Université Paris Dauphine - PSL

Master Ingénierie Economique et Financière

Forecasting stock market crisis events using deep and statistical machine learning techniques

Natanaël BINYEGUI KOUAMO
Ludovic DE VILLELONGUE
Amine MOUNAZIL
Emmanuel ZHENG

Gestion Quantitative

Feburary 2022

Abstract

The objective of this paper is to provide a critical analysis of the article: "Forecasting stock market crisis events using deep and statistical machine learning techniques" written by Sotirios P. Chatzis, Vassilis Siakoulis, Anastasios Petropoulos, Evangelos Stavroulakis and Nikos Vlachogiannakis and published in the journal Expert Systems with Applications in 2018.

The first part will be devoted to the notions and concepts used in the article, to the results of the latter as well as highlighting its main limitations. Then we will reproduce the results of the article. This reproduction has a double objective: to implement the methodological framework proposed by the authors and to ensure the reproducibility of such a method.

Outline

Introduction

In recent years, the world has experienced numerous financial crises. These crises started locally in specific regions and then spread to a global scale. The current goal of financial actors is to anticipate as much as possible the occurrence of a possible crisis in order to take adequate measures very quickly. The current technological evolution allows us to store huge amounts of historical data. This data coupled with the use of modern forecasting tools can allow us to anticipate these possible crises. Our work will therefore consist in applying statistical methods of deep and machine learning on a series of historical data in order to establish the best possible prediction.

Introduction

In order to collect data, we decided to create an interactive scraper in python to retrieve live information from the API of Investing.com. It can be useful to perform several analyses and change the datasets whenever needed. However, it is a constraint to retrieve data from long periods, which is why all the collected data in this project are beginning in 2018 and ending in 2021. The fact that the Covid pandemic took place during this period was appropriate to identify crisis events. To compensate the possible lack of data on the period considered, the percentile of the empirical distribution linked to the negative co-exceedances was increased from 1% to 5% and 10%. The threshold of 5% corresponds to variables with a score of 3 or plus and the 10% to variables with a score of 4 over 4.

Introduction

The forecasting models used in this project are the following:

- Elastic net penalized logistic regression
- Optimal penalized logistic regression
- Broyden–Fletcher–Goldfarb–Shanno Logistic Regression
- Maximal deicision tree
- Optimal decision tree
- Maximal Random Forest
- Optimal Random Forest
- Optimal SVM
- Neural Network
- XGBoost
- MLP neural networks

To compare those models, two metrics files are collected, the matrix confusion for each one

Crisis and systemic risk

A systemic risk is the risk that many market participants are simultaneously affected by severe losses, which spread through the system. It can be triggered by systemic risk-taking, contagion and amplification mechanisms.

A crisis can be represented as a systemic loop, gathering several sources of risk. In the figure, each edge represents a risk transmission channel, whose strength is given by the label on the edge. The sensitivity of a j defaults to the system wide loss is measured by α_j^s while j's contribution to system-wide losses depends on y_j^s.

Crisis and systemic risk

Measures of systemic risk include:

- SRisk
- CoVaR
- ΔCoVar
- Marginal Expected Shortfall
- Multivariate GARCH

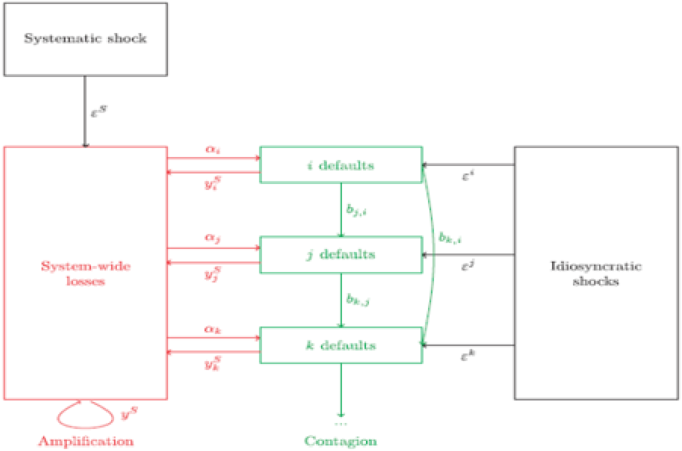
Crisis and systemic risk

Many methodologies are now available to identify different sources of systemic risk and will produce new regulatory or policy tools. What is less clear however is how to link the measures produced by these tools to regulatory interventions.

The quest for a global risk measure that encompasses different sources of systemic risk and yet produces a single/simple metric that can directly be used for regulation (tax or capital surcharge but risk thermometer as well) is still ongoing. Such an approach could align systemic risk banks' interests with social optimum.

Crisis and systemic risk

Figure



Statistical Machine Learning Data Treatment

Data Preprocessing

Some columns of a dataset can contain a large number of missing values, which make them difficult to use for prediction. Missing values are an issue as they can't be processed by ML algorithms. A possible solution to handle missing values is to input a replacement value like the mean or median of the feature.

Statistical Machine Learning Data Treatment

Training and Test Set

Training a deep/machine learning model amounts to finding the model parameter values that allow to generate the predictions that best match the actual labels. The best in-sample predictions is obtained by training the model on the whole dataset.

An increase in the size of a dataset increases the accuracy of the prediction. Yet, there is no guarantee on how general the model is and how it will perform out-of-sample, that is on a new dataset, using data the model has never seen before.

