Deep Learning project: Bankruptcy Prediction

Ahmed Abdelazeem, Chung-Thing Huang, Omar Jarir, and Pedro Mora Gomes

***Index Terms*—Bankruptcy prediction, Deep Learning, Binary Cross Entropy, Focal Loss, Over Sampling, Down Sampling.**

I. Introduction

THE objective of this project is to build deep learning models to predict if a company is going to go bankrupt. The data set was extracted from Kaggle. It is composed of 6,819 rows and 96 variables (95 input features + 1 output feature) concerning companies' financial indicators/ratios from the Taiwan Economic Journal between 1999 and 2009. The Taiwan Stock Exchange business regulations were used to define company bankruptcy.

The observations in the data set include industrial and electronics companies (346), service companies (39), and others.

The data set is relatively clean and does not contains any missing values or duplicates.

II. Modeling

A. Preprocessing

The data set is split into train and test sets, where the size of the test set corresponds to 30% of the original data set size. Stratified sampling is used to ensure training and testing data set have the same distribution of non-bankruptcy vs. bankruptcy cases.

Preprocessing is needed to ensure good quality of the data. Using the train data set, the correlation coefficients are computed between the variables. Variables whose absolute correlation coefficient is greater than 0.9 are considered highly correlated. One of the variables is randomly dropped from both the train and test sets. This reduces the number of variables from 95 to 73.

Knowing that Random forests are efficient at detecting outliers in high dimensional data sets, an isolation forest algorithm is fit on the training data set, then used to remove outliers from the train and test sets.

To select the features to keep, we calculated the mutual information between the explanatory features and the target variable, and also performed an ANOVA F-test. Mutual information is an entropy based method that can find non-linear dependency, whereas ANOVA F-test can find linear dependency. The selected features are the union of features whose mutual information or F-statistics are above some predefined thresholds. This reduces the number of variables from 73 to 42. Features selection is of capital importance as it allows to avoid over-fitting.

Ahmed Abdelazeem, e-mail: m20210433@novaims.unl.pt.
Chung-Ting Huang , e-mail: m20210437@novaims.unl.pt.
Omar Jarir, e-mail: m20201378@novaims.unl.pt.
Pedro Moura Gomes, e-mail: m20200322@novaims.unl.pt.

To remove the potential bias caused by the range of the features and speed up convergence, the data was standardized by subtracting the mean to each feature and scaling it to unit variance.

B. Cross validation

The data set under study is small with very few samples of bankruptcy. To mitigate over-fitting, reduce the metrics variance and make sure our model is not biased by the validation data set, we use 3-fold cross validation with shuffling.

C. Hyper parameters

Using gradient descent or Newton optimization to optimize a function for which we don't have an analytic expression is impossible, Bayesian optimization allows to mitigate this issue as it allows to optimize a black box possibly non convex function.

The Bayesian optimization tuner from Keras Tuner was used to perform hyper-parameters tuning. The objective is to find the hyper-parameters that result in the best F-1 score on 3-fold validation data. The main hyper-parameters we will consider include:

1) Optimizer: We choose to stick with the Adam optimizer as it is known to provide good results, and tune the learning rate values between 10^{-5} to 10^{-3} using logarithmic sampling.

2) Loss function: Deep learning models' goal is to determine the adequate values of the weights and biases using the back-propagation algorithm, which involves finding the global minimum of an adequate loss function.

In the case of a binary classification problem, Binary cross entropy is often used as the loss function. It is formulated as:

$$CE\left(p_t\right) = -log\left(p_t\right)$$

Where $p_t = p$ if $y_{target} = 1$ and $p_t = 1 - p$ otherwise. However, when dealing with class imbalance the focal loss function is known to yield better results [8]. It is derived from the binary cross entropy loss function, by multiplying it by a modulating factor.

$$FL\left(p_t\right) = -\left(1 - p_t\right)^{\gamma} log\left(p_t\right)$$

When $\gamma = 0$ the focal loss coincides with the Binary Cross entropy loss function. The higher the values of γ the lower the loss of well classified observations. We used the tuner to select γ in $\{0.5, 2.0, 3.0\}$.

