Predicting Tomorrow’s Gold Price Direction

University of Colorado at Boulder

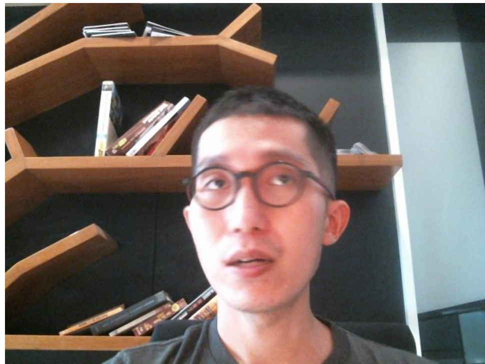
DTSA 5509: Intro to Machine Learning: Supervised Learning

Final Project, Nov 2024

Github Repository Link
https://github.com/AndreeTSendjaja/Predicting-Tomorrow-s-Gold-Price-Direction

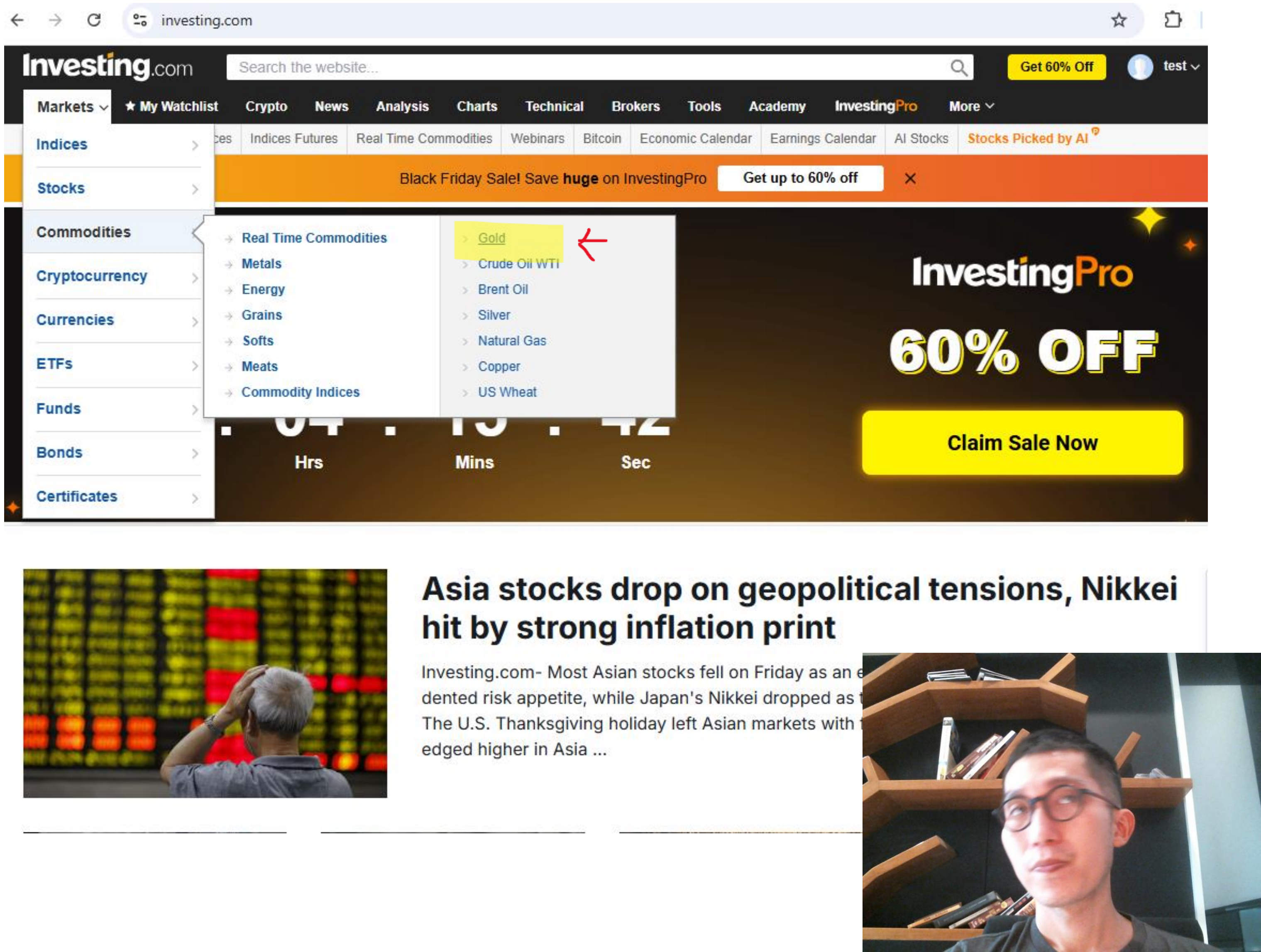
Contents

- Dataset
- Models
  - Logistic Regression
  - KNN
  - Decision Tree
  - Random Forest
  - SVM
  - Logistic Regression, with P hacking
- Results
- Rooms for Improvements



Dataset

Response Variable



Source:
https://www.investing.com/

Dataset

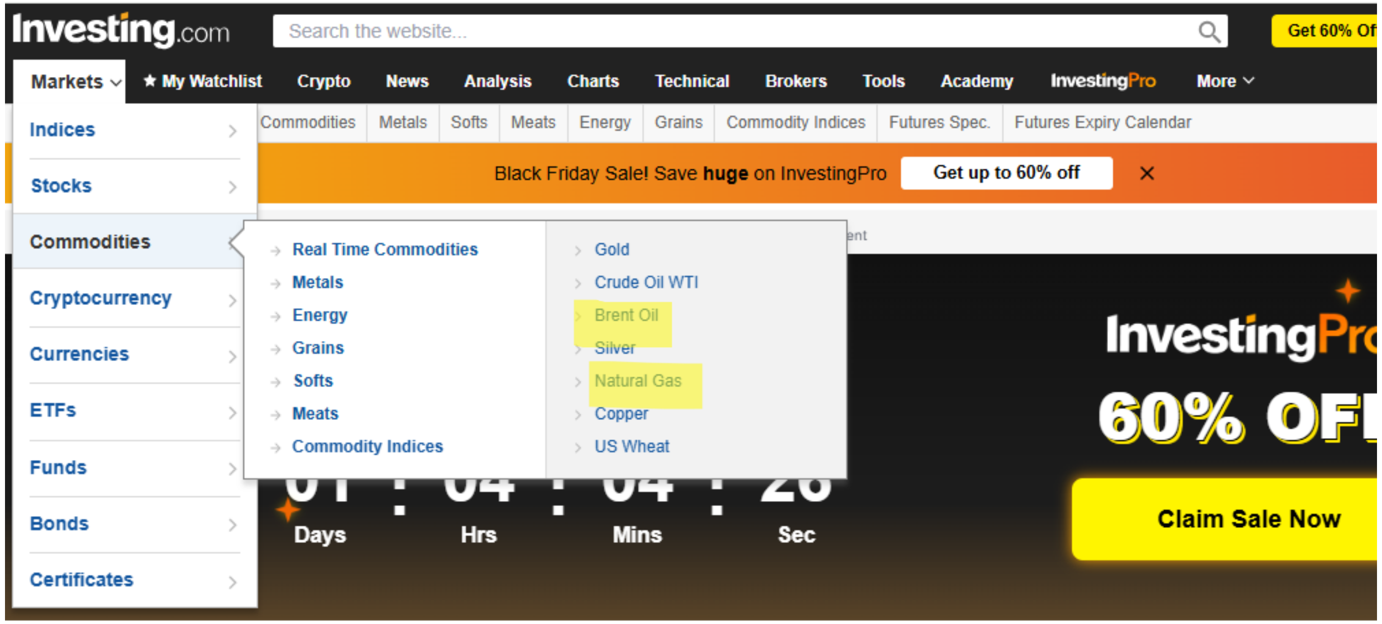
Predictors

- Brent Oil Futures
- Natural Gas

- Dow Jones
- Nasdaq 100
- Nikkei 225
- S&P 500

- EUR/USD
- GBP/USD
- USD/JPY

- US Holidays



Source:

https://www.investing.com/commodities/gold-historical-data

Dataset
Predictors

- Brent Oil Futures
- Natural Gas

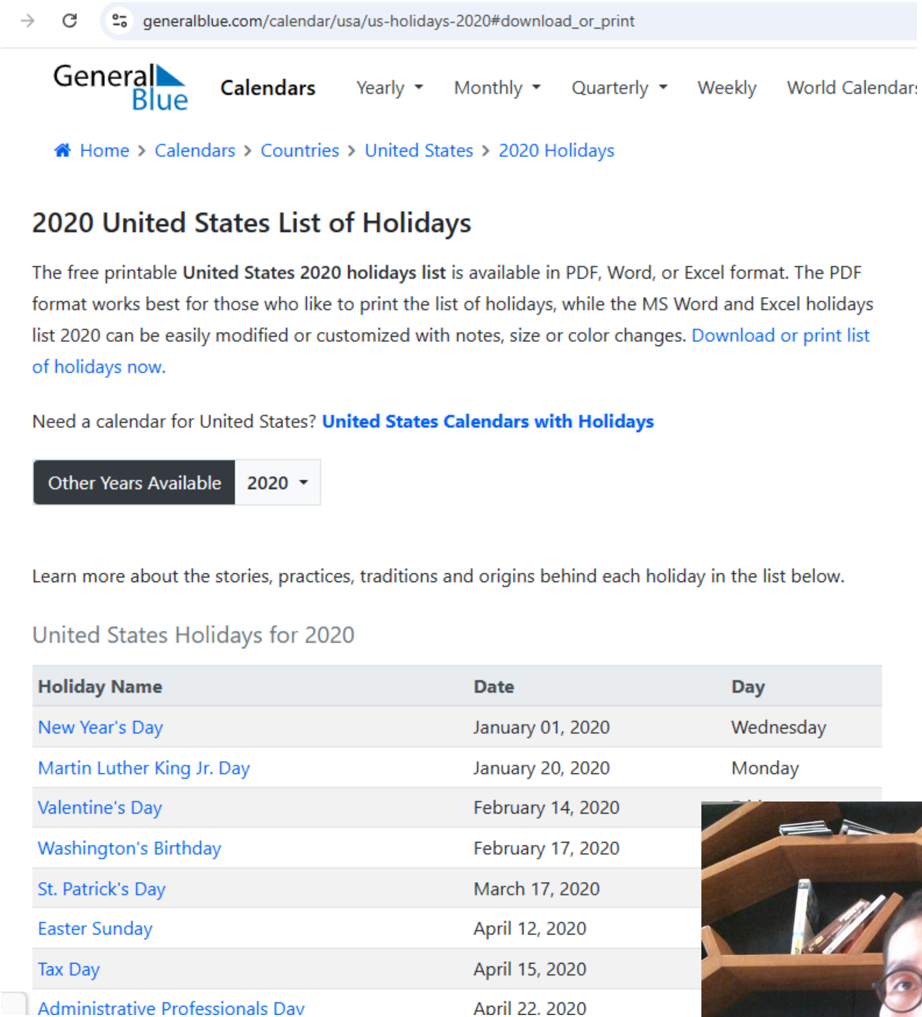
- Dow Jones
- Nasdaq 100
- Nikkei 225
- S&P 500

- EUR/USD
- GBP/USD
- USD/JPY

- US Holidays

Source:

https://www.generalblue.com/calendar/usa/us-holidays-2020#download_or_print



generalblue.com/calendar/usa/us-holidays-2020#download_or_print

General Blue Calendars Yearly Monthly Quarterly Weekly World Calendar:

Home > Calendars > Countries > United States > 2020 Holidays

2020 United States List of Holidays

The free printable **United States 2020 holidays list** is available in PDF, Word, or Excel format. The PDF format works best for those who like to print the list of holidays, while the MS Word and Excel holidays list 2020 can be easily modified or customized with notes, size or color changes. Download or print list of holidays now.

Need a calendar for United States? **United States Calendars with Holidays**

Other Years Available 2020

Learn more about the stories, practices, traditions and origins behind each holiday in the list below.