Statistical Machine Learning Data Treatment

Training and Test Set

For that reason, in every ML project, the original dataset is always split into 2 sets at least:

- A training set: the dataset on which training is performed and models are fine-tuned
- A testing set: the dataset containing unseen data, which is used to assess the performance of the final model

Statistical Machine Learning Data Treatment

Feature Scaling

Machine Learning algorithms do not perform well in general when features greatly differ in magnitude or are expressed using different scales. Two standard procedures allow to transform features so that they get all expressed in the same scale:

- Min-max scaling or normalization
  $x_j^{(i)} = \frac{x^{(i)} - min(x_j)}{(x_j) - min(x_j)} => x_j^{(i)} \in [0, 1]$

- Standardization
  $x_j'^{(i)} = \frac{x^{(i)} - \overline{x_j}}{(x_j) - min(x_j)} => x_j'^{(i)} = 0, (x_j') = 1$

Statistical Machine Learning Data Treatment

Training and Test Set

When scaling is performed, the scaling parameters must be computed using the train set only to avoid any look-ahead bias. Scaling is thus performed in three steps:

- compute the min and max of each feature in the train set
- perform feature scaling on the train set
- perform feature scaling on the test set using values found when computing the min and max

The scaling is applied to features only, not the labels.

Statistical Machine Learning Data Treatment

Data balancing

The issue with machine learning algorithm for classification in the case of crisis prediction is the fact that there are more events when there is no crash. Training the machine learning model on an unbalanced dataset result in the fact that parameters will be biased towards predicting the majority class (no crash). A dataset with an equal proportion of crash and no crash events will avoid this issue.

Statistical Machine Learning Data Treatment

Data balancing

Dataset balancing can be performed in two ways:

- Undersampling: randomly remove no crash events in order to get the same 50% proportion of crash and no crash events
- Oversampling:
  - Naive: ramdomly duplicate examples in the minority class
  - SMOTE (Synthetic Minority Oversampling Technique): generate artificial examples based on k-nearest neighbors from instances in the minority class.

It is possible to combine under and over sampling.

Types of classification algorithms

A classification algorithm takes as input a new set of features and tell whether it belongs to a class or not. The class here is modelized by a binary indicator, constituted of two predicted/dependent variables. The indicator takes the value of 1 in case there is a global stock market crash event in the next day, or in the next 20 days, respectively, and take the value of zero otherwise.

Those algorithms are relying on supervised learning : $x \in \mathrm{R}^{\mathrm{p}} \rightarrow y = f_\theta(x) \in 0, 1,$ to distinghuisg from regression $x \in \mathrm{R}^p \rightarrow y = f_\theta(x) \in \mathrm{R}$.

Types of classification algorithms

Standard and Stochastic Gradient Descent Logistic Regression

The first task consists in retaining variables yielding a score equal to 4. on each step. Once it is done, variables for which the statistical significance test yields p-values more than 15% are dropped. The model is then refitted on the remainder variables.

Types of classification algorithms

Standard Logistic Regression

It is possible to estimate logistic regression parameters in a standard way, without running Gradient Descent or LBFGS methods. In that case, the generalized linear models method is used. It performs model fitting through Iteratively Reweighed Least Squares.

Types of classification algorithms

Logistic Cost Function

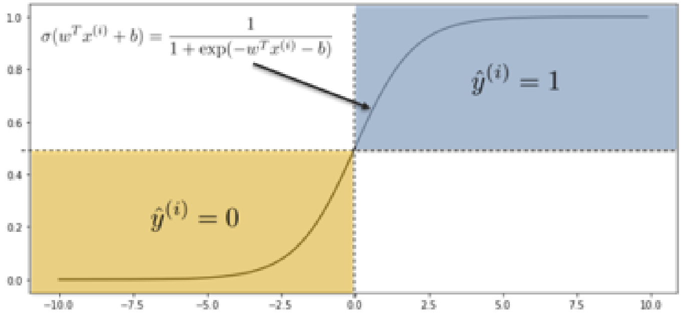
The prediction of a logistic regression will be given by the logistic function. With the logistic function, conditional probabilities are computed as follows :
$\hat{p}(y^{(i)} = 1|w, b, x^{(i)}) = \sigma(w^T x^{(i)} + b)$ $\hat{p}(y^{(i)} = 0|w, b, x^{(i)}) = 1 - \sigma(w^T x^{(i)} + b)$
where :

- $x^{(i)}$ is a column vector with n component
- w is a column vector of n weights (coefficients)
- b is a bias term (intercept / constant)
- σ is the logistic function, $\sigma(t) = \frac{1}{1+\exp(-t)}$

Types of classification algorithms

Figure

Replacing the probability prediction by its expression as computed using the logistic function gives the following:



Batch and Mini Batch Gradient Descent

Stochastic Gradient Descent

SGD logistic regression is performed using the SGDClassifier estimator. The SGDClassifier estimator can implement several ML models, depending on the type of loss that is used. To perform logistic regression, the loss value must be set to 'log'. To avoid regularization, the penalty parameter must be set to 'none'. Finally, SGDClassifier implements a learning rate schedule. It is possible to use a constant rate using learning rate = 'constant'. In that case, the earning rate value must be specified (eta0 = 0.01).

