University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Damian Dąbrowski
Student no. 439954

Heorhii Lopatin
Student no. 456366

Ivan Gechu
Student no. 439665

Krzysztof Szostek
Student no. 440011

Large language models for forecasting market behaviour

Bachelor's thesis
in COMPUTER SCIENCE

Supervisor:
dr Piotr Hofman
Instytut Informatyki

Warsaw, May 2024

Abstract

This thesis concerns research into the use of machine learning and large language models in market analysis, focusing on market predictions.

Keywords

machine learning, large language models, time series forecasting, market prices

Thesis domain (Socrates-Erasmus subject area codes)

11.4 Sztuczna inteligencja

Subject classification

D. Software
D.127. Blabalgorithms
D.127.6. Numerical blabalysis

Tytuł pracy w języku polskim

Duże modele językowe w przewidywaniu giełdy

Contents

Chapter 1

Introduction

1.1. Overview

In the world of stock markets a major problem is the apparent incalculability of the complex network of factors e.g. how stock prices of one company affect those of another. As the environment of stock markets becomes more and more complex, the ability to analise and confidently predict its future becomes of crucial importance for traders, investors and researchers.

With the recent advent of generative AI and the demonstrable power of Large Language Models a question arises of how these can be used to accurately analise and predict time series market prices in different environments. Therefore, this thesis presents our work on the subject.

1.2. Contributions

1.3. Outline

First, we look at what work has already been done in the field of LLM time series prediction, in particular what techniques of fine-tuning and input data transformation were used. Then we look at how different, smaller machine learning models deal with time series prediction.

We describe the datasets we used for testing small models and LLMs.

Subsequently, we discuss our own methodology; different aplied methods and techniques of input reprogramming, use of prompts and context, and LLM fine-tuning. Next, we present the results we have achieved on the chosen datasets (and compare them to some other known solutions).

Finally, we speculate on the significance of our work, its potential applications in forecasting price time-series.

Chapter 2

Related work

Chapter 3

Other models

Here we present our results from trying to use the following models to extrapolate a time series. Classification models output in binary categories: increase or decrease in value, and are therefore less precise.

3.1. Random forest

A random forest is a machine learning model for classification and regression tasks introduced by Leo Breiman in 2001. A random forest is an ensemble of individual tree predictors, each of which depends on a random vector, chosen independently and with the same distribution for all trees. The results from individual trees are then aggregated into the overall result of the model - for classification tasks it is the mode of individual classifications and for prediction tasks it is the mean average of individual predictions.

The error of forest prediction converges as such as the quantity of trees in the forest increases. The error of forest prediction depends negatively on the accuracy of individual trees and correlation between them.

The `num_lags` parameter is the width of data taken for individual predictions. The larger the `num_lags` the more of past data the model takes into account.

The `n_estimators` parameter (number of estimators) describes the number of trees grown in the forest. Increasing the number of trees increases the accuracy of the model, but it also increases the computational cost.

The `max_features` parameter (number of features) specifies the the number of features of the data considered for a split at each node while growing an individual tree. A higher value of `max_features` may capture more information about the data at the ris of overfitting and decreased randomness of the model.

The `criterion` parameter describes the function used by the model to calculate a quality of a split at a given node

The model was trained and tested on the simple property sales dataset. The model was trained to classify whether the next price will be higher or lower than the previous one, based on the sequence of prices spanning the last `num_lags` days. The combinations of following parameters were tried

- `num_lags`: [1, 5, 10, 13, 25, 40, 50]
- `n_estimators`: [20, 50, 100]
- `max_features`: [2, 4, 8]

- `criterion`: ["gini", "entropy", "log_loss"]

The accuracy of the models ranged from 0.749 to 0.827. The main influence seems to be the lag number - the optimal being around 10. Then slightly better were those models with `max_features` equal to 4, and number of trees greater or equal to 50. The criterion didn't seem to play a significant role. See figure 3.1 below.

