Choosing Regression Techniques with Cryptocurrency

John Neilson and William Hart

**Abstract**

This project explores cryptocurrency price prediction using regression analysis on

dataset containing many crypto currencies. Ten currencies, selected for high and low

standard deviations respectively, undergo training and testing with various regression

models. Notably, Multi-linear regression performs well for stable currencies, while

Support Vector and Lasso regression excel for less stable ones. Unexpectedly, Polynomial

regression underperforms. Evaluation metrics include Mean Squared Error and R-squared

values. The study provides insights into selecting suitable regression models based on

cryptocurrency stability, contributing to informed investment decisions in the dynamic

crypto market.

**1 Introduction**

**1.1 Motivation**

Foreseeing the price of cryptocurrencies can be valuable when looking to invest in a certain coin, therefore, being able to predict the exchange rate of a specific crypto currency using a time series regression can be a valuable tool. Additionally, when using regression analysis, there are multitudes of models. Thus, finding out which models perform better in differing datasets (with differing standard deviations and trends) can be a key asset when starting a regression analysis.

**1.2 Objectives**

Select a group of 10 currencies from our dataset, that have a diverse set of properties, to use for training and testing. Out of a handful of regression models, we will select the regression predictor model that performs the best across the ten different currencies. We will analyze and try to understand why the best performing model is the best for that specific currency time series.

**1.3 Approach**

Select 10 currencies from the overall dataset with high percent standard deviation and low percent standard deviation. In order to select the best model for each currency we will train each model through 80 percent of the data and test its performance on the last 20 percent. Then we’ll evaluate each model by calculating the Mean Squared Error (MSE) and R-squared value between the original and predicted test values.

**2 Related Works**

**2.1 Work 1**

In the academic paper titled, *Support vector regression model for predicting the sorption capacity of lead (II)*, written by Nusrat Parveen, Sdaf Zaidi, and Mohammad Danish, [11] they use Support Vector Regression (SVR) multiple linear regression (MLR) to predict the sorption capacity of lead ions with the independent variable being the initial lead concentration. SVR was mainly used with the radial basis function (RBF) kernel and optimized with 10-fold cross validation. They then compare between MLR and SVR-based models to determine the best model for the research. In result, SVR was observed performing better by the following metrics: average absolute relative error (AARE), R-squared, root mean squared error (RMSE), and mean relative error (MRE). In conclusion, they stated that the SVR model is more accurate and generalized for prediction of the sorption capacity of lead ions.

**2.2 Work 2**

The academic paper titled, *The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation*, written by Davide Chicco, Matthijs J. Warrens, and Giuseppe Jurman, [12] goes over what evaluation metrics are more valuable when conducting regression analysis. In their writing they discuss the drawback to evaluation metrics like MSE, RMSE, MAE, and MAPE; their bounds are from zero to infinity. Thus, a single value of them does not say much about the performance of the regression with respect to the distribution of the ground truth instances. So, they compared two metrics that generate a high score only when the majority of the instances within the ground truth are correctly predicted: R-squared and Symmetric mean absolute percentage error (SMAPE). They evaluated each metric by examining their mathematical properties and using each of them in real life medical scenarios. Their results conclude that R-squared is more informative and truthful than SMAPE, and of course that R-squared doesn’t have the limitations MSE, RMSE, MAE, and MAPE do. Thus, they suggest that R-squared should be used as a standard metric to evaluate regression analysis in any scientific domain.

**2.3 Work 3**

The academic paper titled, *Development of Multilinear Regression Models for Online Voltage Stability Margin Estimation.* This paper investigates the use of reactive power reserves (RPR) as an indicator to estimate voltage stability margin (VSM) in an online environment. The methodology relies upon the relationship between system-wide RPRs and VSM. Statistical multilinear regression models (MLRM) are utilized in order to express how variations in RPRs can be transformed into direct information about VSM. Data regarding RPRs and system VSM are obtained through an offline voltage stability assessment (VSA) and stored in a database for further MLRM development. Different load increase directions and a comprehensive list of contingencies were considered to account for uncertainty present in real-time operations. Once properly designed and validated, the MLRMs are ready to be used in the online environment. The methodology is tested on the IEEE 30-bus system and a real size test system containing 1648 buses. Preliminary results show that MLRMs can be successfully employed in online VSM estimation.

**3 Methodology**

**3.1 Data Collection**

**3.11 Dataset Contents & Structure**

The dataset originally comprised 876 different currencies with a total of 645,785 entries. Each entry consisted of a date, symbol, open price, high, low, close price, volume, market cap. Also for organization there was an entry number that would mark the specific entry for the currency.

**3.12 Dataset Collection method**

We had obtained a csv file containing all of the data that was used in the project. For picking specific data we had to obtain the average value of each currency over the time frame in the csv. Then data was picked based on the value of the percent standard deviation and ranked so that we could take the 5 highest values and 5 lowest values.

**3.21 Dataset Outlier Analysis**

When compiling the set of currencies to be used for training, there was an issue with the currencies that were to be selected for low standard deviations. As the dataset was not filtered there was a group of five currencies that had zero standard deviation from the mean. As this would not be good for regression analysis we opted to use the first five currencies that had low standard deviations that were not zero.

**3.2 Data preprocessing**

**3.21 Data Segmentation**

Before beginning the experiment we split the data into a training set and a testing set for each currency. When performing the split we did it based on the date of the entry so that we can use the earlier points to predict later points. The percentage of the splits was 80 percent for training and 20 percent for training.

