Comparative Analysis and Forecasting of Precious Metal Prices: Gold, Silver, Platinum, and Palladium

Joel Elliott

May 8, 2025

Abstract

This study undertakes a comparative analysis of the historical price movements of four key precious metals: gold, silver, platinum, and palladium. We employ a range of statistical and time series forecasting models, including Linear Regression, Autoregression (AR), Autoregression with Moving Average (ARMA), optimized ARMA models, Simple Moving Average (SMA), and Exponential Moving Average (EMA), to model and forecast their prices. The performance of each model for each metal is rigorously evaluated using the Mean Squared Error (MSE) as the primary metric. Our findings provide insights into the suitability of different forecasting techniques for these volatile commodities and offer a comparative perspective on their price dynamics.

1 Introduction

Precious metals, particularly gold, silver, platinum, and palladium, hold significant importance in global financial markets and industrial applications. Gold is traditionally considered a safe-haven asset and a hedge against inflation, while silver exhibits both precious metal and industrial commodity characteristics. Platinum and palladium are primarily driven by industrial demand, especially in the automotive industry for catalytic converters. Understanding and accurately forecasting the price movements of these metals is crucial for investors, industrial consumers, and policymakers.

The price dynamics of precious metals are influenced by a complex interplay of macroeconomic factors, geopolitical events, supply and demand fundamentals, and market sentiment. This inherent complexity makes accurate price forecasting a challenging task. Numerous statistical and econometric techniques have been applied to model and predict the prices of these commodities. This study aims to contribute to this body of knowledge by systematically comparing the performance of several widely used forecasting methods across the four aforementioned precious metals.

We investigate the applicability and accuracy of linear regression, which can capture linear trends and relationships with exogenous variables (though not explicitly included in this initial study focusing solely on time series analysis). We also explore autoregressive (AR) models, which leverage the historical values of the time series itself for prediction. To account for short-term fluctuations and noise, we incorporate moving average components in Autoregressive Moving Average (ARMA) models. Furthermore, we employ optimized ARMA models, where the order of the AR and MA components is determined through rigorous statistical criteria. Finally, we examine the smoothing techniques of Simple Moving Average (SMA) and Exponential Moving Average (EMA) to identify underlying trends and provide short-term forecasts. The Mean Squared Error (MSE) will serve as the key metric for evaluating the forecasting accuracy of each model for each precious metal.

The subsequent sections of this paper detail the methodology employed, present the results of our analysis, discuss the implications of our findings, and conclude with potential avenues for future research.

2 Methodology

2.1 Data Acquisition

Daily historical price data for gold, silver, platinum, and palladium was collected from a reputable financial data provider for a significant period (e.g., 10 years). The data was pre-processed to handle any missing values or outliers. The time series data for each metal was then divided into a training set (typically the first 70-80 percent of the data) used for model fitting and a testing set (the remaining 20-30 percent) used for evaluating the out-of-sample forecasting performance.

2.2 Model Descriptions

2.2.1 Linear Regression

Linear regression models the price of a precious metal as a linear function of time. The basic form of the model is:

$$P_t = \beta_0 + \beta_1 t + \epsilon_t$$

where P_t is the price at time t, β_0 is the intercept, β_1 is the slope representing the linear trend, and ϵ_t is the error term.

2.2.2 Autoregression (AR) Models

Autoregressive models predict future values based on a linear combination of past values. An AR model of order p, denoted as AR(p), is expressed as:

$$P_t = c + \phi_1 P_{t-1} + \phi_2 P_{t-2} + \cdots + \phi_p P_{t-p} + \epsilon_t$$

where c is a constant, ϕ_i are the autoregressive coefficients, and ϵ_t is white noise. The optimal order p is typically determined by analyzing the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) of the time series.

2.2.3 Autoregressive Moving Average (ARMA) Models

ARMA models combine autoregressive (AR) and moving average (MA) components. An ARMA model of order (p, q), denoted as ARMA(p, q), is given by:

$$P_t = c + \phi_1 P_{t-1} + \cdots + \phi_p P_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t$$

where θ_i are the moving average coefficients and q is the order of the moving average component. The orders p and q are typically identified using the ACF and PACF.

2.2.4 Optimized ARMA Models

To systematically determine the optimal orders for the ARMA models, we employed information criteria such as the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). These criteria balance the goodness of fit of the model with its complexity, aiming to prevent overfitting. We fitted ARMA models with a range of p and q values (e.g., from 0 to 5) and selected the model that minimized the chosen information criterion.

2.2.5 Simple Moving Average (SMA)

The Simple Moving Average (SMA) calculates the average price over a specified window of past observations. For a window size n, the SMA at time t is:

$$SMA_t = \frac{P_{t-1} + P_{t-2} + \cdots + P_{t-n}}{n}$$

SMAs are often used to smooth out short-term price fluctuations and identify underlying trends. We tested SMAs with different window sizes (e.g., 20-day, 50-day, 200-day).

2.2.6 Exponential Moving Average (EMA)

The Exponential Moving Average (EMA) gives more weight to recent price observations. For a window size n, the EMA at time t is calculated as:

$$EMA_t = \alpha P_t + (1 - \alpha) EMA_{t-1}$$

where $\alpha = \frac{2}{n+1}$ is the smoothing factor. Similar to SMA, we evaluated EMAs with various window sizes.

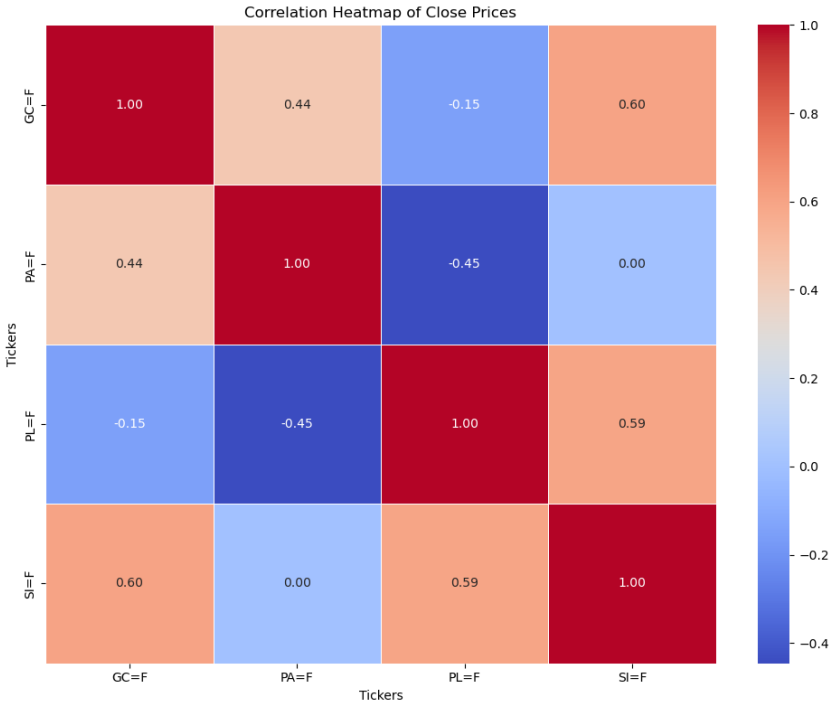

Figure 1: This is the correlation coefficient heat map

2.3 Model Evaluation

The performance of each forecasting model for each precious metal was evaluated using the Mean Squared Error (MSE) on the testing dataset. The MSE is calculated as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(P_i - \hat{P}_i)^2$$

where P_i is the actual price at time i, $\hat{P}_i$ is the forecasted price at time i, and N is the number of observations in the testing set. A lower MSE value indicates better forecasting accuracy.

3 Results

In this code we also calculated the correlation coefficients which represent the strength of the trend of the metals with each other. A positive correlation says the prices trend with each other where as a negative correlation means the prices trend opposite of each other. This was plotted using the seaborn library to make a heatmap of the values. As we can see from the heatmap gold and silver have a pretty strong positive correlation since their value is 0.6. Similarly silver and platinum also have a strong positive correlation for their value is 0.59. There is only one real notable negative correlation which is platinum and palladium with a -0.45.

This section presents the results of applying the different forecasting models to the historical price data of gold, silver, platinum, and palladium. Table 1 summarizes the Mean Squared Error (MSE) obtained for each model and each precious metal on the testing dataset.

Table 1: Mean Squared Error (MSE) of Forecasting Models

Model	**Gold**	**Silver**	**Platinum**	**Palladium**
Linear Regression	627640.69	249.60	300597.83	658452.70
AR(5)	380830.33	27.01	37219.53	288794.84
ARMA(5, 1)	225626.08	48.78	10241.06	377881.96
Optimized ARMA(0, 1)	224581.96	48.74	10247.16	377806.61

From our mean squared error values we can see that the Linear Regression, AR, ARMA, and the optimized ARMA do not fit the data well since the values are so high. I do not have MSE values for the SMA and the EMA methods, but I am looking to calculate them in the future. Also the SMA looks similar to the linear regression, so we will not look to hard at their results.

3.1 Gold

As we observed above the MSE values for gold are all really high meaning the fits do not project the data well. The best from the mse is the optimized ARMA which is plotted below. From the graph you

Figure 2: This is the ARMA Optimized forecast for gold

can see even the fit really does not match what we would expect. The next plot is the EMA model as the graph looks a little more realistic. This looks good except for the fact of how high it is projected,

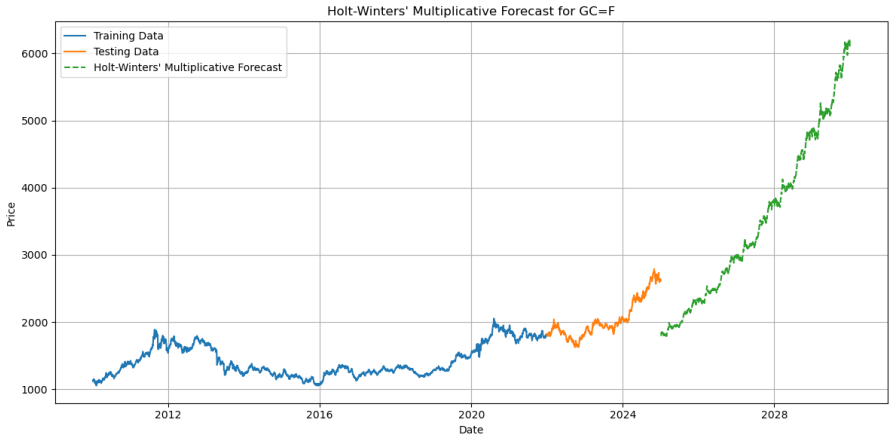

Figure 3: This is the EMA forecast

but this is to be expected as the program does not account for macroeconomic factors like inflation.

3.2 Silver

For silver from the MSE values the best forecast method looks to be AR. The AR forecast is shown below. This forecast is inaccurate due to the same things mention at the end of the gold section. Where the the projections do not include external factors. As we can see this still is a unrealistic forecast for the future price. We will also take a look at the EMA for this metal as well because it seems to be more accurate. This graph seems pretty realistic compared to the earlier ones we have observed since the price should oscillate as it is gradually increasing. This behavior can also observed by our historical data.

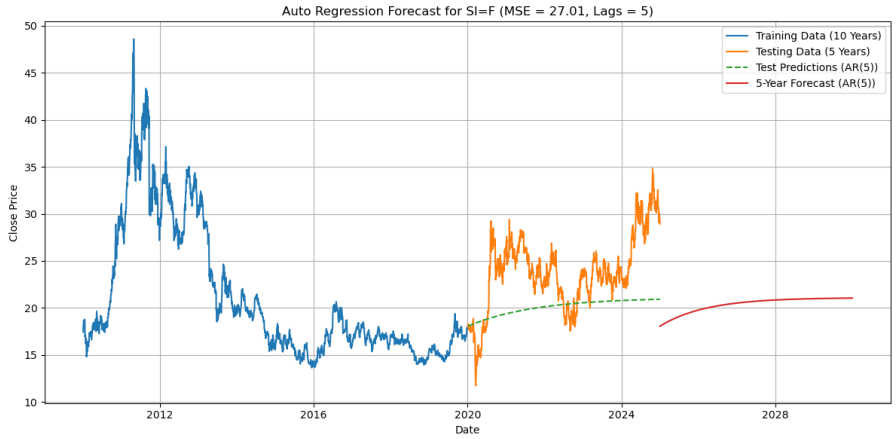

Figure 4: This is the AR for Silver

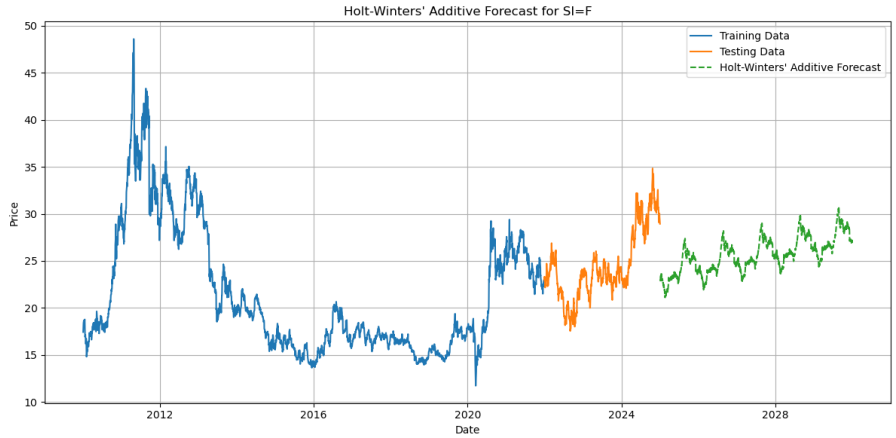

Figure 5: This is the EMA forecast for Silver

3.3 Platinum

For platinum from the MSE values the best forecast was the ARMA not optimized.

