SPECIALIST CERTIFICATE IN DATA ANALYTICS FOR FINANCE

MACHINE LEARNING ALGORITHMS FOR STOCKS PURCHASE

Candidate: Giuseppe Difato

GitHub Repository: https://github.com/GiuseppeDifato/Project\_UCD\_DifatoG.git

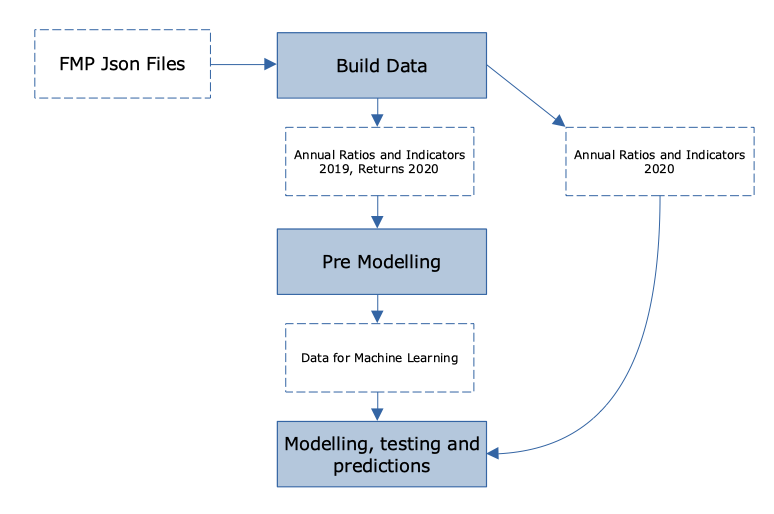
# Executive Summary

Is it possible to base a stock purchase on the information contained in the annual financial statements, balance sheets and other publicly available reports?

The aim of this project is to build and test some machine learning models that would predict whether a specific stock is worth buying based on financial ratios and indicators available at a specific year-end.

The API “Financial Modeling Prep” has been used as a data provider.

The project can be divided in three stages: data building, pre-modelling analysis and modelling. For each step a different Jupyter notebook has been used.



# **Phase 1 – Building Data**

The first phase of the project uses the Jupyter notebook “BuildData” to scrape a number of json files obtained from Financial Modeling Prep and save the data in a unique csv format file. This file contains the year end 2019 indicators[[1]](#footnote-2) and the stocks returns during 2020.

After setting some parameters, five functions are defined.

1. *get\_json\_data:* takes a url as input and allows the json file’s reading;
2. *find\_sector: takes a json file as input and scrape a dictionary returning the company’s symbol and sector;*
3. *find\_fin\_indicators: takes two inputs, a json file and an indicator name, and return the company’s symbol and a specific indicator with its value. This function is used for different categories of indicators as the shape of their json files are exactly the same (list of dictionaries). A generator creates a list of list where symbols, indicators and values are saved. This list of list is then changed into a flat list which is eventually looped only for the indicators to be included in the analysis;*
4. *find\_ratios\_indicators: the scraping of the ratios json file required a new function as its shape is slightly different (a dictionary with a nested list of dictionaries). The embedded list of dictionaries is scraped only for some sub-categories and the function eventually returns the symbol, a ratio and its value;*
5. *get\_price\_var: this last function leverages on the Pandas DataReader to extract the stocks prices at specific dates from Yahoo Finance and calculates the returns over the period.*

*For the ease of convenience the urls of the json files were saved in a text file; the notebook leverages on the functionality of Regex to look up a specific one.*

*Once the list of stocks is filled, it is filtered to the ones belonging to the New York Stock Exchange and the Nasdaq Global. The flag\_random allows to select randomly only 100 of them; this functionality will be useful when using the model for future predictions. The sector is then added to every symbol. A progress bar shows the status of the information loading (tqdm library used).*

*At this stage, a text file containing all the indicators names is loaded and a chunk of code uses the functions 3 and 4 to retrieve them from the json files. The resulting list is transformed into a DataFrame and manipulated to set the symbols as index and all the indicators as columns.*

*Finally the following year stocks returns are read in and merged to the DataFrame by symbol, if required[[2]](#footnote-3). In fact a boolean flag\_return allows the user not to extract the returns. Again, this will be the case when dealing with extraction of data for future model’s usage.*

# **Phase 2 – Pre Modelling Analysis**

The second phase of the project uses the Jupyter notebook “PreModelling” to

* read in the raw data extracted during phase one
* change the return into a binary variable “buy\_ignore” based on its value (1 if greater than 0, 0 otherwise) to be used in the modelling as the response (or dependent) variable
* transform the sectors into a numeric variable “sector\_code”
* clean and standardize data
* analyse the presence of multi-collinearity and remove highly correlated predictor variables (the response variable “return” is clearly excluded). A correlation of 0.80 has been set as threshold for the variables drop.