**3.22 Data Outlier Detection & Replacement**

To identify outliers within each currencies time series we used the isolation forest algorithm. This algorithm runs in a linear time complexity, similar to K-nearest neighbors. The algorithm works by using the characteristics of an outlier to its advantage; there will only be few outliers and they are always very different. Isolation forest uses binary search trees (BSTs) to isolate an instance. The algorithm creates an ensemble of BSTs that recursively generate partitions by randomly selecting a split value for our currencies feature (Open value in USD). In theory, the outlier will be the easiest to split, usually leading to no additional leaves under its node. Each instance will be given a score on how easily they are isolated after X amounts of rounds. The data points that have abnormal scores will then be marked as anomalies.

After finding an outlier within our data we had to find a measurement to set it to, as we didn’t want to leave gaps in our time series. We decided to replace each outlier with the median value of each time series, as replacing it with the mean value could skew more in the outliers favor.

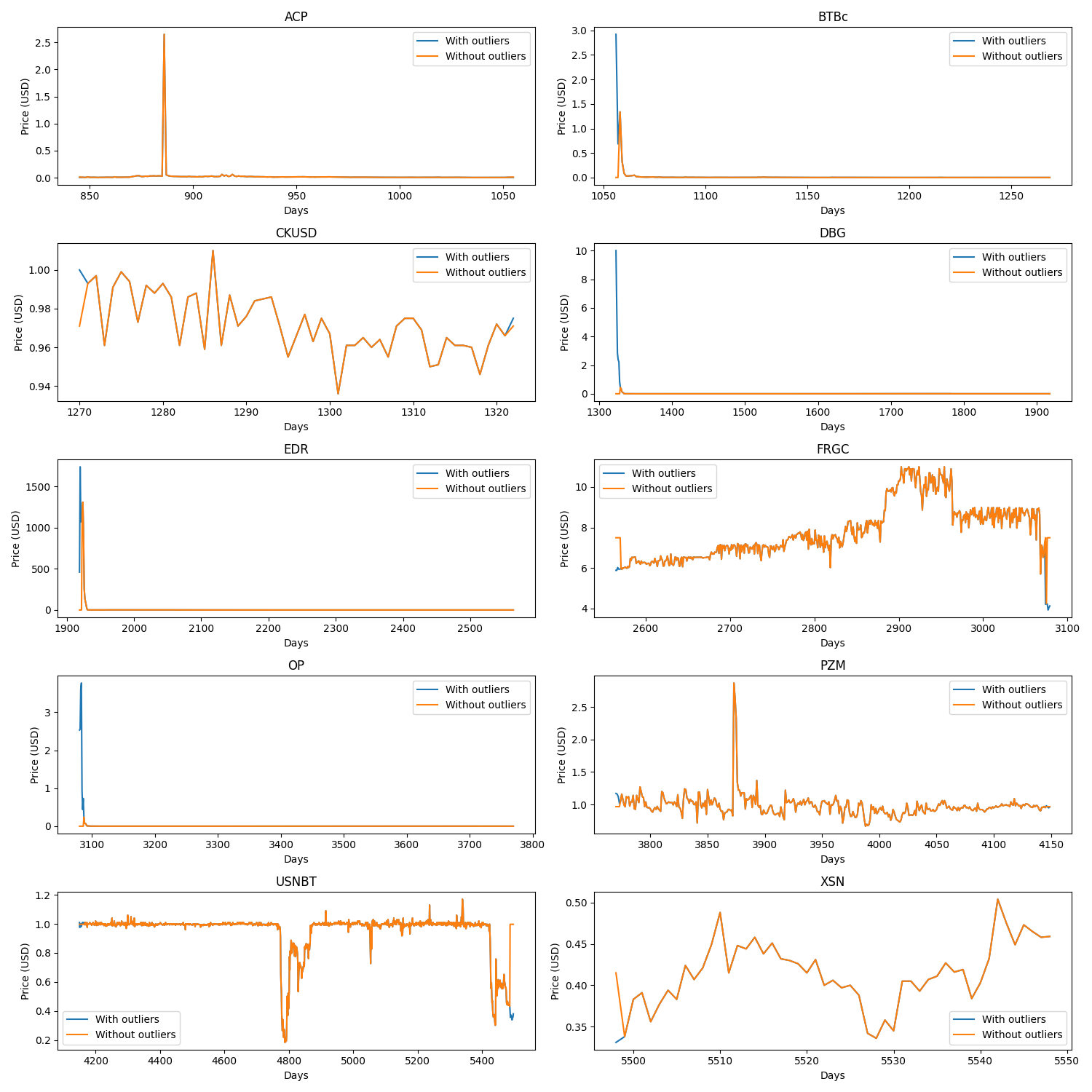


Figure 1: Plot showing each currencies time series; first without outliers (blue), then without outliers replaced with median (orange)

**3.3 Regression techniques**

**3.31 Linear Regression**

The first and most basic regression model we wanted to use on our data was linear regression. Linear regression fits a straight line that attempts to minimize the discrepancies between instances within a time series. The linear model does this by modeling the relationship between two variables and fitting a linear equation to observed data (our training data). For our data, our two variables were a series of days in the year (independent) and a series of prices in USD of a certain crypto currency (dependent). The linear equation is written as follows:

The Y represents the dependent variable, beta-zero represents the y-intercept, beta-one represents the slope coefficient, and x represents the independent variable