3) Dropout: We let the tuner decide if dropout should be applied at a given layer, and if so, what is the optimal dropout probability. This technique sets the weights of some inputs to 0 to regularize the network.

4) Class Imbalance: We use Keras Tuner to determine if it is necessary to perform down-sampling or over-sampling of classes on the training set. For oversampling we use Synthetic Minority Oversampling Technique (SMOTE) to generate more observations of the minority class. For down-sampling we use the NearMiss version 1 and 2 algorithms to drop some observations from the majority class.

D. Metrics

Due to the class imbalance issue (96.8% of companies did not go bankrupt) and the nature of the problem at hand using accuracy as an evaluation metric is irrelevant because a simple model can achieve about 97% accuracy but predicting that all the companies in the data set did not got bankrupt.

It is of vital importance to predict if a company would go bankrupt, because in this case actions must be taken, the most adequate metrics to assess the models are the Recall and the F1 score. We also kept track of the AUC, the precision, and the accuracy.

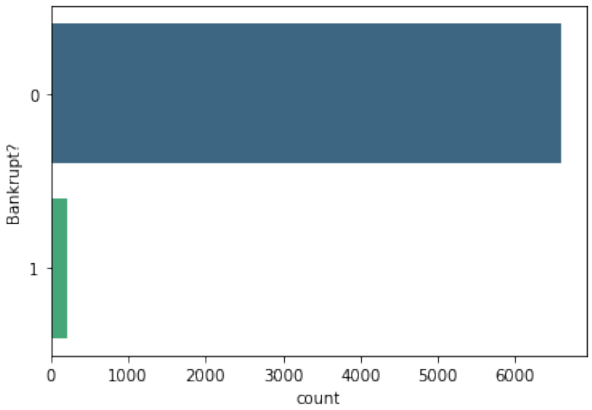

Fig. 1. Distribution of the target variable, on the original data set.

III. Results

A. Toy models

To get a sense of the number of hidden layers needed, we build four toy models. The first one is a sequence of 4 dense layers of size equal to $[64, 32, 16, 1]$ respectively, the second is composed of 3 dense layers of size $[64, 32, 1]$, the third model has 3 layers of size $[64, 16, 1]$, and the final one contains 4 layers of sizes $[32, 16, 4, 1]$.

The batch size is set to 16, and the number of epochs to 20. The figure 2 shows the confusion matrices of the models.

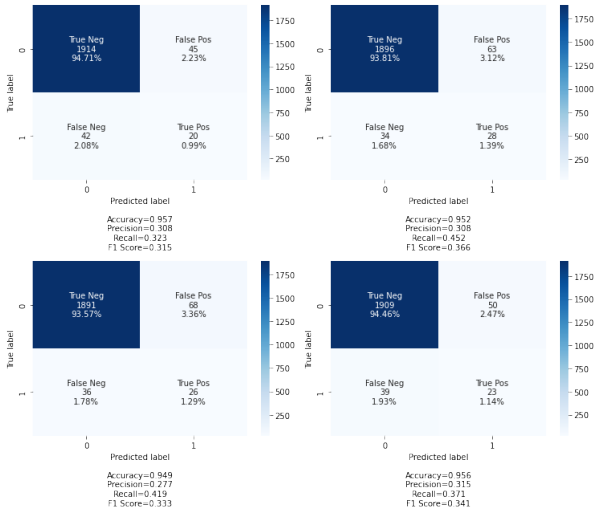

Fig. 2. Confusion matrices, of the four toy models.

The highest number of true positives is obtained by the second model, other models achieve similar results to the first one.

The figure 3 shows the ROC curve of our dummy models.