Batch and Mini Batch Gradient Descent

Broyden–Fletcher–Goldfarb–Shanno algorithm

L-BFGS is the pinnacle of second order methods. It is an algorithm of choice for logistic regression. Its characteristics are the following:

- Low memory overhead
- Very fast convergence (quadratic convergence)
- No stochastic version

Batch and Mini Batch Gradient Descent

Penalized Logistic Regression

It is impossible to build a model with a good quality of fit on the train dataset that also generalizes well on new data due to a mandatory tradeoff between bias and variance. High-variance models are able to represent their training set well but are more prone to overfitting as they are more likely to account for noisy or unrepresentative training data, called outliers. High-bias models fail to capture important regularities in the data because they are prone to underfitting. Yet, they can generate predictions with a lower variance when they are applied to unseen data.

Batch and Mini Batch Gradient Descent

Penalized Logistic Regression

A potential issue is that some parameters are given excessive weights because the model attempts to fit details of the training set and accounts for features that are noisy factors which accidentally correlate with the labels. To avoid the resulting overfitting, it is possible to add regularization terms aimed at penalizing large weights. It is possible to combine linearly the L1 and the L2 regularization, which is called the Elastic Net regularization.

There exist two types of regularizations:

- Ridge or L2 regularization that aims at keeping parameters as small as possible
- Lasso (Least Absolute Shrinkage and Selection Operator) or L1 regularization which narrows down the number of non-zero parameters

Batch and Mini Batch Gradient Descent

Penalized Logistic Regression

With L2 regularization only, the cost function expresses :

$$J_{L2}(w,b) = J(w,b) + \lambda \sum_{k=1}^{p} \theta_k^2$$

Batch and Mini Batch Gradient Descent

Penalized Logistic Regression

With L1 regularization only, the cost function expresses :

$$J_{L1}(w, b) = J(w, b) + \lambda \sum_{k=1}^{p} |\theta_k|$$

Batch and Mini Batch Gradient Descent

Penalized Logistic Regression

With Elastic Net, the cost function expresses :

$$J_{EN}(w, b) = J(w, b) + \alpha \sum_{k=1}^{p} [(1 - \lambda)\theta_k^2 + \lambda|\theta_k|]$$

Types of classification algorithms

Decision Tree

A decision tree can be used for classifications tasks. The principle is to separate progressively a set of training data using a top-down approach known as the CART (Classification And Regression Trees) algorithm. Using first the whole dataset, the algorithm creates two subsets using 1 feature k and a threshold t_k. The (k, t_k) pair is the one that generates the purest subsets (weighted by their size).

Types of classification algorithms

Decision Tree

The algorithm seeks to minimize the following cost function:

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

where $G_{left/right}$ measures te impurity of the left/right subset and $m_{left/right}$ is the number of instances in the left/right subset. Once the original dataset has been split in two subsets, the resulting subsets are split using the same procedure. The algorithm stops once it has reached the maximum depth set by the modeler (or another stopping criterion).

Types of classification algorithms

Decision Tree

Impurity is computed using either Gini impurity or entropy :

- Gini impurity : $G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$ where $p_{i,k}$ is the ratio of class k instances amoung the training instances in the i^{th} node
- Entropy : $H_i = \sum_{k=1}^{n} p_{i,k} log_2(p_{i,k})$ for $p_{i,k} \neq 0$

Types of classification algorithms

Decision Tree

The CART algorithm is a greedy algorithm: it searches the best split at the top level and proceeds in the same way for each level. Using a different split at the top (and at each depth) could result in better data partitioning and better classification. Therefore, there is no guarantee that the solution found by CART is optimal It is possible to control the behavior of the algorithm through various hyperparameters:

- max_depth: the maximum depth of the tree
- min_sample_split: the minimum number of instances requires to split an internal node
- min_samples_leaf: the minimum number of instances required to be at a leaf node

Types of classification algorithms

Decision Tree

Decision trees enjoy the advantage of following a set of simple rules, resulting in a fast inference. It also easy to apprehend why the model made a prediction, thanks to its visualization characteristics. On the downside, they are prone to overfitting and a selection bias towards covariates with many possible splits. In fact, fitting complicated datasets might require very deep trees.

Types of classification algorithms

Random Forest

Instead of training a single decision tree, training several ones is also a method to consider. Thanks to an aggregation of the predictions of all individual decision trees, the class that gets the highest number of votes is defined as the predicted class. Very often, a voting classifier will generate better results than the best classifier in the ensemble. To train different decision trees, random forests generate several decision trees by randomly sampling a subset of the initial features for each decision tree. If sampling is performed with replacement, the method is called bagging. It is called pasting when sampling is performed without replacement. Besides randomly sampling the features, it is possible also to randomly sample the instances.

Types of classification algorithms

Random Forest

Random forests are usually robust to overfitting, since each forest is only presented with a subset of all the available features. On the other hand, the aforementioned bagging process facilitates strong generalization capacity. By construction, the predictive ability of RFs increases as the inter-tree correlation decreases. Thus, a large number of predictors can provide increased generalization capacity, which is the case with our model.

Types of classification algorithms

Random Forest

To predict features importance, the mean squared error ranking was used as a first method. MSE is normally used for regression algorithms. Technically it can be used for classification, but the MSE function is non-convex for binary classification. Thus, if a binary classification model is trained with MSE Cost function, it is not guaranteed to minimize the Cost function. Also, using MSE as a cost function assumes the Gaussian distribution which is not the case for binary classification. A permutation should be used with a classification scoring parameter. As for the node impurity, calculated as the difference between the Residual Sum of Squares (RSS), it should be replaced by the Gini Importance or Mean Decrease in Impurity (MDI).