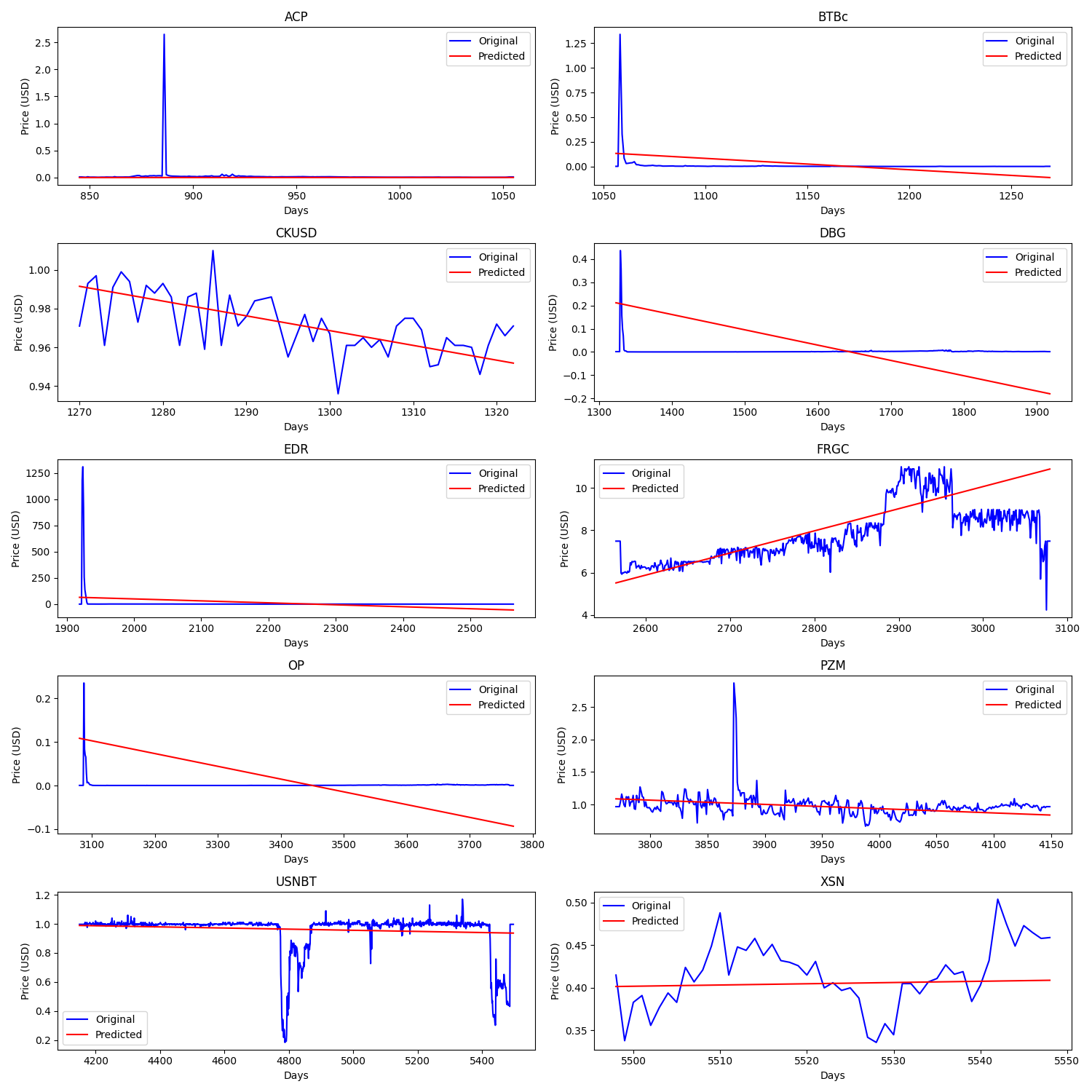


Figure 2: Plot graphing each currencies time series (blue) and linear regression prediction (red)

**3.32 Lasso Regression**

Lasso regression was the second regression model we chose, its name coming from the term Least Absolute Shrinkage and Selection Operator regularization. Lasso regression is an extension of linear regression that invokes adding penalties to the cost function in linear regression during training. This in turn encourages simpler models that have smaller coefficients. Lasso regression specifically includes a L1 penalty, which in turn has the effect of shrinking the coefficients for the input variables that do not contribute much to the prediction task. The coefficients of the model are found via an optimization process that seeks to minimize the sum squared error between the predictions and the actual values. This allows some coefficients to shrink down to zero, essentially taking them out of the equation when predicting. This type of regression can be used with multiple input attributes or just with a single attribute, making the relationship a line. We implemented Lasso with a single attribute, making it easily comparable to linear regression with a slightly altered coefficient value. The Lasso regression equation goes as follows:

Where y represents our predicted value from linear regression, x represents the independent variable, lambda represents the regularization parameter that controls the amount of regularization applied, and beta represents the different coefficients. However, since we only used Lasso with a single independent variable, there will only be one coefficient for our formula.