The figure below shows the correlation matrix of the remaining variables



As expected, there is not evidence anymore of high correlated variables. The resulting dataset is saved into a csv format file to be loaded and used in the modelling phase[[3]](#footnote-4).

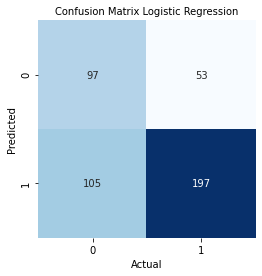
# Phase 3 – Modelling

The third phase of the project is the bulk of this work: some machine learning models have been selected, tested, fine tuned and eventually used for real predictions with 2020 data.

Before proceeding with the modelling itself the data have been split into training and testing, with a testing size of 0.2, and stratified by the dependent variable.

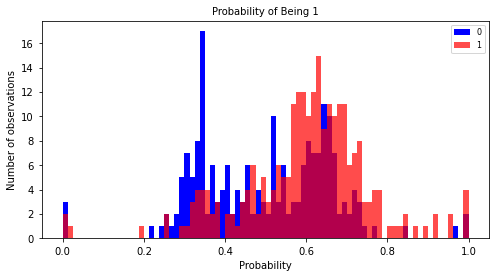
## Logistic Regression

Given the binary nature of the response variable, a logistic regression seems to be an appropriate choice. After a first trial run of the model, a manual fine tuning is performed on the strength of regularization (from default 1 to 10). The figure below shows the model confusion matrix



A first comment is on the number of *false buy (*bottom left square of the matrix*)*: the model predicted 105 buys for elements with an actual ignore.

Another very effective representation of the results can be the frequency of the probability of the testing elements of being 1



Also from this plot is very clear how a number of testing elements with actual response 0 (ignore) have significant probabilities of being 1 (blue bars behind reds).

The model report confirms the comments above with a recall[[4]](#footnote-5) of 0.48 and a f1-score[[5]](#footnote-6) of 0.55 for elements with an actual ignore as response. However, the overall model scores[[6]](#footnote-7) can be considered quite satisfying 0.65.

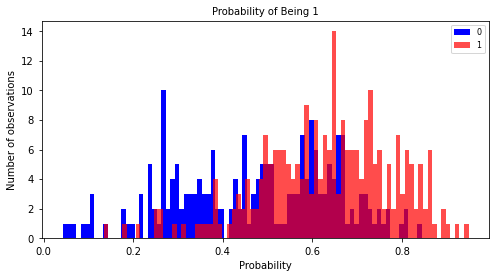
Finally, to get some insights on the most significant coefficients of the regression, the top five positive are printed out (Dividend Payout Ratio, Total assets, Net Income per Share, Revenue per Share, Gross Profit Growth).

## **Random Forests Classifier**

Random Forests are an ensemble method for classification that operates by constructing a multitude of decision trees.

After a first run where an accuracy of 0.66 was obtained, a Grid Search cross validator has been used to fine tune the hyper-parameters significantly improving the score to 0.70 (best estimator result).

An important finding from this model is the reduction of the “false buy” with respect to the Logistic Regression (45 as opposed to 105). This is also confirmed from the graph of the probabilities of being 1: a clearer distinction between blue and red bars is noted



To confirm the efficiency of the Grid Search optimization, the scores of the training and testing data split by the grid parameters are shown, the so called Validation Curves[[7]](#footnote-8)

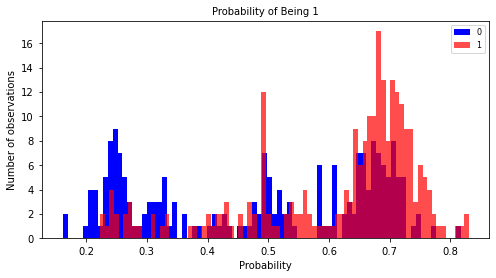
It is possible to say that the optimal number of estimators lies around 30 and the maximum depth, relying on the training data, is 8. The other 2 parameters seem to give equal results whatever the chosen feature is.

## **Support Vector Machine**

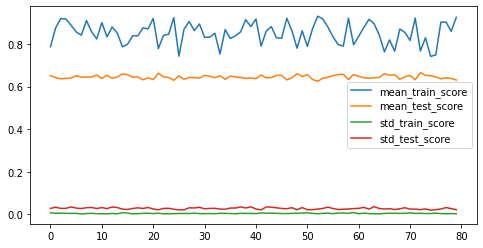
The third model tested is the Support Vector Machine, a non-probabilistic binary linear classifier. After a first run, the Randomized Search cross validator optimizes the parameters Cs and Gammas.