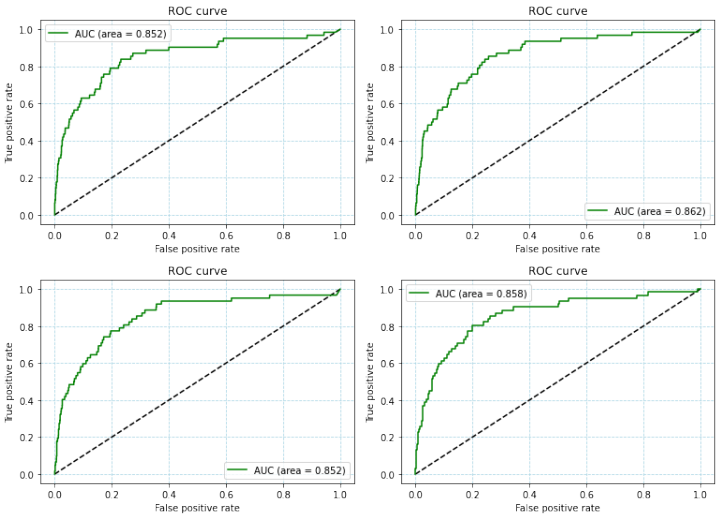

Fig. 3. ROC curves, of the four dummy models.

The second model achieve the highest value of AUC, while other models have close values.

Figure 4 displays the loss functions of the four models.

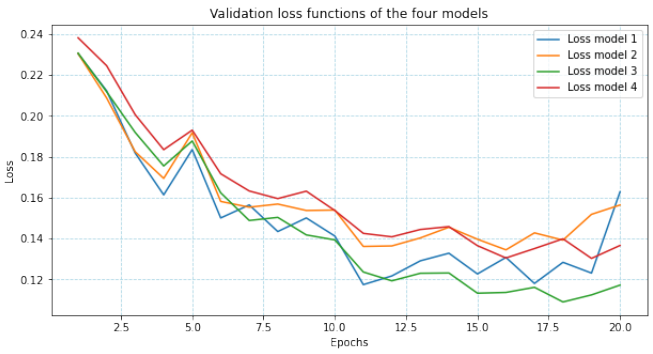

Fig. 4. Validation loss, of the four models.

Despite the second model $[64, 32, 1]$ achieves the best results, its loss function is not the best, model's one loss is the best till epoch 7, while the third model has the best loss after 12 epochs.

From all the above we conclude that a small network in terms of the number of layers seems to generalize well.

B. Tuned model

To tune a pre-selected set of hyper-parameters, a class (MyHyperModel) that inherits the Keras Tuner HyperModel class is implemented. The fit method was adapted to perform K-fold cross validation and over-sampling/under-sampling. It takes into consideration the weights attributed to both the minority and majority classes after over-sampling/under-sampling.

The table I summarizes the obtained hyper parameters using a Bayesian optimization search, with a number of max trials set to 30.

TABLE I
SET OF THE BEST HYPER PARAMETERS

First layer	64
First layer activation function	Softplus [1]
Second layer	32
Second layer activation function	tanh
Second layer dropout	True
Second layer dropout rate	0.2
Kernel initializer	"glorot_uniform"
Loss function	Focal loss
Gamma	3.0
Learning rate	10^{-5}
Batch size	16
Sample	over
Over sample rate	0.2
Shuffle	False

To assess the model constructed using the best set of hyper-parameters, we retrain the model using 5-fold cross validation and plot the averages of the metrics values on each epoch. Figure 5.

The Figure 5 shows that the metrics' values are improving with each epochs both on the train and validation sets.

Finally, we fit the model on the whole training data without considering any validation set and use a callback which is EarlyStopping which monitors the value of the F1-Score.

[1]The softplus activation function formula is as follow $softplus\,(x) = log\,(e^x + 1)$

The results are summarized in the figures below.