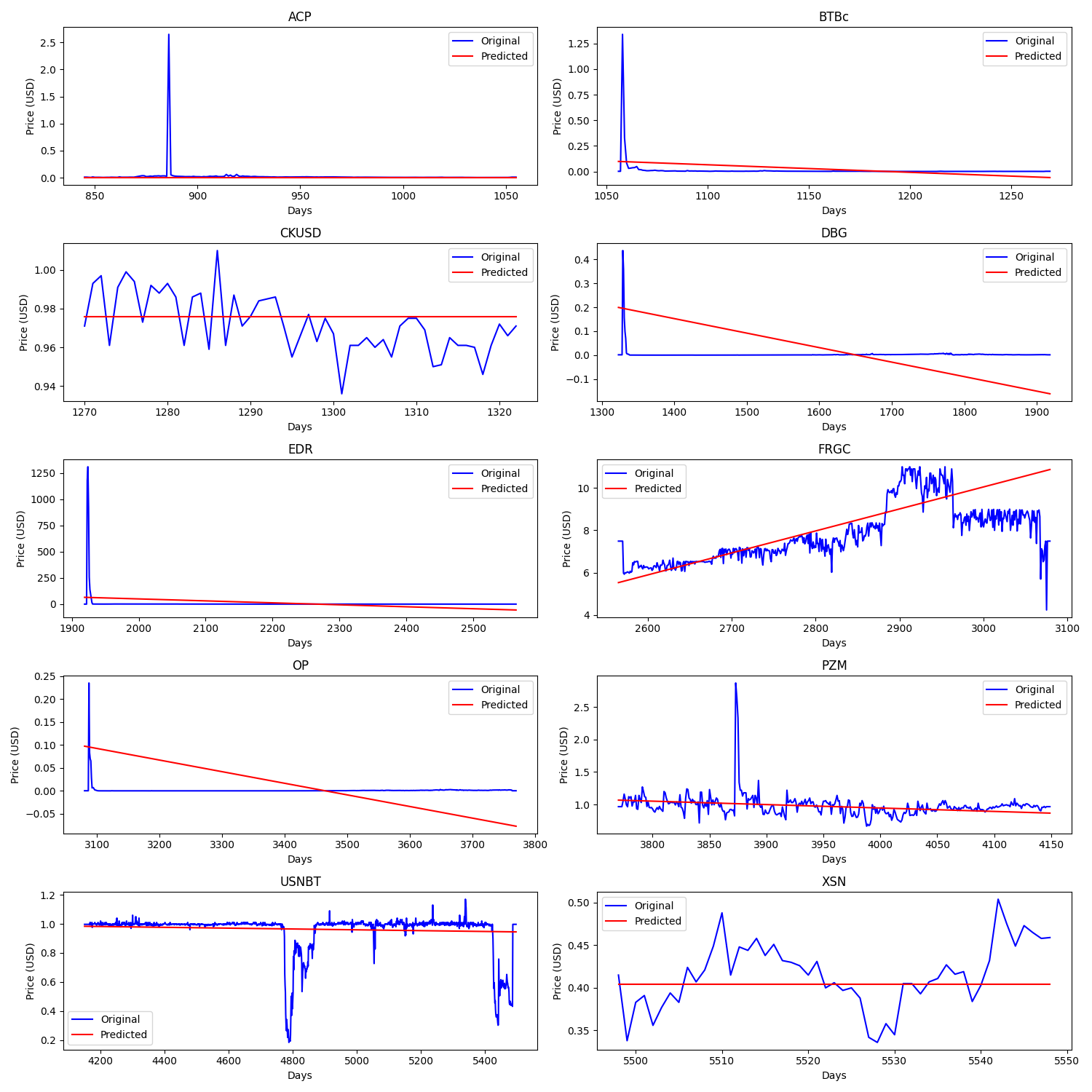


Figure 3: Plot graphing each currencies time series (blue) and lasso regression prediction (red)

**3.33 Polynomial Regression**

Polynomial regression is a non-linear regression technique that is utilized when the relationship between the dependent variable and independent variable is non-linear. This type of regression creates the relationship between the dependent and independent variables based on a nth degree polynomial. Thus, the best fit line is more like a best fit curve. When using polynomial regression, as we increase the degree in the model, it tends to increase the performance of the model; however, increasing the degrees of the model also increases the risk of overfitting and underfitting the data. Since we used 3rd degree polynomial regression the equation would look as follows:

Where y is our predicted value, beta zero is our y-intercept and the following betas are our degree coefficients, x subscript i is the x independent variable to the nth degree