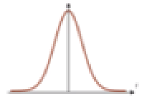
Types of classification algorithms

Support Vector Machine

SVM reduce the possibility of overfitting and alleviate the need of tedious cross-validation for the purpose of appropriate hyper-parameter selection. The main drawbacks of SVMs stem from the fact that they constitute black-box models, thus limiting their potential of offering deeper intuition and visualization of the obtained results and inference procedure. Evaluate soft-margin SVM classifiers using linear, radial basis function (RBF), polynomial, and sigmoid kernels, and retain the model configuration yielding optimal performance.

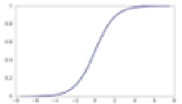
Types of classification algorithms

Support Vector Machine



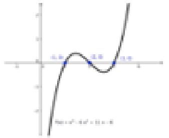
Gaussian RBF Kernel

$$K(\vec{x},\vec{l}^{i}) = e^{-\frac{\|\vec{x}-\vec{l}^{i}\|^2}{2\sigma^2}}$$



Sigmoid Kernel

$$K(X,Y) = \tanh(\gamma \cdot X^T Y + r)$$

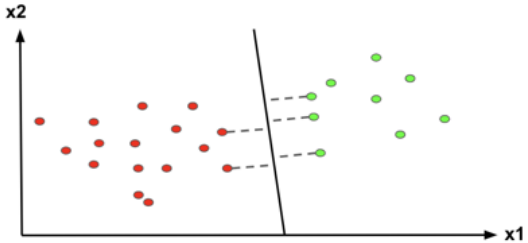


Polynomial Kernel

$$K(X,Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0$$

Types of classification algorithms

Support Vector Machine

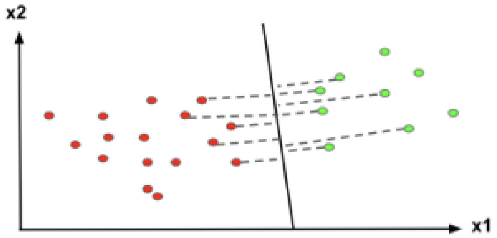


High Gamma

- only near points are considered.

Types of classification algorithms

Support Vector Machine



Low Gamma

- far away points are also considered

Types of classification algorithms

Support Vector Machine

Select the hyperparameters of the evaluated kernels, as well as the cost hyperparameter, C, of the SVM (related to the adopted soft margin), we also resort to cross-validation. The candidate values of these hyperparameters are selected based on a grid-search algorithm

Types of classification algorithms

Support Vector Machine

Grid Search is good when we work with a small number of hyperparameters. However, if the number of parameters to consider is particularly high and the magnitudes of influence are imbalanced, the better choice is to use the Random Search.

Types of classification algorithms

Support Vector Machine

As we observe, a large cost parameter gives low bias, as it penalizes the cost of misclassification a lot. However, it leads to high variance, so that the algorithm is forced to explain the fitting data stricter, and potentially overfit. On the other hand, a small misclassification cost allows more bias and lower variance. Regarding , when it is very small the model is too constrained and cannot capture the complexity of the data. In this case, two points can be classified the same, even if they are far from each other. On the other hand, a large allows two points to be classified the same, only if they are close to each other.

Types of classification algorithms

Neural Networks

Neural networks are a powerful way to go beyond linear models by providing a sequence of layers. One layer maps p inputs $(x1, ..., x_p)$ to q outputs $(y_1, ..., y_p)$.

Types of classification algorithms

Neural Networks

Parameters:

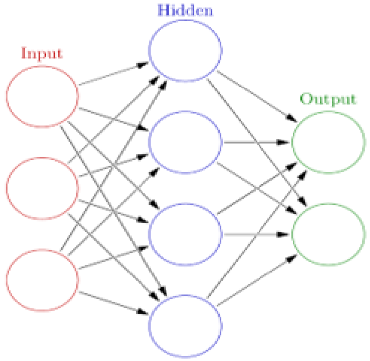
- Weights of p coefficients $w, \quad i = 1 \dots q$
- Biases b^i
- Non linearity $\sigma(e.g \;\; \sigma = \tanh, ...)$

$$y_i = \sigma(\mathrm{w}_1^i * x_1 + ... + w_p^i * x_p + b^i)$$

Types of classification algorithms

Neural Networks

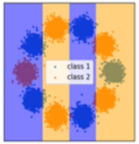
Layers are stacked. A neural network with one hidden layer is a universal approximator. With enough hidden units, it can approximate any transform. The role of adding layers is to learn more complicated functions with fewer hidden units.