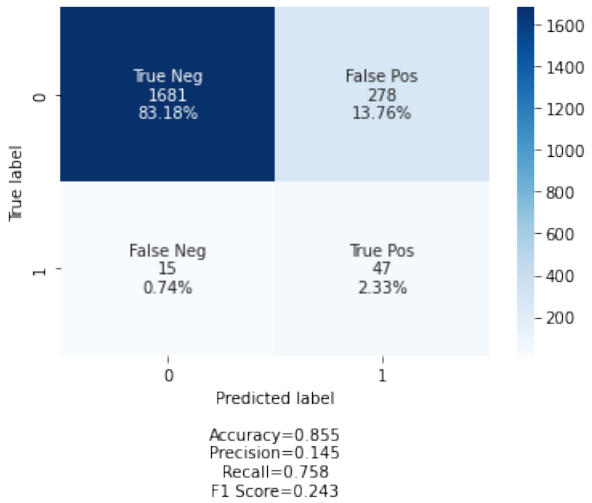

Fig. 6. Confusion matrix of the best model.

The confusion matrix 6 shows that the final model performs better than the toy models, the number of true positives is equal to 47, while the number of False negatives is equal to 15. The metrics displayed bellow the confusion matrix concern on only the bankrupt class.

The figure 7 summarizes the metrics obtained on the test set.

	precision	recall	f1-score	support
No Default	0.99	0.86	0.92	1959
Default	0.14	0.76	0.24	62
accuracy			0.86	2021
macro avg	0.57	0.81	0.58	2021
weighted avg	0.97	0.86	0.90	2021

Fig. 7. Classification report of the best model.

We can see that we get a satisfactory value of Recall for the default class.

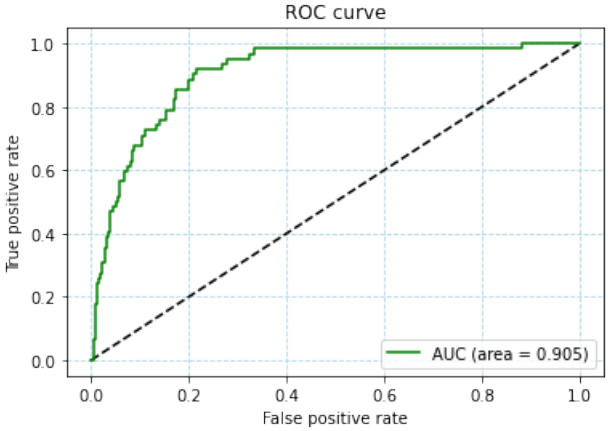

Fig. 8. ROC Curve of the best model.

The ROC curve 8 where we also indicated the AUC value.

