LaTeX Template for ECE417 Final Project Report

Konstantinos Milas kmilas@e-ce.uth.gr
Athanasios Papanikolaou atpapanikolaou@e-ce.uth.gr
Antonios Tragoudaras tantonis@e-ce.uth.gr

Abstract

In a world full of economic uncertainty and unpredictable changes some experts predict that the solution to stability and economic prosperity might lie in the crypto currencies. Cryptocurrencies are to most people something strange and unusual and to many totally unknown, which makes it even more difficult to be trusted as this miraculous solution of the experts. However, anyone that dives into this strange topic can find a lot of opportunities that euros or dollars cannot provide due to their nature, the only question is, how can we use cryptocurrencies to stabilise and even evolve our economy when their prices of these cryptocoins are not stable themselves? The solution is one ... predicting them.

1. Introduction

Our idea was to make as much as possible accurate predictions for the price of the ethereum cryptocoin. It was something that we were thinking for a lot of time especially after we started attending the lecture of the professor Manolis Vavalis and his subject of Blockchain Technology. However, no matter how hard we visualized this idea some specific articles and work made us to transform it into a full fledged project.

1.1. The beginning

Starting with the most important one, Cryptocurrency Analysis and Predictions Utilizing Deep Learning by Thanasis Zoumpekas, Manolis Vavalis and our professor of this subject Elias Houstis. This work was very important for us so to understand how to properly present such a project and a detailed introduction into this idea.

1.2. Articles for regression techniques

The next two articles (https://towardsdatascience.com/stock-market-analysis-using-arima-8731ded2447a) and (https://medium.com/@randerson112358/build-a-bitcoin-price-prediction-program-using-machine-learning-and-python-89f3dc6cb3b1) are both from the amazing site of Medium and they provided us with a lot of practical programming knowledge of how to actually make these predictions of the prices.

1.3. Related work for deep learning

Following the same steps, another great article from Medium was (https://towardsdatascience.com/cryptocurrency-price-prediction-using-deep-learning-70cfca50dd3a) helping in the same way like before but this time for deep learning algorithms. Lastly the code from two people from the github uploads specifically (https://github.com/sudharsan13296/Bitcoin-price-Prediction-using-LSTM/blob/master/Bitcoin%20price%20prediction%20(Time%20series)%20using%20LSTM%20RNN.ipynb?fbclid=IwAR2qfzWC1Sh6xDWENtZ3mpfcxxqDDdL14Z7rZ\YJwVm3CKCpWRWIi6v5Nxkk) and (https://github.com/neocortex/lstm-bitcoin-prediction/blob/master/lstm-bitcoin-prediction.ipynb?fbclid=IwAR3l8r9swRrm23Ky67jb1WVQHWPRyn6X4yZc-0fHtuQuhBlnBAfFUjdsIv0) were very helpful as well.

2. Software

Usually for these projects we use jupyter notebook from anaconda3 but this time due to the time-consuming LSTM algorithm we used google collab which builds the network structure, feed forward and back propagation of the algorithm much faster. Apart from google collab we also used scikit-learn and keras for machine/deep learning frameworks and pandas and numpy for data preprocessing,

3. Hardware

There is nothing notable to mention from the hardware point of view, only that through google collab we use their GPU (12GB NVIDIA Tesla K80) power instead of ours which speeds everything up (hardware acceleration).

4. Dataset

The dataset that we used for the ethereum cryptocoin prices was provided from (https://uk.finance.yahoo.com/quote/ETH-USD/history?p=ETH-USD) which is daily updated and always accurate for years . The lines are the days the prices were documented and the columns are the features some of which we used for the predictions. More specifically:

Open : Opening price on the given day

High : Highest price on the given day

Low : Lowest price on the given day

Close : Closing price on the given day

Volume : Volume of transactions on the given day

Market Cap : Market capitalization in USD

5. Proposed Methods

To start let's briefly mention and describe the algorithms used in the project as mentioned previously.