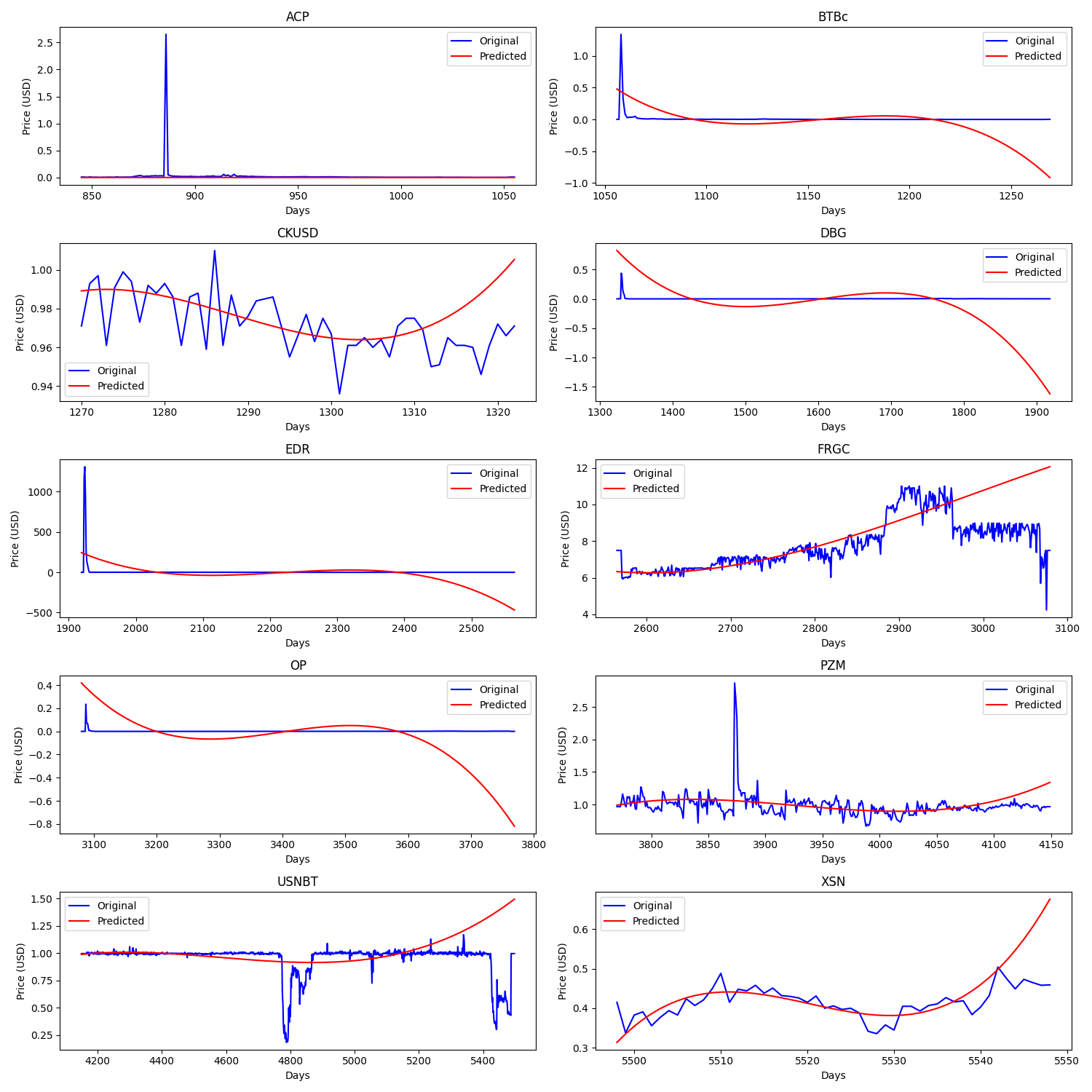


Figure 4: Plot graphing each currencies time series (blue) and polynomial regression prediction (red)

**3.34 Support Vector Regression**

Support Vector Regression (SVR) is a type of support vector machine (SVM) that is used for regression tasks. Its purpose is to find a function that best predicts the output values for a given input value. SVR can be used with both linear and non-linear kernels (we used only non-linear kernel ‘rbf’). These kernels are functions that attempt to determine the similarity between two input vectors, or in our case, the days of the year, and the cost of the coin. A linear kernel is just a dot product between these two vectors. In SVRs, the margin similar to a SVM is called the error tolerance of the model. The error tolerance allows for some deviation from the hyperplane (line in the middle of margin bounds) without being counted as an error. Its goal is to find the best fit error tolerance that accurately predicts the target variable while staying relatively simple, as complex SVMs/SVRs are prone to overfitting. The hyperplane will be our prediction line, and since we are using non-linear SVR, the equation is written as follows:

Where y is our predicted value, a is our weight vector with a subscript i being the ith element in the vector, K is our kernel function (our kernel was ‘rbf’), x is our test vector with x subscript i being our ith test sample, and b is our y-intercept.

