the problem of high recall and low accuracy. I tried to find a solution to reduce my false positives but without satisfactory results.

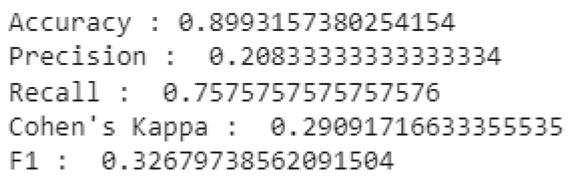
2.2 Neural Network in Keras

```
model = Sequential()
model.add(Dense(units=11, input_shape=(n_cols,), activation='relu', kernel_regularizer=l2(0.07)))
model.add(Dense(units=5, activation='relu',kernel_regularizer=l2(0.07)))
model.add(Dense(units=1, activation='sigmoid'))


model.compile(optimizer='adam', loss='binary_crossentropy', metrics='accuracy')


model.fit(X_train_scaled, y_train_resampled, epochs=10, batch_size=512, verbose=1)


y_pred = (model.predict(X_test_scaled) > 0.57).astype(int)
```

Initially I added more layers in order to train my model better. However, I very quickly realised that during the test phase all my predictions did not detect any bankrupt companies. I think that my overcomplicated model was overfitting and had difficulty generalising on new variables. That's why I decided to keep only one hidden layer. Given the large number of variables, I also applied an l2 penalty. This penalty means that the weight of certain variables will be close to 0, which makes the model lighter. I've also decided to set a probability threshold at which my company will go bankrupt or not. This helps me to manage my high number of false positives, i.e. to reduce the number of companies considered to be bankrupt when in fact they are not. I've also limited the number of epochs and increased the number of batch sizes to avoid overfitting with this model.

2.2.1 Results





```
Accuracy : 0.8993157380254154
Precision :  0.20833333333333334
Recall :  0.7575757575757576
Cohen's Kappa :  0.29091716633355535
F1 :  0.32679738562091504
```

Once again, I obtained results that were very similar to the previous ones. Note that the recall here is particularly high for a very low precision. We remain still with the same problem of too much false positives.

3. Discussions

For this project I analysed the models where I get the best results. In each model I tried to bring a new variant specific to each model in order to try to obtain different results. For the random forest model it's hyperparameter tuning and the development of the model with these. For logistic regression I introduced an unserpervised feature reduction method. As for my Artificial Neural Networks, it was mainly penalties with hyperparameter tuning.

Nevertheless, it's really interesting to note that the four models show really similar results. I'm going to put forward some hypotheses as to the causes and see how we could perhaps improve the models.

First of all, the problem may come from the lack of data for bankrupt companies. Basically, our database contains only 3% of bankrupt companies. It is possible that even using rebalancing methods, the data created is not truly representative of reality. This could influence our model to create too many false positives.

Then maybe I'd have to completely change the variable selection techniques. In my work I used an embedded method for the random forest and an unsupervised feature reduction because these are the methods that gave me the best results. I know that other more complex methods exist, but these are beyond my capabilities. It may be that the techniques I've used are too simple or inefficient with my variables.

Finally, I could try to revise the hyperparameters of my models or opt for other classification models. Specifically for my neural networks, I could try adjusting them or making them a little more complex. However, with several small changes I had the impression that my models were overfitting and classifying almost everything as a company in good financial health, which clearly shows a generalisation problem.

4. Conclusion

In conclusion, I note that it is not so easy to predict exactly when a company will fail. In fact, the 4 models used show very similar results. The biggest problem remains the management of false positives. I think it would be useful to update the database with more failing companies in order to be able to use the rebalancing techniques effectively. Finally, to improve the models it would be necessary to look for new methods of feature selection or reduction.

References

Bibliographie

Chauhan.A. (2021). *Step-by-Step Basic Understanding of Neural Networks with Keras in Python.* Retrieved from https://pub.towardsai.net/step-by-step-basic-understanding-of-neural-networks-with-keras-in-python-94f4afd026e5

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). *SMOTE: Synthetic Minority Over-sampling Technique.* Retrieved from https://arxiv.org/abs/1106.1813

Jaadi.Z. (2023). *When and Why to Standardize Your Data.* Retrieved from https://builtin.com/data-science/when-and-why-standardize-your-data

Mercckt.T, V. d. (n.d.). *Dimensionality Reduction & Parameter Estimation .* Univerisité Libre de Bruxelles.

Pilotti.G. (2020). *Oversampling and Undersampling: ADASYN vs ENN.* Retrieved from https://medium.com/quantyca/oversampling-and-undersampling-adasyn-vs-enn-60828a58db39

Provost.F, & Fawcett.T. (2013). *Data Science for Business.* Retrieved from https://www.researchgate.net/publication/256438799_Data_Science_for_Business

Rosidi.N. (2023). *Advanced Feature Selection Techniques for Machine Learning Models.* Retrieved from https://www.kdnuggets.com/2023/06/advanced-feature-selection-techniques-machine-learning-models.html

Yatnish.V. (2022). *Loss Functions and Their Use In Neural Networks.* Retrieved from https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9

Yiu.T. (2019). *Understanding Random Forest.* Retrieved from https://towardsdatascience.com/understanding-random-forest-58381e0602d2

Yu, L., Ye, J., & Liu, H. (2013). *Dimensionality reduction for data mining.* Retrieved from https://www.slideserve.com/jacob/dimensionality-reduction-for-data-mining-techniques-applications-and-trends