Combining the FT-Transformer with the LSTM model to predict stock prices

Eber Sluis

July 1, 2023

Abstract

The main goal of any investor is to predict what will happen to the stock prices. This emphasises the need for good forecasts. In recent years, a wide range of novel Deep Learning architectures have found competitive results on tabular data and financial time series. This study aims to combine two Deep Learning networks into a model that specialises forecasting stock prices. The proposed FT-LSTM model is trained and tested on a dataset of FRED containing 9 financial variables, of which the S&P 500 is the target variable. The model is compared to four widely used Deep Learning models and thoroughly evaluated on multiple forecasting horizons. The model is found to significantly outperform the other models in most regression forecasting tasks. Furthermore, the robustness, overfitting and tuning are addressed.

BSc Econometrics and Operational Research Thesis[1]
Erasmus School of Economics

Supervisor: M. Mueller PhD candidate, econometrics
Second Assessor: dr. C. Cavicchia

[1]The views stated in this thesis are those of the author and not necessarily those of the supervisor, second assessor, Erasmus School of Economics or Erasmus University Rotterdam.

1 Introduction

Predicting the future is a gift many would like to have, from sports bettors to political leaders. Especially in the field of finance this is an often discussed subject. Financial investors try to earn money by investing in assets and gaining returns when the asset moves in the predicted way. One popular way of doing this is through stocks. Stocks reflect the economic success of a company and one can become owner of a small part of the company by buying one. Not only for investors themselves is it important to predict the stock market, also for almost all other institutions in the market. Companies would want to predict it, because they make money, lend and grow through the stock market. Pension funds need to invest the money of their clients in order to eventually be able to pay out the pensions. Even for the Federal Reserve it is important. They are the ones to set the interest rate and the predicted stock market is one of the indicators on which they base their decisions.

Technology is evolving rapidly, where econometrically conventional methods such as the GARCH, logit or even the OLS model were booming in the past, Artificial Intelligence (AI) is taking over. Machine Learning (ML) models have shown to be very effective in finding patterns in big data, this has lead to better results than its conventional counterparts. They do this by letting the machine "learn" from the data. Recently, the arrival of Deep Learning (DL), a specific type of Machine Learning, has opened up the eyes of researchers. DL models use a structure or architecture that is based on the human brain, which is why it is often called a Deep Neural Network. The impressive achievements have been shown by DL based programs like translators, image recognition and chatbots like CHAT GPT.

There are many different types of DL models, each one being better at a different task. Where a simple Plain Vanilla architecture works well on image recognition, a Residual Network works better on translation. Econometricians see the possibilities of these models and want to implement them into their own field of research. Usually the data used is then economic data, which is often tabular. To obtain the benefits of DL on tabular data, a so far state-of-the-art model has been proposed by [Gorishniy et al., 2021]. The authors came up with the FT-Transformer model, that works well on multiple tabular datasets compared to other DL models. Most of these datasets, however, do not include financial data. This is where another DL model comes in, the LSTM. This model can find patterns in time series data, by using the fact that there are relations in sequences of observations. These "feedback" connections make the LSTM model ideal for data prediction. Wanting to take characteristics of these two models together, to get the best of both worlds w.r.t. financial data forecasting, is why the following research questions will be answered in this paper:

Can a combination of the FT-Transformer and LSTM model predict the stock market well, compared to other Deep Learning methods?

This will be done by first examining the FT-Transformer thoroughly and replicating the model of Gorishny et al. After this, the specific component that makes the model work well on tabular data will be combined with the component of the LSTM model that captures the time series relations. This is the motivation as to why this model should do well on stock market data, which is both tabular and time related. The proposed model is called the FT-LSTM model.

The relevance of this research is explained by the relevance of the stock market. Not only does it provide means for companies to raise money and fund operations, it also creates and sustains wealth for the individual investor. What's more, the stock market is a means of a free-market economy and ensures transparency and liquidity. Therefore it is relevant that one can predict the price indices. Next to that, this research will add to the current Deep Learning research and extend the available knowledge of DL on financial data.

The data used in this paper is the S&P 500 price index as the main target variable to predict. There will be eight other variables used as features of the target. These include a wide range of financial and macroeconomic variables. The variables are analysed by means of a plot and correlation matrix.

The paper will go on by a literary review of previous studies in section 2, explanation of the data in section 3, description of the used methodology in section 4, a breakdown of the obtained results in section 5 and finally a conclusion in section 6.

2 Literature Review

The effectiveness of Deep Learning has been shown in many different fields. [Wu et al., 2015] proved the advantages of DL models in image recognition, [Young et al., 2018] showed the recent trends in Deep Learning based language processing and [Ignatov et al., 2019] came up with a benchmark of AI models to use on smartphones.

In recent years, a wide range of DL models on tabular data, as well as on financial data have been proposed. For tabular data the main benchmark paper is the revisit by [Gorishniy et al., 2021]. This paper has, however, been criticised by [Shwartz-Ziv and Armon, 2022], where they show that it is not only DL that one needs for tabular data. Other findings on tabular data include: [Hazimeh et al., 2020] and [Popov et al., 2019], who invented ensemble methods on tabular data. The ensemble methods have mixed results when it comes to financial data regression, being sometimes outperformed and sometimes not by DL. This contradicts the proposed model in this paper, which is consistent and robust. [Huang et al., 2020] focuses more on the Attention based methods. In our experiments, it is shown that a properly assembled temporal model outperforms the ResNet model, which in turn outperformed the Attention based methods.

In the financial field of research the LSTM model is the benchmark. [Nikou et al., 2019], [Tao et al., 2019], [Sharaf et al., 2021] and [Katayama et al., 2019] all use the LSTM model on the stock market in different ways. [Siami-Namini and Namin, 2018] found that the LSTM model outperformed conventional methods like the ARIMA model. This is the main reason why this model contributes to the proposed model in this paper. Compared to this study, the research with LSTM models has mainly been done with one variable and no other features. This shows the advantage of adding the feature tokeniser, when having more explanatory variables. Other DL models used on the financial market include the stacked autoencoder by [Heaton et al., 2017] and the MLP model by [Zhong and Enke, 2019]. These two papers show, just like our investigation, examples of specifically created DL models for the data. The data used in both cases are stocks in specific sectors of the market instead of targeting the market as a whole.

The community has also proposed several combinations of DL models, the so called hybrid models. [Ji et al., 2021] target the Australian stock market by using an LSTM model with improved particle swarm optimisation (IPSO). Unlike our model, this hybrid does not add changes to the architecture itself. [Niu et al., 2020] test multiple different hybrid LSTM models and find that they work slightly better than singular models when forecasting stock prices. Another version of an LSTM Transformer hybrid has been introduced by [Andayani et al., 2022]. Albeit not on the stock market, they do find good performance. Finally, [Lim et al., 2021] came up with the Temporal Fusion Transformer (TFT). This version of the Transformer model captures temporal features of multivariate time series. By adding recurrent layers to self-attention mechanism they obtain high-performance, but unlike our model they lack the achievement of operating at the feature level of the object.

3 Data

3.1 California Housing

For the replication part, the California Housing real estate dataset is used. The data pertains to the houses found in a given California district and some summary statistics about them based on the 1990 census data. The target variable is the housing price, and there are 8 features for each observation. These features include: longitude, latitude, age, number of rooms, number of bedrooms, population, income, ocean proximity. In total there are 20640 observations. This dataset serves as an excellent introduction to implementing Deep Learning algorithms because it has an easily understandable list of variables and sits at an optimal size between being too simple and too cumbersome. The prepocessing/transformations of this data are the same as in the replicated paper [Gorishniy et al., 2021].