The score result improved from 0.64 to 0.65 (best estimator). With respect to the Random Forests, this one shows a higher number of the “false ignore” (top right square of the confusion matrix): 117 as opposed to 93 of the Random Forests.

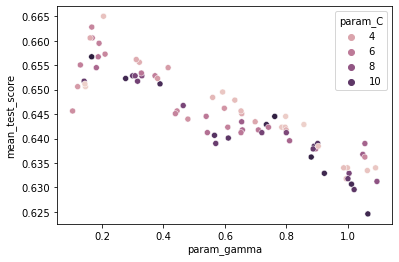
Again, this is confirmed by the probabilities of being 1, where it is noted a wider spread of the red bars



These other following two plots represent the mean and standard deviations of the scores on the training and testing data



and the mean of the scores on the testing data by the parameters randomly searched via the cross validator

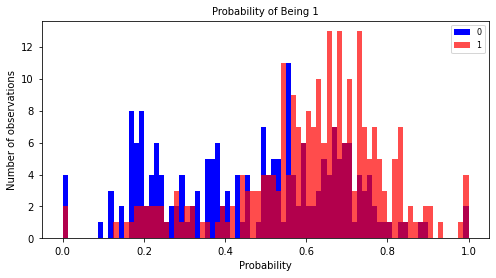


The optimal model should have a C < 4 and a gamma close to 0.2.

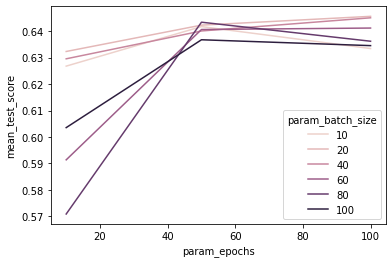
## **Keras Classifier**

This model belongs to the family of the Neural Networks. Two hidden layers have been added and a Grid Search cross validator executed on the batch size and the epochs.

The best estimator score is 0.66 but also this model seems to show a high number of “false ignore”, 101. As a results the red bars on the probabilities of being 1 are spread with some peaks towards low probability values



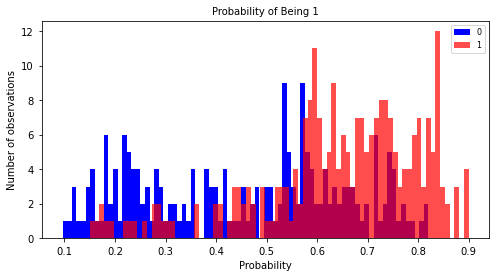
The plot of the scores by batch sizes and epochs on the testing data shows the choice of 20 for the batch size and 100 for the epochs



## **Stacking Model**

The last model is an ensemble of some of the ones just analysed. The base model is comprised of a Logistic Regression, a Support Vector Machine and a Random Forests. The meta learner is again a Logistic Regression.

The resulting accuracy is slightly higher than 0.69 and there is not much evidence of high false buy or false ignore. In fact, the probabilities of being 1 for actual ignores and actual buys are pretty much separated (red bars towards greater probabilities and vice versa for blues)



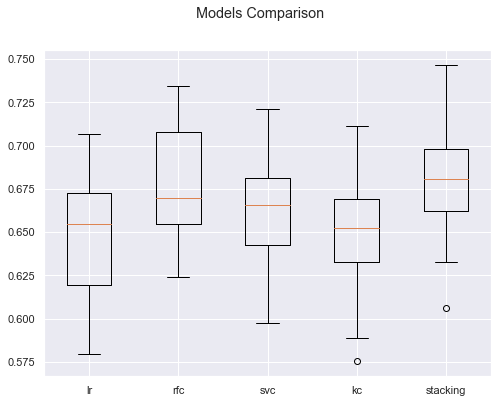
Also the recalls and f1 scores show better results than other models

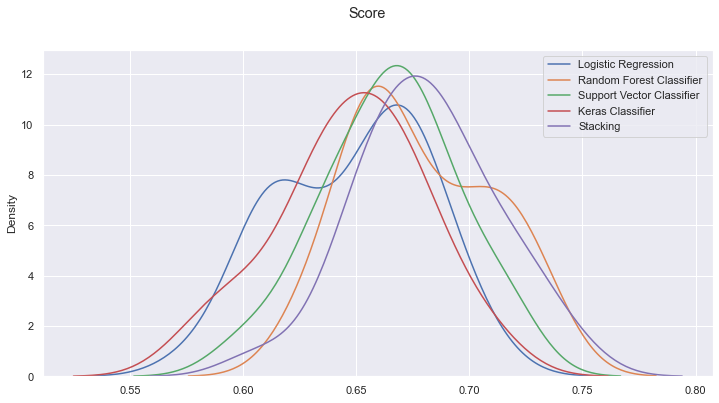
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1 score | support |
|  |  |  |  |  |
| IGNORE | 0.73 | 0.53 | 0.61 | 202 |
| BUY | 0.69 | 0.84 | 0.76 | 250 |
|  |  |  |  |  |
| accuracy |  |  | 0.70 | 452 |
| macro avg | 0.71 | 0.68 | 0.68 | 452 |
| weighted avg | 0.71 | 0.70 | 0.69 | 452 |

## **Repeated Stratified K-Fold Validation**

To get additional comfort on the reliability of the models, a repeated K-Fold cross validator has been used: every model undergoes 3 repeats of 10 cross-validation stratified on the response variable.