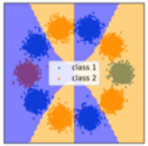
Types of classification algorithms

Neural Networks

1 layer, 2 hidden units



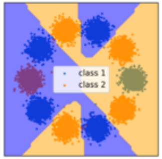
1 layer, 3 hidden units



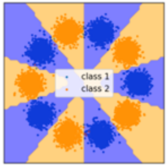
Types of classification algorithms

Neural Networks

1 layer, 4 hidden units

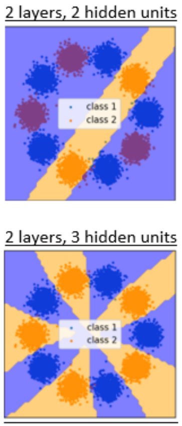


1 layer, 5 hidden units



Types of classification algorithms

Neural Networks



Neural nets have lots of parameters, so you need many samples to train them, otherwise they

Types of classification algorithms

XGBoost

XGBoost stands for "Extreme Gradient Boosting", where the term "Gradient Boosting" originates from the paper Greedy Function Approximation: A Gradient Boosting Machine, by Friedman. XGBoost is used for supervised learning problems, where we use the training data (with multiple features) to predict a target variable. Example of supervised learning tasks are Regression, Classification, and Ranking. It is built on the principles of gradient boosting framework and designed to "push the extreme of the computation limits of machines to provide a scalable, portable and accurate library. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest.

Types of classification algorithms

MLP Neural Network

A feedforward artificial neural network called a multilayer perceptron (MLP) is a type of feedforward artificial neural network (ANN). The name MLP is ambiguous; it can be used to refer to any feedforward ANN, or it can refer to networks made up of many layers of perceptrons (with threshold activation). Multilayer perceptrons, especially those with a single hidden layer, are commonly referred to as "vanilla" neural networks. There are at least three levels of nodes in an MLP: an input layer, a hidden layer, and an output layer. Each node, with the exception of the input nodes, is a neuron with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training.

Types of classification algorithms

MLP Neural Network

A perceptron is a linear classifier; that is, it is an algorithm that classifies input by separating two categories with a straight line. Input is typically a feature vector x multiplied by weights w and added to a bias $b : y = w * x + b$
A perceptron produces a single output based on several real-valued inputs by forming a linear combination using its input weights (and sometimes passing the output through a nonlinear activation function). Here's how you can write that in math: where w denotes the vector of weights, x is the vector of inputs, b is the bias and ϕ is the non-linear activation function.

Model's evaluation

Validation Measures

For each instance i, the model has to return the estimated probability $\hat{p}^{(i)}$ that there is a crash event conditional on the corresponding vector of features $x^{(i)}$.

Model's evaluation

Validation Measures

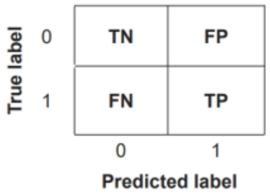
The prediction rule from a classification algorithm is the following:

$$y^{(i)} = \{\mathbf{0} \ if\, \hat{p}^{(i)} < 0.5; \mathbf{1} if \ \hat{p}^{(i)} \geq 0.5\}$$

Model's evaluation

Confusion Matrix

The accuracy score gives only a partial view of the classifier ability to identify crash and non-crash events. To go further into the analysis, it is recommended to compute the confusion matrix of the predictions. The confusion matrix is a table with two rows and two columns (since we have 2 classes) that reports the number of true positives, true negatives, false positives and false negatives.



Model's evaluation

Confusion Matrix

Based on the confusion matrix, it is possible to compute various scores that help understanding how the classifier behaves:

- Accuracy: fraction of correctly classified periods

  $$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

- Precision: fraction of correct predictions when the classifier identifies an instance as a crash event

  $$Precision = \frac{TP}{TP+FP}$$

- Recall (sensitivity/true positive rate): percentage of crash events the classifier is able to identify

  $$Recall = \frac{TP}{TP+FN}$$

Model's evaluation

Confusion Matrix

Based on the confusion matrix, it is possible to compute various scores that help understanding how the classifier behaves:

- F1 Score: harmonic mean of precision and recall

  $F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$

- Specificity: fraction of no crash event identified among all no crash events

  $Specificity = \frac{TN}{TN+FP}$

- False Positive Rate: fraction of no crash events that are incorrectly classified as crash events

  $FPR = \frac{FP}{FP+TN}$

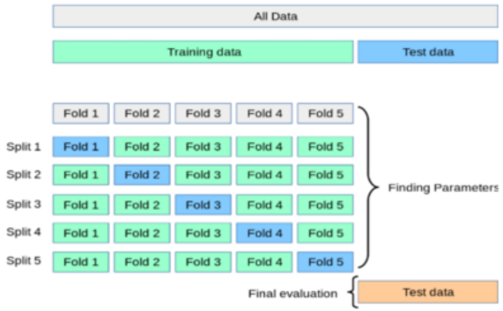
Model's evaluation

Cross-Validation

The scores computed are obtained using the train dataset. There is no guarantee however that these scores will be the same on the test dataset and in real conditions. In order not to use the test set at this stage and explore alternatives, the k-fold cross validation technique can be useful. In k-fold cross-validation, the training set is (randomly) split into k smaller sets. Then the following procedure is applied for each of the k folds. The model is trained using k 1 of the folds as training data. Then the resulting model is validated on the remaining part of the data, used as a pre test set to compute performance score. 5 folds are used in our case.