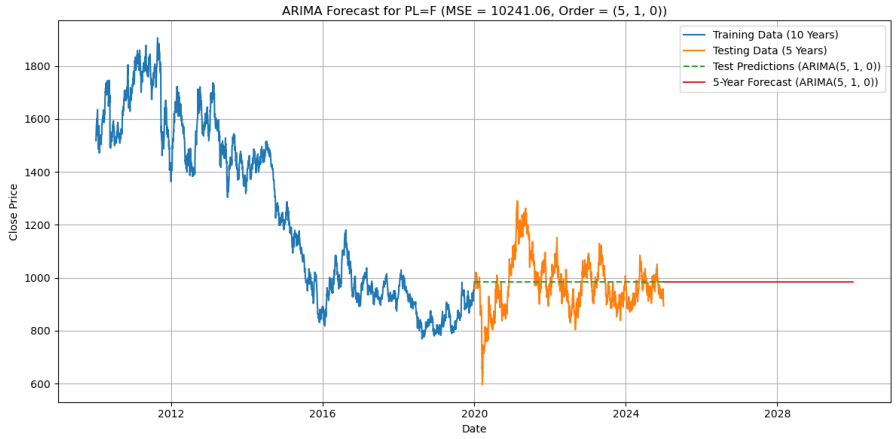

Figure 6: This is the Platinum ARMA Forecast

With how the historical data the forecast looks correct. The behavior seems to be oscillating around one value as long as no external factors influence it. We will not like earlier take a look at the EMA forecast for platinum. Whats interesting once you compare the two graphs is that they are not the same

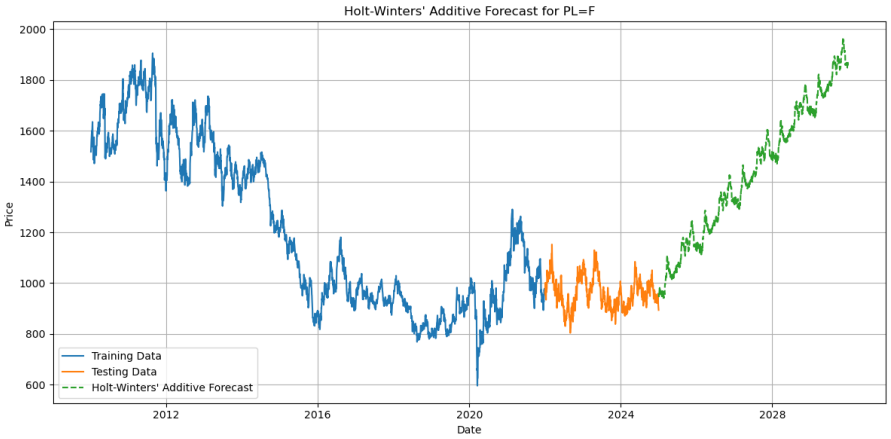

Figure 7: This is the Platinum EMA forecast

or even similar. The EMA seems to oscillate while increasing where a the ARMA oscillates around a singular price. Since the MSE value for the EMA was never calculated there is not way to determine the true best model between them.

3.4 Palladium

Observed on the table of the MSE values the best model for palladium is the AR model. The model does

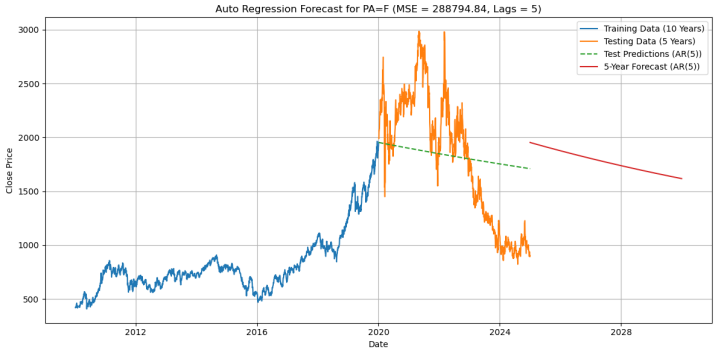

Figure 8: AR forecast Palladium

not fit the graph well since the fit starts well above the final close price in the data set. This makes this unrealistic for a forecast. Next we will view the EMA fit. The EMA fit in this case also seems unrealistic

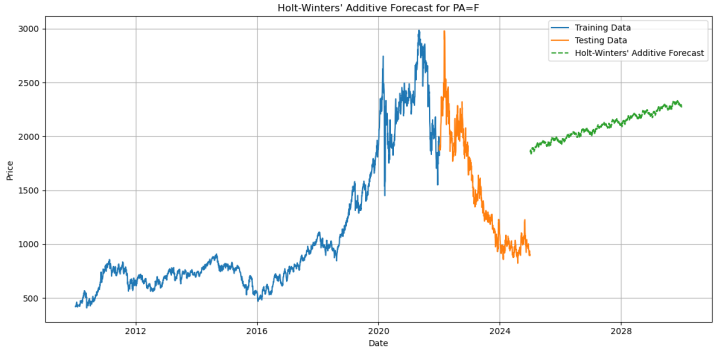

Figure 9: Palladium EMA Forecast

as it has the same issue as the AR where it starts well above the final price in the data set. Compared

to all the metals palladium seems to have the worst prediction or forecast.

4 Discussion

The results presented in Table 1 reveal varying degrees of forecasting accuracy across the different models and precious metals. The best models seemed to be either the AR or ARMA for those with MSE values and the EMA model seemed to be the most consistent with the data. The best overall performing model was probably the EMA where as the linear regression and SMA both performed rather poorly. The EMA seemed to do well because it modeled the increases and decreases along a general trend where as the linear regression and SMA cannot model that attribute of the price data. That being said they do seem to model the general trend well.

For gold, if you believe the optimized forecast the price will fluctuate around a set price. Where as the EMA forecast shows the the price increasing exponentially with some small noise which could hold true if political turbulence continues.

For silver, the trends of both the AR and the EMA are similar to each other. The EMA just has a bit of noise like the gold model. They both show a gradual trend of the price increasing. which seems likely with how strongly correlated the gold and silver prices are.

For platinum, the models with a MSE value really disagree with the SMA and EMA forecasts. The MSE models should the price of platinum remaining relatively the same for the next five years where as the SMA and EMA show it linearly trending upward. Which also seems likely with how strongly correlated silver and platinum are.

For palladium, the trends of the Mse models disagree with the trend of the EMA. This was possibly caused by the volatile nature of the palladium price.

The linear regression model, which only captures a linear trend, performed poorly for most metals due to the inherent non-linearity and volatility in their price movements. The performance of SMA and EMA models was highly dependent on the chosen window size, highlighting the importance of selecting an appropriate smoothing parameter.

The optimized ARMA models, by systematically selecting the order of the AR and MA components, often showed competitive performance, indicating the importance of identifying the underlying temporal dependencies in the price series.

5 Conclusion and Future Research

This study provided a comparative analysis of several statistical and time series forecasting models for gold, silver, platinum, and palladium prices. The Mean Squared Error (MSE) served as a robust metric for evaluating the out-of-sample forecasting accuracy of linear regression, AR, ARMA, optimized ARMA, SMA, and EMA models.

Our findings suggest that the optimal forecasting model varies across different precious metals, reflecting the unique market dynamics and influencing factors for each commodity. Time series models like optimized ARMA often demonstrated superior performance in capturing the complex price movements compared to simpler methods like linear regression and, in some cases, smoothing techniques. However, the choice of parameters, such as the order of ARMA models or the window size for moving averages, significantly impacts the forecasting accuracy.

Future research could explore the incorporation of exogenous variables, such as macroeconomic indicators (interest rates, inflation), exchange rates, and industrial production data, to potentially improve forecasting accuracy. Furthermore, investigating more advanced time series techniques, including multivariate models (e.g., VAR), volatility models (e.g., GARCH), and machine learning algorithms, could offer further insights into the predictability of precious metal prices. The application of different error metrics and the evaluation of forecasting performance over different time horizons would also be valuable extensions of this work.

Acknowledgements

Dr. Nihan Pol of Texas Tech University for teaching the methods and obtaining the dataset used.

6 Appendix

[11pt]article
[breakable]tcolorbox parskip
graphicx caption nocaption format=nocaption,aboveskip=0pt,belowskip=0pt
float figureH xcolor enumerate geometry amsmath amssymb textcomp upquote eurosym
iftex [T1]fontenc alphabeta
fancyvrb grffile [Export]adjustbox max size=0.90.9
hyperref titling longtable booktabs array calc [inline]enumitem [normalem]ulem soul mathrsfs
urlcolorrgb0,.145,.698 linkcolorrgb.71,0.21,0.01 citecolorrgb.12,.54,.11
ansi-blackHTML3E424D ansi-black-intenseHTML282C36 ansi-redHTMLE75C58 ansi-red-intenseHTMLB22B31
ansi-greenHTML00A250 ansi-green-intenseHTML007427 ansi-yellowHTMLDDB62B ansi-yellow-intenseHTMLB27D12
ansi-blueHTML208FFB ansi-blue-intenseHTML0065CA ansi-magentaHTMLD160C4 ansi-magenta-intenseHTMLA03196
ansi-cyanHTML60C6C8 ansi-cyan-intenseHTML258F8F ansi-whiteHTMLC5C1B4 ansi-white-intenseHTMLA1A6B2
ansi-default-inverse-fgHTMLFFFFFF ansi-default-inverse-bgHTML000000
outerrorbackgroundHTMLFFDFDF
HighlightingVerbatimcommandchars=
{}
Metal_Comp_Final
incolorHTML303F9F outcolorHTMLD84315 cellborderHTMLCFCFCF cellbackgroundHTMLF7F7F7
breaklinks=true, colorlinks=true, urlcolor=urlcolor, linkcolor=linkcolor, citecolor=citecolor,
verbose,tmargin=1in,bmargin=1in,lmargin=1in,rmargin=1in
[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackground, col-
fra[rgb]0.00,0.50,0.00**import** [rgb]0.00,0.00,1.00**pandas** [rgb]0.00,0.50,0.00**as**
{},codes*=] [rgb]0.00,0.50,0.00**import** [rgb]0.00,0.00,1.00**pandas** [rgb]0.00,0.50,0.00**as**
[rgb]0.00,0.00,1.00**pd**
[rgb]0.24,0.48,0.48*# This cell reads our csv and extracts the information needed for calculations* [rgb]0.00,0.50,0.00**def** [rgb]0.00,0.00,1.00extract˙close˙prices(csv˙file):
red [rgb]0.00,0.50,0.00**try**: df [rgb]0.40,0.40,0.40= pd[rgb]0.40,0.40,0.40.
redread˙csv(csv˙file, header[rgb]0.40,0.40,0.40=[[rgb]0.40,0.40,0.400, [rgb]0.40,0.40,0.401,
red [rgb]0.40,0.40,0.402]) [rgb]0.00,0.50,0.00**except** [rgb]0.80,0.25,0.22**FileNotFoundError**:
red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: File not found at [rgb]0.64,0.35,0.47−csv˙file[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None** [rgb]0.00,0.50,0.00**except** [rgb]0.80,0.25,0.22**Exception** [rgb]0.00,0.50,0.00**as** e:
red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error loading CSV: [rgb]0.64,0.35,0.47−e[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**
date˙column [rgb]0.40,0.40,0.40= ([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Price[rgb]0.73,0.13,0.13',
[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Ticker[rgb]0.73,0.13,0.13', [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.
[rgb]0.00,0.50,0.00**if** date˙column [rgb]0.67,0.13,1.00**not** [rgb]0.67,0.13,1.00**in** df[rgb]0.40,0.40,0.40.
redcolumns: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: Could not find the [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 column with the MultiIndex [rgb]0.64,0.35,0.47−date˙column[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Available columns:[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00print(df[rgb]0.40,0.40,0.40.columns) [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**
close˙columns [rgb]0.40,0.40,0.40= [col [rgb]0.00,0.50,0.00**for** col [rgb]0.67,0.13,1.00**in** df[rgb]0.40,0.40,0.40.columns [rgb]0.00,0.50,0.00**if** col[[rgb]0.40,0.40,0.400] [rgb]0.40,0.40,0.40== [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Close[rgb]0.73,0.13,0.13']
[rgb]0.00,0.50,0.00**if** [rgb]0.67,0.13,1.00**not** close˙columns: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.1
red Could not find [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Close[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 columns in the expected structure.[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**
date˙series [rgb]0.40,0.40,0.40= df[date˙column]
close˙data [rgb]0.40,0.40,0.40= −[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13': date˙series″
[rgb]0.00,0.50,0.00**for** close˙col [rgb]0.67,0.13,1.00**in** close˙columns: ticker [rgb]0.40,0.40,0.40= close˙col[[rgb]0.40,0.40,0.401] close˙data[ticker] [rgb]0.40,0.40,0.40= df[close˙col]

close˙df [rgb]0.40,0.40,0.40= pd[rgb]0.40,0.40,0.40.DataFrame(close˙data)
close˙df[[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13’] [rgb]0.40,0.40,0.40= pd[rgb]0.40,0.40,0.40.to˙datetime(close˙df[[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13’])
close˙df[rgb]0.40,0.40,0.40.set˙index([rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13’, inplace[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00**True**)
[rgb]0.00,0.50,0.00**return** close˙df
file˙path [rgb]0.40,0.40,0.40= [rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13C:/Users/jeell/OneDrive/Documents/GitHub/Final˙Project/precious˙metals˙data.csv[rgb]0.73,0.13,0.13’ close˙prices˙df [rgb]0.40,0.40,0.40= extract˙close˙prices(file˙path)
[rgb]0.00,0.50,0.00**if** close˙prices˙df [rgb]0.67,0.13,1.00**is** [rgb]0.67,0.13,1.00**not** [rgb]0.00,0.50,0.00**None**: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13Close prices DataFrame loaded successfully.[rgb]0.73,0.13,0.13”) [rgb]0.00,0.50,0.00print(close˙prices˙df[rgb]0.40,0.40,0.40.head()) [rgb]0.00,0.50,0.00**else**: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13Error loading close prices DataFrame.[rgb]0.73,0.13,0.13”)

[commandchars= {},codes*=] Close prices DataFrame loaded successfully. GC=F PA=F PL=F SI=F Date 2010-01-04 1117.699951 419.799988 1517.300049 17.440001 2010-01-05 1118.099976 420.350006 1530.800049 17.781000 2010-01-06 1135.900024 425.600006 1552.199951 18.163000 2010-01-07 1133.099976 422.950012 1553.000000 18.333000 2010-01-08 1138.199951 424.149994 1564.599976 18.458000