3.2 FRED financial data

The data used in this research paper includes a target variable and eight explanatory variables. All data is obtained from the Federal Reserve Economic Database (FRED). The variable that will be predicted is the S&P 500 stock index, which has ticker SP500 in the FRED. This index includes 500 leading companies in leading industries of the United States economy/equity market. The data is collected daily as the closing price of the stock and includes data from the first of July 2013 till the twelfth of May 2023. This leads to 2439 observations after removal of the not available observations, like public holidays. The data is not seasonally adjusted.

The information of the predictors is summed up in table 1. These are also all daily variables, ranging from the same time period as the target. The lags of these variables are the features of the target variable and will throughout this paper be referred to by their ticker symbols.

In table 2, the correlations of all of the variables are shown. One can see that the S&P 500 index is highly correlated with the other stock indices, and also with the Nominal Broad US dollar index and (negatively) with the interest rate spread. Due to the fact that Deep Learning models have been known to find relations that are not visible upfront, the other variables are also included in the analysis even though the correlation is small.

All of the variables are plotted in figure 10 in the appendix. Most variables show a clear sign of the crises, such as the March 2020 Covid-19 pandemic. Furthermore, there is an upward trend present with the target variable. This will be treated by appropriate prepocessing of the data.

3.2.1 Preprocessing

To ensure proper functioning of the Deep Learning models, it is important to preprocess the data. Because the type of analysis that is being done is a regression, the target variable will be normalised. This transformation includes a subtraction of the mean and thereafter a division by the standard deviation.

Table 1: Independent variables

FRED ticker	Explanation	Unit
DCOILWTICO	West Texas Intermediate, Crude Oil prices	Dollars per barrel
DEXUSEU	US dollar/Euro exchange rate	Dollar per one Euro
DTWEXBGS	Nominal Broad US dollar index	Index
T10Y2Y	Interest rate spread between 10- and 2-year treasury	Percent
T10YIE	10-year breakeven measure of expected inflation	Percent
DJIA	Dow Jones Industrial Average 30 component companies	Index
NASDAQCOM	NASDAQ composite index 3000 common equities	Index
VIXCLS	Expectation of volatility, conveyed by stock indices	Index

Table 2: Correlation matrix of the data

	DCOIL	*DEXUSEU*	*DTWEX*	*T10Y2Y*	*T10YIE*	*DJIA*	*NASDAQ*	*SP500*	*VIXCLS*
DCOILWTICO	1								
DEXUSEU	0.4329	1							
DTWEXBGS	-0.4156	-0.8890	1						
T10Y2Y	0.2081	0.7237	-0.8590	1					
T10YIE	0.8076	0.1623	-0.0893	-0.0182	1				
DJIA	0.1358	-0.3909	0.6492	-0.6623	0.5042	1			
NASDAQCOM	0.1114	-0.3245	0.5877	-0.5205	0.4845	0.9660	1		
SP500	0.1620	-0.3826	0.6223	-0.5950	0.5286	0.9870	0.9876	1	
VIXCLS	-0.1315	-0.2729	0.4449	-0.3139	-0.1205	0.3179	0.3896	0.3547	1

By default, we transform the features with the quantile transformation from the Scikit-Learn package [Pedregosa et al., 2011]. This technique ensures that the features follow a gaussian distribution, by making an estimate of the empirical cumulative distribution function of a feature to map the original values. Both of these transformations are fit on the training data only, to make sure that there is no bias or information leakage of the validation or testing set.

3.2.2 Training split

For the evaluation of the models, it is of importance to split the dataset in a training, validation and testing set. If the model is only trained on the training set and tuned on the validation set, one ensures that the model is less likely to suffer from overfitting. This is especially important for this research, since DL models are known to find (unexplainable) relations between the data that work extremely well on the training data and not on the testing, because they are too specific. Throughout this paper the 80-10-10 split is used. This means that the first 80% is used for training, the next 10% for validation/tuning and the last 10% for testing. This corresponds to a split at 25 May 2021 and 19 May 2022. The reasoning behind this data split and the possible dangers of overfitting are further explained in section 5.3.3.

4 Methodology

In this section the Deep Learning architectures used to forecast the S&P 500 are explained. Due to the fact that different Deep Learning models are better at certain tasks than others, the focus will be on why the used models are effective at tabular financial data forecasting.

4.1 Multilayer Perceptron

The Multilayer Perceptron (MLP) model, also called a plain vanilla architecture is one of the oldest and simplest Deep Neural Networks there exists. This common model was originally proposed by [Ivakhnenko et al., 1965] and became the foundation of many DL models. The body of this network is shown in figure 1.

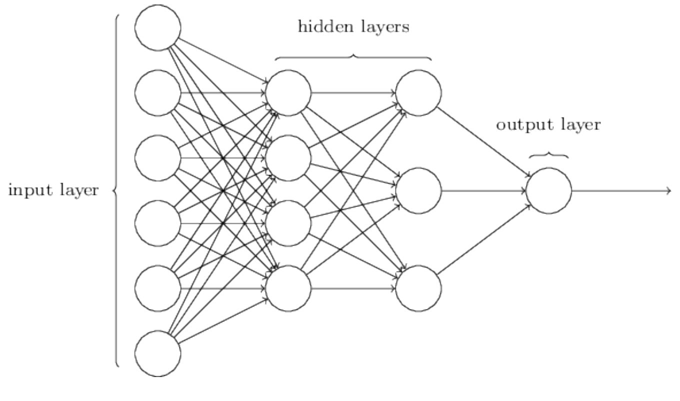


Figure 1: MLP network [source: Medium]

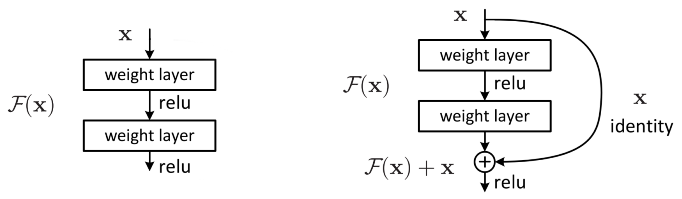
This network is representing the way of coming to a forecast for each observation. The input layer has one node for each feature of the data and the value of the node is called the activation a_i for node i. In this paper the features correspond to the lagged variables introduced in section 3. Every line between the different nodes is the weight $w_{i,j}$. The next (hidden) layer of nodes is activated by

$$a^{(1)} = act(Wa^{(0)} + b)$$

Where W represents a matrix of the weights and b a vector of biases. A bias of inactivity is added when one wants nodes to only activate after a certain point. *act* is the activation function. The activation function decides whether a neuron should be activated or not. It is necessary in a network to prevent linearity. Some activation functions do this by mapping the values between zero and one. In this research the Rectified Linear Unit (ReLU) is used for activation. This is a commonly used piecewise linear function that will output the input directly when possible (see [Nair and Hinton, 2010]). All the values before the hurdle point (usually the origin) are mapped to zero, and after that are mapped as their original values. The MLP model tries the find the optimal weights and biases by calculating the loss for every observation and minimising the loss by gradient descent and backpropagating through the network. The loss function used in this paper is the Root Mean Squared Error (RMSE). Multiple layers are necessary to capture complexity, the more layers in a network the more the model can find complex structures in the data. The amount of layers is therefore an important aspect of the model and is preferably tuned to perfection. The MLP model used in this paper uses three layers. These aspects of activation, backpropagation and loss function define the training procedure and are the same for most DL models.