5.1. SVM

Support-vector machines (SVMs) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis, but mostly for classification objectives. The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points. In this algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss: $c(x, y, f(x)) = (1 - y * f(x))_+$

5.2. AutoML

Automated machine learning (AutoML) is the process of automating the process of applying machine learning to any problem. AutoML provides the user with the complete set that he needs, from handling the raw data to deploying the machine learning model. This algorithm was created in order to make the machine learning problems easier to handle by everyone even if they are not an expert in the field, something that helped a lot of companies and individual make predictions for their work and future.

5.3. XGBoost

XGBoost is a decision-tree-based Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small to medium structured data, decision tree based algorithms are considered the best right now.

5.4. Random Forest

Random forest is a supervised learning algorithm which is used for both classification as well as regression but mainly used for classification problems. The algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

5.5. ARIMA

ARIMA stands for Autoregressive Integrated Moving Average models. Univariate (single vector) ARIMA is a forecasting technique that projects the future values of a series based entirely on its own inertia. Its main application is in the area of short term forecasting requiring at least 40 historical data points. It works best when the data exhibits a stable or consistent pattern over time with a minimum amount of outliers. It is usually superior to exponential smoothing techniques when the data is reasonably long and the correlation between past observations is stable.

5.6. LSTM

LSTM (Long Short-Term Memory network) is a type of recurrent neural network, used in the field of deep learning, capable of remembering the past information and while predicting the future values, it takes this past information into account, unlike standard feedforward neural networks. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). This type of networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.

6. Experiments

6.1. Data Preprocessing for most regression models

First of all we have to work on the preprocessing of the data, more specifically we drop all the columns except the closing price of the cryptocoin of each day. We move on to create a variable called future-days and one column called prediction that contains the price of the coin future-days from the current price.

We create a copy of this dataframe, transform it into a numpy array and drop the prediction column and the

the last future-days rows, we call it the independent X dataset and it will be used later for the train-split of the data. On the other hand we create another copy of the original dataframe, again make it into a numpy array, but this time we drop the price column and keep the prediction one minus again the last future-days rows, we call this the dependent Y dataset which will be also used in the train-split of the data. Having the X and Y datasets ready we split the data into 80% training and 20% testing.

Lastly we create another numpy array called future-days-array which contains only the price column and the last future-days rows that we want to predict.

6.2. Regression

Now that we are ready with the preprocessing part we can start experimenting with SVR, XGBRegressor and Random Forest because they all work with the same preprocessed data.

Starting with the SVR, we use the radial basis function in order to build our model and then we use the fit function to train it.

Next one in the list is the XGBRegressor where we do the same exact thing just this time with the XGBRegressor function and again with the fit one.

Lastly, again the Random Forest model is created with the RandomForestRegressor function and trained with the fit function.

However for the AutoML model we need to work a bit differently for it to function. First of all we use the H2O platform and since the data represents time series we manually add 3 lags for each predictor and feeding in total 15 lagged variables into the model. In order to create the model we use the H20AutoML function and to train it we use the train function. When the model is created it presents us some methods that it used from the best to the worst.

Lastly, like the AutoML model, the ARIMA model needs different data compared to the others. We keep the original dataframe intact without dropping any columns or rows and we make a different type of train-split specific for this model. After that we create the model using very specific numbers for the p q and d parameters.

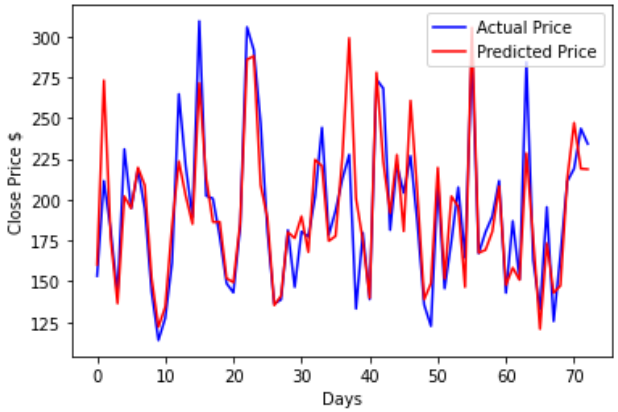
6.3. Neural Network

We have finally finished all the regression models, time to move on to the neural network and more specifically to LSTM model. From our dataset we keep only the date and price columns and with the help of the MinMaxScaler function we use a scaler to normalize the data. Once more we create 2 datasets X and Y, X for the actual prices and Y for the predicted prices and in the next step we split them for training and testing. The only difference this time is that the 2 datasets act like lists where the X dataset begins with the first 40 days and the predicts the next 3 days which go to the Y list. This sequence continues until we get the last day in the Y list.