United States Holidays for 2020

Holiday Name	Date	Day
New Year's Day	January 01, 2020	Wednesday
Martin Luther King Jr. Day	January 20, 2020	Monday
Valentine's Day	February 14, 2020	
Washington's Birthday	February 17, 2020	
St. Patrick's Day	March 17, 2020	
Easter Sunday	April 12, 2020	
Tax Day	April 15, 2020	
Administrative Professionals Day	April 22, 2020	

Dataset

Shape – One Commodity

Month	Day	Year	Gold_Price_Avg	Gold_Price_Ope	Gold_Price_High	Gold_Price_Low	Gold_Vol. (M)	Gold_PriceChange
4	1	2,020	1,584.10	1,581.70	1,605.00	1,574.80	0.35	-0.33%
5	1	2,020	1,700.90	1,693.50	1,714.40	1,676.00	168.02	0.40%
6	1	2,020	1,743.30	1,743.00	1,752.70	1,730.50	0.71	0.02%
7	1	2,020	1,779.90	1,798.90	1,807.70	1,767.90	263.25	-1.14%
9	1	2,020	1,970.80	1,965.90	1,992.50	1,961.20	10.95	0.02%
10	1	2,020	1,912.30	1,888.00	1,913.90	1,886.00	0.75	1.11%
12	1	2,020	1,816.60	1,777.50	1,818.30	1,776.40	1.66	2.17%
1	2	2,020	1,528.10	1,521.00	1,534.00	1,519.70	270.55	0.33%
3	2	2,020	1,594.80	1,592.80	1,612.10	1,576.30	443.53	1.79%
4	2	2,020	1,630.70	1,599.00	1,635.80	1,590.50	0.44	2.94%
6	2	2,020	1,728.90	1,743.50	1,750.00	1,724.10	0.69	-0.83%
7	2	2,020	1,790.00	1,779.00	1,791.70	1,766.30	186.31	0.57%
9	2	2,020	1,936.90	1,967.90	1,972.40	1,931.50	10.84	-1.72%
10	2	2,020	1,903.80	1,907.00	1,919.00	1,893.00	0.81	-0.44%
11	2	2,020	1,892.50	1,877.00	1,897.10	1,873.30	164.94	0.67%
12	2	2,020	1,827.60	1,814.60	1,833.00	1,808.10	1.75	0.61%
1	3	2,020	1,552.40	1,531.70	1,556.60	1,530.40	436.74	1.59%
2	3	2,020	1,579.50	1,594.60	1,595.10	1,570.50	1.60	-0.34%
3	3	2,020	1,644.40	1,586.00	1,650.50	1,585.90	466.53	3.11%
4	3	2,020	1,637.60	1,629.70	1,642.00	1,616.50	0.51	0.42%
6	3	2,020	1,701.00	1,729.70	1,729.80	1,686.90	1.72	-1.61%
7	3	2,020	1,793.50	1,787.90	1,799.00	1,779.20	184.65	0.20%
8	3	2,020	1,969.50	1,979.70	1,992.10	1,958.50	2.82	0.08%
9	3	2,020	1,930.20	1,941.80	1,948.40	1,919.70	8.94	-0.35%
11	3	2,020	1,910.40	1,896.40	1,912.20	1,887.60	171.37	0.95%
12	3	2,020	1,838.40	1,830.50	1,844.00	1,824.00	1.37	0.59%
2	4	2,020	1,552.70	1,579.50	1,580.70	1,550.00	1.88	-1.70%
3	4	2,020	1,643.00	1,640.10	1,654.30	1,632.60	313.34	-0.09%
5	4	2,020	1,713.30	1,711.20	1,726.00	1,700.30	148.73	0.73%

<u>Date Range</u>

Jan 02 2020

to

Nov 21 2024

<u>Approximately</u>

1260

rows

Dataset

Shape – US Holidays

generalblue.com/calendar/usa/us-holidays-2020#download_or_print

General Blue **Calendars** Yearly Monthly Quarterly Weekly World Calendar

Home › Calendars › Countries › United States › 2020 Holidays

2020 United States List of Holidays

The free printable **United States 2020 holidays list** is available in PDF, Word, or Excel format. The PDF format works best for those who like to print the list of holidays, while the MS Word and Excel holidays list 2020 can be easily modified or customized with notes, size or color changes. Download or print list of holidays now.

Need a calendar for United States? **United States Calendars with Holidays**

Other Years Available 2020

Learn more about the stories, practices, traditions and origins behind each holiday in the list below.

United States Holidays for 2020

Holiday Name	Date	Day
New Year's Day	January 01, 2020	Wednesday
Martin Luther King Jr. Day	January 20, 2020	Monday
Valentine's Day	February 14, 2020	Friday
Washington's Birthday	February 17, 2020	Monday
St. Patrick's Day	March 17, 2020	Tuesday
Easter Sunday	April 12, 2020	Sunday
Tax Day	April 15, 2020	Wednesday
Administrative Professionals Day	April 22, 2020	Wednesday

Date	Holiday
January 1, 2020	New Year's Day
January 20, 2020	Martin Luther King Jr. Day
February 14, 2020	Valentine's Day
February 17, 2020	Washington's Birthday
March 17, 2020	St. Patrick's Day
April 12, 2020	Easter Sunday
April 15, 2020	Tax Day
April 22, 2020	Administrative Professionals
May 10, 2020	Mother's Day
May 10, 2020	Memorial Day
June 21, 2020	Father's Day
July 3, 2020	Independence Day (substitu
July 4, 2020	Independence Day
September 7, 2020	Labor Day
October 12, 2020	Columbus Day
October 31, 2020	Halloween
November 3, 2020	Election Day
November 11, 2020	Veterans Day
November 26, 2020	Thanksgiving Day
November 27, 2020	Day after Thanksgiving Day
December 24, 2020	Christmas Eve
December 25, 2020	Christmas Day
December 31, 2020	New Year's Eve
January 1, 2021	New Year's Day
January 18, 2021	Martin Luther King Jr. Day
February 14, 2021	Valentine's Day
February 15, 2021	Washington's Birthday
March 17, 2021	St. Patrick's Day
April 4, 2021	Easter Sunday

Date	DayType
January 1, 2020	Holiday
January 2, 2020	DayAfterHoliday
January 3, 2020	regularDay
January 4, 2020	regularDay
January 5, 2020	regularDay
January 6, 2020	regularDay
January 7, 2020	regularDay
January 8, 2020	regularDay
January 9, 2020	regularDay
January 10, 2020	regularDay
January 11, 2020	regularDay
January 12, 2020	regularDay
January 13, 2020	regularDay
January 14, 2020	regularDay
January 15, 2020	regularDay
January 16, 2020	regularDay

Dataset

Preprocessing – Collating to Build Dataset

Month	Day	Year	Date	US_Holiday?	Day_of_Week	Gold_Price_Av	Gold_Vol.	Gold_PriceCha	BrentOil_Price	BrentOil_Vol.	NatGas_Price_	NatGas_Vol.	EurUSD_Price	GBPUSD_Pric	USDJPY_Price	S&P_Price_Av	Nasdaq_Price_	Nasdaq_Vol.	DowJones_Pri	DowJones_Vol	Nikkei_Price_A	Nikkei_Vol.
4	1	2,020	4/1/2020	regularDay	4.00	1,584.10	350	-0.33%	28.04	180520	1.712	59130	1.0962	1.2376	107.15	2,470.50	7,486.29	261150000	20,943.51	508310000	18,065.41	1060000000
5	1	2,020	5/1/2020	regularDay	6.00	1,700.90	168020	0.40%	28.07	107670	2.134	73910	1.0983	1.2502	106.93	2,830.70	8,718.18	221880000	23,723.69	421590000	19,619.35	866440000
6	1	2,020	6/1/2020	regularDay	2.00	1,743.30	710	0.02%	38.43	124700	1.871	58660	1.1134	1.2492	107.58	3,055.70	9,598.89	163310000	25,475.02	341220000	22,062.39	721660000
7	1	2,020	7/1/2020	regularDay	4.00	1,779.90	263250	-1.14%	42.11	122360	1.721	76380	1.125	1.247	107.46	3,115.90	10,279.25	167520000	25,734.97	373980000	22,121.73	673600000
9	1	2,020	9/1/2020	regularDay	3.00	1,970.80	10950	0.02%	46	138080	2.901	120380	1.191	1.3381	105.95	3,526.70	12,292.86	228040000	28,645.66	428660000	23,138.07	595680000
10	1	2,020	10/1/2020	regularDay	5.00	1,912.30	750	1.11%	41.44	123190	3.062	91070	1.1747	1.2889	105.5	3,380.80	11,583.20	191860000	27,816.90	375650000	23,185.12	0
12	1	2,020	12/1/2020	regularDay	3.00	1,816.60	1660	2.17%	47.4	120980	2.868	39750	1.207	1.3415	104.3	3,662.40	12,455.33	227090000	29,823.92	435440000	26,787.54	732390000
1	2	2,020	1/2/2020	DayAfterHolida	5.00	1,528.10	270550	0.33%	65.56	81190	2.093	73010	1.117	1.3144	108.57	3,257.80	8,872.22	152650000	28,868.80	254290000	#N/A	#N/A
3	2	2,020	3/2/2020	regularDay	2.00	1,594.80	443530	1.79%	51.75	256950	1.797	94820	1.1132	1.275	108.3	3,090.20	8,877.98	350570000	26,703.32	637200000	21,344.08	1240000000
4	2	2,020	4/2/2020	regularDay	5.00	1,630.70	440	2.94%	32.1	266630	1.672	70800	1.0856	1.2392	107.9	2,526.90	7,635.66	248650000	21,413.44	534670000	17,818.72	1070000000
6	2	2,020	6/2/2020	regularDay	3.00	1,728.90	690	-0.83%	39.66	137540	1.876	41600	1.1169	1.2549	108.66	3,080.80	9,657.31	196140000	25,742.65	355240000	22,325.61	778910000
7	2	2,020	7/2/2020	DayBeforeHolic	5.00	1,790.00	186310	0.57%	43.19	88820	1.785	62060	1.1238	1.2466	107.48	3,130.00	10,341.89	167920000	25,827.36	350290000	22,145.96	735690000
9	2	2,020	9/2/2020	regularDay	4.00	1,936.90	10840	-1.72%	44.87	184150	2.931	151060	1.1853	1.3352	106.18	3,580.80	12,420.54	273590000	29,100.50	542540000	23,247.15	526370000

Dataset

Preprocessing – Collating to Build Dataset

Month	Day	Year	Date	US_Holiday?	Day_of_Week	Gold_Price_Av	Gold_Vol.	Gold_PriceCha	BrentOil_Price	BrentOil_Vol.	NatGas_Price_	NatGas_Vol.	EurUSD_Price	GBPUSD_Pric	USDJPY_Price	S&P_Price_Av	Nasdaq_Price_	Nasdaq_Vol.	DowJones_Pri	DowJones_Vol	Nikkei_Price_A	Nikkei_Vol.
4	1	2,020	4/1/2020	regularDay	4.00	1,584.10	350	-0.33%	28.04	180520	1.712	59130	1.0962	1.2376	107.15	2,470.50	7,486.29	261150000	20,943.51	508310000	18,065.41	1060000000
5	1	2,020	5/1/2020	regularDay	6.00	1,700.90	168020	0.40%	28.07	107670	2.134	73910	1.0983	1.2502	106.93	2,830.70	8,718.18	221880000	23,723.69	421590000	19,619.35	866440000
6	1	2,020	6/1/2020	regularDay	2.00	1,743.30	710	0.02%	38.43	124700	1.871	58660	1.1134	1.2492	107.58	3,055.70	9,598.89	163310000	25,475.02	341220000	22,062.39	721660000
7	1	2,020	7/1/2020	regularDay	4.00	1,779.90	263250	-1.14%	42.11	122360	1.721	76380	1.125	1.247	107.46	3,115.90	10,279.25	167520000	25,734.97	373980000	22,121.73	673600000
9	1	2,020	9/1/2020	regularDay	3.00	1,970.80	10950	0.02%	46	138080	2.901	120380	1.191	1.3381	105.95	3,526.70	12,292.86	228040000	28,645.66	428660000	23,138.07	595680000
10	1	2,020	10/1/2020	regularDay	5.00	1,912.30	750	1.11%	41.44	123190	3.062	91070	1.1747	1.2889	105.5	3,380.80	11,583.20	191860000	27,816.90	375650000	23,185.12	0
12	1	2,020	12/1/2020	regularDay	3.00	1,816.60	1660	2.17%	47.4	120980	2.868	39750	1.207	1.3415	104.3	3,662.40	12,455.33	227090000	29,823.92	435440000	26,787.54	732390000
1	2	2,020	1/2/2020	DayAfterHolida	5.00	1,528.10	270550	0.33%	65.56	81190	2.093	73010	1.117	1.3144	108.57	3,257.80	8,872.22	152650000	28,868.80	254290000	#N/A	#N/A
3	2	2,020	3/2/2020	regularDay	2.00	1,594.80	443530	1.79%	51.75	256950	1.797	94820	1.1132	1.275	108.3	3,090.20	8,877.98	350570000	26,703.32	637200000	21,344.08	1240000000
4	2	2,020	4/2/2020	regularDay	5.00	1,630.70	440	2.94%	32.1	266630	1.672	70800	1.0856	1.2392	107.9	2,526.90	7,635.66	248650000	21,413.44	534670000	17,818.72	1070000000
6	2	2,020	6/2/2020	regularDay	3.00	1,728.90	690	-0.83%	39.66	137540	1.876	41600	1.1169	1.2549	108.66	3,080.80	9,657.31	196140000	25,742.65	355240000	22,325.61	778910000
7	2	2,020	7/2/2020	DayBeforeHolic	5.00	1,790.00	186310	0.57%	43.19	88820	1.785	62060	1.1238	1.2466	107.48	3,130.00	10,341.89	167920000	25,827.36	350290000	22,145.96	735690000
9	2	2,020	9/2/2020	regularDay	4.00	1,936.90	10840	-1.72%	44.87	184150	2.931	151060	1.1853	1.3352	106.18	3,580.80	12,420.54	273590000	29,100.50	542540000	23,247.15	526370000