Model's evaluation

Cross-Validation



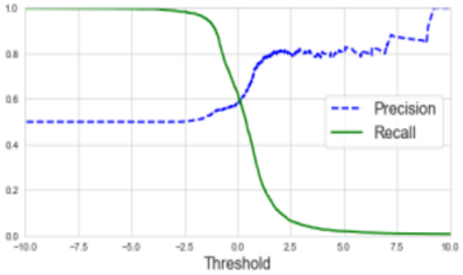
Model's evaluation

ROC Curve and Area under curve

The classifier assigns an instance to one of the two classes based on a decision function. For each instance the classifier computes a decision score : $S^{(i)} = w^T x^{(i)} + b$
Denoting T a threshold value, $S^{(i)} \geq T (S^{(i)} < T)$, the period i is classified as a crash event (no crash event). By default, the threshold value is set to 0. Varying the threshold value impacts the way the instances are classified, and therefore precision and recall scores. It is possible to get the precision/recall tradeoff of the classifier by computing the value of the 2 metrics for different threshold levels.

Model's evaluation

ROC Curve and Area under curve



Model's evaluation

ROC Curve and Area under curve

The recall always decreases with the threshold whereas the evolution of the precision is non-monotonic.

The Receiver Operating Characteristic (ROC) curve is a plot that displays the diagnostic ability of a binary classifier when the threshold is varied. It is created by plotting the true positive rate (TPR) or Recall against the false positive rate (FPR).

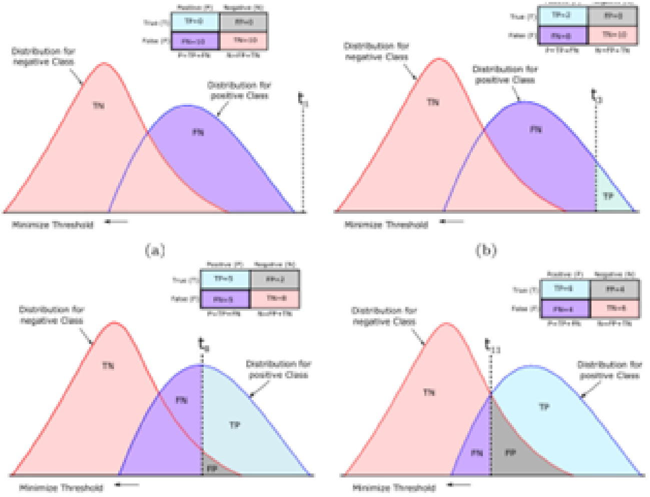
The ROC curve can also be thought of as a plot of the power (probability of detecting a true positive) against the Type I error (rejection of the fact that an event is a no crash event although it is). Very often the area under the ROC Curve or AUC is used to compare models and to select the best one (the one with the highest AUC)

Model's evaluation

ROC Curve and Area under curve

TP = 0	FP=0
FN = 10	TN=10

$$TPR = 0$$
$$FPR = 0$$



Distribution for negative Class
Distribution for positive Class
TN
FN
TP
FP
Minimize Threshold
(a)
(b)

TP = 2	FP=0
FN = 8	TN=10

$$TPR = \frac{2}{2+8} = 0.2$$
$$FPR = 0$$

TP = 5	FP=2
FN = 5	TN=8

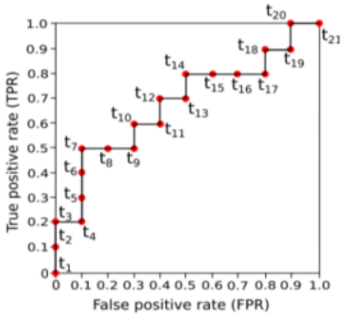
$$TPR = \frac{5}{5+5} = 0.5$$
$$FPR = \frac{2}{2+8} = 0.2$$

TP = 6	FP=4
FN = 4	TN=6

$$TPR = \frac{6}{6+4} = 0.6$$
$$FPR = \frac{4}{4+6} = 0.4$$

Model's evaluation

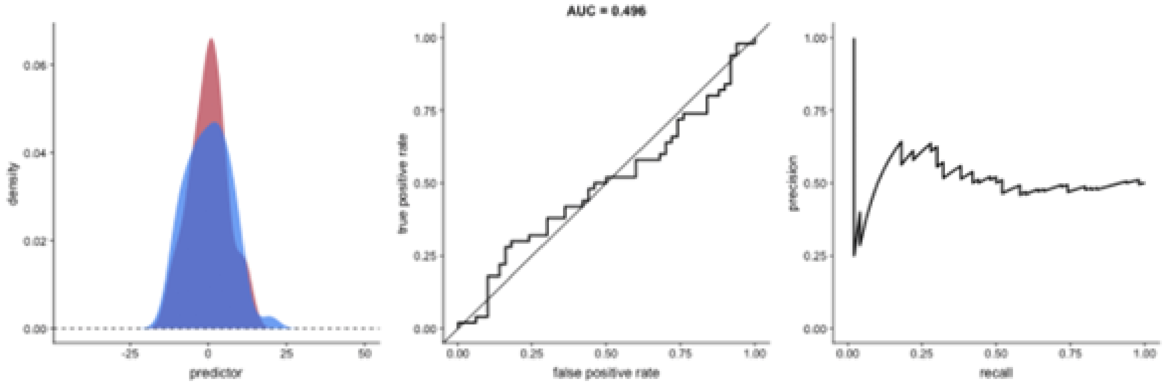
ROC Curve and Area under curve



The ROC AUC indicates how well the classifier separates the probabilities from the positive classes to those from the negative classes.

