Introduction:

This project consists of 3 parts:

- The first part accesses, downloads, formats and saves data.
- The second part processes the data from the first part, extracts desired aspects and compiles them.
- The third part uses the prepared data from the second and performs linear regression on it.

Presentation of results:

- There is the option to display a chart of the processed data for the second part.
- There is the option to display a chart of the predicted values on the actual values for the third part.
- The third part presents its data in a regression table prepared similar to Excel's regression report.

Selected area of study and hypothesis:

- I have chosen to study financial data with technical indicators as predictors. As Bitcoin is one of the most technically responsive currencies, I have chosen its price as my outcome variable. As for my covariates I have chosen spot gold (XAU) prices and technical indicators derived from Bitcoin data.

The derived technical indicators are:

- Moving averages with 20 and 50 windows, and more specifically, their difference. This is commonly used to infer whether the trend is upward or downward, with a positive difference indicating an upward trend.
- Bollinger bands with 3 standard deviations in each direction from the 20 period moving average. As a movement of 3 standard deviations is seen as an extreme, Bollinger bands are used to measure the intensity of the asset's movements. I am using the distance of the price from the upper and lower bands as a pull or push factor with the moving average as the equilibrium point.
- Fibonacci retracement lines of 1.61, 1.0, 0.61, 0.38, 0.23, 0.0 and "-0.0" are also used to create pull and push factors. Fibonacci retracement levels are commonly used to infer possible targets of price movements and are seen as support or resistance barriers. Fibonacci levels were created automatically based on the maximum and minimum levels of the last 500 periods.
- Volume of traded Bitcoin is also directly used as a covariate.

Notes on derivation of variables:

- The windows of 20 and 50 for the moving averages was taken at random, as was the 3 standard deviations and the last 500 periods. Although these values are arbitrary, they are often used in financial analysis.
- The used Fibonacci levels are also standardly used levels.
- To calculate the pull and push factors, I have taken the difference of price from the target levels and I have squared that distance.
- I have also provided the option to shift the data by a desired amount of days. My aim was to shift it by 7 days and kind of predict prices 1 week in advance, however this has resulted in unexpected results as will be discussed later.
- As Bitcoin operates 7 days of a week as opposed to spot gold's 5 days in a week, there was missing XAU data on weekends. I have filled these values by forwarding the last price, which would be the price on the respective Friday, to the weekend.
- I used backward fill in cases where the data is shifted by any number of days.

Part 1 - Getting the Data:

I am using investing.com as my data provider. I am not using an API but rather sending an XmlHttpRequest to the server, specifying what kind of data I want. This information can be found by using web browsers' inspection tools that appear by clicking F12 while in browser window.

To send the XHR, I am using a library called "requests". This request returns, in this case, an HTML content in form of a String. I feed this String into BeautifulSoup in order to parse it as an HTMLDocument. I can then use bs4 methods to filter for the related tables, rows and cells.

I transform and store the information of the HTML in a pandas DataFrame. I then export this information to csv files, one for BTC and one for XAU. These are then read by the data processor in the second step.

Part 2 - Processing the Data:

Most of this section was covered above in the first page. I will only mention additionally that I read the DataFrames from the two csv files created in part 1. The dates are sorted recent-to-old, which I reverse in this part. After performing the operations discussed previously, I save the file in a csv named "regression". This is later read in part 3.

Part 3 - Regression and Reporting:

We read the file prepared in part 2 into a DataFrame. From this DataFrame we extract the outcome variable and the index, which is "Dates", leaving only the covariates. We store the outcome variable in a numpy array. The columns property of the DataFrame is used to extract the names of the covariates to be used in reporting. Using the remaining DataFrame and the outcome variables I find the coefficients of the covariates using the matrix multiplication formulas provided. I further calculate elements present in Excel's regression analysis tool as I wanted to reproduce a similar output. After all elements are calculated I put them in 3 DataFrames, 1 for each Excel regression report portion. To reproduce the t-Stats, p-Values and Significance F, I used the SciPy library.

Before the regression is executed, a function checks for multicollinearity and dimensions. If either is not satisfied, the regression table is never formed and a statement is printed indicating the error. To check for multicollinearity I simply check if either column is the multiple of another column. I do this by dividing every column by every other column and making sure the divisions are not equal for every row. We control for multicollinearity to avoid singular matrix error and because statistically it means one of the variables is useless.