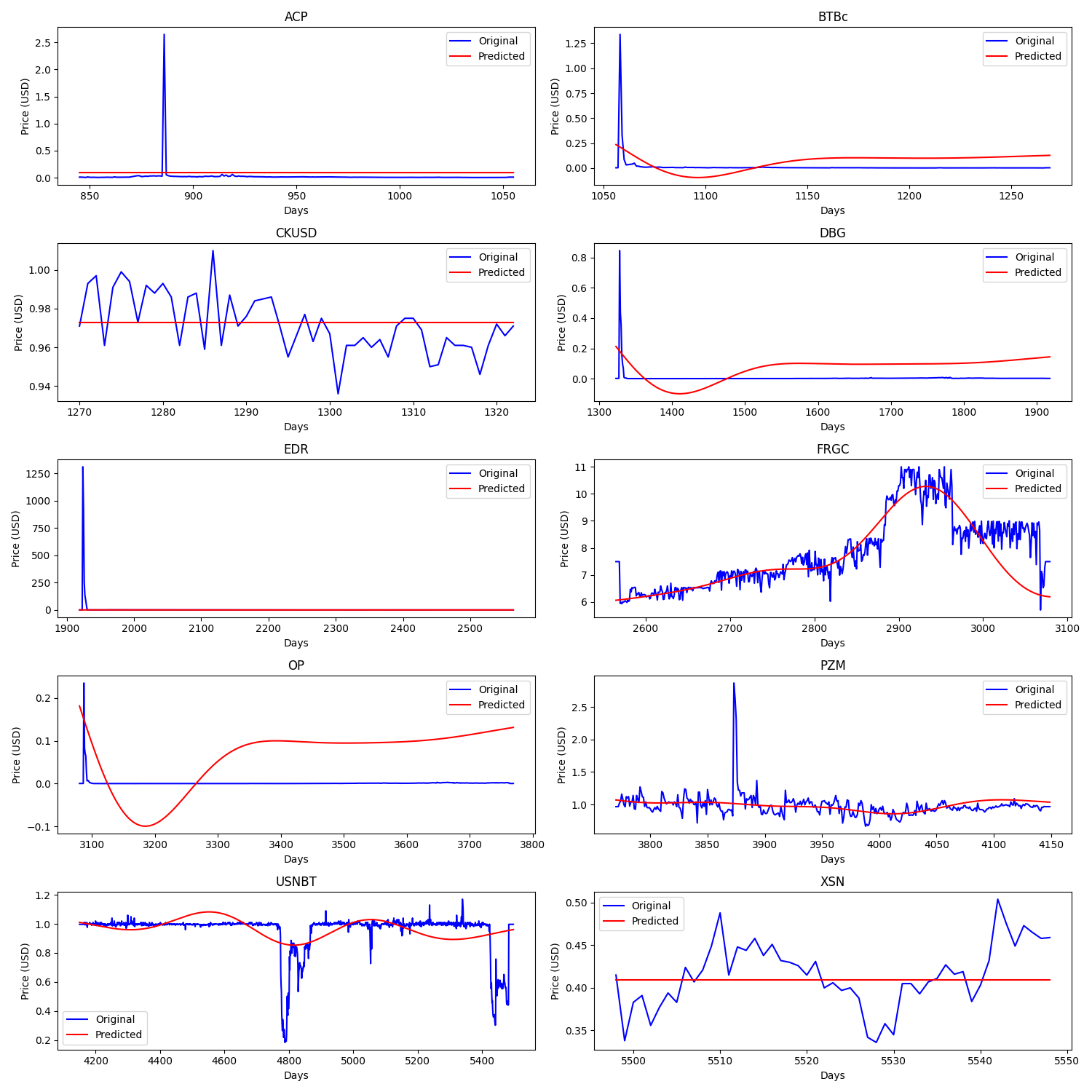


Figure 5: Plot graphing each currencies time series (blue) and SVR prediction (red)

**3.35 Multi-linear Regression**

The final regression technique we wanted to use was Multilinear-regression. It is an extension of linear regression and follows the same principles of trying to minimize a relationship between a set of independent input variables and a single dependent variable. For our dataset our main independent variable across all forms of regression was the instance, and for this form of regression specifically we added in the close price for each day. These would be combined into a linear combination to produce a predicted value for the currency. The form of the equation would be as such:

Where are the weights assigned to the given inputs, are the independent inputs, and is the prediction of the dependent variable.

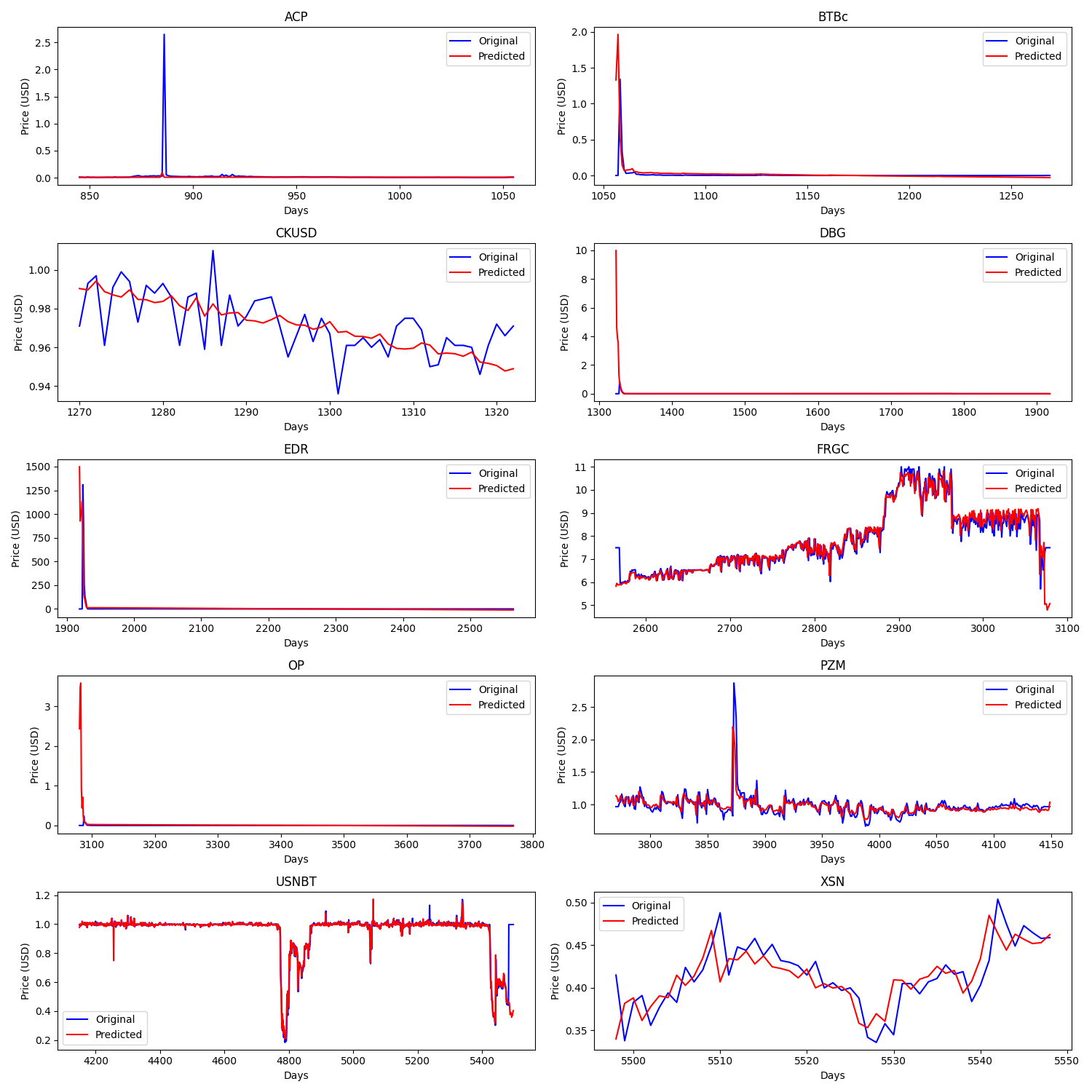
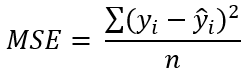