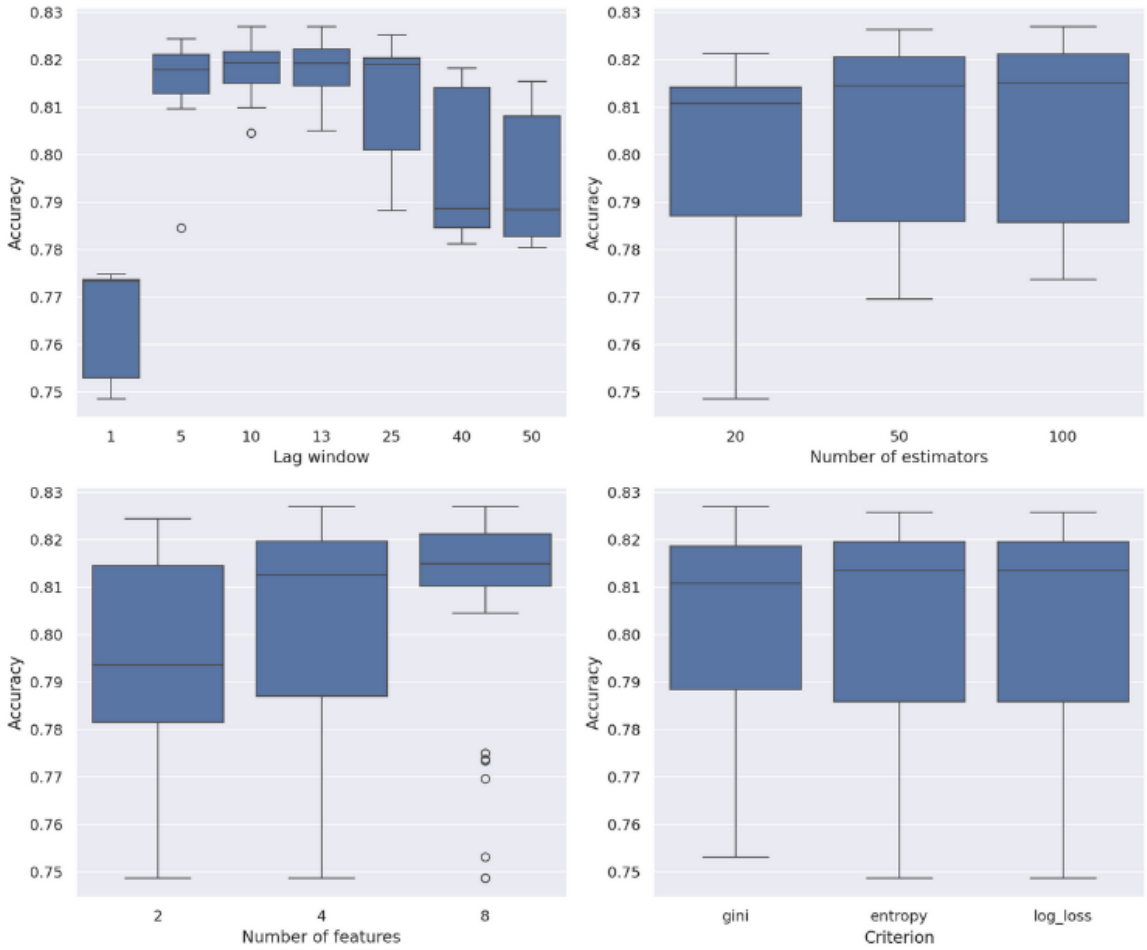


Figure 3.1: Results of random forest experiment.

3.2. Logistic regression

Logistic regression is a statistical model used for binary classification. It calculates the probability of whether a datapoint classifies to one class. During training of the model a sigmoid function of features of data is calculated that best fits the provided training dataset.