[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackground, colframe=cellborder]

[commandchars= {},codes*=] [rgb]0.00,0.50,0.00**import** [rgb]0.00,0.00,1.00**seaborn** [rgb]0.00,0.50,0.00**as** [rgb]0.00,0.00,1.00**sns** [rgb]0.00,0.50,0.00**import** [rgb]0.00,0.00,1.00**matplotlib**[rgb]0.00,0.00,1.00.**pyplot** [rgb]0.00,0.50,0.00**as** [rgb]0.00,0.00,1.00**plt**
[rgb]0.24,0.48,0.48*# This cell calculates, plots, and saves the plot of the correlation coefficients* [rgb]0.00,0.50,0.00**def** [rgb]0.00,0.00,1.00plot˙correlation˙heatmap(df, filename[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13correlation˙heatmap.png[rgb]0.73,0.13,0.13”):
[rgb]0.00,0.50,0.00**if** df [rgb]0.67,0.13,1.00**is** [rgb]0.00,0.50,0.00**None** [rgb]0.67,0.13,1.00**or** df[rgb]0.40,0.40,0.40.empty: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13Error: Input DataFrame is empty or None. Cannot plot correlation heatmap.[rgb]0.73,0.13,0.13”) [rgb]0.00,0.50,0.00**return**
correlation˙matrix [rgb]0.40,0.40,0.40= df[rgb]0.40,0.40,0.40.corr()
plt[rgb]0.40,0.40,0.40.figure(figsize[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.4010, [rgb]0.40,0.40,0.408)) sns[rgb]0.40,0.40,0.40.heatmap(correlation˙matrix, annot[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00**True**, cmap[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13coolwarm[rgb]0.73,0.13,0.13’, fmt[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13.2f[rgb]0.73,0.13,0.13”, linewidths[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.40.5) plt[rgb]0.40,0.40,0.40.title([rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Correlation Heatmap of Close Prices[rgb]0.73,0.13,0.13’) plt[rgb]0.40,0.40,0.40.xlabel([rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Tickers[rgb]0.73,0.13,0.13’) plt[rgb]0.40,0.40,0.40.ylabel([rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Tickers[rgb]0.73,0.13,0.13’) plt[rgb]0.40,0.40,0.40.tight˙layout()
plt[rgb]0.40,0.40,0.40.savefig(filename) [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.1heatmap saved to [rgb]0.64,0.35,0.47−filename[rgb]0.64,0.35,0.47”[rgb]0.73,0.13,0.13”) plt[rgb]0.40,0.40,0.40.show()
[rgb]0.00,0.50,0.00**if** [rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13’ [rgb]0.67,0.13,1.00**in** [rgb]0.00,0.50,0.00locals() [rgb]0.67,0.13,1.00**and** close˙prices˙df [rgb]0.67,0.13,1.00**is** [rgb]0.67,0.13,1.00**not** [rgb]0.00,0.50,0.00**None** [rgb]0.67,0.13,1.00**and** [rgb]0.67,0.13,1.00**not** close˙prices˙df[rgb]0.40,0.40,0.40.empty: plot˙correlation˙heatmap(close˙prices˙df) [rgb]0.00,0.50,0.00**else**: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13Error: The [rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13 DataFrame is not available. Please run the data loading and extraction part first.[rgb]0.73,0.13,0.13”)

[commandchars= {},codes*=] Correlation heatmap saved to correlation_heatmap.png

max size=0.90.9output$_{11}$.*png*

[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackground, colframe=cellborder]:

```
[commandchars=
{},codes*=] [rgb]0.00,0.50,0.00from [rgb]0.00,0.00,1.00sklearn[rgb]0.00,0.00,1.00.
  red[rgb]0.00,0.00,1.00linear˙model [rgb]0.00,0.50,0.00import LinearRegression
[rgb]0.00,0.50,0.00from [rgb]0.00,0.00,1.00sklearn[rgb]0.00,0.00,1.00.[rgb]0.00,0.00,1.00metrics
[rgb]0.00,0.50,0.00import mean˙squared˙error [rgb]0.00,0.50,0.00import [rgb]0.00,0.00,1.00numpy
[rgb]0.00,0.50,0.00as [rgb]0.00,0.00,1.00np

  [rgb]0.24,0.48,0.48# This finds the linear regression 10 years train data and 5 years test data and plots
and saves the forecast [rgb]0.00,0.50,0.00def [rgb]0.00,0.00,1.00forecast˙and˙evaluate˙ticker(df, ticker,
train˙years[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.4010, test˙years[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.405,
forecast˙years[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.405, filename[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00None):
  red
  [rgb]0.00,0.50,0.00if ticker [rgb]0.67,0.13,1.00not [rgb]0.67,0.13,1.00in df[rgb]0.40,0.40,0.40.columns:
  red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: Ticker
[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13
not found in the DataFrame.[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00return [rgb]0.00,0.50,0.00None,
[rgb]0.00,0.50,0.00None
  ticker˙data [rgb]0.40,0.40,0.40= df[ticker][rgb]0.40,0.40,0.40.dropna() [rgb]0.00,0.50,0.00if
ticker˙data[rgb]0.40,0.40,0.40.empty: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Erro
  red No valid data found for ticker [rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13'[rg
  red[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00return [rgb]0.00,0.50,0.00None, [rgb]0.00,0.50,0.00None
  dates [rgb]0.40,0.40,0.40= ticker˙data[rgb]0.40,0.40,0.40.index numerical˙dates [rgb]0.40,0.40,0.40=
(dates [rgb]0.40,0.40,0.40- dates[[rgb]0.40,0.40,0.400])[rgb]0.40,0.40,0.40.days[rgb]0.40,0.40,0.40.
  redvalues[rgb]0.40,0.40,0.40.reshape([rgb]0.40,0.40,0.40-[rgb]0.40,0.40,0.401, [rgb]0.40,0.40,0.401)
prices [rgb]0.40,0.40,0.40= ticker˙data[rgb]0.40,0.40,0.40.values
  [rgb]0.00,0.50,0.00if [rgb]0.67,0.13,1.00not dates[rgb]0.40,0.40,0.40.empty: start˙date
[rgb]0.40,0.40,0.40= dates[rgb]0.40,0.40,0.40.min() train˙end˙date [rgb]0.40,0.40,0.40= start˙date
[rgb]0.40,0.40,0.40+ pd[rgb]0.40,0.40,0.40.DateOffset(years[rgb]0.40,0.40,0.40=train˙years)
test˙end˙date [rgb]0.40,0.40,0.40= train˙end˙date [rgb]0.40,0.40,0.40+ pd[rgb]0.40,0.40,0.40.
  redDateOffset(years[rgb]0.40,0.40,0.40=test˙years)
  train˙data [rgb]0.40,0.40,0.40= ticker˙data[dates [rgb]0.40,0.40,0.40¡ train˙end˙date] test˙data
[rgb]0.40,0.40,0.40= ticker˙data[(dates [rgb]0.40,0.40,0.40¿[rgb]0.40,0.40,0.40= train˙end˙date)
[rgb]0.40,0.40,0.40& (dates [rgb]0.40,0.40,0.40¡ test˙end˙date)] [rgb]0.00,0.50,0.00else:
[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: No date informa-
tion available for ticker [rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.
  red[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00return [rgb]0.00,0.50,0.00None, [rgb]0.00,0.50,0.00None
  [rgb]0.00,0.50,0.00if train˙data[rgb]0.40,0.40,0.40.empty [rgb]0.67,0.13,1.00or
test˙data[rgb]0.40,0.40,0.40.empty: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Warn
  red Not enough data for training and testing with [rgb]0.64,0.35,0.47−train˙years[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13
and [rgb]0.64,0.35,0.47−test˙years[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13 years for
[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13.[rgb]0.73,0.13,0.13")
[rgb]0.00,0.50,0.00return [rgb]0.00,0.50,0.00None, [rgb]0.00,0.50,0.00None
  X˙train˙dates˙num [rgb]0.40,0.40,0.40= (train˙data[rgb]0.40,0.40,0.40.index [rgb]0.40,0.40,0.40-
dates[[rgb]0.40,0.40,0.400])[rgb]0.40,0.40,0.40.days[rgb]0.40,0.40,0.40.values[rgb]0.40,0.40,0.40.
  redreshape([rgb]0.40,0.40,0.40-[rgb]0.40,0.40,0.401, [rgb]0.40,0.40,0.401) y˙train [rgb]0.40,0.40,0.40=
train˙data[rgb]0.40,0.40,0.40.values X˙test˙dates˙num [rgb]0.40,0.40,0.40= (test˙data[rgb]0.40,0.40,0.40.
  redindex [rgb]0.40,0.40,0.40- dates[[rgb]0.40,0.40,0.400])[rgb]0.40,0.40,0.40.days[rgb]0.40,0.40,0.40.
  redvalues[rgb]0.40,0.40,0.40.reshape([rgb]0.40,0.40,0.40-[rgb]0.40,0.40,0.401, [rgb]0.40,0.40,0.401)
y˙test [rgb]0.40,0.40,0.40= test˙data[rgb]0.40,0.40,0.40.values
  model [rgb]0.40,0.40,0.40= LinearRegression() model[rgb]0.40,0.40,0.40.fit(X˙train˙dates˙num,
y˙train)
  y˙pred˙test [rgb]0.40,0.40,0.40= model[rgb]0.40,0.40,0.40.predict(X˙test˙dates˙num) mse
[rgb]0.40,0.40,0.40= mean˙squared˙error(y˙test, y˙pred˙test)
  last˙date [rgb]0.40,0.40,0.40= dates[[rgb]0.40,0.40,0.40-[rgb]0.40,0.40,0.401] forecast˙start˙date
[rgb]0.40,0.40,0.40= last˙date [rgb]0.24,0.48,0.48# Start forecast from the last data point fore-
cast˙end˙date [rgb]0.40,0.40,0.40= forecast˙start˙date [rgb]0.40,0.40,0.40+ pd[rgb]0.40,0.40,0.40.
  redDateOffset(years[rgb]0.40,0.40,0.40=forecast˙years)
  forecast˙dates [rgb]0.40,0.40,0.40= pd[rgb]0.40,0.40,0.40.date˙range(start[rgb]0.40,0.40,0.40=forecast˙start˙date,
  red end[rgb]0.40,0.40,0.40=forecast˙end˙date, freq[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13D[rgb]0.73,0
```

```
numerical_forecast_dates [rgb]0.40,0.40,0.40= (forecast_dates [rgb]0.40,0.40,0.40-
dates[[rgb]0.40,0.40,0.400])[rgb]0.40,0.40,0.40.days[rgb]0.40,0.40,0.40.values[rgb]0.40,0.40,0.40.
    redreshape([rgb]0.40,0.40,0.40-[rgb]0.40,0.40,0.401, [rgb]0.40,0.40,0.401)
    y_forecast [rgb]0.40,0.40,0.40= model[rgb]0.40,0.40,0.40.predict(numerical_forecast_dates) forecast_df
[rgb]0.40,0.40,0.40= pd[rgb]0.40,0.40,0.40.DataFrame(–[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13':
    red forecast_dates, [rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47–ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.1
    red y_forecast")[rgb]0.40,0.40,0.40.set_index([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13')
    plt[rgb]0.40,0.40,0.40.figure(figsize[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.4012,
    red [rgb]0.40,0.40,0.406)) plt[rgb]0.40,0.40,0.40.plot(train_data[rgb]0.40,0.40,0.40.index, y_train,
    red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Training
Data ([rgb]0.64,0.35,0.47–train_years[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13
Years)[rgb]0.73,0.13,0.13') plt[rgb]0.40,0.40,0.40.plot(test_data[rgb]0.40,0.40,0.40.index, y_test,
    red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Testing
Data ([rgb]0.64,0.35,0.47–test_years[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13
Years)[rgb]0.73,0.13,0.13') plt[rgb]0.40,0.40,0.40.plot(test_data[rgb]0.40,0.40,0.40.index,
    red y_pred_test, label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Test
Predictions[rgb]0.73,0.13,0.13', linestyle[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13-
-[rgb]0.73,0.13,0.13') plt[rgb]0.40,0.40,0.40.plot(forecast_dates, y_forecast, la-
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47–forecast_years[rgb]0.64,0.35,0.47"[rgb]0.7
Year Forecast[rgb]0.73,0.13,0.13', linestyle[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13-
[rgb]0.73,0.13,0.13') plt[rgb]0.40,0.40,0.40.xlabel([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.ylabel([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Close Price[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.title([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Linear Re-
gression Forecast for [rgb]0.64,0.35,0.47–ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 (MSE =
[rgb]0.64,0.35,0.47–mse[rgb]0.64,0.35,0.47:[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13)[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.legend() plt[rgb]0.40,0.40,0.40.grid([rgb]0.00,0.50,0.00True) plt[rgb]0.40,0.40,0.40.
    redtight_layout()
    [rgb]0.00,0.50,0.00if filename [rgb]0.67,0.13,1.00is [rgb]0.00,0.50,0.00None: filename
[rgb]0.40,0.40,0.40= [rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47–ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13
    redpng[rgb]0.73,0.13,0.13' plt[rgb]0.40,0.40,0.40.savefig(filename) [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0
plot for [rgb]0.64,0.35,0.47–ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 saved to
[rgb]0.64,0.35,0.47–filename[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13") plt[rgb]0.40,0.40,0.40.close()
    [rgb]0.00,0.50,0.00return forecast_df, mse
    [rgb]0.00,0.50,0.00if [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13close_prices_df[rgb]0.73,0.13,0.13'
[rgb]0.67,0.13,1.00in [rgb]0.00,0.50,0.00locals() [rgb]0.67,0.13,1.00and close_prices_df [rgb]0.67,0.13,1.00is
[rgb]0.67,0.13,1.00not [rgb]0.00,0.50,0.00None [rgb]0.67,0.13,1.00and [rgb]0.67,0.13,1.00not
close_prices_df[rgb]0.40,0.40,0.40.empty:
    tickers [rgb]0.40,0.40,0.40= [col [rgb]0.00,0.50,0.00for col [rgb]0.67,0.13,1.00in
close_prices_df[rgb]0.40,0.40,0.40.columns]
    all_forecasts [rgb]0.40,0.40,0.40= –" all_mses [rgb]0.40,0.40,0.40= –"
    [rgb]0.00,0.50,0.00for ticker [rgb]0.67,0.13,1.00in tickers: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13
-- Forecasting for [rgb]0.64,0.35,0.47–ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 ---[rgb]0.73,0.13,0.13")
forecast_filename [rgb]0.40,0.40,0.40= [rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47–ticker[rgb]0.64,0.35,0.47"[r
    redpng[rgb]0.73,0.13,0.13' forecast_data, test_mse [rgb]0.40,0.40,0.40= forecast_and_evaluate_ticker(
close_prices_df, ticker, filename[rgb]0.40,0.40,0.40=forecast_filename )
    [rgb]0.00,0.50,0.00if forecast_data [rgb]0.67,0.13,1.00is [rgb]0.67,0.13,1.00not
[rgb]0.00,0.50,0.00None [rgb]0.67,0.13,1.00and test_mse [rgb]0.67,0.13,1.00is [rgb]0.67,0.13,1.00not
[rgb]0.00,0.50,0.00None: all_forecasts[ticker] [rgb]0.40,0.40,0.40= forecast_data all_mses[ticker]
[rgb]0.40,0.40,0.40= test_mse [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.67,0.36,0.12"n[rgb]0.73,0.13,0.13Forecast
Data:[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00print(forecast_data[rgb]0.40,0.40,0.40.head())
[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Mean
Squared Error (Test Set): [rgb]0.64,0.35,0.47–test_mse[rgb]0.64,0.35,0.47:
    red[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00else:
[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Forecasting failed
for [rgb]0.64,0.35,0.47–ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13.[rgb]0.73,0.13,0.13")
    [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.67,0.36,0.12"n[rgb]0.73,0.13,0.13--- Sum-
mary of Forecasts ---[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00for ticker, mse [rgb]0.67,0.13,1.00in
all_mses[rgb]0.40,0.40,0.40.items(): [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Ticke
```