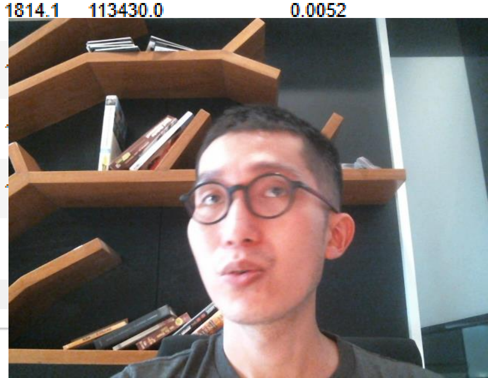
```
gold_df = pd.read_excel('gold_currencies_indices_v2.xlsx')
```

```
In [2]: gold_df
```

Out[2]:

	Month	Day	Year	Date	US_Holiday?	Day_of_Week	Gold_Price_Avg	Gold_Vol.	Gold_PriceChange	BrentOil_Pr
0	4	1	2020	2020-04-01	regularDay	4	1584.1	350.0	-0.0033	
1	5	1	2020	2020-05-01	regularDay	6	1700.9	168020.0	0.0040	
2	6	1	2020	2020-06-01	regularDay	2	1743.3	710.0	0.0002	
3	7	1	2020	2020-07-01	regularDay	4	1779.9	263250.0	-0.0114	
4	9	1	2020	2020-09-01	regularDay	3	1970.8	10950.0	0.0002	
...	...	...	...	...	...	...	...	...	...	
1258	12	30	2020	2020-12-30	DayBeforeHoliday	4	1893.4	133660.0	0.0069	
1259	12	30	2021	2021-12-30	regularDay	5	1814.1	113430.0	0.0052	
1260	12	30	2022	2022-12-30	DayBeforeHoliday	6				
1261	12	31	2020	2020-12-31	Holiday	5				
1262	12	31	2021	2021-12-31	DayBeforeHoliday	6				

1263 rows × 23 columns



Dataset

Preprocessing – Next Day's Gold Price Direction

First thing's first, the raw master dataset does not include next days' gold price change direction; the author needs to create that.

```
gold_df = gold_df.sort_values(by = "Date")

# Turning d+1 daily gold price percentage change to daily gold price change direction
gold_df["Gold_PriceChange_D+1"] = gold_df["Gold_PriceChange"].shift(-1)
## https://www.geeksforgeeks.org/create-a-new-column-in-pandas-dataframe-based-on-the-existing-colu
gold_df['Gold_Price_D+1_Direction'] = np.where(gold_df['Gold_PriceChange_D+1'] > 0, 1, 0)
```

Dataset

Preprocessing – Day of Week and Holidays

The next low hanging fruit in preprocessing seems to be converting day-of-week & holidays into dummy variables.

n [5]:

```
def dow(value):
    if value == 1:
        return "Sun"
    elif value == 2:
        return "Mon"
    elif value == 3:
        return "Tue"
    elif value == 4:
        return "Wed"
    elif value == 5:
        return "Thu"
    elif value == 6:
        return "Fri"
    elif value == 7:
        return "Sat"

gold_df['dow_spelled'] = gold_df['Day_of_Week'].map(dow)

# Turning categorical data, day of week and holiday, into dummies
## https://www.geeksforgeeks.org/how-to-convert-categorical-data-to-binary-data-in-python/
holiday_df = pd.get_dummies(gold_df['US_Holiday?'])
dow_df = pd.get_dummies(gold_df['dow_spelled'])
```

Dataset

Preprocessing – Lag Data

The author suspects tomorrow's price is not only related to today's price & volume, but also the past several days. Here I create columns of shifted past days data to be included in building the model.

```
# https://www.statology.org/pandas-lag/#:~:text=You%20can%20use%20the%20shift,lagged%20values%20of%

gold_df["Gold_Price_Avg_D-1"] = gold_df["Gold_Price_Avg"].shift(1)
gold_df["Gold_Price_Avg_D-2"] = gold_df["Gold_Price_Avg"].shift(2)
gold_df["Gold_Price_Avg_D-3"] = gold_df["Gold_Price_Avg"].shift(3)
gold_df["Gold_Price_Avg_D-4"] = gold_df["Gold_Price_Avg"].shift(4)
gold_df["Gold_Price_Avg_D-5"] = gold_df["Gold_Price_Avg"].shift(5)

gold_df["Gold_Vol._D-1"] = gold_df["Gold_Vol."].shift(1)
gold_df["Gold_Vol._D-2"] = gold_df["Gold_Vol."].shift(2)
gold_df["Gold_Vol._D-3"] = gold_df["Gold_Vol."].shift(3)
gold_df["Gold_Vol._D-4"] = gold_df["Gold_Vol."].shift(4)
gold_df["Gold_Vol._D-5"] = gold_df["Gold_Vol."].shift(5)

gold_df["BrentOil_Price_Avg_D-1"] = gold_df["BrentOil_Price_Avg"].shift(1)
gold_df["BrentOil_Price_Avg_D-2"] = gold_df["BrentOil_Price_Avg"].shift(2)
gold_df["BrentOil_Price_Avg_D-3"] = gold_df["BrentOil_Price_Avg"].shift(3)
gold_df["BrentOil_Price_Avg_D-4"] = gold_df["BrentOil_Price_Avg"].shift(4)
gold_df["BrentOil_Price_Avg_D-5"] = gold_df["BrentOil_Price_Avg"].shift(5)

gold_df["BrentOil_Vol._D-1"] = gold_df["BrentOil_Vol."].shift(1)
gold_df["BrentOil_Vol._D-2"] = gold_df["BrentOil_Vol."].shift(2)
gold_df["BrentOil_Vol._D-3"] = gold_df["BrentOil_Vol."].shift(3)
gold_df["BrentOil_Vol._D-4"] = gold_df["BrentOil_Vol."].shift(4)
gold_df["BrentOil_Vol._D-5"] = gold_df["BrentOil_Vol."].shift(5)

gold_df["NatGas_Price_Avg_D-1"] = gold_df["NatGas_Price_Avg"].shift(1)
gold_df["NatGas_Price_Avg_D-2"] = gold_df["NatGas_Price_Avg"].shift(2)
gold_df["NatGas_Price_Avg_D-3"] = gold_df["NatGas_Price_Avg"].shift(3)
gold_df["NatGas_Price_Avg_D-4"] = gold_df["NatGas_Price_Avg"].shift(4)
gold_df["NatGas_Price_Avg_D-5"] = gold_df["NatGas_Price_Avg"].shift(5)

gold_df["NatGas_Vol._D-1"] = gold_df["NatGas_Vol."].shift(1)
gold_df["NatGas_Vol._D-2"] = gold_df["NatGas_Vol."].shift(2)
gold_df["NatGas_Vol._D-3"] = gold_df["NatGas_Vol."].shift(3)
gold_df["NatGas_Vol._D-4"] = gold_df["NatGas_Vol."].shift(4)
gold_df["NatGas_Vol._D-5"] = gold_df["NatGas_Vol."].shift(5)

gold_df["EurUSD_Price_Avg_D-1"] = gold_df["EurUSD_Price_Avg"].shift(1)
```

Dataset

Preprocessing, Cleaning – Dropping Redundant Columns

Dropping out redundant columns as they have been turned to dummy variables. Also, dropping out dummy variables to allow sufficient degree of freedoms and avoid multicollinearities.

7]:

```
Processed_gold_df = Processed_gold_df.drop(['Day_of_Week','US_Holiday?','dow_spelled', 'Date',
                                            'Fri', 'regularDay', 'Gold_PriceChange_D+1','Gold_PriceChange' ], axis = 1)
```

Dataset

Preprocessing

Let's see how the data looks like now:

```
Processed_gold_df
```

[8]:

	Month	Day	Year	Gold_Price_Avg	Gold_Vol.	BrentOil_Price_Avg	BrentOil_Vol.	NatGas_Price_Avg	NatGas_Vol.	EurUSD_Price_Avg	...	Nikkei_Vol._
7	1	2	2020	1528.1	270550.0	65.56	81190.0	2.093	73010.0	1.1170	...	N
16	1	3	2020	1552.4	436740.0	67.76	202660.0	2.112	61290.0	1.1158	...	N
41	1	6	2020	1568.8	558970.0	68.07	152510.0	2.134	79910.0	1.1193	...	N
50	1	7	2020	1574.3	435870.0	67.57	142790.0	2.153	82700.0	1.1151	...	N
59	1	8	2020	1560.2	813410.0	64.79	270030.0	2.134	170260.0	1.1103	...	N
...	...	...	...	...	...	...	...	...	...	...	...	
1162	11	15	2024	2570.1	179890.0	71.04	307470.0	2.823	181090.0	1.0541	...	1.630000e+
1174	11	18	2024	2614.6	195290.0	73.30	333850.0	2.973	188340.0	1.0599	...	1.500000e+
1177	11	19	2024	2631.0	202240.0	73.31	354130.0	2.998	211930.0	1.0595		
1180	11	20	2024	2651.7	182010.0	72.81	260510.0	3.193	225690.0	1.0543		
1183	11	21	2024	2670.7	0.0	73.62	0.0	3.356	0.0	1.0542		