More precisely, the model calculates the function $Y(X) = \frac{1}{1+e^{-z}}$, where

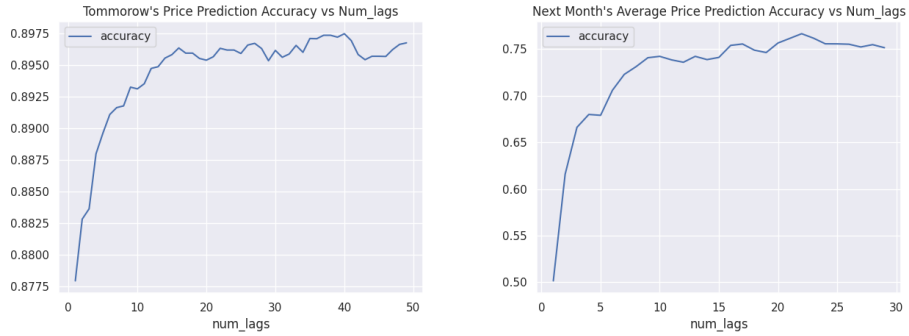
- $z = B_0 + B_1 \cdot X_1 + \ldots + B_n \cdot X_n$,
- $X_1, \ldots, X_n$ are features of data X,
- $B_0, B_1, \ldots, B_n$ are parameters of the model.

Function Y assumes values only in the range $(0, 1)$. If $Y(X) >= 0.5$, the model classifies the datapoint as 1 (in our model below - the price will be higher). If the converse is true, the datapoint is classified as 0 (the price will be lower).

During training the parameters $B_0, B_1, \ldots, B_n$ are chosen using Maximum Likelihood Estimation function, so that the results of the Y function best fit the training dataset.

Two models have been trained. The first one is trained to predict if the next price in the dataset wil be higher than the previous one, provided with a sequence of prices spanning over the last `num_lags` days. The second one does the same but operates on monthly averages instead of single records (for this, the dataset was averaged over subsuquent months). The first model proved to be more precise with an accuracy of 82% for an optimal value of k of around 40 (Fig 3.2a). At the same time the second model only scored 75% (Fig 3.2b).

However, simply predicting whether the price will be higher or lower is much easier then predicting the actual next price. Additionally, logistic regression assumes an independence of datapoints from each other, wherefore it performs poorly for time series datasets. Therefore this model is insufficient for our purpose.



(a) Day-by-day prediction.

(b) Month-by-month prediction.

Figure 3.2: Plots of two models of logistic regression.

3.3. Support vector machine

Support vector machines are a widely popular model for machine learning classification problems, due to their high generalization abilities and applicability to high-dimensional data.

The model construes datapoints as high-dimensional vectors and finds the best hyperplane that divides the two classes the data is classified into. The goal of training is to find the hyperplane with the greatest margin - that is, the greatest distance from the closest vectors, called the support vectors.

The model uses a kernel function to transform the space of data points which are not separable by a hyperplane, into one where they are separable.

The C value specifies how accurate the model should be, that is how much it should avoid misclassifications, versus how wide the margins should be. A lower value of C corresponds to wider margins.

The gamma value specifies how much influence individual training datapoints should have. The `scale` value of gamma means that gamma is scaled automatically to fit the size of the dataset.

The model was tested using two metrics: Mean Squared Error and R-squared.

For the support vector machine there were overall nine models tried:

- three kernels: `rbf`, `poly`, `sigmoid`
- one gamma: `scale`
- three C values: 0.1, 1.0, 10.0

The ‘poly‘ kernel worked much better than both ‘rbf‘ and ‘sigmoid‘, which both worked equally badly. Overall, though, the statistics for every model were terrible. The value of ‘C‘ has had very little impact and only on ‘rbf‘ kernel.

Support Vector Machine model doesn't perform well in this task, as can be seen in the results (Fig 3.3). Overall, Support Vector Machins don't perform well with large datasets, which will be part of our endeavour.