For the creation of the model we used 2 layers with 1024 and 256 neurons respectively after a lot of different trials and errors, because these parameters provided us with the best predictions. Additionally we use the Dropout parameter in order to avoid overfitting and for activation function we used the tanh.

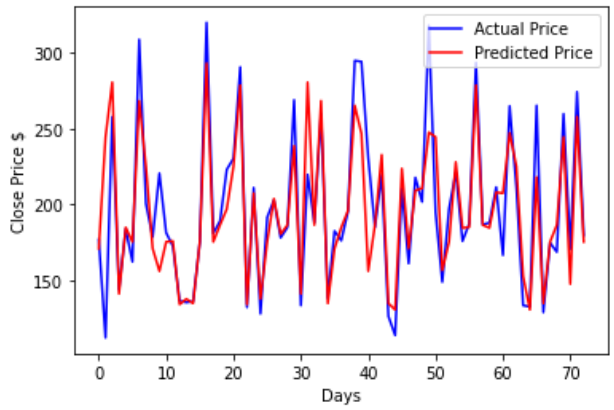
7. Results and Discussion

7.1. SVM



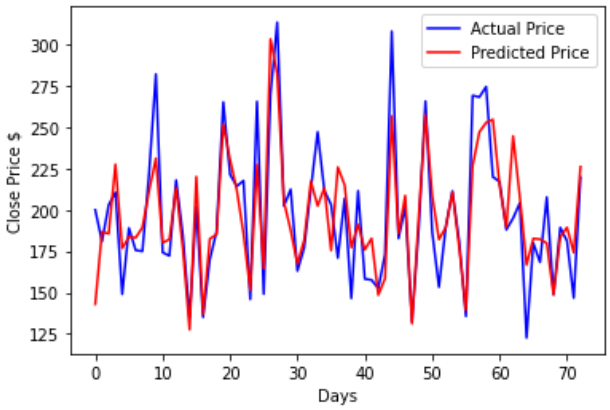
As we can see from the graph, the prediction line follows the line of the actual price every time but fails, usually when the price drops or rises extremely fast, to be identical to it. Thanks to this situation the mse is almost 321 and the smape equals to 6%.

7.2. XGBoost



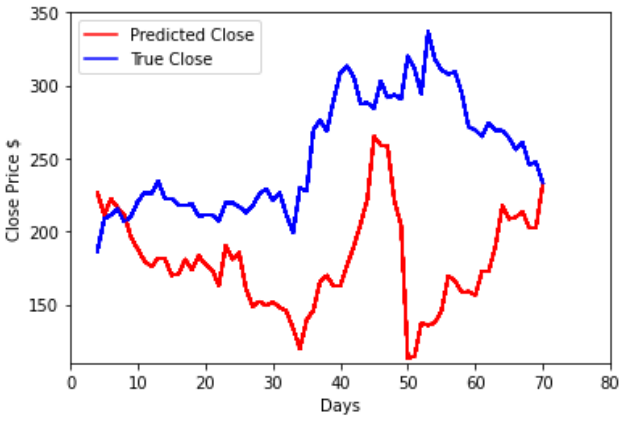
Similarly to the SVR model we can see the same pattern in the graph and almost identical errors. This is happening because the regressions methods work in similar ways so they produce similar results in our case.

7.3. Random Forest



Once again the graph is almost the same with the graphs of the two previous models for the same reason. Only this time it seems the model did a worse job with the prediction but the almost identical errors with the previous algorithms say otherwise.