1263 rows × 107 columns

Dataset

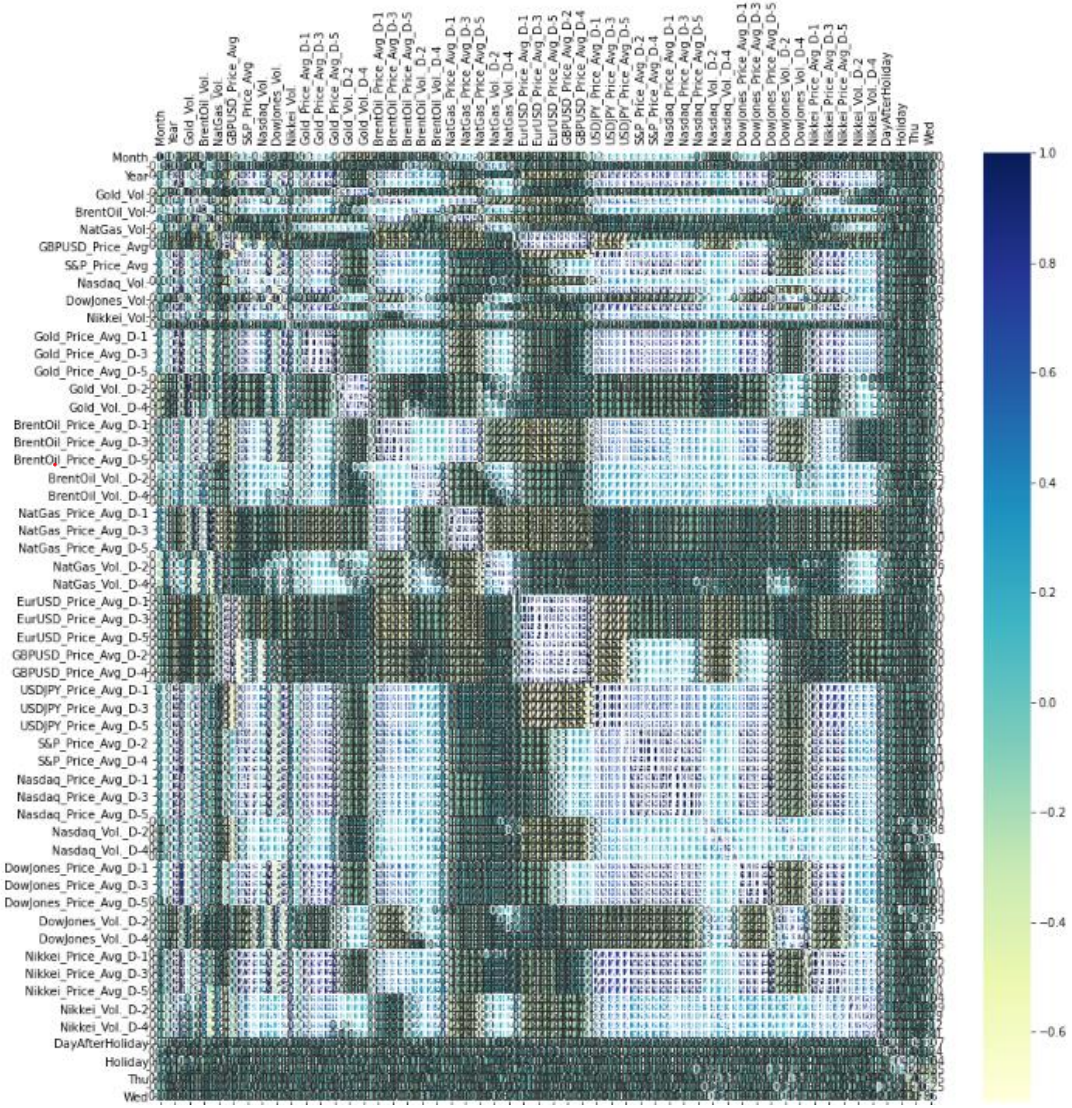
Resulting Columns

```
list(Processed_gold_df.columns)
```

```
['Month',
 'Day',
 'Year',
 'Gold_Price_Avg',
 'Gold_Vol.',
 'BrentOil_Price_Avg',
 'BrentOil_Vol.',
 'NatGas_Price_Avg',
 'NatGas_Vol.',
 'EurUSD_Price_Avg',
 'GBPUSD_Price_Avg',
 'USDJPY_Price_Avg',
 'S&P_Price_Avg',
 'Nasdaq_Price_Avg',
 'Nasdaq_Vol.',
 'DowJones_Price_Avg',
 'DowJones_Vol.',
 'Nikkei_Price_Avg',
 'Nikkei_Vol.',
 'Gold_Price_D+1_Direction',
 'Gold_Price_Avg_D-1',
 'Gold_Price_Avg_D-2',
 'Gold_Price_Avg_D-3',
 'Gold_Price_Avg_D-4',
 'Gold_Price_Avg_D-5',
 'Gold_Vol._D-1',
 'Gold_Vol._D-2',
 'Gold_Vol._D-3',
 'Gold_Vol._D-4',
 'Gold_Vol._D-5',
 'BrentOil_Price_Avg_D-1',
 'BrentOil_Price_Avg_D-2',
 'BrentOil_Price_Avg_D-3',
 'BrentOil_Price_Avg_D-4',
 'BrentOil_Price_Avg_D-5',
 'BrentOil_Vol._D-1',
 'BrentOil_Vol._D-2',
 'BrentOil_Vol._D-3',
 'BrentOil_Vol._D-4',
 'BrentOil_Vol._D-5',
 'NatGas_Price_Avg_D-1',
 'NatGas_Price_Avg_D-2',
 'NatGas_Price_Avg_D-3',
 'NatGas_Price_Avg_D-4',
 'NatGas_Price_Avg_D-5',
 'NatGas_Vol._D-1',
 'NatGas_Vol._D-2',
 'NatGas_Vol._D-3',
 'NatGas_Vol._D-4',
 'NatGas_Vol._D-5',
 'EurUSD_Price_Avg_D-1',
 'EurUSD_Price_Avg_D-2',
 'EurUSD_Price_Avg_D-3',
 'EurUSD_Price_Avg_D-4',
 'EurUSD_Price_Avg_D-5',
 'GBPUSD_Price_Avg_D-1',
 'GBPUSD_Price_Avg_D-2',
 'GBPUSD_Price_Avg_D-3',
 'GBPUSD_Price_Avg_D-4',
 'GBPUSD_Price_Avg_D-5',
 'USDJPY_Price_Avg_D-1',
 'USDJPY_Price_Avg_D-2',
 'USDJPY_Price_Avg_D-3',
 'USDJPY_Price_Avg_D-4',
 'USDJPY_Price_Avg_D-5',
 'S&P_Price_Avg_D-1',
 'S&P_Price_Avg_D-2',
 'S&P_Price_Avg_D-3',
 'S&P_Price_Avg_D-4',
 'S&P_Price_Avg_D-5',
 'Nasdaq_Price_Avg_D-1',
 'Nasdaq_Price_Avg_D-2',
 'Nasdaq_Price_Avg_D-3',
 'Nasdaq_Price_Avg_D-4',
 'Nasdaq_Price_Avg_D-5',
 'DowJones_Price_Avg_D-1',
 'DowJones_Price_Avg_D-2',
 'DowJones_Price_Avg_D-3',
 'DowJones_Price_Avg_D-4',
 'DowJones_Price_Avg_D-5',
 'DowJones_Vol._D-1',
 'DowJones_Vol._D-2',
 'DowJones_Vol._D-3',
 'DowJones_Vol._D-4',
 'DowJones_Vol._D-5',
 'Nikkei_Price_Avg_D-1',
 'Nikkei_Price_Avg_D-2',
 'Nikkei_Price_Avg_D-3',
 'Nikkei_Price_Avg_D-4',
 'Nikkei_Price_Avg_D-5',
 'Nikkei_Vol._D-1',
 'Nikkei_Vol._D-2',
 'Nikkei_Vol._D-3',
 'Nikkei_Vol._D-4',
 'Nikkei_Vol._D-5',
 'DayAfterHoliday',
 'DayBeforeHoliday',
 'Holiday',
 'Mon',
 'Thu',
 'Tue',
 'Wed']
```

Exploratory Data Analysis

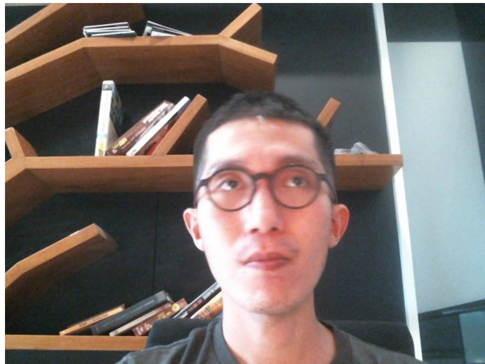
Correlation Matrix



Dataset has

107

columns

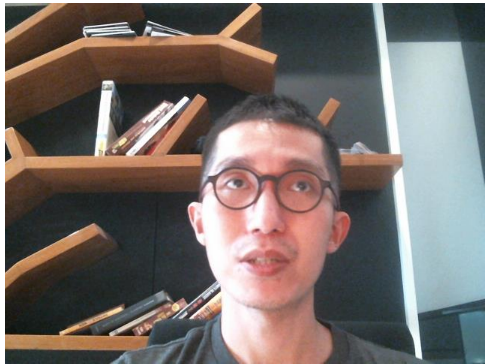


Exploratory Data Analysis

Correlation Matrix – Correlation between Response & each Predictors

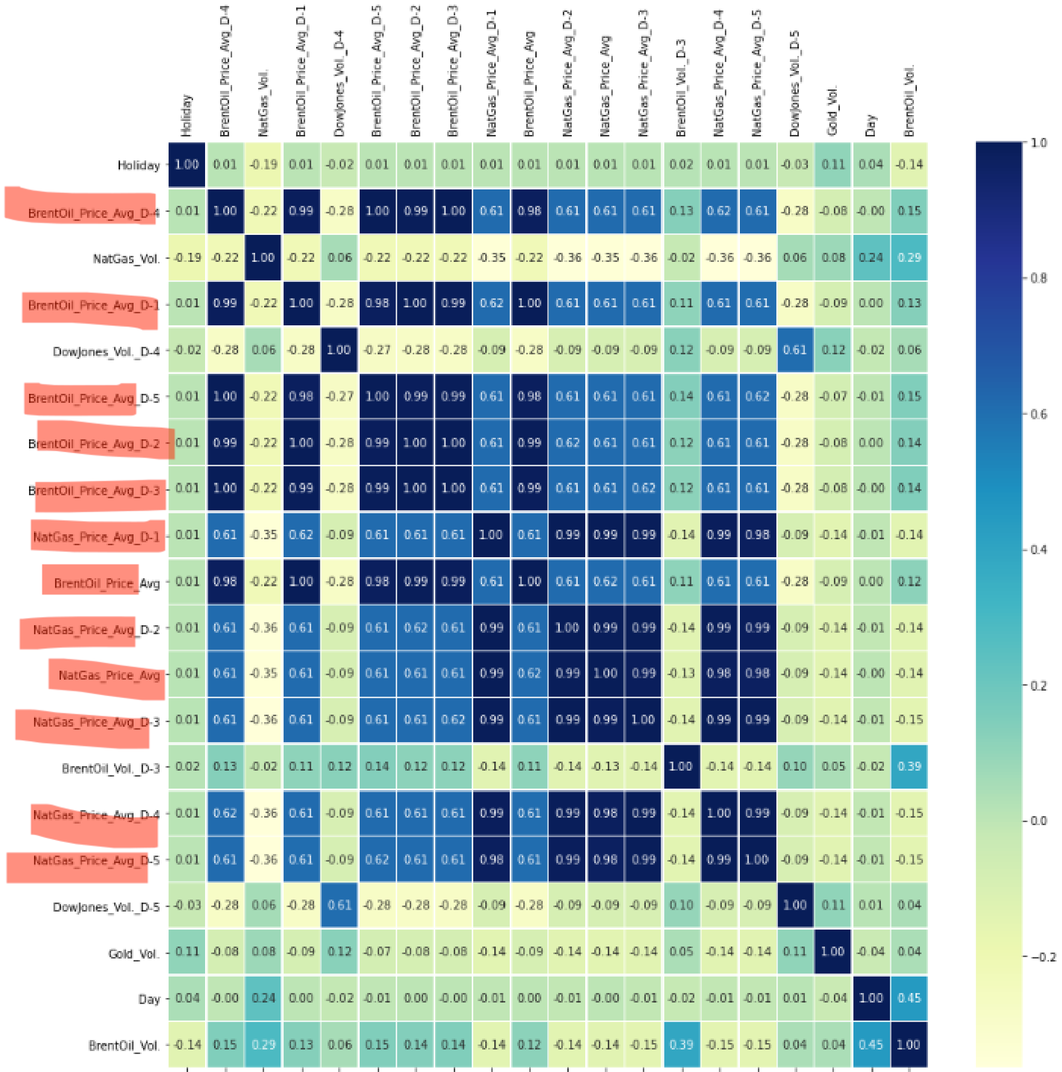
```python
# https://www.geeksforgeeks.org/python-dictionary/

potential_significant_predictors = []
corr_dict = {}

predictors_corr = Processed_gold_df.corr()['Gold_Price_D+1_Direction']

for features_idx in range(0, len(Processed_gold_df.columns )):
    corr_magn = abs(predictors_corr[features_idx])
    corr_dict[features_idx] = corr_magn


# https://www.geeksforgeeks.org/how-to-sort-a-dictionary-by-value-in-python/
# https://www.geeksforgeeks.org/python-sorted-function/

sorted_corr_dict = dict(sorted(corr_dict.items(), key = lambda item: item[1], reverse = True))

for key_idx in range(1, 21):
    potential_significant_predictors.append(Processed_gold_df.columns[list(sorted_corr_dict.keys())[key_idx]])

    cur_predictor = Processed_gold_df.columns[list(sorted_corr_dict.keys())[key_idx]]
    cur_magnitude = list(sorted_corr_dict.values())[key_idx]
    print(cur_predictor, " :", cur_magnitude)
```

```
Holiday  : 0.09343113114048412
BrentOil_Price_Avg_D-4  : 0.046923571584731934
NatGas_Vol.  : 0.04686017465023451
BrentOil_Price_Avg_D-1  : 0.04614732470332546
DowJones_Vol._D-4  : 0.04592757708562616
BrentOil_Price_Avg_D-5  : 0.04565373059195448
BrentOil_Price_Avg_D-2  : 0.04536080230278412
BrentOil_Price_Avg_D-3  : 0.04398866441766228
NatGas_Price_Avg_D-1  : 0.04381668387406184
BrentOil_Price_Avg  : 0.04221902213518596
NatGas_Price_Avg_D-2  : 0.041802682749530404
NatGas_Price_Avg  : 0.04174057168618509
NatGas_Price_Avg_D-3  : 0.03898571086243888
BrentOil_Vol._D-3  : 0.03799187077581839
NatGas_Price_Avg_D-4  : 0.03797592991211697
NatGas_Price_Avg_D-5  : 0.03509820472177668
DowJones_Vol._D-5  : 0.034932646190208104
Gold_Vol.  : 0.033744617250401
Day  : 0.030074742626925874
BrentOil_Vol.  : 0.030030573224653266
```



Exploratory Data Analysis

Correlation Matrix between the More Significant Predictors



Since the highlighted predictors seem to highly correlate with each other,
we only need to include one in the model
with the hope that it would also represent the others.