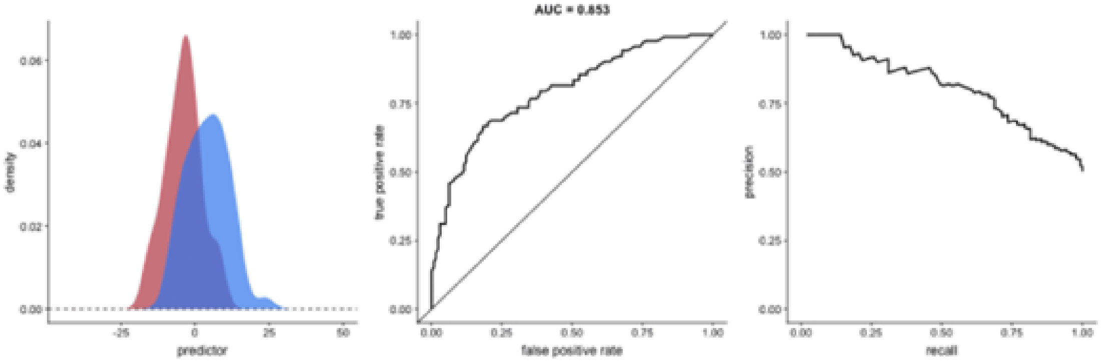
Model's evaluation

ROC Curve and Area under curve



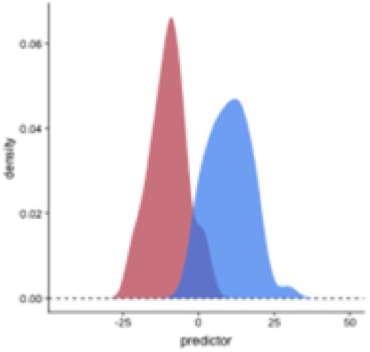
Model's evaluation

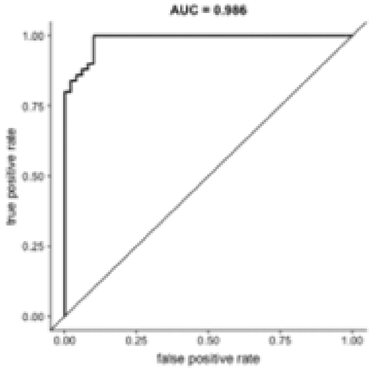
ROC Curve and Area under curve

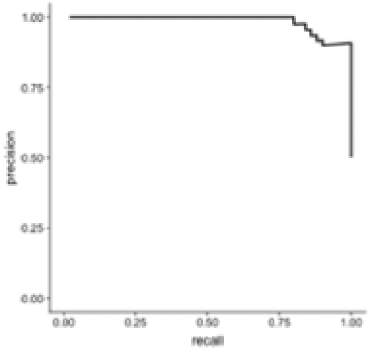


Model's evaluation

ROC Curve and Area under curve







Observations for each model

Models

We ran our python code and models for a set of assets and will analyze our results. Given that we had data restriction with our free API keys, we were only able to collect short term data. Our IP address would get flagged for the huge amount of request per second our code made to feed the models with data. Given this, we noticed that we have an "underfiting" issue.

1. Europe, Bonds, US, Currencies, Asia, Crypto
   - ROC
   - Feature Importance XGBOOST
   - Feature Importance Random Forest

Observations for each model

Models

Superiority of Optim Random Forest, Optim decision tree, Max decision tree, Optim Random Forest, Max random forest, Optimal SVM, XGBoost

1. America Stocks indices lagged 20d
   - ROC: Predict very quickly due to the gap between TPR and FPR except for Max Decision Tree
   - Feature Importance XGBOOST: returns SP Lima, returns Vol FTSE Biva, returns NASDAQ Vol, Returns SP/TSX, Returns Bovespa, Returns SP Lima General Returns SP CLX IPSA
   - Feature Importance Random Forest: Returns SP/TSX, Return Bovespa, Returns SP Lima General, Vol SP/TSX
2. US Stocks indices non lagged
   - ROC: Predict very quickly due to the gap between TPR and FPR
   - Feature Importance XGBOOST: Returns US 10y, Vol US 10Y, Returns US 10Y, Vol Germany 10Y, Vol Norway 10Y...
   - Feature Importance Random Forest: Vol US 10y, Return HK 10y, Returns US 10Y, Returns UK 10Y, Vol Netherland 10Y

Observations for each model

Models

Superiority of XGBoost, Max Random Forest, Optim Random Forest

1. Currencies lagged 20d
   - ROC:Predict very quickly due to the gap between TPR and FPR except for MLP Neural Network
   - Feature Importance XGBOOST: Returns DKK/USD, GBP/USD et Vol EUR/USD significant for Random Forest - Returns EUR/USD, GBP/USD et Vol GBP/USD significant for XGBoost
   - Feature Importance Random Forest: Returns DKK/USD, GBP/USD et Vol EUR/USD significant for Random Forest - Returns EUR/USD, GBP/USD et Vol GBP/USD significant for XGBoost
2. Currencies non lagged
   - ROC: Predict very quickly due to the gap between TPR and FPR except for MLP Neural Network
   - Feature Importance XGBOOST: Returns DKK/USD, GBP/USD et Vol EUR/USD significant for Random Forest - Returns EUR/USD, GBP/USD et Vol GBP/USD significant for XGBoost
   - Feature Importance Random Forest: Returns DKK/USD, GBP/USD et Vol EUR/USD significant for Random Forest - Returns EUR/USD, GBP/USD et Vol GBP/USD significant for XGBoost