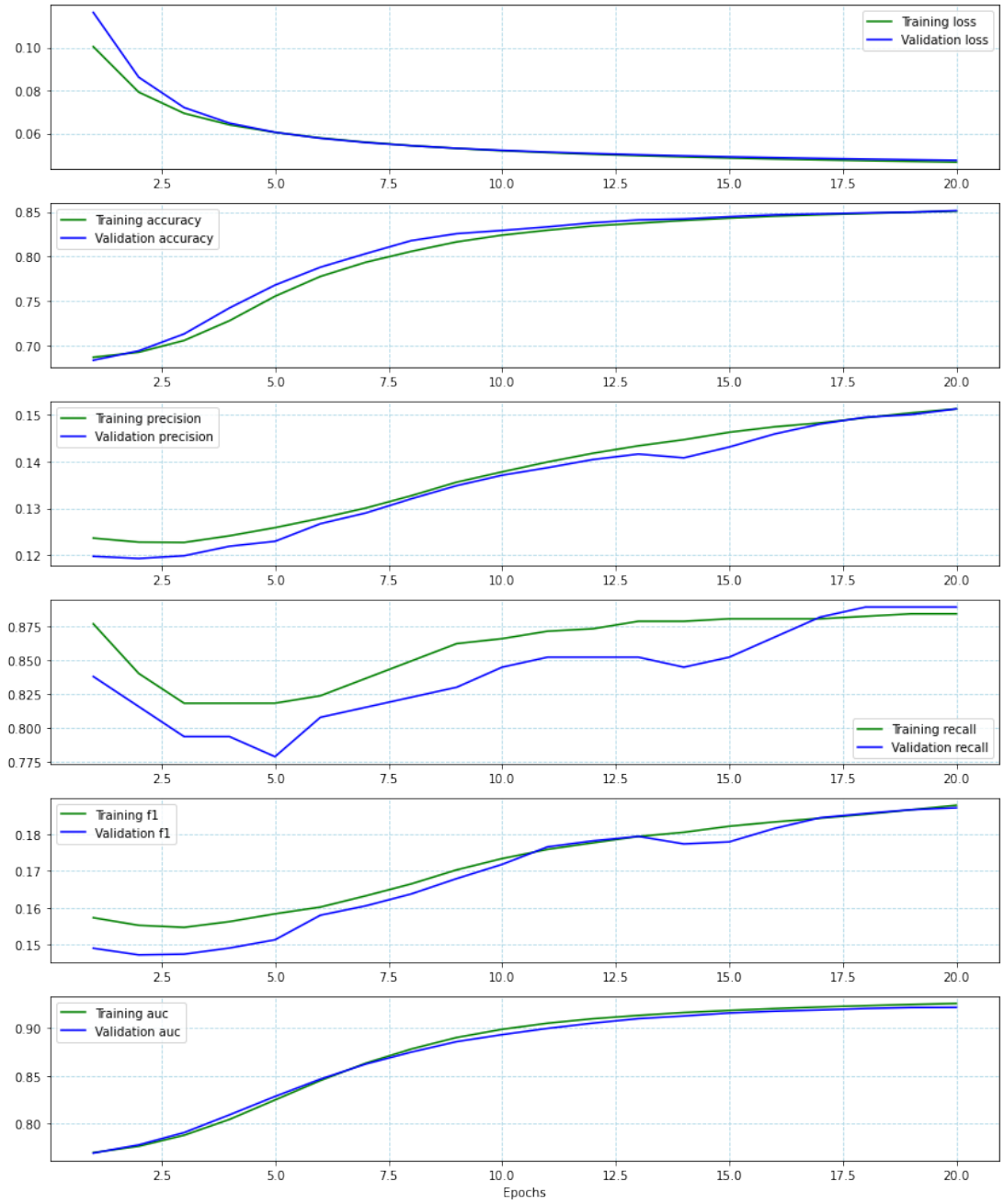

Fig. 5. Evaluation of the average of the metrics by epochs, the metrics are in the following order: Loss, Accuracy, Recall, F1-score, and AUC.

TABLE II
METRICS SUMMARY

	Train set	Test set
Accuracy	0.869409	0.855022
Precision	0.583082	0.567886
Recall	0.886391	0.808078
F1 Score	0.606760	0.581365
AUC	0.886391	0.808078

The table II shows that the final model does not over-fit, this is proven by the values of the macro metrics calculated on both the train and test sets. Taking into consideration the nature of the problem at hand we want a high value of Recall, the obtained results clearly satisfy this.

C. Comparing the Binary Cross Entropy and the Focal loss functions

We retrain two models using the set of the best hyper parameters but using two different loss functions, which are the binary cross entropy and the focal loss.

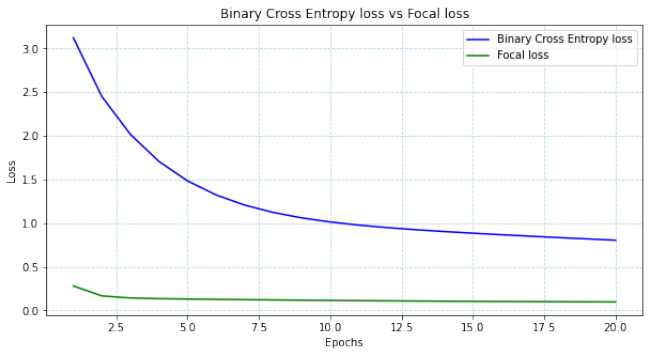

Fig. 9. Binary Cross Entropy vs Focal loss.