4.2 Residual Network

A common problem with Deep Learning models is vanishing or exploding gradients. This problem occurs when a model has many layers and the gradient becomes equal to zero or too big. This implies that when layers are added to a model, the validation and testing losses increase. Because adding layers can still be a good way of improving a model's performance, the Microsoft Research team proposed the Residual Network (ResNet) model [He et al., 2016]. This model introduces the concept of Residual blocks. A Residual block is a certain number of layers connected by a skip connection. This skip connection is a connection between the activation of a layer with a layer from the past. By using these skip connections, the certain neurons will not die out and therefore important information does not get lost.



x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
relu
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2: Normal layer connections versus Residual block [source: linkinpark]

In figure 2, the skip connection made by a residual block is shown. The initial mapping $F(x)$ has changed to $F(x) + x$. Using this type of mapping makes sure that any layer hurting the model will be skipped during the training process. The ResNet architecture is made up of multiple Residual blocks, being able to handle amounts of up to 1000 layers efficiently (in the case of He et al.).

4.3 Feature Tokeniser - Transformer

The Feature Tokeniser - Transformer (FT-Transformer) is, as the name suggests, an adaption of the widely used Transformer architecture. The Transformer architecture was proposed to solve the Recurrent Neural Network (RNN) problem of vanishing gradients when running long sequence tasks by [Vaswani et al., 2017]. They solved this problem through introducing the concept of attention in an encoder/decoder environment. Essentially both the encoder and the decoder are transformer layers. The whole network is shown in figure 9 in the appendix.

So far the Transformer architecture has mainly shown its advantages in the fields of image recognition, text prediction and translation. That is the reason why the authors of the main reference paper have decided to implement a Feature Tokeniser to the model, thereby focusing on the effectiveness with tabular data. Instead of using an encoder and decoder Transformer, they use the Feature Tokeniser and multiple layers of one Transformer.

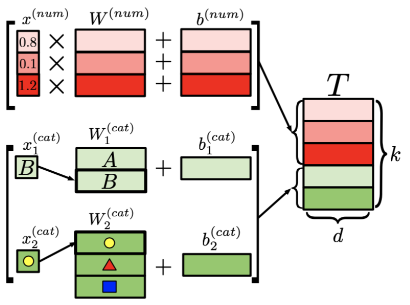


Figure 3: Feature Tokeniser [Gorishniy et al., 2021]

In figure 3, one can observe the feature information extraction by the FT part of the FT-Transformer. The FT transforms input to embeddings and is designed to handle both numerical and categorical features simultaneously. Since the data only includes continuous features, the embeddings are obtained via

$$T_j = b_j + f_j(x_j) \in \mathbb{R}^d \qquad f_j = \mathbb{X}_j \to \mathbb{R}^d$$

After this, the classification (cls) token is added, to eventually use its final representation for making the prediction.

Both the embeddings and the cls token are then fed into the stack of Transformer layers. One layer is similar to the encoder Transformer in figure 9. The Transformer part starts with a Multi-Head Attention layer. Multi-Head Attention is a way of running through multiple levels of singular attention mechanisms at once. The attention mechanism mimics cognitive attention when, for example, reading a sentence. Just like the works of [Chorowski et al., 2015] have shown. In a nutshell, an attention mechanism consists of three parts. Firstly, a process reads the embeddings and divides them into distributed vectors. Then a list of feature are memorised as a sequence of facts, which can be retrieved later. Finally, the part of the memory that is needed to most for the prediction (or e.g. translation) is exploited. This is where the name attention comes from.

After the Multi-Head Attention mechanism, the data passes the Feed Forward function. This is an MLP network, with an activation function. The used activation function is based on the ReLU. Throughout the network, there is a residual skip connection runs past both layers.

For more information on the details behind this architecture, please refer to the original paper [Gorishniy et al., 2021].

4.4 Long Short-Term Memory

The Long Short-Term Memory (LSTM) architecture is a specific type of RNN model. It was proposed by [Hochreiter and Schmidhuber, 1997] to recognise patterns in sequences of data (short-term), while keeping the general features of the data (long-term) into consideration. The LSTM model has shown to be successful at this, with implementations from multinationals like Apple and Google (see [Schmidhuber, 2021]). The architecture used is shown in figure 4.

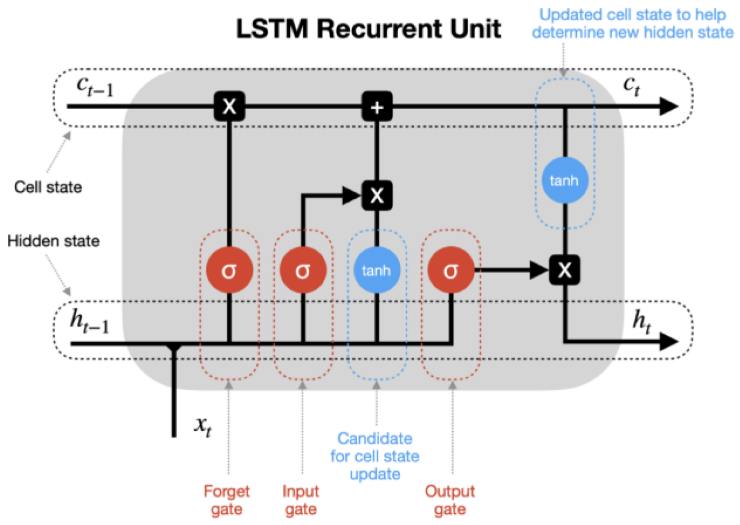


Figure 4: LSTM network [source: knowledgehut]

This network works by using two types of information. Short-term memory, the cell state c_t and long-term memory, the hidden state h_t. The network is build up of three different parts. The first part is the Forget Gate, this is the bottom-left quadrant in the diagram. Here the input x_t is combined with the previous short-term information c_{t-1}. To ignore unimportant information, the combined vector is then multiplied with an ignore factor. Then the Input Gate decides what part of the information should be forgotten, and what part should be added to the long-term memory. This is combined with the hyperbolic tangent function to indulge non-linearity. Finally the Output Gate decides what part of the information should be passed on to the next hidden state h_t, the new long-term memory. This is being done by a combination of the Forget and Input Gate output and the updated short-term memory c_t via the hyperbolic tangent function.

4.5 FT-LSTM

In this section the network of the proposed Feature Tokeniser - Long Short-Term Memory architecture is explained. The notation used in the model is:

- n: Number of observation of the input, in this case number of trading days
- d: Number of d_tokens created by the FT
- cls: Classification token
- s: Sequence length
- h: Size of the hidden state output of the LSTM

Due to the fact that this architecture needs multiple observations (for the sequences) to create one output, the proposed architecture takes all observations[2] as input. This is different to most architecture diagrams, that depict the network for only one observation. Furthermore, every small circle stands for a feature of the data. In this case there are nine features. The architecture consists of 5 layers:

1. FT: This is the Feature Tokeniser layer, which embeds the data into tokens and attaches the cls token like the FT-Transformer.
2. Linear: This is a simple linear layer like the ones in an MLP network. The Linear layer concatenates the tokenised data.
3. Sequence split: Here multiple consecutive observations are split into a sequence.
4. LSTM: This is an LSTM layer, that extracts the temporal information of the sequences.
5. ReLU Linear: Another linear layer, activated by the ReLU.

The full architecture is given in figure 5.

On the left, input is fed into the model, whereafter the FT extracts the relevant information on the feature level and adds the classification token. This information is then combined by a linear layer into one vector. The sequence split method conjugated these different vectors into sequences of size s. This is important, as it preserves the temporal correlations of the data. Since the sequences will be used for LSTM forecasting, we end up with n minus s sequences. In the diagram s is depicted as 3, and in the real program s can range from 2 till 20. These sequences, including the cls token are fed through an LSTM network. This ends up as a hidden state output of size h, which was 72 after optimisation. Then finally the information is formed into $n - s$ forecasts by means of an ReLU activated linear layer.