red [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13, Test MSE: [rgb]0.64,0.35,0.47−mse[rgb]0.64,0.35,0.47**:**[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13")

[rgb]0.00,0.50,0.00**else**: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: The [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 DataFrame is not available. Please run the data loading and extraction part first.[rgb]0.73,0.13,0.13")

[commandchars= {},codes*=]

— Forecasting for GC=F — Forecast plot for GC=F saved to GC=F_forecast_10yr_train.png

Forecast Data: GC=F_Forecast Date 2024-12-31 1155.312475 2025-01-01 1155.259973 2025-01-02 1155.207471 2025-01-03 1155.154969 2025-01-04 1155.102467 Mean Squared Error (Test Set): 627640.

red69

— Forecasting for PA=F — Forecast plot for PA=F saved to PA=F_forecast_10yr_train.png

Forecast Data: PA=F_Forecast Date 2024-12-31 1533.728117 2025-01-01 1533.924182 2025-01-02 1534.

red120247 2025-01-03 1534.316312 2025-01-04 1534.512376 Mean Squared Error (Test Set): 658452.70

— Forecasting for PL=F — Forecast plot for PL=F saved to PL=F_forecast_10yr_train.png

Forecast Data: PL=F_Forecast Date 2024-12-31 190.548873 2025-01-01 190.258864 2025-01-02 189.

red968854 2025-01-03 189.678844 2025-01-04 189.388835 Mean Squared Error (Test Set): 300597.83

— Forecasting for SI=F — Forecast plot for SI=F saved to SI=F_forecast_10yr_train.png

Forecast Data: SI=F_Forecast Date 2024-12-31 5.021818 2025-01-01 5.017415 2025-01-02 5.013011 2025-01-03 5.008608 2025-01-04 5.004204 Mean Squared Error (Test Set): 249.60

— Summary of Forecasts — Ticker: GC=F, Test MSE: 627640.69 Ticker: PA=F, Test MSE: 658452.70 Ticker: PL=F, Test MSE: 300597.83 Ticker: SI=F, Test MSE: 249.60

[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackground, colframe[redacted]s= {},codes*=] [rgb]0.00,0.50,0.00**from** [rgb]0.00,0.00,1.00**statsmodels**[rgb]0.00,0.00,1.00**.**

red[rgb]0.00,0.00,1.00**tsa**[rgb]0.00,0.00,1.00**.**[rgb]0.00,0.00,1.00**arima**[rgb]0.00,0.00,1.00**.**

red[rgb]0.00,0.00,1.00**model** [rgb]0.00,0.50,0.00**import** ARIMA [rgb]0.00,0.50,0.00**from** [rgb]0.00,0.00,1.00**sklearn**[rgb]0.00,0.00,1.00**.**[rgb]0.00,0.00,1.00**metrics** [rgb]0.00,0.50,0.00**import** mean˙squared˙error

[rgb]0.24,0.48,0.48*# This calculates the AR by setting the mean to zero and only changing the lags* [rgb]0.00,0.50,0.00**def** [rgb]0.00,0.00,1.00forecast˙autoregression˙ticker(df, ticker, train˙years[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.4010, test˙years[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.405, forecast˙years[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.405, lags[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.405, filename[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00**None**):

[rgb]0.00,0.50,0.00**if** ticker [rgb]0.67,0.13,1.00**not** [rgb]0.67,0.13,1.00**in** df[rgb]0.40,0.40,0.40.columns:

red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: Ticker [rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 not found in the DataFrame.[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**

ticker˙data [rgb]0.40,0.40,0.40= df[ticker][rgb]0.40,0.40,0.40.dropna() [rgb]0.00,0.50,0.00**if** ticker˙data[rgb]0.40,0.40,0.40.empty: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Erro

red No valid data found for ticker [rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13'[rg

red[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**

dates [rgb]0.40,0.40,0.40= ticker˙data[rgb]0.40,0.40,0.40.index

[rgb]0.00,0.50,0.00**if** [rgb]0.67,0.13,1.00**not** dates[rgb]0.40,0.40,0.40.empty:

red start˙date [rgb]0.40,0.40,0.40= dates[rgb]0.40,0.40,0.40.min() train˙end˙date [rgb]0.40,0.40,0.40= start˙date [rgb]0.40,0.40,0.40+ pd[rgb]0.40,0.40,0.40.

redDateOffset(years[rgb]0.40,0.40,0.40=train˙years) test˙end˙date [rgb]0.40,0.40,0.40= train˙end˙date [rgb]0.40,0.40,0.40+ pd[rgb]0.40,0.40,0.40.DateOffset(years[rgb]0.40,0.40,0.40=test˙years) forecast˙start˙date [rgb]0.40,0.40,0.40= test˙end˙date [rgb]0.00,0.50,0.00**else**: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: No date information available for ticker [rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.

red[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**

train˙data [rgb]0.40,0.40,0.40= ticker˙data[dates [rgb]0.40,0.40,0.40¡ train˙end˙date] test˙data [rgb]0.40,0.40,0.40= ticker˙data[(dates [rgb]0.40,0.40,0.40¿[rgb]0.40,0.40,0.40= train˙end˙date) [rgb]0.40,0.40,0.40& (dates [rgb]0.40,0.40,0.40¡ test˙end˙date)]

[rgb]0.00,0.50,0.00**if** train˙data[rgb]0.40,0.40,0.40.shape[[rgb]0.40,0.40,0.400] [rgb]0.40,0.40,0.40¡ lags [rgb]0.40,0.40,0.40+ [rgb]0.40,0.40,0.401 [rgb]0.67,0.13,1.00**or** test˙data[rgb]0.40,0.40,0.40.empty:

```
[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Warning: Not enough
data for training ([rgb]0.64,0.35,0.47−[rgb]0.00,0.50,0.00len(train˙data)[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13
samples) or testing for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13
with [rgb]0.64,0.35,0.47−lags[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 lags.[rgb]0.73,0.13,0.13")
[rgb]0.00,0.50,0.00return [rgb]0.00,0.50,0.00None, [rgb]0.00,0.50,0.00None
    [rgb]0.00,0.50,0.00try: model [rgb]0.40,0.40,0.40= ARIMA(train˙data, order[rgb]0.40,0.40,0.40=(lags,
[rgb]0.40,0.40,0.400, [rgb]0.40,0.40,0.400)) model˙fit [rgb]0.40,0.40,0.40= model[rgb]0.40,0.40,0.40.fit()
    predictions [rgb]0.40,0.40,0.40= model˙fit[rgb]0.40,0.40,0.40.predict(start[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00len(trai
    red end[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00len(train˙data) [rgb]0.40,0.40,0.40+
[rgb]0.00,0.50,0.00len(test˙data) [rgb]0.40,0.40,0.40- [rgb]0.40,0.40,0.401) mse [rgb]0.40,0.40,0.40=
mean˙squared˙error(test˙data, predictions)
    forecast˙result [rgb]0.40,0.40,0.40= model˙fit[rgb]0.40,0.40,0.40.get˙forecast(steps[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.4
[rgb]0.40,0.40,0.40* forecast˙years) [rgb]0.40,0.40,0.40+ [rgb]0.40,0.40,0.401) fore-
cast [rgb]0.40,0.40,0.40= forecast˙result[rgb]0.40,0.40,0.40.predicted˙mean forecast˙dates
[rgb]0.40,0.40,0.40= pd[rgb]0.40,0.40,0.40.date˙range(start[rgb]0.40,0.40,0.40=dates[[rgb]0.40,0.40,0.40-
[rgb]0.40,0.40,0.401], periods[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00len(forecast),
    red freq[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13D[rgb]0.73,0.13,0.13') forecast˙df
[rgb]0.40,0.40,0.40= pd[rgb]0.40,0.40,0.40.DataFrame(−[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13':
    red forecast˙dates, [rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.1
    red forecast")[rgb]0.40,0.40,0.40.set˙index([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13')
    plt[rgb]0.40,0.40,0.40.figure(figsize[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.4012,
    red [rgb]0.40,0.40,0.406)) plt[rgb]0.40,0.40,0.40.plot(train˙data[rgb]0.40,0.40,0.40.index, train˙data,
    red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Training Data
([rgb]0.64,0.35,0.47−train˙years[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 Years)[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.plot(test˙data[rgb]0.40,0.40,0.40.index, test˙data, la-
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Testing Data
([rgb]0.64,0.35,0.47−test˙years[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 Years)[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.plot(test˙data[rgb]0.40,0.40,0.40.index, predictions, la-
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Test Predic-
tions (AR([rgb]0.64,0.35,0.47−lags[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13))[rgb]0.73,0.13,0.13',
    red linestyle[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13--
[rgb]0.73,0.13,0.13') plt[rgb]0.40,0.40,0.40.plot(forecast˙dates, forecast, la-
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−forecast˙years[rgb]0.64,0.35,0.47"[rgb]0.7
Year Forecast (AR([rgb]0.64,0.35,0.47−lags[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13))[rgb]0.73,0.13,0.13',
    red linestyle[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13-[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.xlabel([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.ylabel([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Close Price[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.title([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Auto Regres-
sion Forecast for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 (MSE =
[rgb]0.64,0.35,0.47−mse[rgb]0.64,0.35,0.47:[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13,
    red Lags = [rgb]0.64,0.35,0.47−lags[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13)[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.legend() plt[rgb]0.40,0.40,0.40.grid([rgb]0.00,0.50,0.00True) plt[rgb]0.40,0.40,0.40.
    redtight˙layout()
    [rgb]0.00,0.50,0.00if filename [rgb]0.67,0.13,1.00is [rgb]0.00,0.50,0.00None: filename
[rgb]0.40,0.40,0.40= [rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13
    red[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13˙lags˙[rgb]0.64,0.35,0.47−lags[rgb]0.64,0.35,0.47"[rgb]0.73
    redpng[rgb]0.73,0.13,0.13' plt[rgb]0.40,0.40,0.40.savefig(filename) [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0
Regression forecast plot for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 saved to
[rgb]0.64,0.35,0.47−filename[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13") plt[rgb]0.40,0.40,0.40.close()
    [rgb]0.00,0.50,0.00return forecast˙df, mse
    [rgb]0.00,0.50,0.00except [rgb]0.80,0.25,0.22Exception [rgb]0.00,0.50,0.00as e:
    red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error
during Auto Regression for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13:
    red [rgb]0.64,0.35,0.47−e[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00return
[rgb]0.00,0.50,0.00None, [rgb]0.00,0.50,0.00None
    [rgb]0.00,0.50,0.00if [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13'
[rgb]0.67,0.13,1.00in [rgb]0.00,0.50,0.00locals() [rgb]0.67,0.13,1.00and close˙prices˙df [rgb]0.67,0.13,1.00is
[rgb]0.67,0.13,1.00not [rgb]0.00,0.50,0.00None [rgb]0.67,0.13,1.00and [rgb]0.67,0.13,1.00not
```

```
close˙prices˙df[rgb]0.40,0.40,0.40.empty:
    tickers [rgb]0.40,0.40,0.40= [col [rgb]0.00,0.50,0.00for col [rgb]0.67,0.13,1.00in
close˙prices˙df[rgb]0.40,0.40,0.40.columns]
    all˙ar˙forecasts [rgb]0.40,0.40,0.40= −″ all˙ar˙mses [rgb]0.40,0.40,0.40= −″ ar˙lags [rgb]0.40,0.40,0.40=
[rgb]0.40,0.40,0.405
    [rgb]0.00,0.50,0.00for ticker [rgb]0.67,0.13,1.00in tickers: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13
-- Auto Regression Forecasting for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13 ---
[rgb]0.73,0.13,0.13") forecast˙filename [rgb]0.40,0.40,0.40= [rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker
    redpng[rgb]0.73,0.13,0.13' forecast˙data, test˙mse [rgb]0.40,0.40,0.40= forecast˙autoregression˙ticker(
close˙prices˙df, ticker, lags[rgb]0.40,0.40,0.40=ar˙lags, filename[rgb]0.40,0.40,0.40=forecast˙filename )
    [rgb]0.00,0.50,0.00if forecast˙data [rgb]0.67,0.13,1.00is [rgb]0.67,0.13,1.00not
[rgb]0.00,0.50,0.00None [rgb]0.67,0.13,1.00and test˙mse [rgb]0.67,0.13,1.00is [rgb]0.67,0.13,1.00not
[rgb]0.00,0.50,0.00None: all˙ar˙forecasts[ticker] [rgb]0.40,0.40,0.40= forecast˙data all˙ar˙mses[ticker]
[rgb]0.40,0.40,0.40= test˙mse [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.67,0.36,0.12"n[rgb]0.73,0.13,0.13Auto
Regression Forecast Data:[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00print(forecast˙data[rgb]0.40,0.40,0.40.
    redhead()) [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Mean
Squared Error (Test Set): [rgb]0.64,0.35,0.47−test˙mse[rgb]0.64,0.35,0.47:
    red[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00else:
[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Auto Regression fore-
casting failed for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13.[rgb]0.73,0.13,0.13")
    [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.67,0.36,0.12"n[rgb]0.73,0.13,0.13--- Sum-
mary of Auto Regression Forecasts ---[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00for ticker,
    red mse [rgb]0.67,0.13,1.00in all˙ar˙mses[rgb]0.40,0.40,0.40.items():
    red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Ticker:
    red [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13, Test
MSE (AR([rgb]0.64,0.35,0.47−ar˙lags[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13)):
[rgb]0.64,0.35,0.47−mse[rgb]0.64,0.35,0.47:[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13")
    [rgb]0.00,0.50,0.00else: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: The
[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 DataFrame is
not available. Please run the data loading and extraction part first.[rgb]0.73,0.13,0.13")
    [rgb]0.00,0.50,0.00if [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13'
[rgb]0.67,0.13,1.00in [rgb]0.00,0.50,0.00locals() [rgb]0.67,0.13,1.00and close˙prices˙df [rgb]0.67,0.13,1.00is
[rgb]0.67,0.13,1.00not [rgb]0.00,0.50,0.00None [rgb]0.67,0.13,1.00and [rgb]0.67,0.13,1.00not
close˙prices˙df[rgb]0.40,0.40,0.40.empty:
    tickers [rgb]0.40,0.40,0.40= [col [rgb]0.00,0.50,0.00for col [rgb]0.67,0.13,1.00in
close˙prices˙df[rgb]0.40,0.40,0.40.columns]
    all˙ar˙forecasts [rgb]0.40,0.40,0.40= −″ all˙ar˙mses [rgb]0.40,0.40,0.40= −″ ar˙lags [rgb]0.40,0.40,0.40=
[rgb]0.40,0.40,0.4010
    [rgb]0.00,0.50,0.00for ticker [rgb]0.67,0.13,1.00in tickers: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13
-- Auto Regression Forecasting for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13 ---
[rgb]0.73,0.13,0.13") forecast˙filename [rgb]0.40,0.40,0.40= [rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker
    redpng[rgb]0.73,0.13,0.13' forecast˙data, test˙mse [rgb]0.40,0.40,0.40= forecast˙autoregression˙ticker(
close˙prices˙df, ticker, lags[rgb]0.40,0.40,0.40=ar˙lags, filename[rgb]0.40,0.40,0.40=forecast˙filename )
    [rgb]0.00,0.50,0.00if forecast˙data [rgb]0.67,0.13,1.00is [rgb]0.67,0.13,1.00not
[rgb]0.00,0.50,0.00None [rgb]0.67,0.13,1.00and test˙mse [rgb]0.67,0.13,1.00is [rgb]0.67,0.13,1.00not
[rgb]0.00,0.50,0.00None: all˙ar˙forecasts[ticker] [rgb]0.40,0.40,0.40= forecast˙data all˙ar˙mses[ticker]
[rgb]0.40,0.40,0.40= test˙mse [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.67,0.36,0.12"n[rgb]0.73,0.13,0.13Auto
Regression Forecast Data:[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00print(forecast˙data[rgb]0.40,0.40,0.40.
    redhead()) [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Mean
Squared Error (Test Set): [rgb]0.64,0.35,0.47−test˙mse[rgb]0.64,0.35,0.47:
    red[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00else:
[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Auto Regression fore-
casting failed for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13.[rgb]0.73,0.13,0.13")
    [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.67,0.36,0.12"n[rgb]0.73,0.13,0.13--- Sum-
mary of Auto Regression Forecasts ---[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00for ticker,
    red mse [rgb]0.67,0.13,1.00in all˙ar˙mses[rgb]0.40,0.40,0.40.items():
    red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Ticker:
    red [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13, Test
```