The distributions of the models accuracy confirm the results obtained with a test size of 0.20 and the better performance of the stacking model. The random forests and the support vector show similar scores averages; however the support vector seems to have a much smaller variance.





## **Results**

To wrap everything up, a portfolio has been built for each model predictions (symbol with a prediction of 1 added to the portfolio, ignored otherwise), the 2020 profit and loss have been calculated and compared with the actual

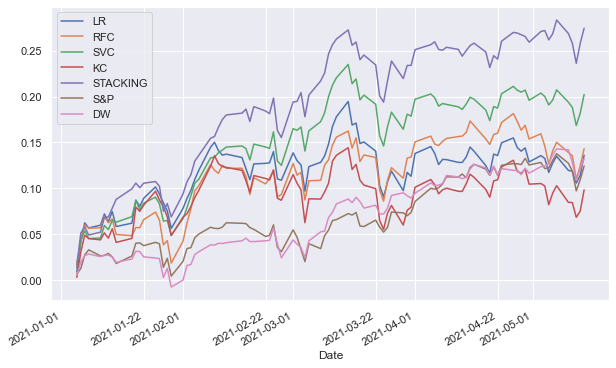
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **p/l\_lr** | **p/l\_rfc** | **p/l\_svc** | **p/l\_kc** | **p/l\_stacking** | **p/l\_actual** |
| **17.28** | **18.92** | **17.70** | **16.89** | **19.17** | **27.64** |

As expected the better the model’s accuracy, the higher the return.

Now, what if one were to use these models with 2020 annual data to make decisions about what stock to buy and hold for 2021?

To answer this question, with the notebook BuildData (and for a sample of symbols), the 2020 data have been extracted for the validated set of indicators (i.e. the ones defined within the notebook PreModelling)[[8]](#footnote-9).

Once the models had produced the predictions, the 2021 portfolios performances have been pulled together and compared to two US stock indexes, S&P and DJ.



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **LR** | **RFC** | **SVC** | **KC** | **STACKING** | **S&P** | **DW** |
| **Jan 21** | 5.64 | 1.88 | 4.92 | 4.86 | 6.90 | 0.46 | -0.73 |
| **Feb 21** | 10.89 | 9.34 | 12.51 | 8.71 | 15.53 | 3.11 | 2.45 |
| **Mar 21** | 9.76 | 11.11 | 16.44 | 6.01 | 21.95 | 7.34 | 9.53 |
| **Apr 21** | 12.88 | 15.38 | 19.59 | 10.46 | 25.91 | 12.55 | 11.66 |
| **May 21** | **13.62** | **14.32** | **20.23** | **9.84** | **27.43** | **12.44** | **13.20** |
| **Nr Of Stocks** | 19 | 13 | 34 | 16 | 23 | n/a | n/a |

## **Insights**

1. Ratios profit expected / profit actual on the testing data broadly in line with models scores.

2. Four of the five models outperformed the S&P and the DJ indexes in the first 5 months of 2021.

3. The stacking confirmed as top performer: score and low number of false ignore and false buy envisaged the outcome.

4. Support Vector Machine outperformed Random Forests in 2021 despite their scores were pretty similar: apparently the random forests resulted in a higher number of false ignore.

5. The models were built to buy and hold the securities for the whole year; 2021 results only account for YTD returns. It would be interesting to monitor the portfolios until the year end.

1. Full list of indicators contained in the text file “indicators.txt”. [↑](#footnote-ref-2)
2. Raw data with full year 2019 indicators and 2020 returns “DATA\_RAW\_2019.csv” [↑](#footnote-ref-3)
3. Final dataset to be used for machine learning “DATA\_ML.csv” [↑](#footnote-ref-4)
4. True0 / (True0 + False1) [↑](#footnote-ref-5)
5. 2 True0 / (2 True0 + False0 + False1) [↑](#footnote-ref-6)
6. Percentage of correct predictions [↑](#footnote-ref-7)
7. To build up the volatility bands of the curves, the pooled variance has been use with a size 5 for each group [↑](#footnote-ref-8)
8. “DATA\_RAW\_2020” [↑](#footnote-ref-9)