To summarise, this architecture is specifically created to forecast stock indices. It does this by means of the following. The FT parts captures feature information of the input. This ensures that, because the data is tabular, one essentially has good benchmark embeddings. The splitting into sequences makes sure that these embeddings will keep related to the time series. The LSTM model then is able to capture as much information as possible of each sequence and its state-of-the-art short term memory network can hopefully require optimal forecasts. In the next section, all models are deliberately tested and compared.

[2] or all observations of the batch, if batches are used

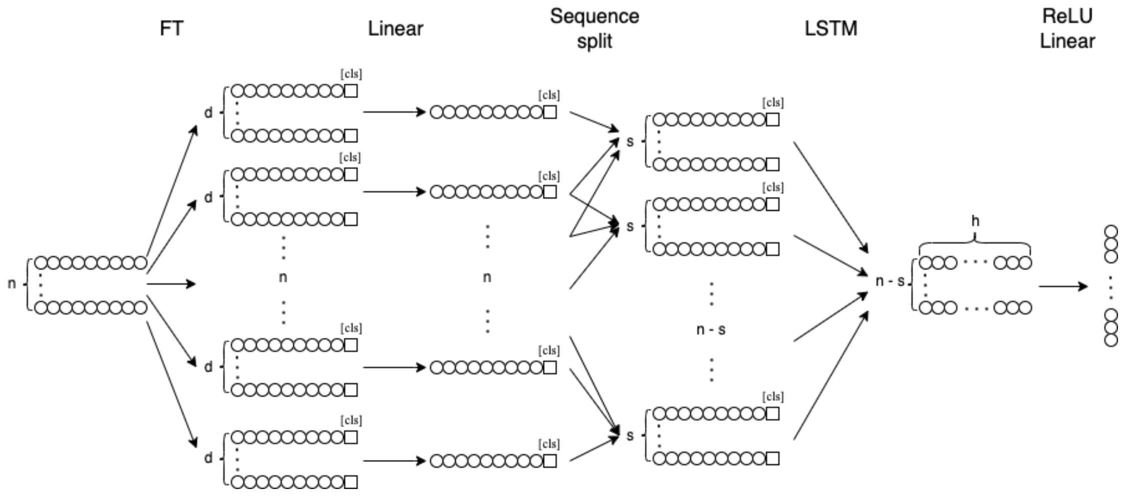


Figure 5: FT-LSTM network

5 Results

5.1 Reproduction

In this section some of the results of the main reference paper are replicated. The dataset used for the replication is the California Housing dataset and all of parameters and settings are similar to the ones used by [Gorishniy et al., 2021], after tuning. The three tested architectures are: Multilayer Perceptron (MLP), Residual Network (ResNet) and Feature Tokeniser (FT) Transformer. The metric used for evaluation is the root mean squared error (RMSE). The results can be seen in table 3.

Table 3: Replication RMSE values

	MLP	ResNet	FT-Transformer
Training set	0.464	0.422	0.358
Validation set	0.588	0.673	0.574
Testing set	0.634	0.658	**0.572**

Compared to the results of the original paper there is a slight difference in accuracy. The values of the reproduction are 0.572, 0.658 and 0.634 for the FT-Transformer, ResNet and MLP, respectively, for the testing set. Whereas these values are 0.459, 0.486 and 0.499 in the original paper for the same models. The RMSE values for all three models are higher in the reproduction. This difference can be explained by multiple reasons. Firstly, the authors could have used different seeds and/or settings during the programming. Or, they have tested and optimised the models more vigorously and therefore found results that are more accurate. Nevertheless, the main takeaway from the paper can still be seen from the replication values. The FT-Transformer clearly outperforms the ResNet and the MLP model.

5.2 Baseline setup

Deep Learning models are complex and can be good at specific, but different tasks. Making small changes to the model input, can lead to extensive changes in the output. These facts are the reason why each DL model will have a different data input for the baseline setup. This is unlike one would do for conventional econometric models, where it can create bias. DL models all handle information differently and also need different kind of input[3]. In this section, the baseline data, specifically the amount of lags used, will be explained and deliberated for each model.

Table 4: MLP data inclusion

	1-day lag	2-days lag	10-days lag
Training loss	83.68	100.81	153.52
Validation loss	127.44	**113.31**	157.65
Testing loss	174.63	226.69	814.32

Table 5: FT-Transformer data inclusion

	1-day lag	2-days lag	10-days lag
Training loss	80.21	122.46	61.96
Validation loss	241.83	**169.09**	328.73
Testing loss	672.57	303.81	559.01

Table 6: LSTM data inclusion

	1-day lag	5-days lag	10-days lag
Training loss	164.87	106.31	24.11
Validation loss	114.20	**107.24**	196.30
Testing loss	195.59	391.25	606.41

In tables 4, 5 and 6, the amount of lags included in the input data is altered for the MLP, FT and LSTM model and the respective RMSEs are shown. Doing the same analysis for the ResNet model was thought unnecessary, because of the lack of performance. Due to the way the FT-LSTM network is defined it is not necessary to alter the input lags for this model. The model requires all observations as input and the lags are created within the model architecture.

For both the MLP model and the FT-Transformer, the optimal baseline setting is using two days of lagged variables. For the FT-Transformer this not only leads to the lowest validation RMSE, but also causes the least amount of overfitting. The difference between the three RMSEs is the lowest for 2-days of lag. This makes sense, in a way that the necessary information can be found in the first two lags. Where ten lags give so much information, that the results can be a pattern of overfit on the training. The LSTM model finds its best setup when using sequences of 5-days lag. It is important to note that in this case the chosen amount does not lead to the best testing results. Choosing 1 day of lag here would be unbiased, because optimisation is always done with regards to the validation losses. However, this does show that perhaps the LSTM model has more potential.

[3]e.g. the LSTM model needs sequences and the MLP model only needs the features.

5.3 Financial data results

For the analysis of the DL architecture we use lags of all the variables, the amount of which is specified in subsection 5.2. The hyperparameter optimisation is done using 5 fold cross-validation in Optuna [Akiba et al., 2019], where the accuracy is measured using the validation set. The time horizon on which the target is forecast, is one day and ten days (two weeks of trading). We use inverse transformation to counteract the preprocessing transformations. In this way, the results are visualised as and the RMSEs are calculated as values that are representative to the real index.

Table 7: 1-day ahead forecast RMSE values

	MLP	ResNet	FT-Transformer	LSTM	FT-LSTM
Training loss	100.81	560.33	122.46	106.31	62.97
Validation loss	113.31	291.27	169.09	107.24	129.57
Testing loss	226.69	905.29	303.81	391.52	**115.58**

Looking at the results of the one-day ahead forecasts in table 7, it becomes clear that with this setting all the models are not able to predict the S&P very well. The results of models are more than a hundred RMSE points off the target[4]. This is, given that we are only forecasting one day ahead, a large error. It is more important, however, to compare the different models against each other.

The proposed model clearly outperformed the other models, with the testing loss of 115.58 being substantially lower than the others. The forecasts of the MLP model and the FT-LSTM model have been compared with the Diebold and Mariano (DM) test[5], which resulted in a p-value of 0.000 in favour of the FT-LSTM. Because the MLP model is the second best performing model, we can conclude that the outperformance of the proposed model is significant. Particularly interesting to note is that the FT-Transformer and LSTM are both not the second best performing model on its own, but when combined become the best. It is also the model with the RMSEs of the validation and testing set being closest to each other. They differ 14 points for our model and over 100 points for the others. This is a good sign, because it means that the patterns found in the training set work well for both the validation and testing (and are thus more likely to be real patterns of the target variable). Furthermore, the ResNet model does way poorer than the other models, in all three of the data sets. The losses are over five times as large. This leads to believe that the ResNet model is not able to capture the behaviour of the stock index. This statement is confirmed later on, when looking at the forecast plots.