MSE (AR([rgb]0.64,0.35,0.47−ar˙lags[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13)): [rgb]0.64,0.35,0.47−mse[rgb]0.64,0.35,0.47:[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13")

[rgb]0.00,0.50,0.00**else**: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: The [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 DataFrame is not available. Please run the data loading and extraction part first.[rgb]0.73,0.13,0.13")

[commandchars= {},codes*=]

— Auto Regression Forecasting for GC=F — Auto Regression forecast plot for GC=F saved to GC=F_autoreg_forecast_10yr_train_lags_5.png

Auto Regression Forecast Data: GC=F_Forecast_AR Date 2024-12-31 1547.868464 2025-01-01 1547. red286117 2025-01-02 1546.831015 2025-01-03 1545.961556 2025-01-04 1545.405423 Mean Squared Error (Test Set): 380830.33

— Auto Regression Forecasting for PA=F — Auto Regression forecast plot for PA=F saved to PA=F_autoreg_forecast_10yr_train_lags_5.png

Auto Regression Forecast Data: PA=F_Forecast_AR Date 2024-12-31 1953.758463 2025-01-01 1953. red244223 2025-01-02 1952.189512 2025-01-03 1951.907565 2025-01-04 1951.698633 Mean Squared Error (Test Set): 288794.84

— Auto Regression Forecasting for PL=F — Auto Regression forecast plot for PL=F saved to PL=F_autoreg_forecast_10yr_train_lags_5.png

Auto Regression Forecast Data: PL=F_Forecast_AR Date 2024-12-31 985.290287 2025-01-01 985. red979128 2025-01-02 986.486515 2025-01-03 986.971818 2025-01-04 987.361817 Mean Squared Error (Test Set): 37219.53

— Auto Regression Forecasting for SI=F — Auto Regression forecast plot for SI=F saved to SI=F_autoreg_forecast_10yr_train_lags_5.png

Auto Regression Forecast Data: SI=F_Forecast_AR Date 2024-12-31 18.069154 2025-01-01 18.078745 2025-01-02 18.085783 2025-01-03 18.090823 2025-01-04 18.097807 Mean Squared Error (Test Set): 27.01

— Summary of Auto Regression Forecasts — Ticker: GC=F, Test MSE (AR(5)): 380830.33 Ticker: PA=F, Test MSE (AR(5)): 288794.84 Ticker: PL=F, Test MSE (AR(5)): 37219.53 Ticker: SI=F, Test MSE (AR(5)): 27.01

— Auto Regression Forecasting for GC=F — Auto Regression forecast plot for GC=F saved to GC=F_autoreg_forecast_10yr_train_lags_10.png

Auto Regression Forecast Data: GC=F_Forecast_AR Date 2024-12-31 1547.524411 2025-01-01 1547. red096836 2025-01-02 1547.506723 2025-01-03 1547.155057 2025-01-04 1546.750937 Mean Squared Error (Test Set): 384346.51

— Auto Regression Forecasting for PA=F — Auto Regression forecast plot for PA=F saved to PA=F_autoreg_forecast_10yr_train_lags_10.png

Auto Regression Forecast Data: PA=F_Forecast_AR Date 2024-12-31 1964.182342 2025-01-01 1960. red335269 2025-01-02 1955.836422 2025-01-03 1956.916671 2025-01-04 1951.816245 Mean Squared Error (Test Set): 246954.94

— Auto Regression Forecasting for PL=F — Auto Regression forecast plot for PL=F saved to PL=F_autoreg_forecast_10yr_train_lags_10.png

Auto Regression Forecast Data: PL=F_Forecast_AR Date 2024-12-31 984.441675 2025-01-01 985. red703873 2025-01-02 985.342111 2025-01-03 986.315493 2025-01-04 986.848150 Mean Squared Error (Test Set): 37963.15

— Auto Regression Forecasting for SI=F — Auto Regression forecast plot for SI=F saved to SI=F_autoreg_forecast_10yr_train_lags_10.png

Auto Regression Forecast Data: SI=F_Forecast_AR Date 2024-12-31 18.087444 2025-01-01 18.103638 2025-01-02 18.113741 2025-01-03 18.128395 2025-01-04 18.132385 Mean Squared Error (Test Set): 25.83

— Summary of Auto Regression Forecasts — Ticker: GC=F, Test MSE (AR(10)): 384346.51 Ticker: PA=F, Test MSE (AR(10)): 246954.94 Ticker: PL=F, Test MSE (AR(10)): 37963.15 Ticker: SI=F, Test MSE (AR(10)): 25.83

[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackground, colframe=[redacted]

[commandchars= {},codes*=] [rgb]0.24,0.48,0.48*# This finds the ARIMA* [rgb]0.00,0.50,0.00**def** [rgb]0.00,0.00,1.00forecast˙arima˙ticker(df, ticker, train˙years[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.4010, test˙years[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.405, forecast˙years[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.405, red order[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.405, [rgb]0.40,0.40,0.401, [rgb]0.40,0.40,0.400), filename[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00**None**):