Exploratory Data Analysis
Checking for NAs

Let's use VIF to further avoid multicollinearity. For starters, let's pick the ones that do not correlate with one another while prioritizing ones that have high correlations with the response variable.

```
potential_significant_predictors = ['Holiday', 'NatGas_Vol.', 'BrentOil_Price_Avg', 'DowJones_Vol._D-4', 'NatGas_Price_Avg',
x_signi_df = pd.DataFrame(data = Processed_gold_df, columns = potential_significant_predictors)

len(x_signi_df) - x_signi_df.describe().loc['count']
```

```
16]: Holiday                0.0
     NatGas_Vol.            0.0
     BrentOil_Price_Avg     0.0
     DowJones_Vol._D-4     36.0
     NatGas_Price_Avg       0.0
     Gold_Vol.              0.0
     Day                    0.0
     BrentOil_Vol.          0.0
     Name: count, dtype: float64
```

Since out of 1263 rows, there are only 36 rows that has NA values, these rows are dropped; 36 rows is arguably not a lot to lose any sleep on, at the sam time, they could cause calculations & models to malfunction.

Exploratory Data Analysis

Variation Inflation Factors VIF results

All 8 *significant* predictors

	feature	VIF
0	Holiday	1.151311
1	NatGas_Vol.	5.766440
2	BrentOil_Price_Avg	19.458267
3	DowJones_Vol._D-4	8.436561
4	NatGas_Price_Avg	9.121030
5	Gold_Vol.	1.734557
6	Day	5.587290
7	BrentOil_Vol.	11.407500

Without BrentOil_Vol

	feature	VIF
0	Holiday	1.121965
1	NatGas_Vol.	5.615108
2	BrentOil_Price_Avg	15.535491
3	DowJones_Vol._D-4	7.974094
4	NatGas_Price_Avg	8.450883
5	Gold_Vol.	1.726774
6	Day	4.440582

Without BrentOil_Price

	feature	VIF
0	Holiday	1.127874
1	NatGas_Vol.	5.522673
2	DowJones_Vol._D-4	8.294477
3	NatGas_Price_Avg	3.777801
4	Gold_Vol.	1.730858
5	Day	5.542320
6	BrentOil_Vol.	9.107754

As AnalyticsVidhya.com explained, "VIF exceeding 5 or 10 indicates high multicollinearity between independent variable and the others".

Exploratory Data Analysis

Variation Inflation Factors VIF results

All 8 *significant* predictors

	feature	VIF
0	Holiday	1.151311
1	NatGas_Vol.	5.766440
2	BrentOil_Price_Avg	19.458267
3	DowJones_Vol._D-4	8.436561
4	NatGas_Price_Avg	9.121030
5	Gold_Vol.	1.734557
6	Day	5.587290
7	BrentOil_Vol.	11.407500

Without BrentOil_Vol

	feature	VIF
0	Holiday	1.121965
1	NatGas_Vol.	5.615108
2	BrentOil_Price_Avg	15.535491
3	DowJones_Vol._D-4	7.974094
4	NatGas_Price_Avg	8.450883
5	Gold_Vol.	1.726774
6	Day	4.440582

Without BrentOil_Price

	feature	VIF
0	Holiday	1.127874
1	NatGas_Vol.	5.522673
2	DowJones_Vol._D-4	8.294477
3	NatGas_Price_Avg	3.777801
4	Gold_Vol.	1.730858
5	Day	5.542320
6	BrentOil_Vol.	9.107754

Without BrentOil_Price
Without BrentOil_Vol

	feature	VIF
0	Holiday	1.114697
1	NatGas_Vol.	5.055551
2	DowJones_Vol._D-4	7.391046
3	NatGas_Price_Avg	3.717591
4	Gold_Vol.	1.714140
5	Day	4.350121

As AnalyticsVidhya.com explained, "VIF exceeding 5 or 10 indicates high multicollinearity between independent variable and the others".

Exploratory Data Analysis
Standardizing Predictors

```
potential_significant_predictors = ['Holiday', 'NatGas_Vol.', 'DowJones_Vol._D-4', 'NatGas_Price_Avg', 'Gold_Vol.', 'Day']
x_signi_df = pd.DataFrame(data = Processed_gold_df, columns = potential_significant_predictors)

object = StandardScaler()
x_signi_df_scale = object.fit_transform(x_signi_df)
y_df = Processed_gold_df['Gold_Price_D+1_Direction']

x_arr = np.asarray(x_signi_df_scale)
y_arr = np.asarray(y_df)
```

Exploratory Data Analysis

Checking for Imbalances

```
sum(y_df)/len(y_df)
```

```
5]: 0.5352335708630246
```

Models

- Logistic Regression
- KNN
- Decision Tree
- Random Forest
- SVM
- Logistic Regression, with P hacking

Models

Logistic Regression

Logistic Regression

```
LogReg = LogisticRegression(random_state = 2)
LogReg.fit(x_train, y_train)
logreg_acc = LogReg.score(x_test, y_test)
print(logreg_acc)
```

```
0.5059288537549407
```

```
y_train_pred_proba = LogReg.predict_proba(x_train)[::,1]
y_test_pred_proba = LogReg.predict_proba(x_test)[::,1]

auc_train = metrics.roc_auc_score(y_train, y_train_pred_proba)
auc_test = metrics.roc_auc_score(y_test, y_test_pred_proba)

print("auc_train: ", auc_train)
print("auc_test: ", auc_test)
```

```
auc_train:  0.5581134814440798
auc_test:  0.5511520160280491
```

```
ypred_logreg = []

for j in range(0, len(y_test_pred_proba)):
    if y_test_pred_proba[j] > 0.5:
        ypred_logreg.append(1)
    elif y_test_pred_proba[j] < 0.5:
        ypred_logreg.append(0)

ypred_logreg = np.array(ypred_logreg)

confusion_matrix(y_test, ypred_logreg)

print(classification_report(y_test, ypred_logreg))
```

	precision	recall	f1-score	support
0.0	0.43	0.11	0.17	121
1.0	0.52	0.87	0.65	132
accuracy			0.51	253
macro avg	0.47	0.49	0.41	253
weighted avg	0.48	0.51	0.42	253

```
param_grid_logreg = [
    {'penalty':['l1','l2','elasticnet','none'],
    'C' : np.logspace(-4,4,20),
    'solver': ['lbfgs','newton-cg','liblinear','sag','saga'],
    'max_iter'  : [1, 10, 100,1000]
}
]

clf_logreg = GridSearchCV(LogReg, param_grid = param_grid_logreg, cv = 3, verbose=True,n_jobs=-1)
clf_logreg.fit(x_train, y_train)
```

```
Fitting 3 folds for each of 1600 candidates, totalling 4800 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 32 concurrent workers.
[Parallel(n_jobs=-1)]: Done 136 tasks      | elapsed:   37.1s
[Parallel(n_jobs=-1)]: Done 386 tasks      | elapsed:   39.1s
[Parallel(n_jobs=-1)]: Done 736 tasks      | elapsed:   41.9s
[Parallel(n_jobs=-1)]: Done 1186 tasks      | elapsed:   45.5s
[Parallel(n_jobs=-1)]: Done 1736 tasks      | elapsed:   49.9s
[Parallel(n_jobs=-1)]: Done 2386 tasks      | elapsed:   55.5s
[Parallel(n_jobs=-1)]: Done 3136 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 3986 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 4800 out of 4800 | elapsed:  1.3min finished
```

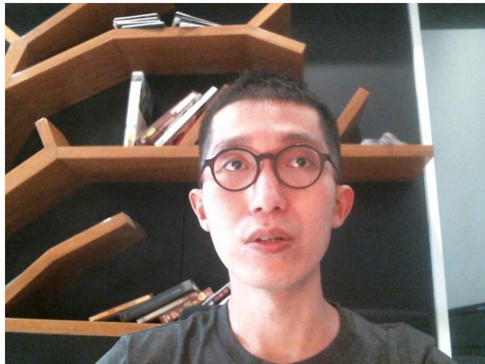
```
GridSearchCV(cv=3, error_score=nan,
             estimator=LogisticRegression(C=1.0, class_weight=None, dual=False,
                                          fit_intercept=True,
                                          intercept_scaling=1, l1_ratio=None,
                                          max_iter=100, multi_class='auto',
                                          n_jobs=None, penalty='l2',
                                          random_state=2, solver='lbfgs',
                                          tol=0.0001, verbose=0,
                                          warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid=[{'C': array([1.00000000e-04, 2.63665...
       2.33572147e-01, 6.15848211e-01, 1.62377674e+00, 4.28133240e+00,
       1.12883789e+01, 2.97635144e+01, 7.84759970e+01, 2.06913808e+02,
       5.45559478e+02, 1.43844989e+03, 3.79269019e+03, 1.00000000e+04]),
                          'max_iter': [1, 10, 100, 1000],
                          'penalty': ['l1', 'l2', 'elasticnet', 'none'],
                          'solver': ['lbfgs', 'newton-cg', 'liblinear', 'sag',
                                     'saga']}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=True)
```

```
print("best score: ", clf_logreg.best_score_)
print("best param: ", clf_logreg.best_params_)
```

```
best score:  0.5693208044840093
best param:  {'C': 0.004832930238571752, 'max_iter': 1, 'penalty': 'l2', 'solver': 'saga'}
```

```
LogReg_V2 = LogisticRegression(C =0.004832930238571752, max_iter = 1, penalty = 'l2', solver = 'saga', random_state = 3).fit
y_train_pred_proba_v2 = LogReg_V2.predict_proba(x_train)[::,1]
y_test_pred_proba_v2 = LogReg_V2.predict_proba(x_test)[::,1]

auc_train_v2 = metrics.roc_auc_score(y_train, y_train_pred_proba_v2)
auc_test_v2 = metrics.roc_auc_score(y_test, y_test_pred_proba_v2)

print("auc_train: ", auc_train)
print("auc_test: ", auc_test)
```

```
auc_train:  0.5581134814440798
auc_test:  0.5511520160280491
```



Models

Logistic Regression, with P hacking

The plan is to start with all initial predictors, fit them into a logistic regression model with train data, throw away all the ones with insignificant p values, repeat these steps until all the lasting predictors have significant p values. Standard 0.05 is used as significance level.

Logistic regression summaries, including p values, are easier to see using statsmodels than sklearn.

Models - Logistic Regression, with P hacking

Data Re-Exploration: Imputing Nas with Median Values

```python
list(len(Processed_gold_df) - x_df.describe().loc['count'])
```

```
[0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 32.0,
 31.0,
 33.0,
 31.0,
 32.0,
 76.0,
 76.0,
 1.0,
 2.0,
 3.0,
 4.0,
 5.0,
 1.0,
 2.0,
 3.0,
 4.0,
 5.0,
 1.0,
 2.0,
 3.0,
 4.0,
 5.0,
 1.0,
 2.0,
 3.0,
 4.0,
 5.0,
 1.0,
 2.0,
 3.0,
 4.0,
 5.0,
 1.0,
 2.0,
 3.0,
```

It seems that there are columns with sizeable NAs, let's broadly impute them with median values

```python
features = Processed_gold_df.columns
imputer = SimpleImputer(strategy = 'median')
Processed_gold_df[features] = pd.DataFrame(imputer.fit_transform(Processed_gold_df[features]), columns = features)

x_p_df = Processed_gold_df.drop(['Gold_Price_D+1_Direction'], axis = 1)

object = StandardScaler()
x_p_df_scale = object.fit_transform(x_p_df)
y_p_df = Processed_gold_df['Gold_Price_D+1_Direction']

#x_p_arr = np.asarray(x_p_df_scale)
#y_p_arr = np.asarray(y_p_df)

x_p_train, x_p_test, y_p_train, y_p_test = train_test_split(x_p_df_scale, y_p_df, random_state = 5, test_size = 0.20, shuffl
```

Models - Logistic Regression, with P hacking

Fitting into a Logistic Regression Model: Iteration 1

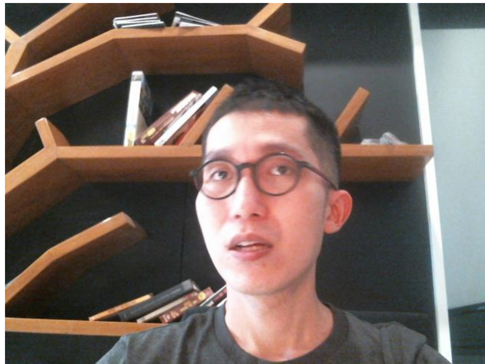
```
random.seed(6)

LogReg_p_1 = sm.Logit(y_p_train, x_p_train).fit()
LogReg_p_1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.644352
         Iterations 7
```