In table 8, the results of the more difficult task of predicting ten days ahead are presented. Since DL models can find information links that other conventional methods cannot, it will be useful to evaluate the models on this harder task where the conventional models would fail.

Here we can see similar results as with the one day ahead predictions. The RMSE values of the models have increased slightly. This is a good sign, given that the task has gotten considerably harder.

When comparing the models, the results are also very similar. The FT-LSTM model now is performing even better than the other models, with the RMSE of 141.27 being at least two times lower than the rest. The p-value of the DM test is again smaller than 0.00. Also for the training

[4]A simple AR(1) model on the S&P 500 obtained 53.58 as RMSE.

[5]The DM test is a way of comparing forecasts, we test the null hypothesis that the forecast errors coming from the two forecasts bring about the same loss [Diebold and Mariano, 2002].

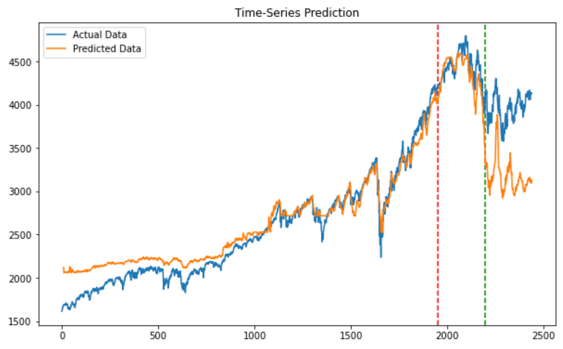
Table 8: 10-day ahead forecast RMSE values

	MLP	ResNet	FT-Transformer	LSTM	FT-LSTM
Training loss	107.66	601.72	118.46	85.72	121.66
Validation loss	201.62	268.71	222.38	155.91	184.63
Testing loss	316.71	524.49	628.30	541.34	**141.27**

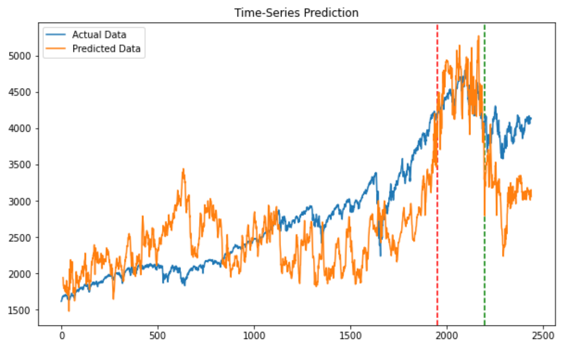
set, the proposed model fits the data best. Interestingly, the FT-Transformer and the LSTM architecture are now the worst performing models on the testing set. These models also show the most evidence of overfitting, because the difference between the training and testing losses is the largest. A possible explanation for this is the fact that both architectures are designed for specific tasks and therefore lack the capability of being consisted in this stock market setting.

In figure 6, the model predictions are plotted against the actual data, with the training split and testing split in the dataset displayed with red and green dotted lines respectively. The MLP model does reasonably well in the second half of the training set and the validation set, but does not seem to be able to capture all of the features of the target in the test set and the first half of the training. The model predicts values that are too high at the start and too low in the end. When looking at figure 6b, one can conclude that the ResNet architecture is not suitable for time series forecasting, with the predictions being way too volatile and all over the place. Both the FT-Transformer and the LSTM model fit the training set well. The Feature Tokaniser based Transformer model does not observe a pattern during the validation period, where the long short-term memory behaves more like the actual data. In the testing period, both of these models predict the index too low. This might be explained by the models having anticipated too much on the covid-19 crisis and therefore predicting something similar would happen when the stocks went down in March 2022, but then overestimated it. When the two models are combined into the proposed model, the best graphical forecasts are obtained. The predictions resemble the actuals nicely, with only small discrepancies at some points.

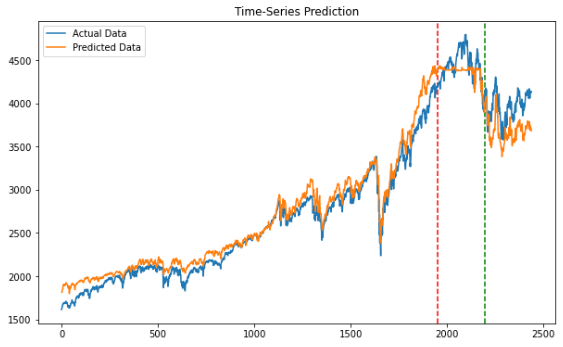
The same plots for the ten day forecasting horizon are given in the appendix in figure 11. The results from the RMSEs correspond well with their visual counterparts. All models fit the data slightly worse than the one day aheads during the testing set. The ResNet model now does better from the red line onwards, but the horrendous fit in the training time period still causes the model to be considered useless. When taking an up-close look at the testing forecasts for the FT-Transformer and LSTM architectures, one can see the 10 day lag, with the forecast values being slightly behind the real values. This is not the case with the FT-LSTM combination, which can be a promising development.



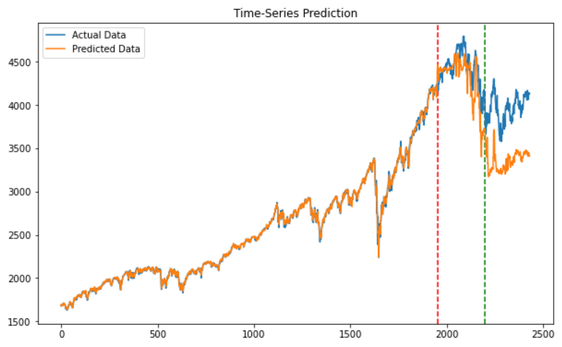
(a) MLP



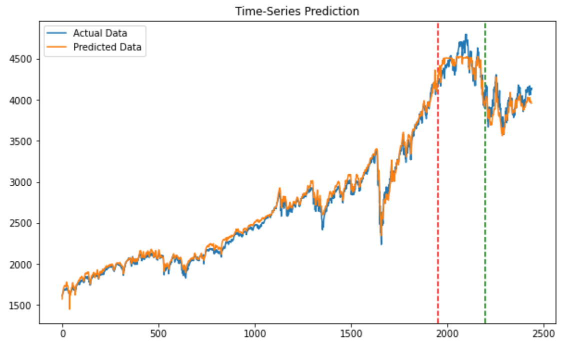
(b) ResNet



(c) FT-Transformer



(d) LSTM



(e) FT-LSTM

Figure 6: One day ahead forecasting plots. The S&P 500 closing price values are plotted against the forecasted values. The horizontal axis displays the amount of observations. The red and green dotted lines denote the testing and validation split, respectively.

5.3.1 Tuning

To get insight in the tuning/training procedure, losses over the training epochs are plotted for the FT-LSTM model in figure 7. Both the training and validation losses decrease towards zero, albeit in a somewhat volatile way. The volatility can be explained by multiple reasons. Firstly, the learning rate and therefore step sizes of the gradient descent are too large. This is not very likely, as this parameter is tuned to obtain the lowest losses. More likely is that the model has not found its optimum yet and therefore act more volatile, which seems to be the case as the losses eventually both steady. Due to the training of the model, the training losses steady sooner than the validation losses. But after 87 epochs the validation losses have also steadied around a value close to zero, just above the training loss. At that point the early stopping mechanism[6] kicks in and the optimal result is achieved.