[rgb]0.00,0.50,0.00**if** ticker [rgb]0.67,0.13,1.00**not** [rgb]0.67,0.13,1.00**in** df[rgb]0.40,0.40,0.40.columns:
red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: Ticker
[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13
not found in the DataFrame.[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**,
[rgb]0.00,0.50,0.00**None**
ticker˙data [rgb]0.40,0.40,0.40= df[ticker][rgb]0.40,0.40,0.40.dropna() [rgb]0.00,0.50,0.00**if**
ticker˙data[rgb]0.40,0.40,0.40.empty: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Erro
red No valid data found for ticker [rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13'[rg
red[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**
dates [rgb]0.40,0.40,0.40= ticker˙data[rgb]0.40,0.40,0.40.index
[rgb]0.00,0.50,0.00**if** [rgb]0.67,0.13,1.00**not** dates[rgb]0.40,0.40,0.40.empty:
red start˙date [rgb]0.40,0.40,0.40= dates[rgb]0.40,0.40,0.40.min() train˙end˙date
[rgb]0.40,0.40,0.40= start˙date [rgb]0.40,0.40,0.40+ pd[rgb]0.40,0.40,0.40.
redDateOffset(years[rgb]0.40,0.40,0.40=train˙years) test˙end˙date [rgb]0.40,0.40,0.40=
train˙end˙date [rgb]0.40,0.40,0.40+ pd[rgb]0.40,0.40,0.40.DateOffset(years[rgb]0.40,0.40,0.40=test˙years)
forecast˙start˙date [rgb]0.40,0.40,0.40= test˙end˙date [rgb]0.00,0.50,0.00**else**:
[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: No date informa-
tion available for ticker [rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.
red[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**
train˙data [rgb]0.40,0.40,0.40= ticker˙data[dates [rgb]0.40,0.40,0.40¡ train˙end˙date] test˙data
[rgb]0.40,0.40,0.40= ticker˙data[(dates [rgb]0.40,0.40,0.40¿[rgb]0.40,0.40,0.40= train˙end˙date)
[rgb]0.40,0.40,0.40& (dates [rgb]0.40,0.40,0.40¡ test˙end˙date)]
[rgb]0.00,0.50,0.00**if** [rgb]0.00,0.50,0.00len(train˙data) [rgb]0.40,0.40,0.40¡ order[[rgb]0.40,0.40,0.400]
[rgb]0.40,0.40,0.40+ order[[rgb]0.40,0.40,0.402] [rgb]0.40,0.40,0.40+ order[[rgb]0.40,0.40,0.401]
[rgb]0.40,0.40,0.40+ [rgb]0.40,0.40,0.401 [rgb]0.67,0.13,1.00**or** test˙data[rgb]0.40,0.40,0.40.empty:
[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Warning: Not enough
data for training ([rgb]0.64,0.35,0.47−[rgb]0.00,0.50,0.00len(train˙data)[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13
samples) or testing for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13 with
ARIMA order [rgb]0.64,0.35,0.47−order[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13.[rgb]0.73,0.13,0.13"))
[rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**
[rgb]0.00,0.50,0.00**try**: model [rgb]0.40,0.40,0.40= ARIMA(train˙data, or-
der[rgb]0.40,0.40,0.40=order) model˙fit [rgb]0.40,0.40,0.40= model[rgb]0.40,0.40,0.40.fit()
predictions [rgb]0.40,0.40,0.40= model˙fit[rgb]0.40,0.40,0.40.predict(start[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00len(trai
red end[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00len(train˙data) [rgb]0.40,0.40,0.40+
[rgb]0.00,0.50,0.00len(test˙data) [rgb]0.40,0.40,0.40- [rgb]0.40,0.40,0.401) mse [rgb]0.40,0.40,0.40=
mean˙squared˙error(test˙data, predictions)
forecast˙result [rgb]0.40,0.40,0.40= model˙fit[rgb]0.40,0.40,0.40.get˙forecast(steps[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.4
[rgb]0.40,0.40,0.40* forecast˙years) [rgb]0.40,0.40,0.40+ [rgb]0.40,0.40,0.401)
[rgb]0.24,0.48,0.48# *Add 1 to include the last point* forecast [rgb]0.40,0.40,0.40=
forecast˙result[rgb]0.40,0.40,0.40.predicted˙mean forecast˙dates [rgb]0.40,0.40,0.40=
pd[rgb]0.40,0.40,0.40.date˙range(start[rgb]0.40,0.40,0.40=dates[[rgb]0.40,0.40,0.40-
[rgb]0.40,0.40,0.401], periods[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00len(forecast),
red freq[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13D[rgb]0.73,0.13,0.13')
[rgb]0.24,0.48,0.48# *Start from the last date* forecast˙df [rgb]0.40,0.40,0.40= pd[rgb]0.40,0.40,0.40.
redDataFrame(−[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13': forecast˙dates,
[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13˙Forecast˙ARIMA[rg
red forecast″)[rgb]0.40,0.40,0.40.set˙index([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.figure(figsize[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.4012,
red [rgb]0.40,0.40,0.406)) plt[rgb]0.40,0.40,0.40.plot(train˙data[rgb]0.40,0.40,0.40.index, train˙data,
red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Training Data
([rgb]0.64,0.35,0.47−train˙years[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13 Years)[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.plot(test˙data[rgb]0.40,0.40,0.40.index, test˙data, la-
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Testing Data
([rgb]0.64,0.35,0.47−test˙years[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13 Years)[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.plot(test˙data[rgb]0.40,0.40,0.40.index, predictions, la-
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Test Predic-
tions (ARIMA[rgb]0.64,0.35,0.47−order[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13)[rgb]0.73,0.13,0.13',
red linestyle[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13--

```
[rgb]0.73,0.13,0.13') plt[rgb]0.40,0.40,0.40.plot(forecast˙dates, forecast, la-
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−forecast˙years[rgb]0.64,0.35,0.47"[rgb]0.7
Year Forecast (ARIMA[rgb]0.64,0.35,0.47−order[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13)[rgb]0.73,0.13,0.13',
    red linestyle[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13-[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.xlabel([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.ylabel([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Close Price[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.title([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13ARIMA
Forecast for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 (MSE =
[rgb]0.64,0.35,0.47−mse[rgb]0.64,0.35,0.47:[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13,
    red Order = [rgb]0.64,0.35,0.47−order[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13)[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.legend() plt[rgb]0.40,0.40,0.40.grid([rgb]0.00,0.50,0.00True) plt[rgb]0.40,0.40,0.40.
    redtight˙layout()
    [rgb]0.00,0.50,0.00if filename [rgb]0.67,0.13,1.00is [rgb]0.00,0.50,0.00None: filename
[rgb]0.40,0.40,0.40= [rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13
    red[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13˙order˙[rgb]0.64,0.35,0.47−order[rgb]0.64,0.35,0.47"[rgb]0
    redpng[rgb]0.73,0.13,0.13' plt[rgb]0.40,0.40,0.40.savefig(filename) [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0
forecast plot for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 saved to
[rgb]0.64,0.35,0.47−filename[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13") plt[rgb]0.40,0.40,0.40.close()
    [rgb]0.00,0.50,0.00return forecast˙df, mse
    [rgb]0.00,0.50,0.00except [rgb]0.80,0.25,0.22Exception [rgb]0.00,0.50,0.00as e:
    red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error
during ARIMA for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13:
    red [rgb]0.64,0.35,0.47−e[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00return
[rgb]0.00,0.50,0.00None, [rgb]0.00,0.50,0.00None
    [rgb]0.00,0.50,0.00if [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13'
[rgb]0.67,0.13,1.00in [rgb]0.00,0.50,0.00locals() [rgb]0.67,0.13,1.00and close˙prices˙df [rgb]0.67,0.13,1.00is
[rgb]0.67,0.13,1.00not [rgb]0.00,0.50,0.00None [rgb]0.67,0.13,1.00and [rgb]0.67,0.13,1.00not
close˙prices˙df[rgb]0.40,0.40,0.40.empty:
    tickers [rgb]0.40,0.40,0.40= [col [rgb]0.00,0.50,0.00for col [rgb]0.67,0.13,1.00in
close˙prices˙df[rgb]0.40,0.40,0.40.columns]
    all˙arima˙forecasts [rgb]0.40,0.40,0.40= −" all˙arima˙mses [rgb]0.40,0.40,0.40= −" arima˙order
[rgb]0.40,0.40,0.40= ([rgb]0.40,0.40,0.405, [rgb]0.40,0.40,0.401, [rgb]0.40,0.40,0.400) [rgb]0.24,0.48,0.48# 
You can adjust the ARIMA order (p, d, q)
    [rgb]0.00,0.50,0.00for ticker [rgb]0.67,0.13,1.00in tickers: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13
-- ARIMA Forecasting for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 ---
[rgb]0.73,0.13,0.13") forecast˙filename [rgb]0.40,0.40,0.40= [rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker
    redpng[rgb]0.73,0.13,0.13' forecast˙data, test˙mse [rgb]0.40,0.40,0.40= fore-
cast˙arima˙ticker( close˙prices˙df, ticker, order[rgb]0.40,0.40,0.40=arima˙order, file-
name[rgb]0.40,0.40,0.40=forecast˙filename )
    [rgb]0.00,0.50,0.00if forecast˙data [rgb]0.67,0.13,1.00is [rgb]0.67,0.13,1.00not
[rgb]0.00,0.50,0.00None [rgb]0.67,0.13,1.00and test˙mse [rgb]0.67,0.13,1.00is
[rgb]0.67,0.13,1.00not [rgb]0.00,0.50,0.00None: all˙arima˙forecasts[ticker]
[rgb]0.40,0.40,0.40= forecast˙data all˙arima˙mses[ticker] [rgb]0.40,0.40,0.40= test˙mse
[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.67,0.36,0.12"n[rgb]0.73,0.13,0.13ARIMA
Forecast Data:[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00print(forecast˙data[rgb]0.40,0.40,0.40.
    redhead()) [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Mean
Squared Error (Test Set): [rgb]0.64,0.35,0.47−test˙mse[rgb]0.64,0.35,0.47:
    red[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00else:
[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13ARIMA forecasting
failed for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13.[rgb]0.73,0.13,0.13")
    [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.67,0.36,0.12"n[rgb]0.73,0.13,0.13--
- Summary of ARIMA Forecasts ---[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00for ticker,
    red mse [rgb]0.67,0.13,1.00in all˙arima˙mses[rgb]0.40,0.40,0.40.items():
    red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Ticker:
    red [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13, Test MSE
(ARIMA[rgb]0.64,0.35,0.47−arima˙order[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13):
[rgb]0.64,0.35,0.47−mse[rgb]0.64,0.35,0.47:[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13")
    [rgb]0.00,0.50,0.00else: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: The
```

[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 DataFrame is not available. Please run the data loading and extraction part first.[rgb]0.73,0.13,0.13")

[commandchars=
{},codes*=]

— ARIMA Forecasting for GC=F — ARIMA forecast plot for GC=F saved to GC=F_arima_forecast_10yr_train_order_(5, 1, 0).png

ARIMA Forecast Data: GC=F_Forecast_ARIMA Date 2024-12-31 1548.383284 2025-01-01 1548. red332035 2025-01-02 1548.380379 2025-01-03 1548.018543 2025-01-04 1547.851361 Mean Squared Error (Test Set): 225626.08

— ARIMA Forecasting for PA=F — ARIMA forecast plot for PA=F saved to PA=F_arima_forecast_10yr_train_order_(5, 1, 0).png

ARIMA Forecast Data: PA=F_Forecast_ARIMA Date 2024-12-31 1954.418463 2025-01-01 1954. red130180 2025-01-02 1954.291449 2025-01-03 1954.849424 2025-01-04 1955.785661 Mean Squared Error (Test Set): 377881.96

— ARIMA Forecasting for PL=F — ARIMA forecast plot for PL=F saved to PL=F_arima_forecast_10yr_train_order_(5, 1, 0).png

ARIMA Forecast Data: PL=F_Forecast_ARIMA Date 2024-12-31 985.038186 2025-01-01 985.048588 2025-01-02 984.658674 2025-01-03 984.480726 2025-01-04 984.256502 Mean Squared Error (Test Set): 10241.06

— ARIMA Forecasting for SI=F — ARIMA forecast plot for SI=F saved to SI=F_arima_forecast_10yr_train_order_(5, 1, 0).png

ARIMA Forecast Data: SI=F_Forecast_ARIMA Date 2024-12-31 18.061387 2025-01-01 18.064510 2025-01-02 18.063632 2025-01-03 18.063129 2025-01-04 18.064483 Mean Squared Error (Test Set): 48.78

— Summary of ARIMA Forecasts — Ticker: GC=F, Test MSE (ARIMA(5, 1, 0)): 225626.08 Ticker: PA=F, Test MSE (ARIMA(5, 1, 0)): 377881.96 Ticker: PL=F, Test MSE (ARIMA(5, 1, 0)): 10241.06 Ticker: SI=F, Test MSE (ARIMA(5, 1, 0)): 48.78

[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackground, colframe=cellborder] [commandchars=
{},codes*=] [rgb]0.24,0.48,0.48# *This finds the optimized AIC/BIC* [rgb]0.00,0.50,0.00**def** [rgb]0.00,0.00,1.00optimize˙arima˙and˙forecast(df, ticker, train˙years[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.4010, test˙years[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.405, forecast˙years[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.405, max˙ar[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.405, max˙ma[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.405, filename˙prefix[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00**None**):
red

[rgb]0.00,0.50,0.00**if** ticker [rgb]0.67,0.13,1.00**not** [rgb]0.67,0.13,1.00**in** df[rgb]0.40,0.40,0.40.columns: red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: Ticker [rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47–ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 not found in the DataFrame.[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**

ticker˙data [rgb]0.40,0.40,0.40= df[ticker][rgb]0.40,0.40,0.40.dropna() [rgb]0.00,0.50,0.00**if** ticker˙data[rgb]0.40,0.40,0.40.empty: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Erro
red No valid data found for ticker [rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47–ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13'[rg
red[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**

dates [rgb]0.40,0.40,0.40= ticker˙data[rgb]0.40,0.40,0.40.index

[rgb]0.00,0.50,0.00**if** [rgb]0.67,0.13,1.00**not** dates[rgb]0.40,0.40,0.40.empty: red start˙date [rgb]0.40,0.40,0.40= dates[rgb]0.40,0.40,0.40.min() train˙end˙date [rgb]0.40,0.40,0.40= start˙date [rgb]0.40,0.40,0.40+ pd[rgb]0.40,0.40,0.40. redDateOffset(years[rgb]0.40,0.40,0.40=train˙years) test˙end˙date [rgb]0.40,0.40,0.40= train˙end˙date [rgb]0.40,0.40,0.40+ pd[rgb]0.40,0.40,0.40.DateOffset(years[rgb]0.40,0.40,0.40=test˙years) forecast˙start˙date [rgb]0.40,0.40,0.40= test˙end˙date [rgb]0.00,0.50,0.00**else**: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: No date information available for ticker [rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47–ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.
red[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**

train˙data [rgb]0.40,0.40,0.40= ticker˙data[dates [rgb]0.40,0.40,0.40¡ train˙end˙date] test˙data [rgb]0.40,0.40,0.40= ticker˙data[(dates [rgb]0.40,0.40,0.40¿[rgb]0.40,0.40,0.40= train˙end˙date) [rgb]0.40,0.40,0.40& (dates [rgb]0.40,0.40,0.40¡ test˙end˙date)]