7.4. AutoML

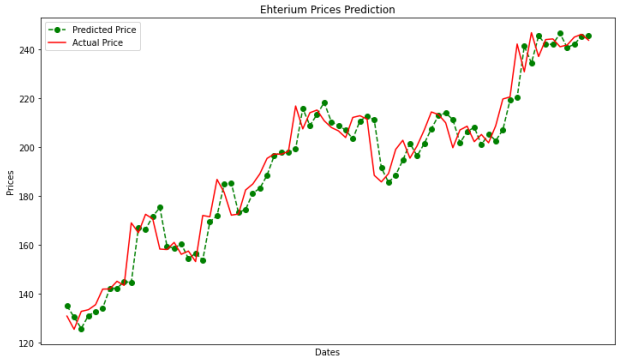


The AutoML model sadly creates a prediction that falls short compared to the actual price. In some occasions the difference between the two prices is small numerically but in some others the difference is way bigger than it should be for an accurate prediction.

Also as mentioned before, the model presents us with a lot of methods and here is the leader board.

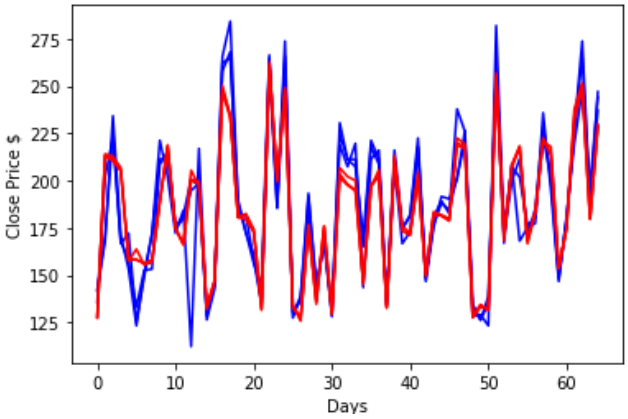
model_id	mean_residual_deviance	rmse	mse	mae	rmsle
StackedEnsemble_BestOfFamily_AutoML_20200609_212209	10.9055	3.30235	10.9055	1.09231	0.0147809
GLM_1_AutoML_20200609_212209	11.0381	3.32237	11.0381	1.39897	0.0154532
XGBoost_grid__1_AutoML_20200609_212209_model_7	12.4059	3.5222	12.4059	1.21858	0.0156249
StackedEnsemble_AllModels_AutoML_20200609_212209	14.1218	3.7579	14.1218	1.50015	0.0164565
XGBoost_grid__1_AutoML_20200609_212209_model_3	15.5001	3.93702	15.5001	1.77531	0.0180584
GBM_grid__1_AutoML_20200609_212209_model_8	17.9588	4.23778	17.9588	1.07688	0.0197538
XGBoost_grid__1_AutoML_20200609_212209_model_5	19.9671	4.46846	19.9671	2.60224	0.0205616
GBM_grid__1_AutoML_20200609_212209_model_14	20.3036	4.50595	20.3036	1.57669	0.0209526
GBM_1_AutoML_20200609_212209	20.7143	4.5513	20.7143	1.87623	0.0214447
GBM_4_AutoML_20200609_212209	22.5357	4.74718	22.5357	1.79832	0.0214722

7.5. ARIMA



As expected the ARIMA model outperformed all the other regression models. Its predictions was spot on, nearly perfect for the 20 days prediction in the future with minimal errors, an astonishing tool for predictions and time series problems.

7.6. LSTM



Similarly to the regression graphs the prediction line is identical with the real prices at some points but at some others the difference is visible. However it is the only model with the lowest, almost 0 mse.

The only other notable thing that we have to mention is that for each day predicted there is a different prediction line depicted in the graph, meaning that we have 3 different blue prediction lines in the same graph, which means it was much more demanding for the prediction.

8. Conclusions

After 6 different models and their respective graphs we have our results. The prediction of the prices was quite a difficult and surely not the most accurate work. The normal regression models were quite good but unfortunately not the best, having large errors and differencies compared to the actual prices. On the other hand ARIMA made one of the best predictions out of all models, showing why it is considered one of the best methods for time series problems.

On the same note the LSTM model also made visually a very good effort for the prediction and if we take into account, as mentioned before, the errors it is possible that it is our best result so far. Compared to the others, the LSTM was pressured into predicting even more days into the future and every time predicting 3 days, something that makes the other models more unstable.

9. Contributions

Antonios: Research, regression methods
Athanasios: Research, project report and poster
Konstantinos: Research, neural netword - lstm model