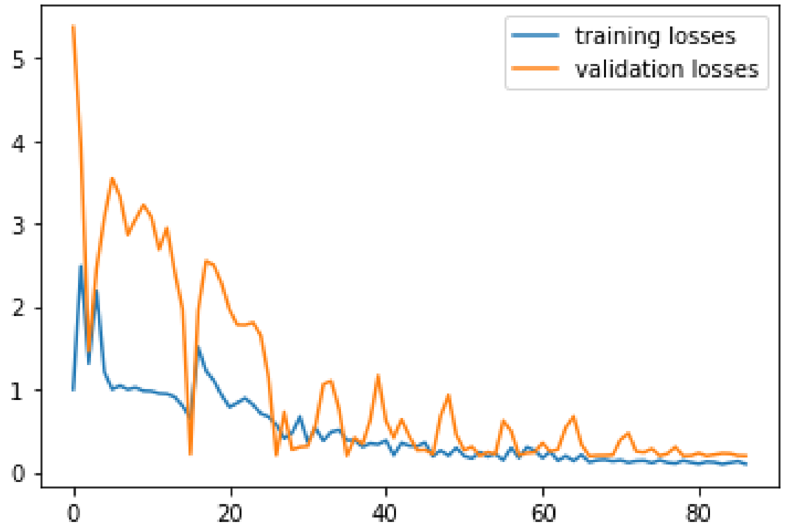


Figure 7: FT-LSTM epoch development. The amount of epochs on the horizontal axis is plotted against the RMSE losses on the vertical axis.

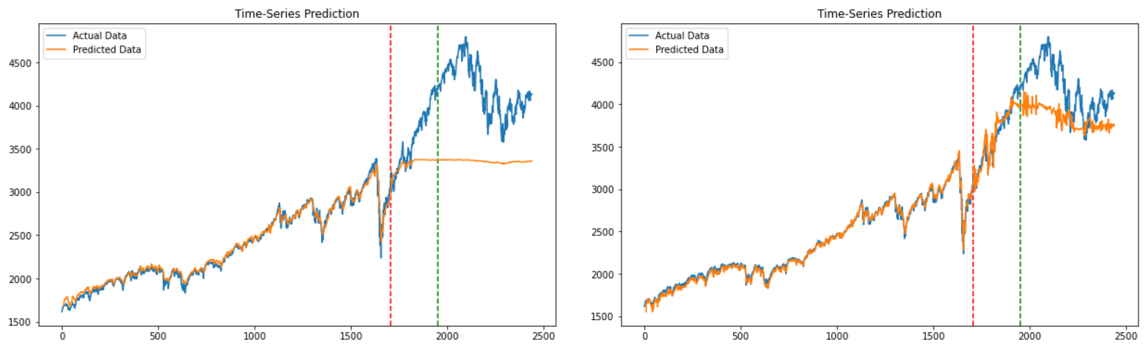
5.3.2 Robustness

With the results, two checks regarding robustness of the FT-LSTM can be done. The first is the multiple horizon test. Due to the fact that the model performed better in both forecasting horizon, the model is robust against this change in task. The second one is the change in training epochs. From figure 7, it turns out that the last ten epochs generate the same performing model, because the losses are all similar to the final epoch. This means that the volatility of the losses has settled and the model is robust to a slight change in the amount of training.

[6]This is a way of not having to run the training longer than necessary, by stopping when the average loss of the validation starts increasing instead of decreasing.

5.3.3 Overfitting

One of the main advantages of Deep Learning is the capability to find structures and patterns in the data that humans or conventional econometric methods cannot. However, these links can be so complex that even something that in real life has no influence on the target at all can get spuriously correlated to obtain lower losses. Overfitting is when this happens too much and when the model predicts the training data too well compared to the testing data. It then fits the errors of the training set as well, instead of only the distribution of the data generating process. This phenomenon is one of the dangers of DL. In this section, overfitting is discussed and explained how it is treated.



(a) Case I (b) Case II

Figure 8: Overfitting of the MLP model predictions. Stock prices on the vertical axis, number of observations on the horizontal.

To get a good understanding of overfitting, we divide it into two cases. Case I is the simple case of overfitting; The model derives good forecasts for the training data, but as soon of the model is applied to data it has not been trained on, it cannot find the same results anymore. To get visual insights, the MLP model forecasts have been plotted in figure 8. Here the 70-10-20 train-validation-test split has been used. In case I, the model predicts the target during the training set and is only able to predict a single value afterwards.

Case II is overfitting the validation set. This usually happens when a model is overly tuned and performs well on the validation set (sometimes even better than one the training set), but does not do so on the testing set. We can see this happening in figure 8b. Even though the model is only trained on the training set, a certain parameter tune could still lead to these spurious relations in the validation set. A solution could be to tune the model less or to tune it on different validation sets. In the main results, one can argue that this happened with the ResNet and FT-Transformer model, when looking at figure 6b and 6d. However, a different explanation for Case II is that the data generating process changed after the validation data set. In this case there is not overfitting. It is always important to be aware of overfitting in DL and to consider it when drawing conclusions.

6 Conclusion

In this section the findings of the paper are summarised and the research question is being answered. The proposed FT-LSTM model contributes to the research, because it is the model that focuses on the prediction of the stock market, therefore having the potential of becoming a good predictive tool for investors. The stock market is represented by the S&P 500 index, which averages 500 individual stocks. The features used for the Deep Learning analysis are the lags of 8 other economic variables.

The FT-LSTM model is a combination of two of the leading DL architectures. The FT-Transformer is an encoder-decoder structured model that captures feature based information. In this paper we found that indeed the FT-Transformer works well on tabular data, as stated by [Gorishniy et al., 2021]. The Feature Tokeniser part is combined with the Long Short-Term Memory model, after introducing a sequence split to the obtained tokenised data. This model is compared to the MLP, ResNet, FT-Transformer and LSTM model itself and we found that it significantly outperformed these models. Both on the one day and ten day horizon the model obtained lower RMSE values. Furthermore, this model is robust to parameter changes and showed the least overfitting. This leads us to answer the research question

Can a combination of the FT-Transformer and LSTM model predict the stock market well, compared to other Deep Learning methods?

with a clear yes.

There were however several shortcomings and downfalls during this research, which led to the following ideas for further research. First of all, the model is only tested on one dataset. The model would be even more robust and widely applicable if the obtained results hold for multiple datasets and target variables. If this is the case, the model could become one of the DL benchmarks. Another shortfall could be the issue with overfitting because of the train, validation and test split. In this paper, we wield a 80-10-10 split of the data, such that the whole covid crisis is part of the training set. Usually with DL analyses one would choose a larger testing set than 10% in order to get a better view of the actual performance and a better comparison between the different models. A suggestion for future research is to choose a different split, e.g. 70-10-20 and perhaps introduce some outlier detection methods and handle the outliers as such. A study on this has been done by [Franses and Haldrup, 1994]. Next, there are limitless possibilities of combining two models into a hybrid model. In this paper only one is given, with educated reasoning behind the combining choices. Two suggestions for model combinations that can be obtained from the FT-LSTM architecture are: Seperating the CLS token before the LSTM network and adding it back in at the end; Or mean pulling the tokens together instead of using a Linear layer. Finally, this model is compared to other Deep Learning methods. In order to know if it has the possibility to become a real benchmark model for stock market predictions it should also be compared to Machine Learning models and conventional methods. Possible Machine Learning models are Random Forest, Gradient Boosting Decision Trees, Support Vector Machine. Which have proven to also be effective to (see [Strader et al., 2020]).