The figure 9 shows that the Focal loss function clearly outperforms the Binary Cross Entropy, emphasizing the fact that focal loss is more suitable for the class imbalance issue [8].

Code, data and models are available at: https://github.com/chungthuang/DeepLearning.

IV. Discussion

Since the data set was obtained from Kaggle, multiple classification attempts using sophisticated deep learning and machine learning models have been conducted.

We selected one approach to cross-check the added value of our model relatively to the available solutions.

The approach starts by normalizing the data and removing correlated columns, ending up with 59 features. After that, a Recursive Feature Elimination using a Logistic Regression is applied, the top 9 best ranking features are selected, using domain knowledge the notebook author decides to drop three additional features, he is left with only 6 features which are:

- ROA(C).
- Net Value Per Share (B).
- Debt ratio %.
- Total Asset Turnover.
- Cash/Total Assets.
- Equity to Liability.

The final models which are a decision tree classifier and a shallow neural network under-perform our model.

However, we decided to leverage the knowledge of the notebook author, and feed the selected features to our model. Using those features only our model has not yielded better results compared to the feature selection method used initially.

V. Conclusion

The objective of this project is to predict bankruptcy. The toy models give us insights about the network architecture, while Keras Tuner API allows us to find the optimal set of hyper-parameters, and prevent the tedious and time-consuming process of manually trying all the possible combinations.

The obtained results are satisfactory and show the importance of hyper-parameters tuning for deep learning models.

A potential improvement of the conducted work would be tuning the optimizer, and using a class balanced loss function [1], yet this implies a high cost of time and computational power.

References

[1] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. type: article.

[2] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings. ISSN: 1938-7228.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Adaptive computation and machine learning. The MIT press.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. type: article.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. type: article.

[6] Deron Liang, Chia-Chi Lu, Chih-Fong Tsai, and Guan-An Shih. Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study | elsevier enhanced reader.

[7] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr DollÃ¡r. Focal loss for dense object detection. 42(2):318–327. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[8] Roshan Nayak. Focal loss : A better alternative for cross-entropy.

Appendix

The figure Figure 10 explains the mutual information between the explanatory variable and the target variable. The figure Figure 11 displays the Anova F-value of each explanatory feature.