User Input:

The user is asked to input the column title as it appears in the csv. The user is also asked to provide a path to a desired file location. The user can choose to display the resulting charts by turning "show_chart" into True. The user can also choose to put a delay in the predictions by changing the delay from 0 to the desired number of periods. Aside from placing a function for my project, I have also provided a generic regression function which can be used for testing. My version includes specific column titles that would cause errors during testing.

Discussion of Results:

Bitcoin claimed to be an alternative to gold, which would suggest a positively correlated movement in their prices. The positive coefficient of gold prices is in agreement with this, however the relation is not apparent on the chart. The small value of the derived covariates is understandable, as they are mostly the distance squared, which is sometimes in the millions. It is pleasing to see that the "pull" elements have positive coefficients while the "push" elements have negative coefficients. This is expected behavior as implied by the names. The closer the price is getting to a barrier the more of a "push" is expected, which would be in the opposite direction of the movement, hence the negative sign. The effect of volume is rather debatable, as a high volume could mean irrationally large movement fuel by "hype" and a small volume could also mean large swings as small volume is more susceptible to price changes. The coefficient of volume is not that large considering the max value of volume is around 200, which would correspond to about 3-5% of the price when multiplied by the coefficient.

The moving average effect had a large p-Value so I repeated the study by omitting it as a variable. This had almost no effect on the R square so we could safely remove moving averages as a covariate. The p-Value of Bollinger pull was also larger than desired and removing the moving averages did not reduce it enough so it could also be omitted, however I think it kind of compliments the Bollinger push. All other p-Values were well within the required limits. The R square value is rather large with 0.98 and it decreased only to 0.89 even with 7 days' delay. I do not know why it is so large, I know it is a good thing but it is unexpected and unexplained. The chart of 7 day delayed predictions is skewed/shifted, however this seems to not affect the R square.

I am not pleased with the predictive abilities of this model, as the 7 days delayed chart looks so skewed. I cannot account for this behavior. However, considering that I do not use any bitcoin price indicators when estimating the price, but merely pull and push factors and the gold prices, I could say the model is surprisingly successful.

Caveats:

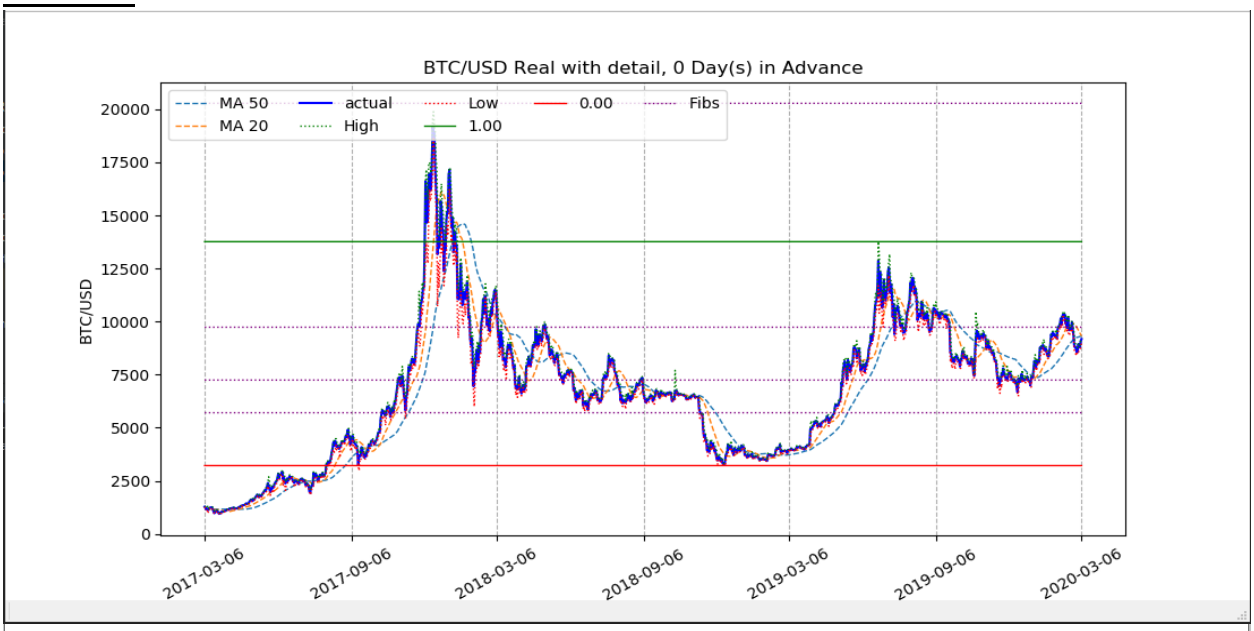
I do not handle division by zero: I am not sure whether my Python or my IDE is taking care of it, but even though I receive a warning that I tried to divide by zero, this does not affect my results or stop my code.