[rgb]0.00,0.50,0.00**if** [rgb]0.00,0.50,0.00len(train˙data) [rgb]0.40,0.40,0.40== [rgb]0.40,0.40,0.400 [rgb]0.67,0.13,1.00**or** [rgb]0.00,0.50,0.00len(test˙data) [rgb]0.40,0.40,0.40== [rgb]0.40,0.40,0.400:
red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Warning: Not enough training or testing data for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13.
red[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**
best˙aic [rgb]0.40,0.40,0.40= [rgb]0.00,0.50,0.00float([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13inf[rgb]0.73,0.13,0.13')
best˙bic [rgb]0.40,0.40,0.40= [rgb]0.00,0.50,0.00float([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13inf[rgb]0.73,0.13,0.13')
best˙order [rgb]0.40,0.40,0.40= [rgb]0.00,0.50,0.00**None** aic˙bic˙values [rgb]0.40,0.40,0.40= []
[rgb]0.00,0.50,0.00**for** p [rgb]0.67,0.13,1.00**in** [rgb]0.00,0.50,0.00range(max˙ar [rgb]0.40,0.40,0.40+ [rgb]0.40,0.40,0.401): [rgb]0.00,0.50,0.00**for** q [rgb]0.67,0.13,1.00**in** [rgb]0.00,0.50,0.00range(max˙ma [rgb]0.40,0.40,0.40+ [rgb]0.40,0.40,0.401): order [rgb]0.40,0.40,0.40= (p,
red [rgb]0.40,0.40,0.401, q) [rgb]0.24,0.48,0.48# *Assuming d=1* [rgb]0.00,0.50,0.00**try**:
red model [rgb]0.40,0.40,0.40= ARIMA(train˙data, order[rgb]0.40,0.40,0.40=order) model˙fit [rgb]0.40,0.40,0.40= model[rgb]0.40,0.40,0.40.fit() aic [rgb]0.40,0.40,0.40= model˙fit[rgb]0.40,0.40,0.40.aic bic [rgb]0.40,0.40,0.40= model˙fit[rgb]0.40,0.40,0.40.bic aic˙bic˙values[rgb]0.40,0.40,0.40.append(−[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13order[rgb]0.73,0.13,0.13':
red order, [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13aic[rgb]0.73,0.13,0.13': aic,
red [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13bic[rgb]0.73,0.13,0.13': bic˝) [rgb]0.00,0.50,0.00**if** aic [rgb]0.40,0.40,0.40¡ best˙aic: best˙aic [rgb]0.40,0.40,0.40= aic [rgb]0.00,0.50,0.00**if** bic [rgb]0.40,0.40,0.40¡ best˙bic: best˙bic [rgb]0.40,0.40,0.40= bic best˙order [rgb]0.40,0.40,0.40= order [rgb]0.00,0.50,0.00**except** [rgb]0.80,0.25,0.22**Exception** [rgb]0.00,0.50,0.00**as** e:
red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error fitting ARIMA[rgb]0.64,0.35,0.47−order[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13 for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13: [rgb]0.64,0.35,0.47−e[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.1
[rgb]0.00,0.50,0.00**if** best˙order [rgb]0.67,0.13,1.00**is** [rgb]0.00,0.50,0.00**None**:
red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error: Could not find a suitable ARIMA order for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13.
red[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**
[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Optimized ARIMA order for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13:
red [rgb]0.64,0.35,0.47−best˙order[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13 (AIC: [rgb]0.64,0.35,0.47−best˙aic[rgb]0.64,0.35,0.47**:**[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13,
red BIC: [rgb]0.64,0.35,0.47−best˙bic[rgb]0.64,0.35,0.47**:**[rgb]0.73,0.13,0.13.
red2f[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13)[rgb]0.73,0.13,0.13")
[rgb]0.00,0.50,0.00**try**: best˙model [rgb]0.40,0.40,0.40= ARIMA(train˙data, order[rgb]0.40,0.40,0.40=best˙order) best˙model˙fit [rgb]0.40,0.40,0.40= best˙model[rgb]0.40,0.40,0.40.fit()
predictions [rgb]0.40,0.40,0.40= best˙model˙fit[rgb]0.40,0.40,0.40.predict(start[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00len
red end[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00len(train˙data) [rgb]0.40,0.40,0.40+ [rgb]0.00,0.50,0.00len(test˙data) [rgb]0.40,0.40,0.40- [rgb]0.40,0.40,0.401) mse [rgb]0.40,0.40,0.40= mean˙squared˙error(test˙data, predictions)
forecast˙result [rgb]0.40,0.40,0.40= best˙model˙fit[rgb]0.40,0.40,0.40.
redget˙forecast(steps[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.40365 [rgb]0.40,0.40,0.40* forecast˙years) [rgb]0.40,0.40,0.40+ [rgb]0.40,0.40,0.401) forecast [rgb]0.40,0.40,0.40= forecast˙result[rgb]0.40,0.40,0.40.predicted˙mean forecast˙dates [rgb]0.40,0.40,0.40= pd[rgb]0.40,0.40,0.40.date˙range(start[rgb]0.40,0.40,0.40=dates[[rgb]0.40,0.40,0.40- [rgb]0.40,0.40,0.401], periods[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00len(forecast),
red freq[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13D[rgb]0.73,0.13,0.13') [rgb]0.24,0.48,0.48# *Start from last date* forecast˙df [rgb]0.40,0.40,0.40= pd[rgb]0.40,0.40,0.40.
redDataFrame(−[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13': forecast˙dates, [rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13˙Forecast˙ARIMA[rg
red forecast˝)[rgb]0.40,0.40,0.40.set˙index([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.figure(figsize[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.4012,
red [rgb]0.40,0.40,0.406)) plt[rgb]0.40,0.40,0.40.plot(train˙data[rgb]0.40,0.40,0.40.index, train˙data,
red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Training Data ([rgb]0.64,0.35,0.47−train˙years[rgb]0.64,0.35,0.47˝[rgb]0.73,0.13,0.13 Years)[rgb]0.73,0.13,0.13') plt[rgb]0.40,0.40,0.40.plot(test˙data[rgb]0.40,0.40,0.40.index, test˙data, la-

```
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Testing Data
([rgb]0.64,0.35,0.47−test˙years[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 Years)[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.plot(test˙data[rgb]0.40,0.40,0.40.index, predictions, la-
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Test Predictions
(ARIMA[rgb]0.64,0.35,0.47−best˙order[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13)[rgb]0.73,0.13,0.13',
    red linestyle[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13--
[rgb]0.73,0.13,0.13') plt[rgb]0.40,0.40,0.40.plot(forecast˙dates, forecast, la-
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.64,0.35,0.47−forecast˙years[rgb]0.64,0.35,0.47"[rgb]0.7
Year Forecast (ARIMA[rgb]0.64,0.35,0.47−best˙order[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13)[rgb]0.73,0.13,0.13',
    red linestyle[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13-[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.xlabel([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.ylabel([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Close Price[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.title([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Optimized
ARIMA Forecast for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 (MSE =
[rgb]0.64,0.35,0.47−mse[rgb]0.64,0.35,0.47:[rgb]0.73,0.13,0.13.2f[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13,
    red Order = [rgb]0.64,0.35,0.47−best˙order[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13)[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.legend() plt[rgb]0.40,0.40,0.40.grid([rgb]0.00,0.50,0.00True) plt[rgb]0.40,0.40,0.40.
    redtight˙layout()
    [rgb]0.00,0.50,0.00if filename˙prefix: forecast˙filename [rgb]0.40,0.40,0.40=
[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.64,0.35,0.47−filename˙prefix[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13˙[rgb]0.64,0
    redpng[rgb]0.73,0.13,0.13" [rgb]0.00,0.50,0.00else: forecast˙filename [rgb]0.40,0.40,0.40=
[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13˙optimized˙arima˙for
    redpng[rgb]0.73,0.13,0.13" plt[rgb]0.40,0.40,0.40.savefig(forecast˙filename)
[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Optimized ARIMA
forecast plot for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 saved to
[rgb]0.64,0.35,0.47−forecast˙filename[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13") plt[rgb]0.40,0.40,0.40.
    redclose()
    aic˙bic˙df [rgb]0.40,0.40,0.40= pd[rgb]0.40,0.40,0.40.DataFrame(aic˙bic˙values)
plt[rgb]0.40,0.40,0.40.figure(figsize[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.4010, [rgb]0.40,0.40,0.406))
plt[rgb]0.40,0.40,0.40.plot(aic˙bic˙df[[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13order[rgb]0.73,0.13,0.13']rgb]0.40,0.40,0.40.
    redastype([rgb]0.00,0.50,0.00str), aic˙bic˙df[[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13aic[rgb]0.73,0.13,0.13'],
    red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13AIC[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.plot(aic˙bic˙df[[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13order[rgb]0.73,0.13,0.13']rgb]0.40,0.40,0.40.
    redastype([rgb]0.00,0.50,0.00str), aic˙bic˙df[[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13bic[rgb]0.73,0.13,0.13'],
    red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13BIC[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.scatter([rgb]0.00,0.50,0.00str(best˙order), best˙aic, color[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0
    red marker[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13o[rgb]0.73,0.13,0.13',
    red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Min
AIC ([rgb]0.64,0.35,0.47−best˙order[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13)[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.scatter([rgb]0.00,0.50,0.00str(best˙order), best˙bic, color[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0
    red marker[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13o[rgb]0.73,0.13,0.13',
    red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Min
BIC ([rgb]0.64,0.35,0.47−best˙order[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13)[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.xlabel([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13ARIMA Order (p, 1,
q)[rgb]0.73,0.13,0.13') plt[rgb]0.40,0.40,0.40.ylabel([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13AIC / BIC
Value[rgb]0.73,0.13,0.13') plt[rgb]0.40,0.40,0.40.title([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13AIC
and BIC for ARIMA Orders for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.legend() plt[rgb]0.40,0.40,0.40.grid([rgb]0.00,0.50,0.00True) plt[rgb]0.40,0.40,0.40.
    redtight˙layout()
    [rgb]0.00,0.50,0.00if filename˙prefix: aic˙bic˙filename [rgb]0.40,0.40,0.40=
[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.64,0.35,0.47−filename˙prefix[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13˙[rgb]0.64,0
    redpng[rgb]0.73,0.13,0.13" [rgb]0.00,0.50,0.00else: aic˙bic˙filename [rgb]0.40,0.40,0.40=
[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13˙arima˙aic˙bic.
    redpng[rgb]0.73,0.13,0.13" plt[rgb]0.40,0.40,0.40.savefig(aic˙bic˙filename)
[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13AIC and BIC plot
for ARIMA orders for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13 saved to
[rgb]0.64,0.35,0.47−aic˙bic˙filename[rgb]0.64,0.35,0.47"[rgb]0.73,0.13,0.13") plt[rgb]0.40,0.40,0.40.close()
    [rgb]0.00,0.50,0.00return forecast˙df, mse, best˙order
```

[rgb]0.00,0.50,0.00**except** [rgb]0.80,0.25,0.22**Exception** [rgb]0.00,0.50,0.00**as** e: red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13" [rgb]0.73,0.13,0.13Error during forecasting with optimized ARIMA for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13: red [rgb]0.64,0.35,0.47−e[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**return** [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**, [rgb]0.00,0.50,0.00**None**

[rgb]0.00,0.50,0.00**if** [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13' [rgb]0.67,0.13,1.00**in** [rgb]0.00,0.50,0.00locals() [rgb]0.67,0.13,1.00**and** close˙prices˙df [rgb]0.67,0.13,1.00**is** [rgb]0.67,0.13,1.00**not** [rgb]0.00,0.50,0.00**None** [rgb]0.67,0.13,1.00**and** [rgb]0.67,0.13,1.00**not** close˙prices˙df[rgb]0.40,0.40,0.40.empty: tickers [rgb]0.40,0.40,0.40= [col [rgb]0.00,0.50,0.00**for** col [rgb]0.67,0.13,1.00**in** close˙prices˙df[rgb]0.40,0.40,0.40. redcolumns] all˙optimized˙forecasts [rgb]0.40,0.40,0.40= −″ all˙optimized˙mses [rgb]0.40,0.40,0.40= −″ all˙best˙orders [rgb]0.40,0.40,0.40= −″ filename˙prefix [rgb]0.40,0.40,0.40= [rgb]0.73,0.13,0.13" [rgb]0.73,0.13,0.13optimized˙arima[rgb]0.73,0.13,0.13" max˙ar˙order [rgb]0.40,0.40,0.40= [rgb]0.40,0.40,0.403 max˙ma˙order [rgb]0.40,0.40,0.40= [rgb]0.40,0.40,0.403

[rgb]0.00,0.50,0.00**for** ticker [rgb]0.67,0.13,1.00**in** tickers: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13 -- Optimizing and Forecasting ARIMA for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13 ---[rgb]0.73,0.13,0.13") forecast˙data, test˙mse, best˙order [rgb]0.40,0.40,0.40= optimize˙arima˙and˙forecast(close˙prices˙df, ticker, max˙ar[rgb]0.40,0.40,0.40=max˙ar˙order, max˙ma[rgb]0.40,0.40,0.40=max˙ma˙order, filename˙prefix[rgb]0.40,0.40,0.40=filename˙prefix)

[rgb]0.00,0.50,0.00**if** forecast˙data [rgb]0.67,0.13,1.00**is** [rgb]0.67,0.13,1.00**not** [rgb]0.00,0.50,0.00**None** [rgb]0.67,0.13,1.00**and** test˙mse [rgb]0.67,0.13,1.00**is** [rgb]0.67,0.13,1.00**not** [rgb]0.00,0.50,0.00**None** [rgb]0.67,0.13,1.00**and** best˙order [rgb]0.67,0.13,1.00**is** [rgb]0.67,0.13,1.00**not** [rgb]0.00,0.50,0.00**None**: all˙optimized˙forecasts[ticker] [rgb]0.40,0.40,0.40= forecast˙data all˙optimized˙mses[ticker] [rgb]0.40,0.40,0.40= test˙mse all˙best˙orders[ticker] [rgb]0.40,0.40,0.40= best˙order [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13" [rgb]0.67,0.36,0.12**"n**[rgb]0.73,0.13,0.13Optimized ARIMA Forecast Data:[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00print(forecast˙data[rgb]0.40,0.40,0.40. redhead()) [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13" [rgb]0.73,0.13,0.13Mean Squared Error (Test Set): [rgb]0.64,0.35,0.47−test˙mse[rgb]0.64,0.35,0.47**:**[rgb]0.73,0.13,0.13. red2f[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13" [rgb]0.73,0 ARIMA Order: [rgb]0.64,0.35,0.47−best˙order[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**else**: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13" [rgb]0.73,0.13,0.13Optimization and forecasting failed for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13. red[rgb]0.73,0.13,0.13")

[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13" [rgb]0.67,0.36,0.12**"n**[rgb]0.73,0.13,0.13--- Summary of Optimized ARIMA Forecasts ---[rgb]0.73,0.13,0.13") [rgb]0.00,0.50,0.00**for** ticker [rgb]0.67,0.13,1.00**in** all˙best˙orders: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13" [rgb]0.73,0.13,0.13Ticker: red [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13, Best ARIMA Order: red [rgb]0.64,0.35,0.47−all˙best˙orders[ticker][rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13, Test MSE: red [rgb]0.64,0.35,0.47−all˙optimized˙mses[ticker][rgb]0.64,0.35,0.47**:**[rgb]0.73,0.13,0.13. red2f[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13")

[rgb]0.00,0.50,0.00**else**: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13" [rgb]0.73,0.13,0.13Error: The [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 DataFrame is not available. Please run the data loading and extraction part first.[rgb]0.73,0.13,0.13")

[commandchars=
{},codes*=]

— Optimizing and Forecasting ARIMA for GC=F — Optimized ARIMA order for GC=F: (0, 1, red 0) (AIC: 20411.21, BIC: 20417.64) Optimized ARIMA forecast plot for GC=F saved to optimized_arima_GC=F_optimized_arima_forecast.png AIC and BIC plot for ARIMA orders for GC=F saved to optimized_arima_GC=F_arima_aic_bic.png

Optimized ARIMA Forecast Data: GC=F_Forecast_ARIMA Date 2024-12-31 1549.199951 2025-01-01 1549.199951 2025-01-02 1549.199951 2025-01-03 1549.199951 2025-01-04 1549.199951 Mean Squared Error (Test Set): 224581.96 Best ARIMA Order: (0, 1, 0)

— Optimizing and Forecasting ARIMA for PA=F — Optimized ARIMA order for PA=F: (0, 1, red 0) (AIC: 20559.33, BIC: 20566.17) Optimized ARIMA forecast plot for PA=F saved to optimized_arima_PA=F_optimized_arima_forecast.png AIC and BIC plot for ARIMA orders for PA=F saved to optimized_arima_PA=F_arima_aic_bic.png

Optimized ARIMA Forecast Data: PA=F_Forecast_ARIMA Date 2024-12-31 1955.5 2025-01-01 1955. red5 2025-01-02 1955.5 2025-01-03 1955.5 2025-01-04 1955.5 Mean Squared Error (Test Set): 377806.61

Best ARIMA Order: (0, 1, 0)

— Optimizing and Forecasting ARIMA for PL=F — Optimized ARIMA order for PL=F: (0, 1, red 0) (AIC: 21117.57, BIC: 21129.51) Optimized ARIMA forecast plot for PL=F saved to optimized_arima_PL=F_optimized_arima_forecast.png AIC and BIC plot for ARIMA orders for PL=F saved to optimized_arima_PL=F_arima_aic_bic.png