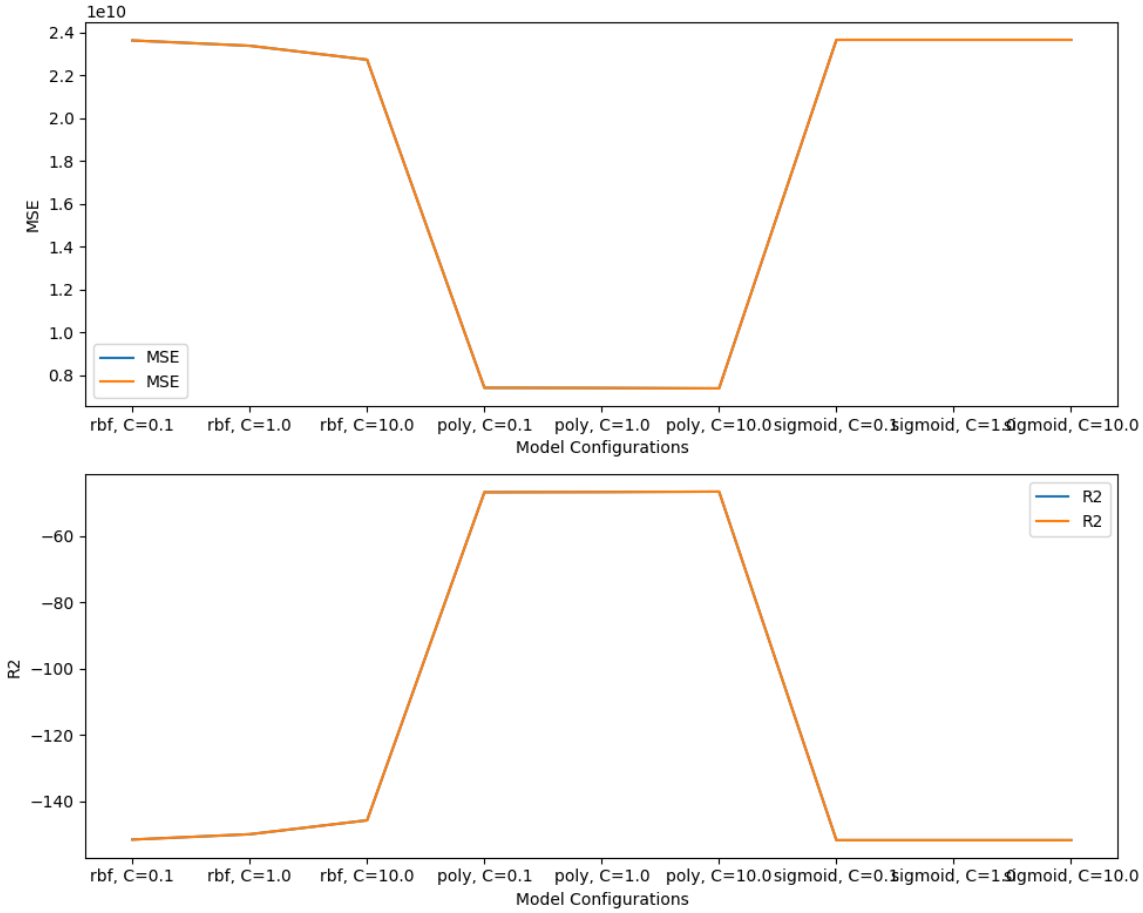


Figure 3.3: Results of support vector machine experiment.

3.4. Multi-layer perceptron

3.5. Convolutional neural network

3.6. Residual neural network

Chapter 4

Methodology

Chapter 5

Main results

Chapter 6

Forecasting applications

Chapter 7

Conclusion

Appendix A

Visualisation

Bibliography

[] https://arxiv.org/pdf/2310.19717v1.pdf

[Bea65] Juliusz Beaman, *Morbidity of the Joll function*, Mathematica Absurdica, 117 (1965) 338–9.

[Blar16] Elizjusz Blarbarucki, *O pewnych aspektach pewnych aspektów*, Astrolog Polski, Zeszyt 16, Warszawa 1916.