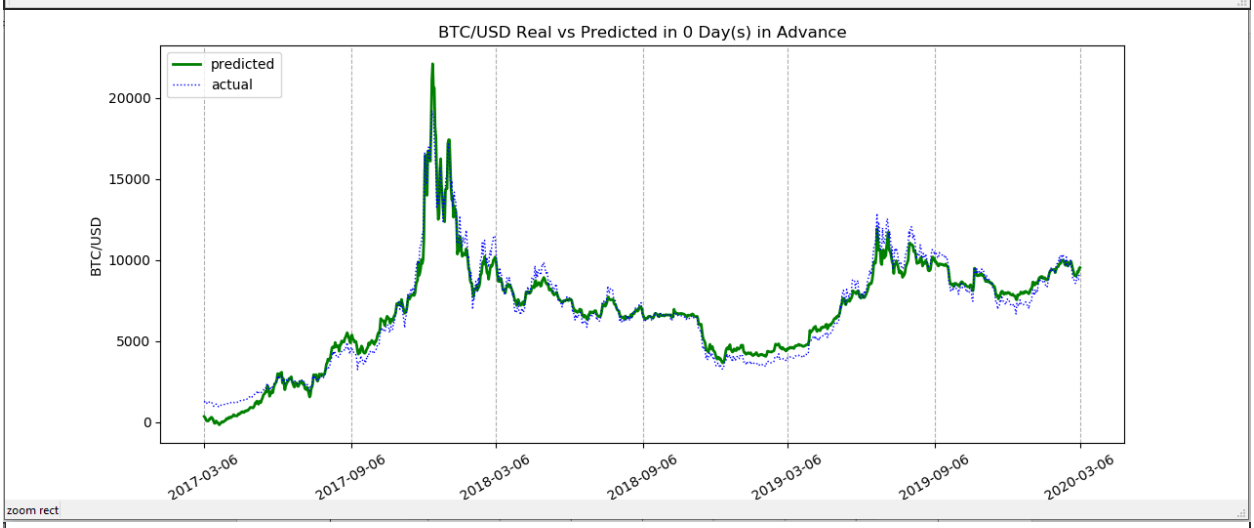
I could also remove covariates with multicollinearity, keeping only one and resuming regression, whereas currently the execution ends and no report is prepared until the user removes the covariate with multicollinearity.

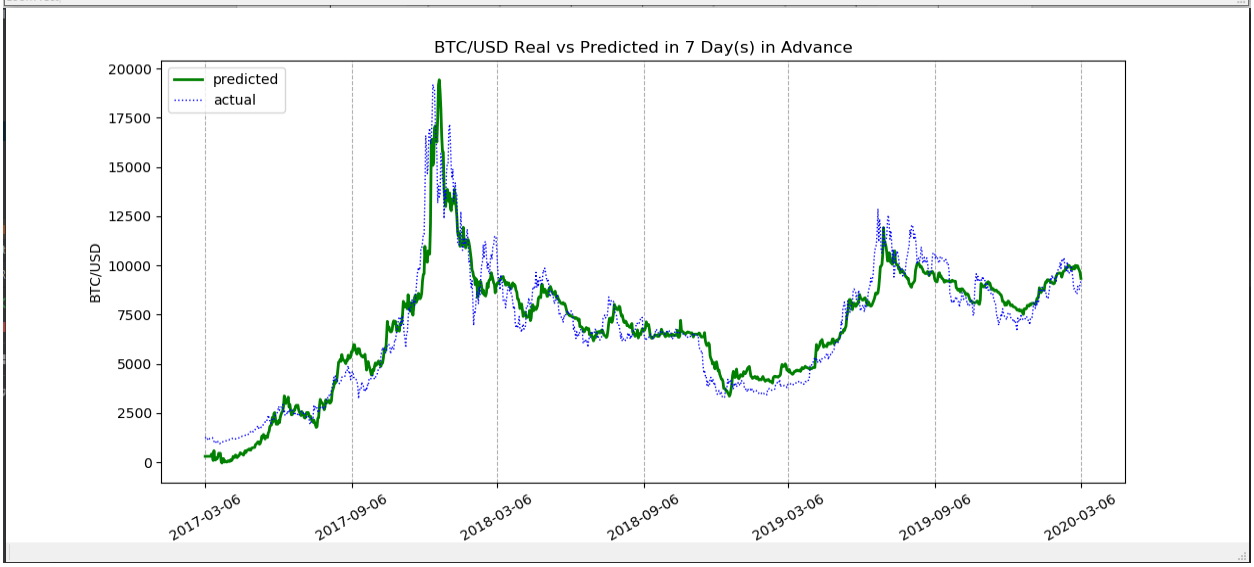
I also do not handle bad user input such as wrong file location or String in the place of data (where it should be numeric) or invalid start and end times. I rely on the user entering reasonable input else the program throws an error.