Logit Regression Results

Dep. Variable:	Gold_Price_D+1_Direction	No. Observations:	1010
Model:	Logit	Df Residuals:	904
Method:	MLE	Df Model:	105
Date:	Fri, 06 Dec 2024	Pseudo R-squ.:	0.06678
Time:	05:51:43	Log-Likelihood:	-650.80
converged:	True	LL-Null:	-697.37
Covariance Type:	nonrobust	LLR p-value:	0.7896

	coef	std err	z	P>|z|	[0.025	0.975]
x1	-0.0724	0.163	-0.443	0.658	-0.392	0.248
x2	0.0304	0.100	0.304	0.761	-0.166	0.227
x3	-0.2931	0.522	-0.561	0.574	-1.316	0.730
x4	-1.7673	1.032	-1.713	0.087	-3.789	0.254
x5	-0.2436	0.142	-1.718	0.086	-0.521	0.034
x6	0.9593	0.810	1.185	0.236	-0.628	2.546
x7	0.0312	0.118	0.265	0.791	-0.200	0.263
x8	0.8819	0.705	1.251	0.211	-0.500	2.264
x9	-0.0431	0.097	-0.443	0.658	-0.234	0.148
x10	2.0335	1.216	1.672	0.095	-0.350	4.417
x11	0.2490	0.997	0.250	0.803	-1.705	2.203
x12	4.4391	1.972	2.251	0.024	0.574	8.304
x13	4.4560	3.410	1.307	0.191	-2.228	11.140
x14	-3.0678	1.930	-1.589	0.112	-6.852	0.716
x15	-0.1913	0.156	-1.224	0.221	-0.497	0.115
x16	-1.4331	1.808	-0.793	0.428	-4.976	2.110
x17	0.1224	0.177	0.693	0.488	-0.224	0.469
x18	-0.2899	0.277	-1.045	0.296	-0.833	0.254
x19	0.1786	0.128	1.397	0.162	-0.072	0.429
x20	1.5692	1.261	1.244	0.214	-0.903	4.042
x21	1.0914	1.160	0.941	0.347	-1.182	3.364
x22	-0.9464	1.153	-0.821	0.412	-3.206	1.313
x23	1.1230	1.250	0.898	0.369	-1.328	3.574
x24	-1.4126	0.995	-1.419	0.156	-3.364	0.538
x25	0.3081	0.171	1.805	0.071	-0.026	0.643
x26	0.0604	0.165	0.365	0.715	-0.264	0.385
x27	-0.2869	0.181	-1.589	0.112	-0.641	0.067
x28	0.2351	0.167	1.405	0.160	-0.093	0.563
x29	-0.0601	0.137	-0.438	0.661	-0.329	0.209
x30	-1.6797	1.117	-1.504	0.132	-3.868	0.509
x31	0.2818	1.151	0.245	0.807	-1.975	2.538
x32	1.5245	1.148	1.328	0.184	-0.725	3.774
x33	-1.9393	1.156	-1.678	0.093	-4.204	0.326
x34	0.6491	0.800	0.811	0.417	-0.919	2.218
x35	-0.0612	0.124	-0.494	0.621	-0.304	0.181
x36	-0.0309	0.129	-0.240	0.810	-0.283	0.222
x37	0.0109	0.127	0.085	0.932	-0.239	0.261
x38	-0.1396	0.126	-1.110	0.267	-0.386	0.107
x39	0.0907	0.109	0.831	0.406	-0.123	0.305
x40	-1.5041	0.925	-1.626	0.104	-3.317	0.309
x41	-0.4946	0.957	-0.517	0.605	-2.370	1.381
x42	0.0766	0.973	0.079	0.937	-1.830	1.983
x43	0.2738	0.979	0.280	0.780	-1.644	2.192
x44	0.7134	0.736	0.970	0.332	-0.728	2.155
x45	-0.0829	0.110	-0.756	0.450	-0.298	0.132
x46	0.0111	0.113	0.098	0.922	-0.210	0.232
x47	0.1290	0.111	1.165	0.244	-0.088	0.346
x48	0.0080	0.111	0.072	0.942	-0.209	0.225
x49	-0.1329	0.097	-1.369	0.171	-0.323	0.057
x50	-2.0355	1.678	-1.213	0.225	-5.324	1.253
x51	-0.6485	1.648	-0.393	0.694	-3.879	2.582
x52	1.7708	1.676	1.056	0.291	-1.514	5.056
x53	0.8635	1.670	0.517	0.605	-2.410	4.137
x54	-1.1192	1.242	-0.901	0.367	-3.553	1.315
x55	-1.4047	1.389	-1.011	0.312	-4.128	1.318
x56	-0.0174	1.343	-0.013	0.990	-2.650	2.616
x57	0.6267	1.362	0.460	0.645	-2.043	3.297
x58	-0.9011	1.376	-0.655	0.513	-3.599	1.796
x59	0.8144	0.987	0.826	0.409	-1.119	2.748
x60	-3.9018	2.458	-1.587	0.112	-8.720	0.916
x61	-0.7761	2.094	-0.371	0.711	-4.880	3.328
x62	1.3683	2.106	0.650	0.516	-2.759	5.496
x63	0.7202	2.457	0.293	0.769	-4.095	5.536
x64	-1.3251	1.986	-0.667	0.505	-5.217	2.567
x65	5.5306	3.782	1.462	0.144	-1.882	12.943
x66	-1.2038	3.798	-0.317	0.751	-8.647	6.239
x67	2.0096	4.241	0.474	0.636	-6.303	10.322
x68	-8.1163	4.085	-1.987	0.047	-16.123	-0.109
x69	1.1915	3.365	0.354	0.723	-5.405	7.788
x70	-2.2305	2.031	-1.098	0.272	-6.212	1.751
x71	0.1156	2.072	0.056	0.955	-3.945	4.176
x72	-0.5564	2.370	-0.235	0.814	-5.202	4.089
x73	4.8475	2.312	2.097	0.036	0.317	9.378
x74	-1.0231	1.883	-0.543	0.587	-4.713	2.667
x75	0.0619	0.159	0.389	0.697	-0.250	0.374
x76	0.1778	0.158	1.128	0.259	-0.131	0.487
x77	-0.1096	0.160	-0.684	0.494	-0.423	0.204
x78	0.0488	0.160	0.306	0.760	-0.264	0.362
x79	-0.1627	0.156	-1.045	0.296	-0.468	0.142
x80	-2.9566	2.120	-1.395	0.163	-7.112	1.198
x81	1.3597	2.083	0.653	0.514	-2.724	5.443
x82	-1.1028	2.187	-0.504	0.614	-5.389	3.183
x83	3.4618	2.087	1.659	0.097	-0.629	7.553
x84	-0.4764	1.757	-0.271	0.786	-3.921	2.968
x85	-0.1773	0.191	-0.928	0.354	-0.552	0.197
x86	-0.0473	0.183	-0.258	0.797	-0.407	0.312
x87	0.0109	0.189	0.058	0.954	-0.359	0.381
x88	0.1366	0.179	0.761	0.446	-0.215	0.488
x89	0.1197	0.181	0.662	0.508	-0.234	0.474
x90	0.0439	0.310	0.142	0.887	-0.563	0.651
x91	-0.0633	0.317	-0.199	0.842	-0.685	0.559
x92	-0.3506	0.283	-1.238	0.216	-0.906	0.205
x93	0.0914	0.263	0.347	0.728	-0.424	0.607
x94	0.2360	0.264	0.893	0.372	-0.282	0.754
x95	-0.0643	0.135	-0.475	0.635	-0.329	0.201
x96	-0.0156	0.140	-0.111	0.911	-0.290	0.259
x97	0.2441	0.143	1.712	0.087	-0.035	0.524
x98	-0.3077	0.134	-2.302	0.021	-0.570	-0.046
x99	0.0481	0.118	0.409	0.683	-0.182	0.278
x100	0.0255	0.078	0.325	0.745	-0.128	0.179
x101	0.0236	0.069	0.339	0.734	-0.113	0.160
x102	-0.1215	0.075	-1.613	0.107	-0.269	0.026
x103	-0.0803	0.099	-0.809	0.418	-0.275	0.114
x104	0.0012	0.098	0.012	0.990	-0.191	0.193
x105	-0.0500	0.102	-0.492	0.623	-0.249	0.149
x106	0.0071	0.101	0.070	0.944	-0.191	0.205



Models - Logistic Regression, with P hacking

Fitting into a Logistic Regression Model: Iteration 1

```
random.seed(6)

LogReg_p_1 = sm.Logit(y_p_train, x_p_train).fit()
LogReg_p_1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.644352
         Iterations 7
```

Logit Regression Results

Dep. Variable:	Gold_Price_D+1_Direction	No. Observations:	1010
Model:	Logit	Df Residuals:	904
Method:	MLE	Df Model:	105
Date:	Fri, 06 Dec 2024	Pseudo R-squ.:	0.06678
Time:	06:51:43	Log-Likelihood:	-650.80
converged:	True	LL-Null:	-697.37
Covariance Type:	nonrobust	LLR p-value:	0.7898

	coef	std err	z	P>|z|	[0.025	0.975]
x1	-0.0724	0.163	-0.443	0.658	-0.392	0.248
x2	0.0304	0.100	0.304	0.761	-0.166	0.227
x3	-0.2931	0.522	-0.561	0.574	-1.316	0.730
x4	-1.7673	1.032	-1.713	0.087	-3.789	0.254
x5	-0.2436	0.142	-1.718	0.086	-0.521	0.034
x6	0.9593	0.810	1.185	0.236	-0.628	2.546
x7	0.0312	0.118	0.265	0.791	-0.200	0.263
x8	0.8819	0.705	1.251	0.211	-0.500	2.264
x9	-0.0431	0.097	-0.443	0.658	-0.234	0.148
x10	2.0335	1.216	1.672	0.095	-0.350	4.417
x11	0.2490	0.997	0.250	0.803	-1.705	2.203
x12	4.4391	1.972	2.251	0.024	0.574	8.304
x13	4.4560	3.410	1.307	0.191	-2.228	11.140
x14	-3.0678	1.930	-1.589	0.112	-6.852	0.716
x15	-0.1913	0.156	-1.224	0.221	-0.497	0.115
x16	-1.4331	1.808	-0.793	0.428	-4.976	2.110
x17	0.1224	0.177	0.693	0.488	-0.224	0.469
x18	-0.2899	0.277	-1.045	0.296	-0.833	0.254
x19	0.1786	0.128	1.397	0.162	-0.072	0.429
x20	1.5692	1.261	1.244	0.214	-0.903	4.042
x21	1.0914	1.160	0.941	0.347	-1.182	3.364
x22	-0.9464	1.153	-0.821	0.412	-3.206	1.313
x23	1.1230	1.250	0.898	0.369	-1.328	3.574
x24	-1.4126	0.995	-1.419	0.156	-3.364	0.538
x25	0.3081	0.171	1.805	0.071	-0.026	0.643
x26	0.0604	0.165	0.365	0.715	-0.264	0.385
x27	-0.2869	0.181	-1.589	0.112	-0.641	0.067
x28	0.2351	0.167	1.405	0.160	-0.093	0.563
x29	-0.0601	0.137	-0.438	0.661	-0.329	0.209
x30	-1.6797	1.117	-1.504	0.132	-3.868	0.509
x31	0.2818	1.151	0.245	0.807	-1.975	2.538
x32	1.5245	1.148	1.328	0.184	-0.725	3.774
x33	-1.9393	1.156	-1.678	0.093	-4.204	0.326
x34	0.6491	0.800	0.811	0.417	-0.919	2.218
x35	-0.0612	0.124	-0.494	0.621	-0.304	0.181
x36	-0.0309	0.129	-0.240	0.810	-0.283	0.222
x37	0.0109	0.127	0.085	0.932	-0.239	0.261
x38	-0.1396	0.126	-1.110	0.267	-0.386	0.107
x39	0.0907	0.109	0.831	0.406	-0.123	0.305
x40	-1.5041	0.925	-1.626	0.104	-3.317	0.309
x41	-0.4946	0.957	-0.517	0.605	-2.370	1.381
x42	0.0766	0.973	0.079	0.937	-1.830	1.983
x43	0.2738	0.979	0.280	0.780	-1.644	2.192
x44	0.7134	0.736	0.970	0.332	-0.728	2.155
x45	-0.0829	0.110	-0.756	0.450	-0.298	0.132
x46	0.0111	0.113	0.098	0.922	-0.210	0.232
x47	0.1290	0.111	1.165	0.244	-0.088	0.346
x48	0.0080	0.111	0.072	0.942	-0.209	0.225
x49	-0.1329	0.097	-1.369	0.171	-0.323	0.057
x50	-2.0355	1.678	-1.213	0.225	-5.324	1.253
x51	-0.6485	1.648	-0.393	0.694	-3.879	2.582
x52	1.7708	1.676	1.056	0.291	-1.514	5.056
x53	0.8635	1.670	0.517	0.605	-2.410	4.137
x54	-1.1192	1.242	-0.901	0.367	-3.553	1.315
x55	-1.4047	1.389	-1.011	0.312	-4.128	1.318
x56	-0.0174	1.343	-0.013	0.990	-2.650	2.616
x57	0.6267	1.362	0.460	0.645	-2.043	3.297
x58	-0.9011	1.376	-0.655	0.513	-3.599	1.796
x59	0.8144	0.987	0.826	0.409	-1.119	2.748
x60	-3.9018	2.458	-1.587	0.112	-8.720	0.916
x61	-0.7761	2.094	-0.371	0.711	-4.880	3.328
x62	1.3683	2.106	0.650	0.516	-2.759	5.496
x63	0.7202	2.457	0.293	0.769	-4.095	5.536
x64	-1.3251	1.986	-0.667	0.505	-5.217	2.567
x65	5.5306	3.782	1.462	0.144	-1.882	12.943
x66	-1.2038	3.798	-0.317	0.751	-8.647	6.239
x67	2.0096	4.241	0.474	0.636	-6.303	10.322
x68	-8.1163	4.085	-1.987	0.047	-16.123	-0.109
x69	1.1915	3.365	0.354	0.723	-5.405	7.788
x70	-2.2305	2.031	-1.098	0.272	-6.212	1.751
x71	0.1156	2.072	0.056	0.955	-3.945	4.176
x72	-0.5564	2.370	-0.235	0.814	-5.202	4.089
x73	4.8475	2.312	2.097	0.036	0.317	9.378
x74	-1.0231	1.883	-0.543	0.587	-4.713	2.667
x75	0.0619	0.159	0.389	0.697	-0.250	0.374
x76	0.1778	0.158	1.128	0.259	-0.131	0.487
x77	-0.1096	0.160	-0.684	0.494	-0.423	0.204
x78	0.0488	0.160	0.306	0.760	-0.264	0.362
x79	-0.1627	0.156	-1.045	0.296	-0.468	0.142
x80	-2.9566	2.120	-1.395	0.163	-7.112	1.198
x81	1.3597	2.083	0.653	0.514	-2.724	5.443
x82	-1.1028	2.187	-0.504	0.614	-5.389	3.183
x83	3.4618	2.087	1.659	0.097	-0.629	7.553
x84	-0.4764	1.757	-0.271	0.786	-3.921	2.968
x85	-0.1773	0.191	-0.928	0.354	-0.552	0.197
x86	-0.0473	0.183	-0.258	0.797	-0.407	0.312
x87	0.0109	0.189	0.058	0.954	-0.359	0.381
x88	0.1366	0.179	0.761	0.446	-0.215	0.488
x89	0.1197	0.181	0.662	0.508	-0.234	0.474
x90	0.0439	0.310	0.142	0.887	-0.563	0.651
x91	-0.0633	0.317	-0.199	0.842	-0.685	0.559
x92	-0.3506	0.283	-1.238	0.216	-0.906	0.205
x93	0.0914	0.263	0.347	0.728	-0.424	0.607
x94	0.2360	0.264	0.893	0.372	-0.282	0.754
x95	-0.0643	0.135	-0.475	0.635	-0.329	0.201
x96	-0.0156	0.140	-0.111	0.911	-0.290	0.259
x97	0.2441	0.143	1.712	0.087	-0.036	0.524
x98	-0.3077	0.134	-2.302	0.021	-0.570	-0.046
x99	0.0481	0.118	0.409	0.683	-0.182	0.278
x100	0.0255	0.078	0.325	0.745	-0.128	0.179
x101	0.0236	0.069	0.339	0.734	-0.113	0.160
x102	-0.1215	0.075	-1.613	0.107	-0.269	0.026
x103	-0.0803	0.099	-0.809	0.418	-0.275	0.114
x104	0.0012	0.098	0.012	0.990	-0.191	0.193
x105	-0.0500	0.102	-0.492	0.623	-0.249	0.149
x106	0.0071	0.101	0.070	0.944	-0.191	0.205

```
signi_pred_p_1 = []

#https://www.statology.org/statsmodels-linear-regression-p-value/
for predictors_idx in range(0, len(x_p_df.columns)):
    if LogReg_p_1.pvalues[predictors_idx] < 0.05:
        signi_pred_p_1.append(x_p_df.columns[predictors_idx])

print(signi_pred_p_1)
```

```
['USDJPY_Price_Avg', 'S&P_Price_Avg_D-4', 'Nasdaq_Price_Avg_D-4', 'Nikkei_Vol._D-4']
```