References

[Akiba et al., 2019] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

[Andayani et al., 2022] Andayani, F., Theng, L. B., Tsun, M. T., and Chua, C. (2022). Hybrid lstm-transformer model for emotion recognition from speech audio files. *IEEE Access*, 10:36018–36027.

[Chorowski et al., 2015] Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. *Advances in neural information processing systems*, 28.

[Diebold and Mariano, 2002] Diebold, F. X. and Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & economic statistics*, 20(1):134–144.

[Franses and Haldrup, 1994] Franses, P. H. and Haldrup, N. (1994). The effects of additive outliers on tests for unit roots and cointegration. *Journal of Business & Economic Statistics*, 12(4):471–478.

[Gorishniy et al., 2021] Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. (2021). Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943.

[Hazimeh et al., 2020] Hazimeh, H., Ponomareva, N., Mol, P., Tan, Z., and Mazumder, R. (2020). The tree ensemble layer: Differentiability meets conditional computation. In *International Conference on Machine Learning*, pages 4138–4148. PMLR.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Heaton et al., 2017] Heaton, J. B., Polson, N. G., and Witte, J. H. (2017). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 33(1):3–12.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Huang et al., 2020] Huang, X., Khetan, A., Cvitkovic, M., and Karnin, Z. (2020). Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*.

[Ignatov et al., 2019] Ignatov, A., Timofte, R., Kulik, A., Yang, S., Wang, K., Baum, F., Wu, M., Xu, L., and Van Gool, L. (2019). Ai benchmark: All about deep learning on smartphones in 2019. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3617–3635. IEEE.

[Ivakhnenko et al., 1965] Ivakhnenko, A. G., Lapa, V. G., et al. (1965). Cybernetic predicting devices. *(No Title)*.

[Ji et al., 2021] Ji, Y., Liew, A. W.-C., and Yang, L. (2021). A novel improved particle swarm optimization with long-short term memory hybrid model for stock indices forecast. *IEEE Access*, 9:23660–23671.

[Katayama et al., 2019] Katayama, D., Kino, Y., and Tsuda, K. (2019). A method of sentiment polarity identification in financial news using deep learning. *Procedia Computer Science*, 159:1287–1294.

[Lim et al., 2021] Lim, B., Arık, S. Ö., Loeff, N., and Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764.

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

[Nikou et al., 2019] Nikou, M., Mansourfar, G., and Bagherzadeh, J. (2019). Stock price prediction using deep learning algorithm and its comparison with machine learning algorithms. *Intelligent Systems in Accounting, Finance and Management*, 26(4):164–174.

[Niu et al., 2020] Niu, H., Xu, K., and Wang, W. (2020). A hybrid stock price index forecasting model based on variational mode decomposition and lstm network. *Applied Intelligence*, 50:4296–4309.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

[Popov et al., 2019] Popov, S., Morozov, S., and Babenko, A. (2019). Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*.

[Schmidhuber, 2021] Schmidhuber, J. (2021). Our impact on the world's most valuable public companies. *AI Blog*.

[Sharaf et al., 2021] Sharaf, M., Hemdan, E. E.-D., El-Sayed, A., and El-Bahnasawy, N. A. (2021). Stockpred: a framework for stock price prediction. *Multimedia Tools and Applications*, 80:17923–17954.

[Shwartz-Ziv and Armon, 2022] Shwartz-Ziv, R. and Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90.

[Siami-Namini and Namin, 2018] Siami-Namini, S. and Namin, A. S. (2018). Forecasting economics and financial time series: Arima vs. lstm. *arXiv preprint arXiv:1803.06386*.

[Strader et al., 2020] Strader, T. J., Rozycki, J. J., Root, T. H., and Huang, Y.-H. J. (2020). Machine learning stock market prediction studies: review and research directions. *Journal of International Technology and Information Management*, 28(4):63–83.

[Tao et al., 2019] Tao, Z., Han, L., Song, Y., and Bai, K. (2019). Stock market reactions to the 2011 off the pacific coast of tohoku earthquake. *International Journal of Disaster Risk Reduction*, 41:101294.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Wu et al., 2015] Wu, R., Yan, S., Shan, Y., Dang, Q., and Sun, G. (2015). Deep image: Scaling up image recognition. *arXiv preprint arXiv:1501.02876*, 7(8):4.

[Young et al., 2018] Young, T., Hazarika, D., Poria, S., and Cambria, E. (2018). Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75.

[Zhong and Enke, 2019] Zhong, X. and Enke, D. (2019). Predicting the daily return direction of the stock market using hybrid machine learning algorithms. *Financial Innovation*, 5(1):1–20.

7 Appendix

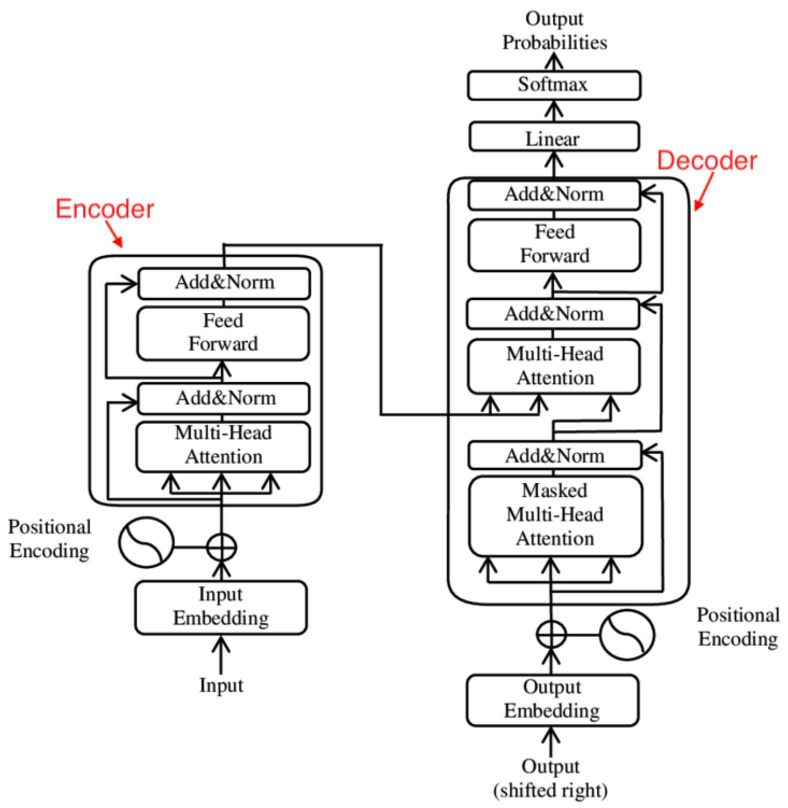


Figure 9: Transformer model architecture [source: wikipedia]