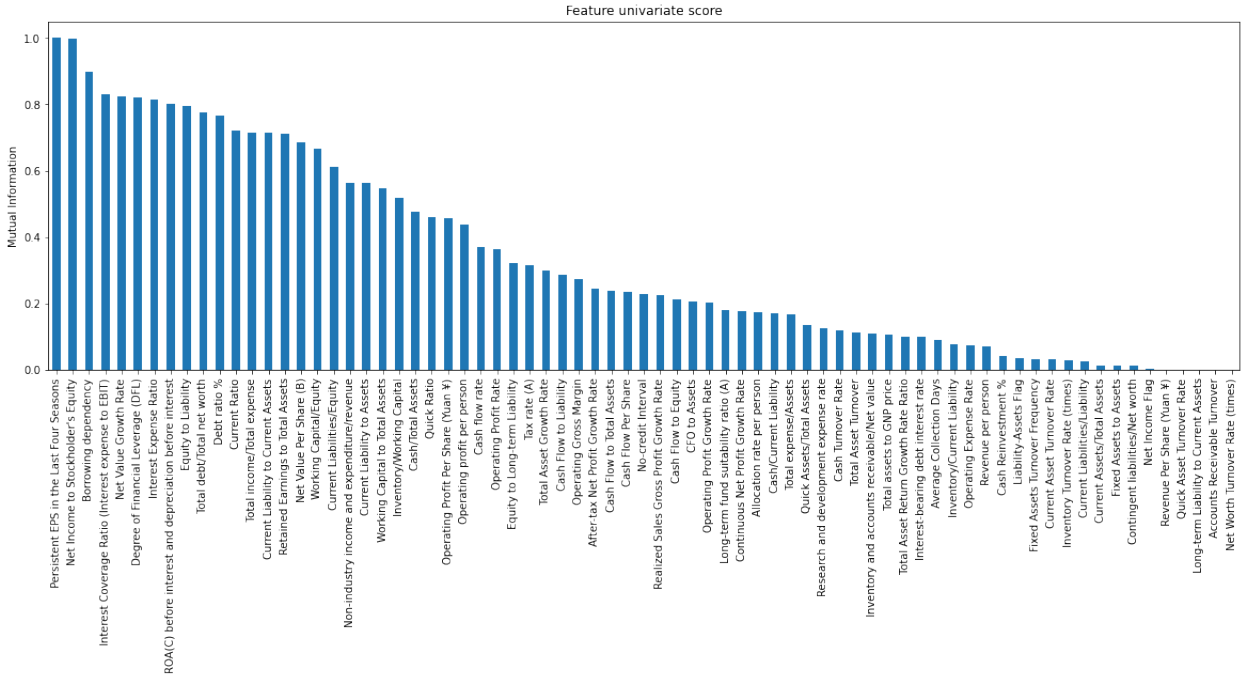

Fig. 10. Mutual information of explanatory features.

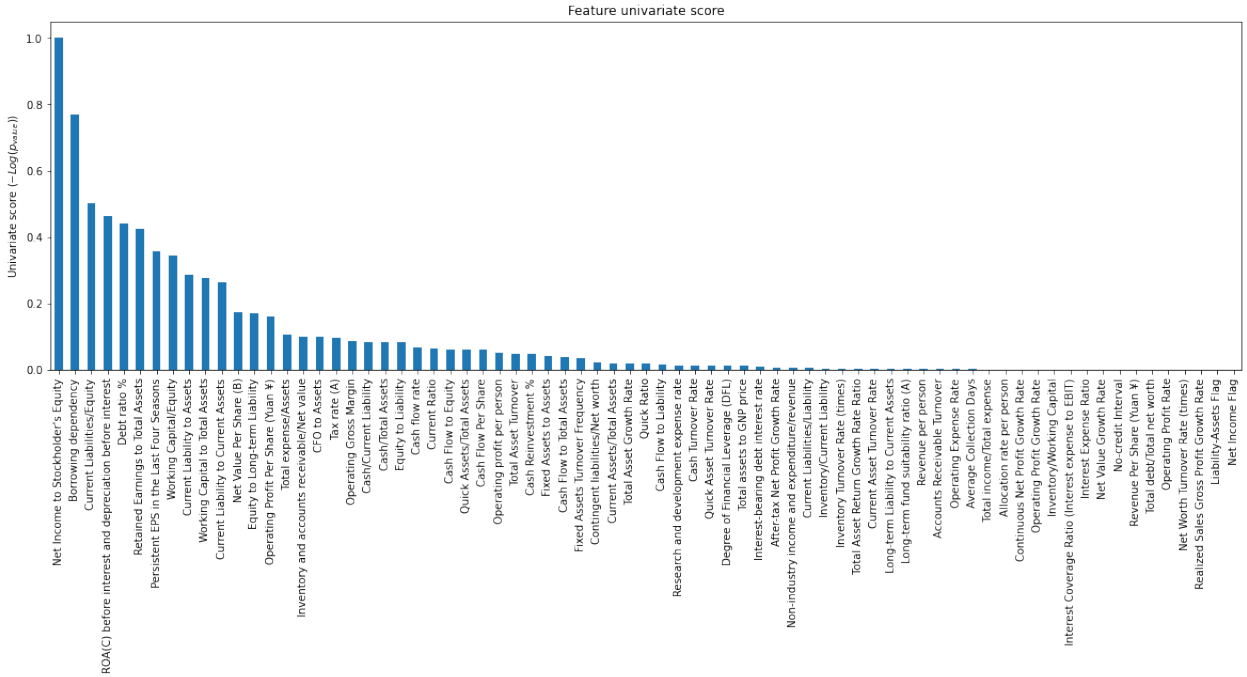

Fig. 11. Anova F-value of explanatory features.