Models - Logistic Regression, with P hacking

Fitting into a Logistic Regression Model: Iteration 2

```python
signi_pred_p_1 = []

#https://www.statology.org/statsmodels-linear-regression-p-value/
for predictors_idx in range(0, len(x_p_df.columns)):
    if LogReg_p_1.pvalues[predictors_idx] < 0.05:
        signi_pred_p_1.append(x_p_df.columns[predictors_idx])

print(signi_pred_p_1)
```

```
['USDJPY_Price_Avg', 'S&P_Price_Avg_D-4', 'Nasdaq_Price_Avg_D-4', 'Nikkei_Vol._D-4']
```

Logit Regression Results

Dep. Variable:	Gold_Price_D+1_Direction	No. Observations:	1010
Model:	Logit	Df Residuals:	1006
Method:	MLE	Df Model:	3
Date:	Fri, 06 Dec 2024	Pseudo R-squ.:	-0.002624
Time:	05:52:14	Log-Likelihood:	-899.20
converged:	True	LL-Null:	-897.37
Covariance Type:	nonrobust	LLR p-value:	1.000

	coef	std err	z	P>|z|	[0.025	0.975]
x1	-0.0691	0.092	-0.754	0.451	-0.249	0.111
x2	0.0958	0.337	0.284	0.776	-0.565	0.756
x3	-0.0485	0.321	-0.151	0.880	-0.679	0.581
x4	-0.0527	0.075	-0.699	0.484	-0.201	0.095

Models - Logistic Regression, with P hacking

Fitting into a Logistic Regression Model: Iteration 2

```python
signi_pred_p_1 = []

#https://www.statology.org/statsmodels-linear-regression-p-value/
for predictors_idx in range(0, len(x_p_df.columns)):
    if LogReg_p_1.pvalues[predictors_idx] < 0.05:
        signi_pred_p_1.append(x_p_df.columns[predictors_idx])

print(signi_pred_p_1)
```

```
['USDJPY_Price_Avg', 'S&P_Price_Avg_D-4', 'Nasdaq_Price_Avg_D-4', 'Nikkei_Vol._D-4']
```

Logit Regression Results

Dep. Variable:	Gold_Price_D+1_Direction	No. Observations:	1010
Model:	Logit	Df Residuals:	1006
Method:	MLE	Df Model:	3
Date:	Fri, 06 Dec 2024	Pseudo R-squ.:	-0.002624
Time:	05:52:14	Log-Likelihood:	-699.20
converged:	True	LL-Null:	-697.37
Covariance Type:	nonrobust	LLR p-value:	1.000

	coef	std err	z	P>|z|	[0.025	0.975]
x1	-0.0691	0.092	-0.754	0.451	-0.249	0.111
x2	0.0958	0.337	0.284	0.776	-0.565	0.756
x3	-0.0485	0.321	-0.151	0.880	-0.679	0.581
x4	-0.0527	0.075	-0.699	0.484	-0.201	0.095

None of the predictors is significant

Results

Summary

Model	AUC	Accuracy	Precision	Recall	F1 score
Logistic Regression	0.551	0.506	0.520	0.870	0.650
Logistic Regression, Hyperparameter tuned	0.551	0.522	0.520	0.900	0.660
Decision Tree	0.505	0.522	0.530	0.870	0.660
Decision Tree, Hyperparameter tuned	0.505	0.522	0.530	0.870	0.660
Random Forest		0.530	0.530	1.000	0.690
Random Forest, Hyperparameter tuned		0.530	0.530	1.000	0.690
KNN, BallPark	-	0.553	0.550	0.650	0.600
KNN, SKLearn	0.526	0.553	0.550	0.580	0.560
SVM		0.522	0.520	1.000	0.690
SVM, Hyperparameter tuned	0.500	0.534	0.530	0.970	0.680
Logistic Regression, with p hacking		-			[illegible]

Results

Summary

Model	AUC	Accuracy	Precision	Recall	F1 score
Logistic Regression	0.551	0.506	0.520	0.870	0.650
Logistic Regression, Hyperparameter tuned	0.551	0.522	0.520	0.900	0.660
Decision Tree	0.505	0.522	0.530	0.870	0.660
Decision Tree, Hyperparameter tuned	0.505	0.522	0.530	0.870	0.660
Random Forest		0.530	0.530	1.000	0.690
Random Forest, Hyperparameter tuned		0.530	0.530	1.000	0.690
KNN, BallPark	-	0.553	0.550	0.650	0.600
KNN, SKLearn	0.526	0.553	0.550	0.580	0.560
SVM		0.522	0.520	1.000	0.690
SVM, Hyperparameter tuned	0.500	0.534	0.530	0.970	0.680
Logistic Regression, with p hacking		-			

Area Under Curve AUC

The probability that the model, if given a randomly chosen positive and negative sample, will rank the positive higher than the negative.

Source

https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

Results
Summary

Model	AUC	Accuracy	Precision	Recall	F1 score
Logistic Regression	0.551	0.506	0.520	0.870	0.650
Logistic Regression, Hyperparameter tuned	0.551	0.522	0.520	0.900	0.660
Decision Tree	0.505	0.522	0.530	0.870	0.660
Decision Tree, Hyperparameter tuned	0.505	0.522	0.530	0.870	0.660
Random Forest		0.530	0.530	1.000	0.690
Random Forest, Hyperparameter tuned		0.530	0.530	1.000	0.690
KNN, BallPark	-	0.553	0.550	0.650	0.600
KNN, SKLearn	0.526	0.553	0.550	0.580	0.560
SVM		0.522	0.520	1.000	0.690
SVM, Hyperparameter tuned	0.500	0.534	0.530	0.970	0.680
Logistic Regression, with p hacking		-			

```
sum(y_df)/len(y_df)
```

```
0.5352335708630246
```

Area Under Curve AUC
The probability that the model, if given a randomly chosen positive and negative sample, will rank the positive higher than the negative.

Source
https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

Results
Summary

Model	AUC	Accuracy	Precision	Recall	F1 score
Logistic Regression	0.551	0.506	0.520	0.870	0.650
Logistic Regression, Hyperparameter tuned	0.551	0.522	0.520	0.900	0.660
Decision Tree	0.505	0.522	0.530	0.870	0.660
Decision Tree, Hyperparameter tuned	0.505	0.522	0.530	0.870	0.660
Random Forest		0.530	0.530	1.000	0.690
Random Forest, Hyperparameter tuned		0.530	0.530	1.000	0.690
KNN, BallPark	-	0.553	0.550	0.650	0.600
KNN, SKLearn	0.526	0.553	0.550	0.580	0.560
SVM		0.522	0.520	1.000	0.690
SVM, Hyperparameter tuned	0.500	0.534	0.530	0.970	0.680
Logistic Regression, with p hacking		-			

Accuracy
The proportion of all classification that were correct whether positive or negative.

Source
https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

Results

Summary

Model	AUC	Accuracy	Precision	Recall	F1 score
Logistic Regression	0.551	0.506	0.520	0.870	0.650
Logistic Regression, Hyperparameter tuned	0.551	0.522	0.520	0.900	0.660
Decision Tree	0.505	0.522	0.530	0.870	0.660
Decision Tree, Hyperparameter tuned	0.505	0.522	0.530	0.870	0.660
Random Forest		0.530	0.530	1.000	0.690
Random Forest, Hyperparameter tuned		0.530	0.530	1.000	0.690
KNN, BallPark	-	0.553	0.550	0.650	0.600
KNN, SKLearn	0.526	0.553	0.550	0.580	0.560
SVM		0.522	0.520	1.000	0.690
SVM, Hyperparameter tuned	0.500	0.534	0.530	0.970	0.680
Logistic Regression, with p hacking		-			

Recall, *True Positive Rate*
The proportion of all actual positives that were correctly classified as positives.

Comments
This metric is almost useless in trading.
Instead of the probability of positive prediction given actual positive value as indicated by recall, the reverse is more desirable - the probability of actual positive given positive prediction

Source
https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

Results

Summary

Model	AUC	Accuracy	Precision	Recall	F1 score
Logistic Regression	0.551	0.506	0.520	0.870	0.650
Logistic Regression, Hyperparameter tuned	0.551	0.522	0.520	0.900	0.660
Decision Tree	0.505	0.522	0.530	0.870	0.660
Decision Tree, Hyperparameter tuned	0.505	0.522	0.530	0.870	0.660
Random Forest		0.530	0.530	1.000	0.690
Random Forest, Hyperparameter tuned		0.530	0.530	1.000	0.690
KNN, BallPark	-	0.553	0.550	0.650	0.600
KNN, SKLearn	0.526	0.553	0.550	0.580	0.560
SVM		0.522	0.520	1.000	0.690
SVM, Hyperparameter tuned	0.500	0.534	0.530	0.970	0.680
Logistic Regression, with p hacking		-			

Precision

The proportion of all the models' positive classifications that are actually positive.