Figure 6: Plot graphing each currencies time series (blue) and multi-linear regression prediction (red)

**3.4 Evaluation Metrics**

**3.41 MSE (Mean Squared Error)**

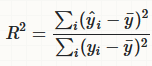
MSE is one of the ways to measure the error between a predicted (regression fit line) and actual values. By squaring the differences between predicted and observed values, MSE emphasizes larger errors, penalizing models that make significant mistakes more heavily. Due to this, MSE is vulnerable to outliers and has no upper bound. However, MSE does easily quantify the error between the predicted and actual values, giving higher values to predictors that create the most error and lower values to predictors that have the best fit. The equation for MSE is as follows:



Where y represents an observed value and y-hat represents its predicted value, with n being the number of instances in the data.

**3.42 R-squared**

Another metric we used for evaluating prediction models is R-squared, or the coefficient of determination. R-squared is used to measure the goodness of fit or best-fit line. Its bounds are between 0 and 1, 0 meaning no correlation and 1 meaning complete correlation. So a higher R-squared value would represent a better fit model and a lower R-squared value would represent a worse fit model. R-squared measures the proportion of the total variation in the dependent variable that can be explained by the independent variables in the model. R-squared equation goes as follows:



Where, again, y represents an observed value and y-hat represents its predicted value. Also we have y-bar, which is the mean of the data.

**4 Results & Discussion**

Having calculated the values and errors for all ten currencies with each of the models we produced which model had the lowest error rate for the given currency. The list of which currency should use which predictor broke down as such for both R-squared error and MSE:

Multi-linear: ACP, FRGC, PZM, USNBT, XSN

Support Vector: DBG, EBR

Lasso: BTBc, OP

Linear: CKUSD

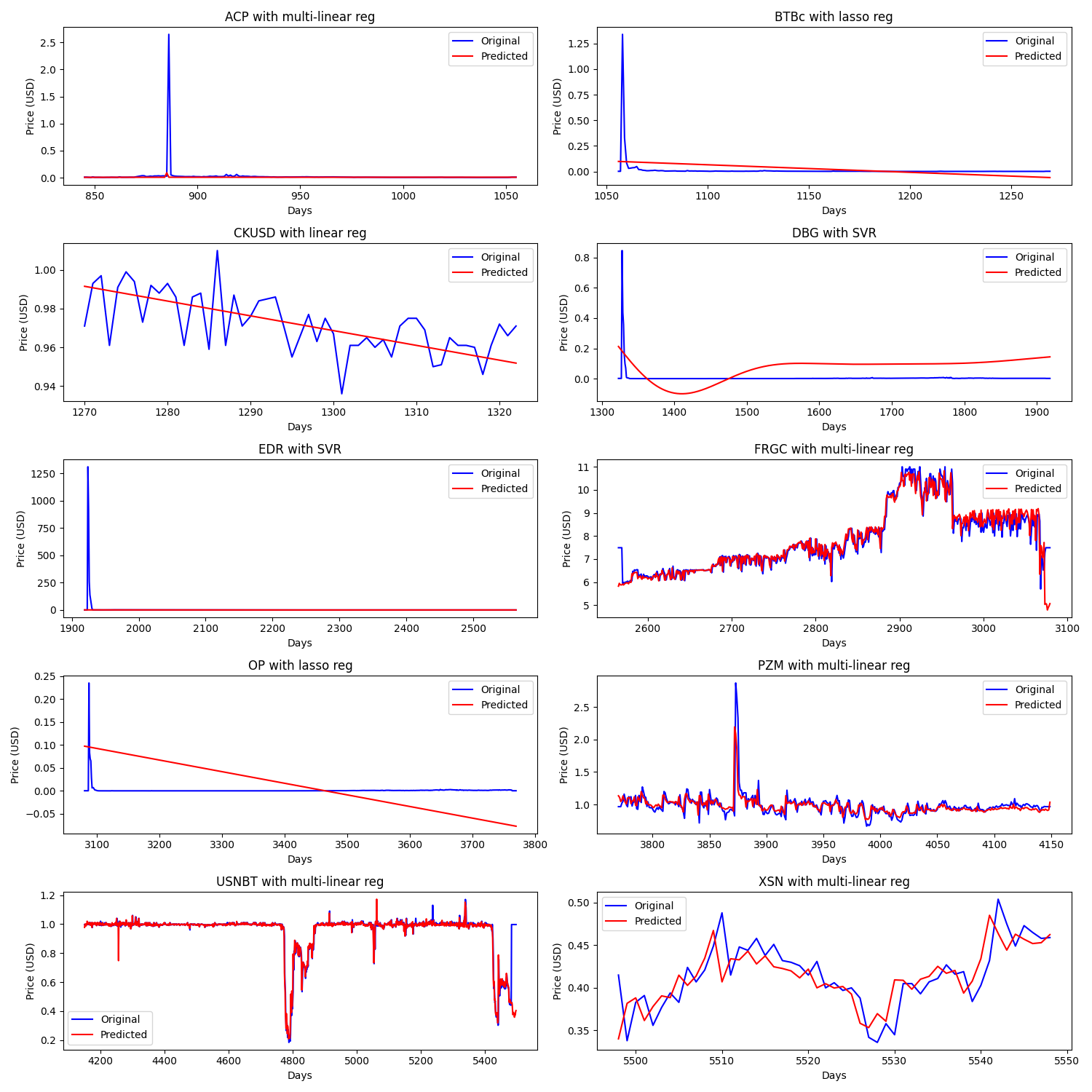


Figure 7: List of all crypto currencies with best predictor model

We discovered that in general multi-linear regression seemed best as it was the one with the lowest error rate for five of the ten currencies. However upon further analysis it was clear that four of those five that were best with multi-linear regression were the currencies with a low standard deviation. Of the less stable currencies it was more varied in what model would be best.

A curious finding that we had was that for both evaluation methods the worst performing model was the polynomial regression model. This was surprising as it has the most flexibility when fitting to the data, but this actually became a detriment to it. Due to the nature of the model and equation it produces it would attempt to fit too tightly to some of the data and produce a curve that does not match the data.

Overall this means that if you know a currency has overall high stability in its value and is not likely to fluctuate drastically then it would be best to use Multi-linear regression. If the currency is less stable and has a high likelihood of fluctuation then the best options would be support vector or lasso regression, but would need to be tested for accuracy. Also it means that in general polynomial regression should be avoided when attempting to train regression models for forecasting the value of a crypto currency.

**5 Conclusions**

In this project we aimed to explore the crypto currency dataset and compare the results of predictors like Linear regression, Lasso regression, Polynomial Regression, Support Vector Regression, and Multi-linear regression. First we selected a set of 10 currencies with both high and low standard deviations that would be good to compare the different models with. Following training the models on this we tested them with the subset of test data in order to evaluate them based on accuracy with two different measures (R-squared and MSE). Initially Multi-linear regression appeared to be the best model. However, the currencies that were best fit by Multi-linear were the more stable of the currencies chosen and then Lasso and Support Vector Regression were better for less stable currencies. Also noted that polynomial regression was the worst overall performing model which was unexpected at the outset.

To make a more definitive choice on the best model for less stable crypto currencies more testing would be required on other available data from either our dataset or from others. Other areas for more work would be factoring more than just the daily open price, but also the close price of each day and the high and lowest for each day.

**References**

[1] N. Parveen, S. Zaidi, and M. Danish, “Support vector regression model for predicting the

sorption capacity of lead (II),” *Perspectives in Science*, vol. 8, pp. 629–631, Sep. 2016, doi: <https://doi.org/10.1016/j.pisc.2016.06.040>.

[2] D. Chicco, M. J. Warrens, and G. Jurman, “The coefficient of determination R-squared is

more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation,” *PeerJ Computer Science*, vol. 7, no. 5, p. e623, Jul. 2021, doi: <https://doi.org/10.7717/peerj-cs.623>.

[3] “Development of Multilinear Regression Models for Online Voltage Stability Margin

Estimation | IEEE Journals & Magazine | IEEE Xplore,” *ieeexplore.ieee.org*. <https://ieeexplore.ieee.org/abstract/document/5484523> (accessed Dec. 10, 2023).

[4] J. Brownlee, “4 Automatic Outlier Detection Algorithms in Python,” Machine Learning

Mastery, Jul. 07, 2020. <https://machinelearningmastery.com/model-based-outlier-detection-and-removal-in-python/>

[5] “sklearn.ensemble.IsolationForest — scikit-learn 0.23.1 documentation,” scikit-learn.org.

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html>

[6] Y. Lim, “Unsupervised Outlier Detection with Isolation Forest,” *MLearning.ai*, Mar. 22,

2022. <https://medium.com/mlearning-ai/unsupervised-outlier-detection-with-isolation-forest-eab398c593b2>

[7] scikit-learn developers, “sklearn.linear\_model.LinearRegression — scikit-learn 0.22

documentation,” *Scikit-learn.org*, 2019. <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html>

[8] “sklearn.linear\_model.Lasso,” *scikit-learn*.

<https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html>

[9] J. Brownlee, “How to Develop LASSO Regression Models in Python,” *Machine Learning*

*Mastery*, Oct. 11, 2020. <https://machinelearningmastery.com/lasso-regression-with-python/>

[10] Zach, “How to Perform Polynomial Regression Using Scikit-Learn,” *Statology*, Jan. 03,

2023. <https://www.statology.org/sklearn-polynomial-regression>

[11] “sklearn.svm.SVR — scikit-learn 0.23.1 documentation,” *scikit-learn.org*.

<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html>

[12] “Python Machine Learning Multiple Regression,” *www.w3schools.com*.

<https://www.w3schools.com/python/python_ml_multiple_regression.asp>

[13] A. Kumar, “Mean Squared Error or R-Squared - Which one to use?,” *Data Analytics*, Sep.

30, 2020. <https://vitalflux.com/mean-square-error-r-squared-which-one-to-use/>