Observations for each model

Models

1. Bonds lagged 20d
   - ROC: Predict very quickly due to the gap between TPR and FPR
   - Feature Importance XGBOOST: volatility US 10Y, return US 10 Y
   - Feature Importance Random Forest: volatility US 10Y, return US 10 Y
2. Bonds non lagged
   - ROC: Predict very quickly due to the gap between TPR and FPR
   - Feature Importance XGBOOST: volatility US 10Y, return US 10 Y
   - Feature Importance Random Forest: volatility US 10Y, return US 10 Y

Observations for each model

Models

1. Europe Indices lagged 20d
   - ROC: Predict very quickly due to the gap between TPR and FPR
   - Feature Importance XGBOOST: return OMXC20
   - Feature Importance Random Forest: return WIG20
2. Europe indices non lagged
   - ROC: Predict very quickly due to the gap between TPR and FPR
   - Feature Importance XGBOOST: return OMXC20, return WIG20
   - Feature Importance Random Forest: return AEX (Amsterdam Exchange), return CAC40

Observations for each model

Models

Superiority of Neural Network, Max Random Forest, XGBOOST

1. Asia Indices lagged 20d
   - ROC: predicts very quickly because the gap between TPR and FPR is large except for Max Decision Tree
   - Feature Importance XGBOOST: Return Kospi, Vol KLCI, Returns Hang Seng, Returns SP/ASX 200
   - Feature Importance Random Forest: Return Kospi, Returns BSE Sensex, Returns PSEi Composite, Returns Taiwan Weighted, Returns Hang Seng
2. Asia Indices non lagged
   - ROC: predicts very quickly because the gap between TPR and FPR is large except for Max Decision Tree Train
   - Feature Importance XGBOOST:
   - Feature Importance Random Forest: Returns Hang Seng, Returns Taiwan Weighted, Returns KLCI, Returns SP/ASX 200

Observations for each model

Models

1. Crypto lagged 20d
   - ROC: predicts very quickly because the gap between TPR and FPR is large
   - Feature Importance XGBOOST: Bitcoin Vol, Bitcoin return
   - Feature Importance Random Forest: Bitcoin Return, small impact of Litecoin returns
2. Crypto non lagged
   - ROC: predicts very quickly because the gap between TPR and FPR is large
   - Feature Importance XGBOOST: return bitcoin, returns Monero, vol bitcoin
   - Feature Importance Random Forest: return bitcoin, vol bitcoin, vol litcoin, return etherum classic

Observations for each model

Ranking

- 1 - XGBOOST
- 2 - MLP Neural Network
- 3 - Optim Random Forest
- 4 - Neural Network
- 5 - Max Random Forest
- 6 - Optim Decision Tree
- 7 - Optimal SVM
- 8 - Max Decision Tree
- 9 - Optimal Penalized Logistic Regression
- 10 - Elastic Net Penalized Logistic Regression
- 11 - Broyden–Fletcher–Goldfarb–Shanno Logistic Regression

Conclusion

The goal of classification algorithms in forecasting stock market crisis is to find accurate periods predictions for crash events. Once it is done, it would be interesting to look at the specific events which took place, and predict, according to the security type, which type of crisis events can be involved. The extensions brought are:

- Use of SGD, L-BFGS and penalized logistic regression to see if less traditional methods can improve results.
- Inclusion of cryptocurrencies

References

- Sotirios P. Chatzis, Vassilis Siakoulis, Anastasios Petropoulos, Evangelos Stavroulakis, Nikos Vlachogiannakis, Forecasting stock market crisis events using deep and statistical machine learning techniques, Expert Systems with Applications, Volume 112, 2018, Pages 353-371, ISSN 0957-4174.
  https://www.sciencedirect.com/science/article/pii/S0957417418303798
- Obthong, Mehtabhorn Tantisantiwong, Nongnuch Jeamwatthanachai, Watthanasak Wills, Gary. (2020).
  A Survey on Machine Learning for Stock Price Prediction: Algorithms and Techniques.
- Michel Ballings, Dirk Van den Poel, Nathalie Hespeels, Ruben Gryp, Evaluating multiple classifiers for stock price direction prediction, Expert Systems with Applications, Volume 42, Issue 20, 2015, Pages 7046-7056.
  https://www.sciencedirect.com/science/article/pii/S0957417415003334
- Roman Moser, Predicting stock market crashes. An attempt with statistical machine, 2019,Towards Data Science.
- Christian Versloot, How to visualize support vectors of your SVM classifier?, Machine Curve, May 2020.
  https://www.machinecurve.com/index.php/2020/05/05/how-to-visualize-support-vectors-of-your-svm-classifier/
- scikit-learn.org documentation: model_selection.GridSearchCV, sklearn.metrics._curve, plot_forest_importances
  https://scikit-learn.org
- Ali Mirzaei, How to use Deep-Learning for Feature-Selection, Python, Keras, Medium.com, 2019.
  https://medium.com/@a.mirzaei69/how-to-use-deep-learning-for-feature-selection-python-keras-24a68bef1e33
- Daniel J. TOTH, Binary Classification: XGBoost Hyperparameter Tuning Scenarios by Non-exhaustive Grid Search and Cross-Validation
  https://towardsdatascience.com/binary-classification-xgboost-hyperparameter-tuning-scenarios-by-non-exhaustive-grid-search-and-c261f4ce098d