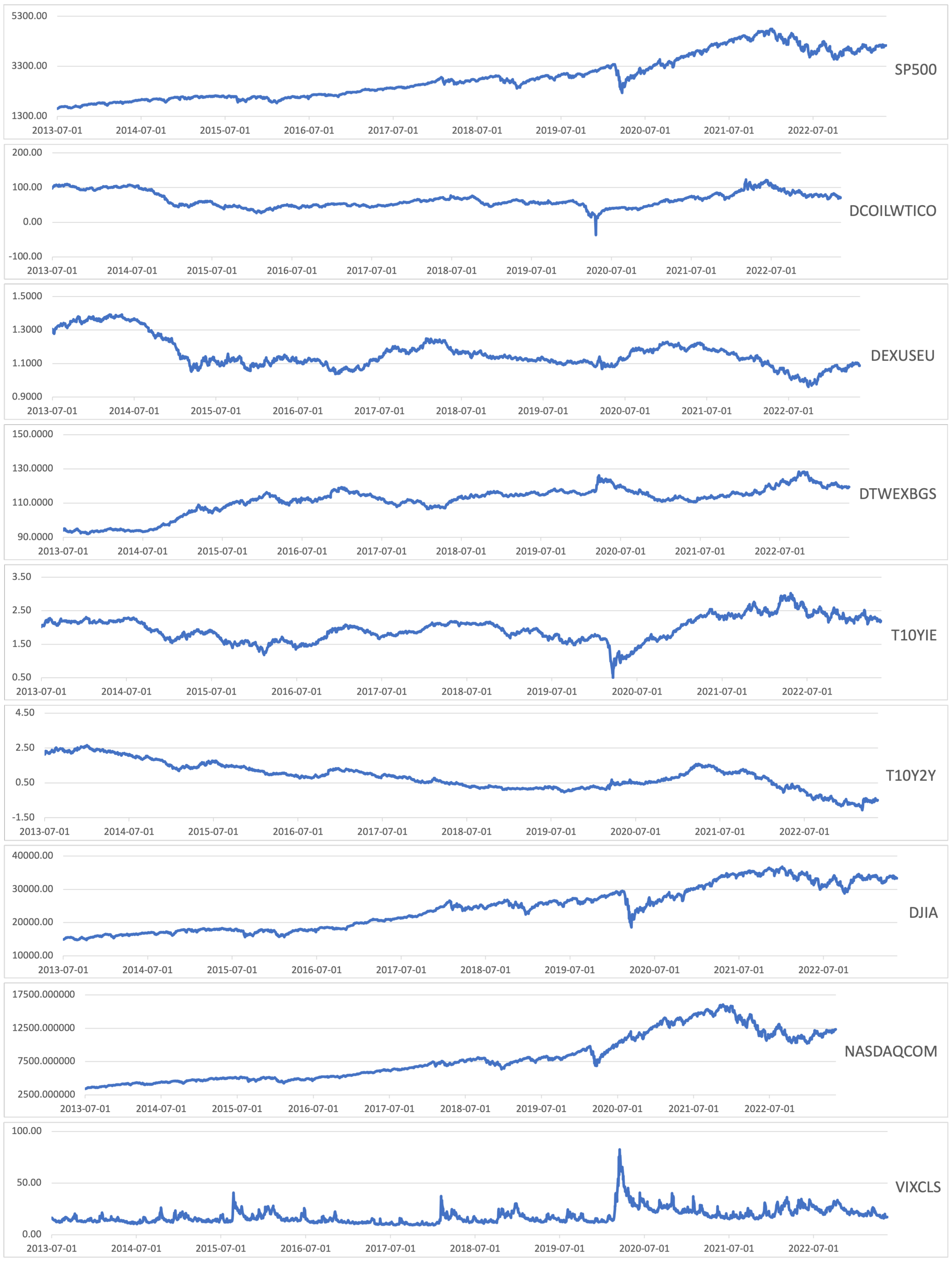
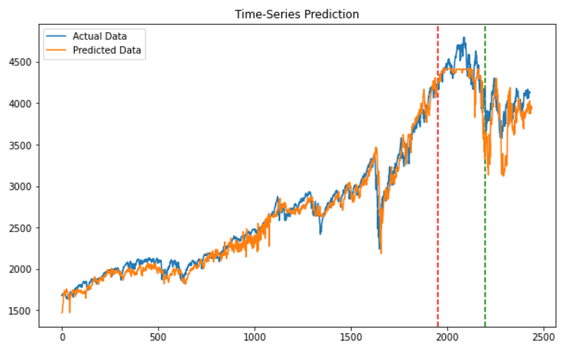
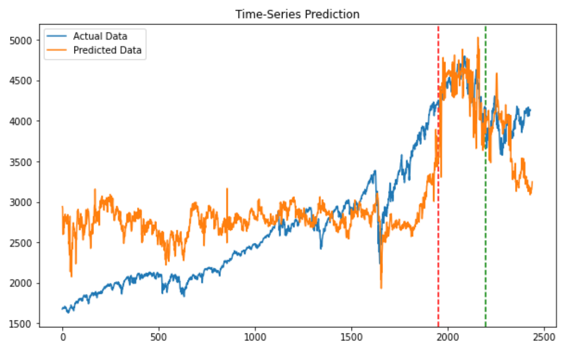


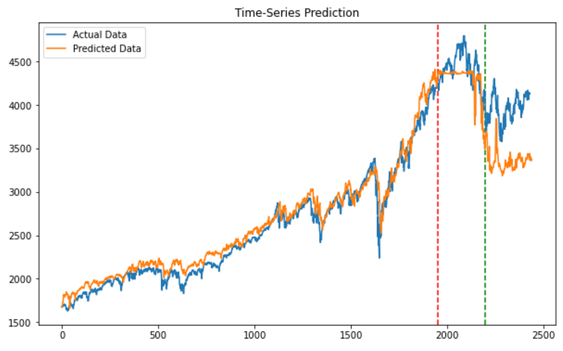
Figure 10: Plots of the data series for all the variables. The y-axis shows the unit of each variable, the x-axis states the time. All variables are described in section 3



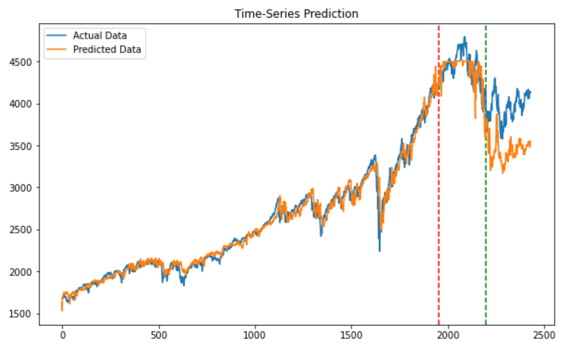
(a) MLP



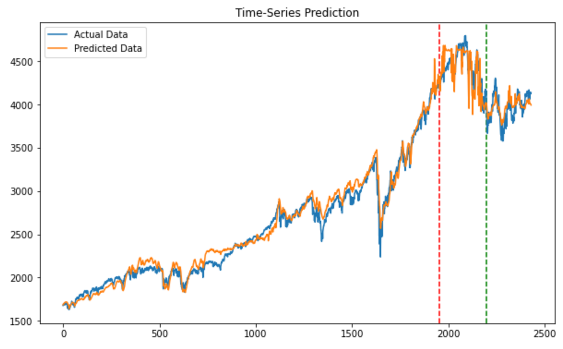
(b) ResNet



(c) FT-Transformer



(d) LSTM



(e) FT-LSTM

Figure 11: Ten day ahead forecasting plots

Table 9: FT-LSTM model default parameter configuration

	One day ahead	Ten days ahead
Lags (s)	3	10
Hidden size (h)	66	72
d_token (d)	144	52
L	469	202
n_heads	8	4
Learning rate	0.073557	0.059692
Weight decay	4.520512e-5	2.197347e-5
Epochs	87	80

7.1 ReadMe

The code for replication can be interpreted in the following way;

1. The used data can either be obtained from FRED, or to get the exact same dataset, the data1.csv file is added in the code. The replication data is obtained from the skicit-learn package in python. To preprocess and split the data in the training-validation-testing sets, the fetchdata.py file is used.

2. Tuning is done in the train-XXX.py files, with XXX corresponding to the model names. In the train-XXX.py files, the parameters to be optimised and the amount of tuning can be specified.

3. With the optimised parameter set, one can test, plot and evaluate the model in the test-XXX.py files[7].

[7]It is recommended to start with the MLP model, since for that model the code has been fully commented. After that, the other models are similar.