Basically, no error handling aside from checking for dimensions and multicollinearity, which are handled by if statements.

Exhibits:


BTC/USD Real with detail, 0 Day(s) in Advance
MA 50
MA 20
actual
High
Low
1.00
0.00
Fibs
BTC/USD
20000
17500
15000
12500
10000
7500
5000
2500
0
2017-03-06
2017-09-06
2018-03-06
2018-09-06
2019-03-06
2019-09-06
2020-03-06


BTC/USD Real vs Predicted in 0 Day(s) in Advance
predicted
actual
BTC/USD
20000
15000
10000
5000
0
2017-03-06
2017-09-06
2018-03-06
2018-09-06
2019-03-06
2019-09-06
2020-03-06
zoom rect


BTC/USD Real vs Predicted in 7 Day(s) in Advance
predicted
actual
BTC/USD
20000
17500
15000
12500
10000
7500
5000
2500
0
2017-03-06
2017-09-06
2018-03-06
2018-09-06
2019-03-06
2019-09-06
2020-03-06

Regression Statistics	
Multiple R	0.98361
R Square	0.96748
Adjusted R Square	0.96727
Standard Error	592.55698
Observations	1097.00000

ANOVA

	df	SS	MS	F	Significance F
Regression	7	11376578758.0	1625225537.0	4628.63999	0.0
Residual	1089	382373788.0	351124.0		
Total	1096	11758952547.0			

	Coefficients	Standard Error	t Stat	P-value	Lower 95%	Upper 95%
Intercept	4530.768430	2.951828e+02	15.349027	2.611470e-48	3.951582e+03	5109.955068
ma_trend	-0.009909	2.326382e-02	-0.425941	1.329766e+00	-5.555564e-02	0.035738
bollinger_pull	0.000004	2.205857e-06	1.759278	7.880937e-02	-4.474608e-07	0.000008
bollinger_push	-0.000007	1.810599e-06	-3.895529	1.999896e+00	-1.060587e-05	-0.000004
fib_pull	0.000004	1.205544e-07	34.418889	1.326827e-176	3.912805e-06	0.000004
fib_push	-0.000003	2.956534e-08	-99.849245	2.000000e+00	-3.010088e-06	-0.000003
Volume	3.905067	8.720078e-01	4.478248	8.313507e-06	2.194075e+00	5.616059
xau_Price	3.152056	2.061502e-01	15.290091	5.510665e-48	2.747563e+00	3.556549

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.98361
R Square	0.96748
Adjusted R Square	0.96727
Standard Error	592.55698
Observations	1097.00000

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	7	11376578758	1625225537	4628.63999	0.0
Residual	1089	382373788	351124		
Total	1096	11758952547			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*
Intercept	4530.768430	2.951828E+02	15.349027	2.781729E-48	3.951577E+03	5109.959768
ma_trend	-0.009909	2.326382E-02	-0.425941	6.702347E-01	-5.555601E-02	0.035738
bollinger_pull	0.000004	2.205857E-06	1.759278	7.881116E-02	-4.474959E-07	0.000008
bollinger_push	-0.000007	1.810599E-06	-3.895529	1.039555E-04	-1.060590E-05	-0.000004
fib_pull	0.000004	1.205544E-07	34.418889	2.812981E-176	3.912803E-06	0.000004
fib_push	-0.000003	2.956534E-08	-99.849245	0.000000E+00	-3.010089E-06	-0.000003
Volume	3.905067	8.720078E-01	4.478248	8.318765E-06	2.194061E+00	5.616073
xau_Price	3.152056	2.061502E-01	15.290091	5.864980E-48	2.747559E+00	3.556553