Source

https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

Results

Summary

Model	AUC	Accuracy	Precision	Recall	F1 score
Logistic Regression	0.551	0.506	0.520	0.870	0.650
Logistic Regression, Hyperparameter tuned	0.551	0.522	0.520	0.900	0.660
Decision Tree	0.505	0.522	0.530	0.870	0.660
Decision Tree, Hyperparameter tuned	0.505	0.522	0.530	0.870	0.660
Random Forest		0.530	0.530	1.000	0.690
Random Forest, Hyperparameter tuned		0.530	0.530	1.000	0.690
KNN, BallPark	-	0.553	0.550	0.650	0.600
KNN, SKLearn	0.526	0.553	0.550	0.580	0.560
SVM		0.522	0.520	1.000	0.690
SVM, Hyperparameter tuned	0.500	0.534	0.530	0.970	0.680
Logistic Regression, with p hacking		-			

F1 Score
The harmonic mean (a kind of average) of precision and recall.

Comments
This metric is normally used in datasets that are imbalanced.
Also, since traders are not usually interested in recall, we can ignore this metric

Source
https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

Conclusions

Model	AUC	Accuracy	Precision	Recall	F1 score
Logistic Regression	0.551	0.506	0.520	0.870	0.650
Logistic Regression, Hyperparameter tuned	0.551	0.522	0.520	0.900	0.660
Decision Tree	0.505	0.522	0.530	0.870	0.660
Decision Tree, Hyperparameter tuned	0.505	0.522	0.530	0.870	0.660
Random Forest		0.530	0.530	1.000	0.690
Random Forest, Hyperparameter tuned		0.530	0.530	1.000	0.690
KNN, BallPark	-	0.553	0.550	0.650	0.600
KNN, SKLearn	0.526	0.553	0.550	0.580	0.560
SVM		0.522	0.520	1.000	0.690
SVM, Hyperparameter tuned	0.500	0.534	0.530	0.970	0.680
Logistic Regression, with p hacking		-			

- The result is depressing

Conclusions

Model	AUC	Accuracy	Precision	Recall	F1 score
Logistic Regression	0.551	0.506	0.520	0.870	0.650
Logistic Regression, Hyperparameter tuned	0.551	0.522	0.520	0.900	0.660
Decision Tree	0.505	0.522	0.530	0.870	0.660
Decision Tree, Hyperparameter tuned	0.505	0.522	0.530	0.870	0.660
Random Forest		0.530	0.530	1.000	0.690
Random Forest, Hyperparameter tuned		0.530	0.530	1.000	0.690
KNN, BallPark	-	0.553	0.550	0.650	0.600
KNN, SKLearn	0.526	0.553	0.550	0.580	0.560
SVM		0.522	0.520	1.000	0.690
SVM, Hyperparameter tuned	0.500	0.534	0.530	0.970	0.680
Logistic Regression, with p hacking		-			

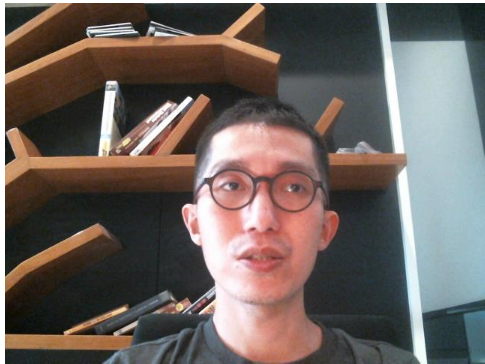
- The result is depressing
- Even when doing LogReg with p hacking, none of the predictors seem to be significant

Conclusions

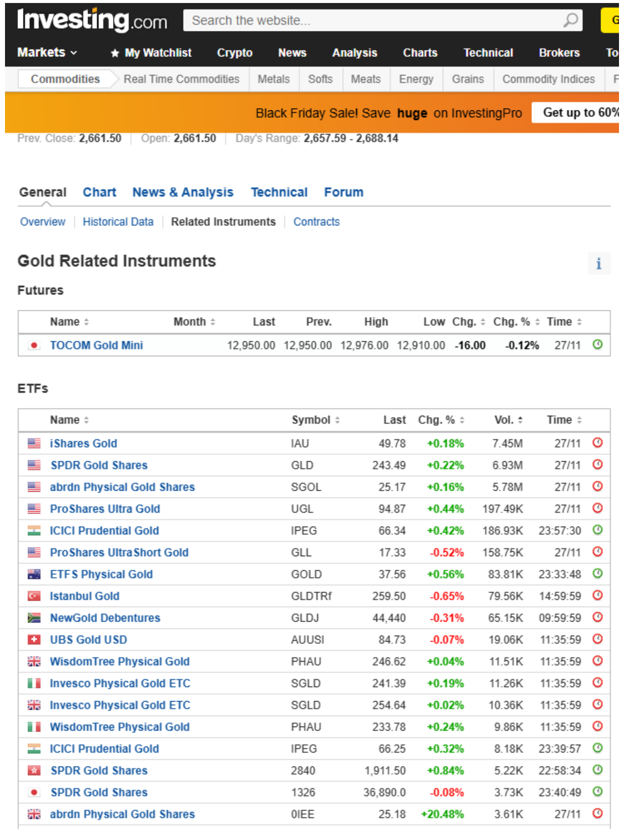
Model	AUC	Accuracy	Precision	Recall	F1 score
Logistic Regression	0.551	0.506	0.520	0.870	0.650
Logistic Regression, Hyperparameter tuned	0.551	0.522	0.520	0.900	0.660
Decision Tree	0.505	0.522	0.530	0.870	0.660
Decision Tree, Hyperparameter tuned	0.505	0.522	0.530	0.870	0.660
Random Forest		0.530	0.530	1.000	0.690
Random Forest, Hyperparameter tuned		0.530	0.530	1.000	0.690
KNN, BallPark	-	0.553	0.550	0.650	0.600
KNN, SKLearn	0.526	0.553	0.550	0.580	0.560
SVM		0.522	0.520	1.000	0.690
SVM, Hyperparameter tuned	0.500	0.534	0.530	0.970	0.680
Logistic Regression, with p hacking		-			

- The result is depressing

- Even when doing LogReg with p hacking, none of the predictors seem to be significant:
  - Not enough predictors
  - Wrong predictors



Rooms for Improvements



Investing.com Search the website...

Markets ★ My Watchlist Crypto News Analysis Charts Technical Brokers To

Commodities Real Time Commodities Metals Softs Meats Energy Grains Commodity Indices F

Black Friday Sale! Save **huge** on InvestingPro Get up to 60%

Prev. Close: 2,661.50 Open: 2,661.50 Day's Range: 2,657.59 - 2,688.14

General Chart News & Analysis Technical Forum

Overview Historical Data Related Instruments Contracts

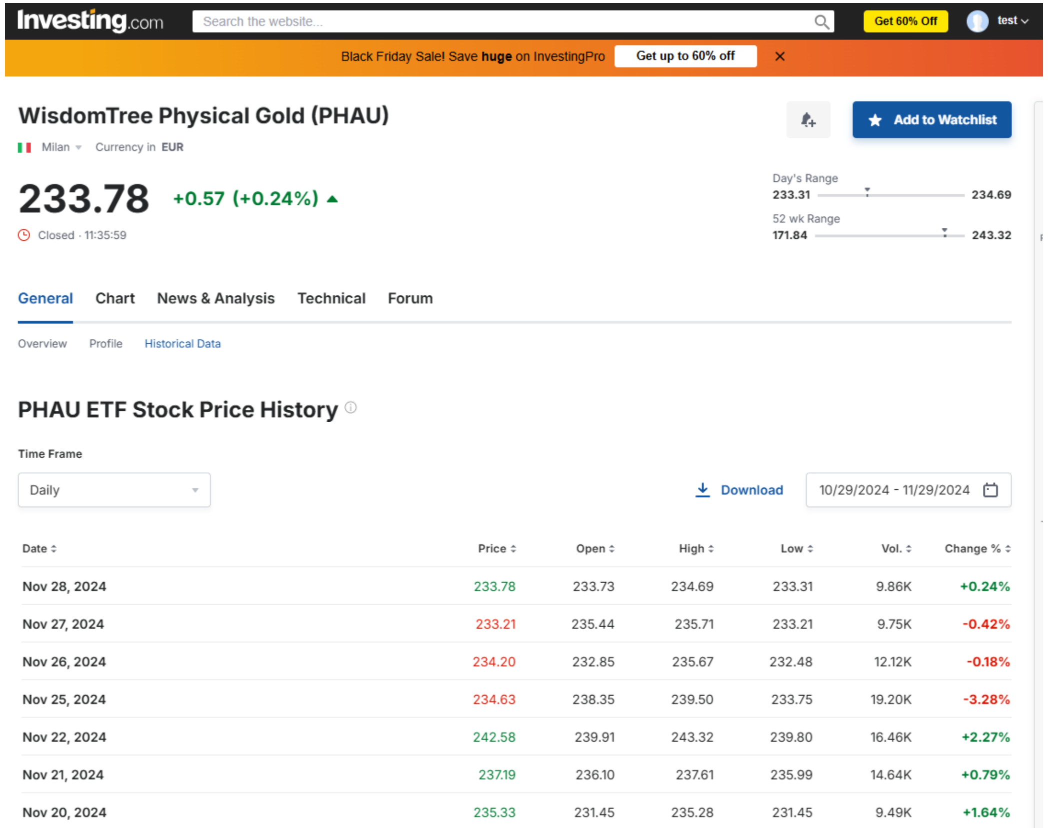
Gold Related Instruments

Futures

Name	Month	Last	Prev.	High	Low	Chg.	Chg. %	Time
TOCOM Gold Mini		12,950.00	12,950.00	12,976.00	12,910.00	-16.00	-0.12%	27/11

ETFs

Name	Symbol	Last	Chg. %	Vol.	Time
iShares Gold	IAU	49.78	+0.18%	7.45M	27/11
SPDR Gold Shares	GLD	243.49	+0.22%	6.93M	27/11
abrdn Physical Gold Shares	SGOL	25.17	+0.16%	5.78M	27/11
ProShares Ultra Gold	UGL	94.87	+0.44%	197.49K	27/11
ICICI Prudential Gold	IPEG	66.34	+0.42%	186.93K	23:57:30
ProShares UltraShort Gold	GLL	17.33	-0.52%	158.75K	27/11
ETFS Physical Gold	GOLD	37.56	+0.56%	83.81K	23:33:48
Istanbul Gold	GLDTRf	259.50	-0.65%	79.56K	14:59:59
NewGold Debentures	GLDJ	44,440	-0.31%	65.15K	09:59:59
UBS Gold USD	AUUSI	84.73	-0.07%	19.06K	11:35:59
WisdomTree Physical Gold	PHAU	246.62	+0.04%	11.51K	11:35:59
Invesco Physical Gold ETC	SGLD	241.39	+0.19%	11.26K	11:35:59
Invesco Physical Gold ETC	SGLD	254.64	+0.02%	10.36K	11:35:59
WisdomTree Physical Gold	PHAU	233.78	+0.24%	9.86K	11:35:59
ICICI Prudential Gold	IPEG	66.25	+0.32%	8.18K	23:39:57
SPDR Gold Shares	2840	1,911.50	+0.84%	5.22K	22:58:34
SPDR Gold Shares	1326	36,890.0	-0.08%	3.73K	23:40:49
abrdn Physical Gold Shares	0IEE	25.18	+20.48%	3.61K	27/11



Investing.com Search the website... Get 60% Off test

Black Friday Sale! Save **huge** on InvestingPro Get up to 60% off

WisdomTree Physical Gold (PHAU)

Add to Watchlist

Milan Currency in EUR

233.78 +0.57 (+0.24%)

Closed · 11:35:59

Day's Range 233.31 – 234.69

52 wk Range 171.84 – 243.32

General Chart News & Analysis Technical Forum

Overview Profile Historical Data

PHAU ETF Stock Price History

Time Frame

Daily

Download 10/29/2024 - 11/29/2024

Date	Price	Open	High	Low	Vol.	Change %
Nov 28, 2024	233.78	233.73	234.69	233.31	9.86K	+0.24%
Nov 27, 2024	233.21	235.44	235.71	233.21	9.75K	-0.42%
Nov 26, 2024	234.20	232.85	235.67	232.48	12.12K	-0.18%
Nov 25, 2024	234.63	238.35	239.50	233.75	19.20K	-3.28%
Nov 22, 2024	242.58	239.91	243.32	239.80	16.46K	+2.27%
Nov 21, 2024	237.19	236.10	237.61	235.99	14.64K	+0.79%
Nov 20, 2024	235.33	231.45	235.28	231.45	9.49K	+1.64%

Image Source:

https://www.investing.com/etfs/etfs---physical-gold?cid=47101

https://www.investing.com/etfs/etfs---physical-gold-historical-data?cid=47101

Thank you