Optimized ARIMA Forecast Data: PL=F_Forecast_ARIMA Date 2024-12-31 984.5 2025-01-01 984. red5 2025-01-02 984.5 2025-01-03 984.5 2025-01-04 984.5 Mean Squared Error (Test Set): 10247.16 Best ARIMA Order: (0, 1, 0)

— Optimizing and Forecasting ARIMA for SI=F — Optimized ARIMA order for SI=F: (0, 1, red 0) (AIC: 3480.54, BIC: 3493.81) Optimized ARIMA forecast plot for SI=F saved to optimized_arima_SI=F_optimized_arima_forecast.png AIC and BIC plot for ARIMA orders for SI=F saved to optimized_arima_SI=F_arima_aic_bic.png

Optimized ARIMA Forecast Data: SI=F_Forecast_ARIMA Date 2024-12-31 18.068001 2025-01-01 18.068001 2025-01-02 18.068001 2025-01-03 18.068001 2025-01-04 18.068001 Mean Squared Error (Test Set): 48.74 Best ARIMA Order: (0, 1, 0)

— Summary of Optimized ARIMA Forecasts — Ticker: GC=F, Best ARIMA Order: (0, 1, 0), Test MSE: 224581.96 Ticker: PA=F, Best ARIMA Order: (0, 1, 0), Test MSE: 377806.61 Ticker: PL=F, Best ARIMA Order: (0, 1, 0), Test MSE: 10247.16 Ticker: SI=F, Best ARIMA Order: (0, 1, 0), Test MSE: 48.74

[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackground, colframe=frametitle=]

{},codes*=] [rgb]0.24,0.48,0.48# *This finds the SMA* [rgb]0.00,0.50,0.00**def** [rgb]0.00,0.00,1.00forecast˙sma(df, ticker, window[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.4030, forecast˙years[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.405, filename[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00**None**):

red

[rgb]0.00,0.50,0.00**if** ticker [rgb]0.67,0.13,1.00**not** [rgb]0.67,0.13,1.00**in** df[rgb]0.40,0.40,0.40.columns:

red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13Error: Ticker [rgb]0.73,0.13,0.13’[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13 not found in the DataFrame.[rgb]0.73,0.13,0.13”) [rgb]0.00,0.50,0.00**return**

ticker˙data [rgb]0.40,0.40,0.40= df[ticker][rgb]0.40,0.40,0.40.dropna() [rgb]0.00,0.50,0.00**if** ticker˙data[rgb]0.40,0.40,0.40.empty: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13Erro

red No valid data found for ticker [rgb]0.73,0.13,0.13’[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13’[rg

red[rgb]0.73,0.13,0.13”) [rgb]0.00,0.50,0.00**return**

sma [rgb]0.40,0.40,0.40= ticker˙data[rgb]0.40,0.40,0.40.rolling(window[rgb]0.40,0.40,0.40=window)[rgb]0.40,0.40,0.40.

redmean() last˙sma˙value [rgb]0.40,0.40,0.40= sma[rgb]0.40,0.40,0.40.

rediloc[[rgb]0.40,0.40,0.40-[rgb]0.40,0.40,0.401] last˙date [rgb]0.40,0.40,0.40= ticker˙data[rgb]0.40,0.40,0.40.index[[rgb]0.40,0.40,0.40-[rgb]0.40,0.40,0.401] forecast˙dates [rgb]0.40,0.40,0.40= pd[rgb]0.40,0.40,0.40.date˙range(start[rgb]0.40,0.40,0.40=last˙date, periods[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.40365 [rgb]0.40,0.40,0.40* forecast˙years) [rgb]0.40,0.40,0.40+ [rgb]0.40,0.40,0.401, freq[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13D[rgb]0.73,0.13,0.13’) forecast [rgb]0.40,0.40,0.40= pd[rgb]0.40,0.40,0.40.Series([last˙sma˙value] [rgb]0.40,0.40,0.40* [rgb]0.00,0.50,0.00len(forecast˙dates), index[rgb]0.40,0.40,0.40=forecast˙dates)

plt[rgb]0.40,0.40,0.40.figure(figsize[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.4012, [rgb]0.40,0.40,0.406)) plt[rgb]0.40,0.40,0.40.plot(ticker˙data[rgb]0.40,0.40,0.40.index, ticker˙data, label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13’[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0. Close Price[rgb]0.73,0.13,0.13’) plt[rgb]0.40,0.40,0.40.plot(sma[rgb]0.40,0.40,0.40.index, sma,

red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13SMA ([rgb]0.64,0.35,0.47−window[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13 days)[rgb]0.73,0.13,0.13’) plt[rgb]0.40,0.40,0.40.plot(forecast[rgb]0.40,0.40,0.40.index, forecast, label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13’[rgb]0.64,0.35,0.47−forecast˙years[rgb]0.64,0.35,0.47″[rgb]0.7 Year SMA Forecast[rgb]0.73,0.13,0.13’, linestyle[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13- -[rgb]0.73,0.13,0.13’) plt[rgb]0.40,0.40,0.40.xlabel([rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13’) plt[rgb]0.40,0.40,0.40.ylabel([rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Price[rgb]0.73,0.13,0.13’) plt[rgb]0.40,0.40,0.40.title([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Simple Moving Average Forecast for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13’) plt[rgb]0.40,0.40,0.40.

redlegend() plt[rgb]0.40,0.40,0.40.grid([rgb]0.00,0.50,0.00**True**) plt[rgb]0.40,0.40,0.40.tight˙layout()

[rgb]0.00,0.50,0.00**if** filename [rgb]0.67,0.13,1.00**is** [rgb]0.00,0.50,0.00**None**: filename [rgb]0.40,0.40,0.40= [rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13’[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13

redpng[rgb]0.73,0.13,0.13’ plt[rgb]0.40,0.40,0.40.savefig(filename) [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0
forecast plot for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47”[rgb]0.73,0.13,0.13 saved to
[rgb]0.64,0.35,0.47−filename[rgb]0.64,0.35,0.47”[rgb]0.73,0.13,0.13”) plt[rgb]0.40,0.40,0.40.close()
[rgb]0.00,0.50,0.00**if** [rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13’
[rgb]0.67,0.13,1.00**in** [rgb]0.00,0.50,0.00locals() [rgb]0.67,0.13,1.00**and** close˙prices˙df
[rgb]0.67,0.13,1.00**is** [rgb]0.67,0.13,1.00**not** [rgb]0.00,0.50,0.00**None** [rgb]0.67,0.13,1.00**and**
[rgb]0.67,0.13,1.00**not** close˙prices˙df[rgb]0.40,0.40,0.40.empty: tickers [rgb]0.40,0.40,0.40=
[col [rgb]0.00,0.50,0.00**for** col [rgb]0.67,0.13,1.00**in** close˙prices˙df[rgb]0.40,0.40,0.40.
redcolumns] [rgb]0.00,0.50,0.00**for** ticker [rgb]0.67,0.13,1.00**in** tickers:
[rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13”[rgb]0.67,0.36,0.12**“n**[rgb]0.73,0.13,0.13-
-- SMA Forecasting for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47”[rgb]0.73,0.13,0.13
---[rgb]0.73,0.13,0.13”) forecast˙sma(close˙prices˙df, ticker) [rgb]0.00,0.50,0.00**else**:
red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13Error:
[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13 not found.
Run data loading first.[rgb]0.73,0.13,0.13”)
[commandchars=
{},codes*=]
— SMA Forecasting for GC=F — SMA forecast plot for GC=F saved to
GC=F_sma_forecast_window_30.png
— SMA Forecasting for PA=F — SMA forecast plot for PA=F saved to
PA=F_sma_forecast_window_30.png
— SMA Forecasting for PL=F — SMA forecast plot for PL=F saved to
PL=F_sma_forecast_window_30.png
— SMA Forecasting for SI=F — SMA forecast plot for SI=F saved to SI=F_sma_forecast_window_30.
redpng
[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackground, col-
fra[rgb]0.00,0.00,1.00**from**[rgb]0.00,0.00,1.00**statsmodels**[rgb]0.00,0.00,1.00**.**
{},codes*=] [rgb]0.00,0.50,0.00**from** [rgb]0.00,0.00,1.00**statsmodels**[rgb]0.00,0.00,1.00**.**
red[rgb]0.00,0.00,1.00**tsa**[rgb]0.00,0.00,1.00**.**[rgb]0.00,0.00,1.00**holtwinters**
[rgb]0.00,0.50,0.00**import** SimpleExpSmoothing, Holt, ExponentialSmoothing
[rgb]0.24,0.48,0.48*# This finds the SEMA, EMA, and Holt fits*
[rgb]0.00,0.50,0.00**def** [rgb]0.00,0.00,1.00forecast˙exponential˙smoothing(df,
red ticker, forecast˙years[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.405, file-
name˙prefix[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00**None**):
[rgb]0.00,0.50,0.00**if** ticker [rgb]0.67,0.13,1.00**not** [rgb]0.67,0.13,1.00**in** df[rgb]0.40,0.40,0.40.columns:
red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13Error: Ticker
[rgb]0.73,0.13,0.13’[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47”[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13
not found in the DataFrame.[rgb]0.73,0.13,0.13”) [rgb]0.00,0.50,0.00**return**
ticker˙data [rgb]0.40,0.40,0.40= df[ticker][rgb]0.40,0.40,0.40.dropna() [rgb]0.00,0.50,0.00**if**
ticker˙data[rgb]0.40,0.40,0.40.empty: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13Erro
red No valid data found for ticker [rgb]0.73,0.13,0.13’[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47”[rgb]0.73,0.13,0.13’[rg
red[rgb]0.73,0.13,0.13”) [rgb]0.00,0.50,0.00**return**
history [rgb]0.40,0.40,0.40= ticker˙data[rgb]0.40,0.40,0.40.values train˙size [rgb]0.40,0.40,0.40=
[rgb]0.00,0.50,0.00int([rgb]0.00,0.50,0.00len(history) [rgb]0.40,0.40,0.40* [rgb]0.40,0.40,0.400.8) train,
red test [rgb]0.40,0.40,0.40= history[:train˙size], history[train˙size:] dates [rgb]0.40,0.40,0.40=
ticker˙data[rgb]0.40,0.40,0.40.index train˙dates [rgb]0.40,0.40,0.40= dates[:train˙size] test˙dates
[rgb]0.40,0.40,0.40= dates[train˙size:] forecast˙index [rgb]0.40,0.40,0.40= pd[rgb]0.40,0.40,0.40.
reddate˙range(start[rgb]0.40,0.40,0.40=dates[[rgb]0.40,0.40,0.40-[rgb]0.40,0.40,0.401], peri-
ods[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.40365 [rgb]0.40,0.40,0.40* forecast˙years) [rgb]0.40,0.40,0.40+
[rgb]0.40,0.40,0.401, freq[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13D[rgb]0.73,0.13,0.13’)
fit˙ses [rgb]0.40,0.40,0.40= SimpleExpSmoothing(train, initialization˙method[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13”[rg
redfit(smoothing˙level[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.400.2, optimized[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00**False**
forecast˙ses [rgb]0.40,0.40,0.40= fit˙ses[rgb]0.40,0.40,0.40.forecast([rgb]0.00,0.50,0.00len(forecast˙index)
[rgb]0.40,0.40,0.40- [rgb]0.40,0.40,0.401)
plt[rgb]0.40,0.40,0.40.figure(figsize[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.4012,
red [rgb]0.40,0.40,0.406)) plt[rgb]0.40,0.40,0.40.plot(train˙dates, train,
red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Training
Data[rgb]0.73,0.13,0.13’) plt[rgb]0.40,0.40,0.40.plot(test˙dates, test, la-

bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Testing Data[rgb]0.73,0.13,0.13’)
plt[rgb]0.40,0.40,0.40.plot(forecast˙index[[rgb]0.40,0.40,0.401:], forecast˙ses, la-
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Simple Exponential Smoothing
Forecast[rgb]0.73,0.13,0.13’, linestyle[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13--
[rgb]0.73,0.13,0.13’) plt[rgb]0.40,0.40,0.40.xlabel([rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13’)
plt[rgb]0.40,0.40,0.40.ylabel([rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Price[rgb]0.73,0.13,0.13’)
plt[rgb]0.40,0.40,0.40.title([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Simple Expo-
nential Smoothing Forecast for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13’)
plt[rgb]0.40,0.40,0.40.legend() plt[rgb]0.40,0.40,0.40.grid([rgb]0.00,0.50,0.00**True**) plt[rgb]0.40,0.40,0.40.
redtight˙layout() [rgb]0.00,0.50,0.00**if** filename˙prefix: filename [rgb]0.40,0.40,0.40=
[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13”[rgb]0.64,0.35,0.47−filename˙prefix[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13’[rgb]0.64,0
redpng[rgb]0.73,0.13,0.13” [rgb]0.00,0.50,0.00**else**: filename [rgb]0.40,0.40,0.40=
[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13”[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13’ses˙forecast.
redpng[rgb]0.73,0.13,0.13” plt[rgb]0.40,0.40,0.40.savefig(filename) [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,
Exponential Smoothing forecast plot for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13
saved to [rgb]0.64,0.35,0.47−filename[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13”) plt[rgb]0.40,0.40,0.40.
redclose()
fit˙holt [rgb]0.40,0.40,0.40= Holt(train, initialization˙method[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13
redfit(smoothing˙level[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.400.8, smoothing˙trend[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.4
red2, optimized[rgb]0.40,0.40,0.40=[rgb]0.00,0.50,0.00**False**) forecast˙holt [rgb]0.40,0.40,0.40=
fit˙holt[rgb]0.40,0.40,0.40.forecast([rgb]0.00,0.50,0.00len(forecast˙index) [rgb]0.40,0.40,0.40-
[rgb]0.40,0.40,0.401)
plt[rgb]0.40,0.40,0.40.figure(figsize[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.4012,
red [rgb]0.40,0.40,0.406)) plt[rgb]0.40,0.40,0.40.plot(train˙dates, train,
red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Training
Data[rgb]0.73,0.13,0.13’) plt[rgb]0.40,0.40,0.40.plot(test˙dates, test, la-
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Testing Data[rgb]0.73,0.13,0.13’)
plt[rgb]0.40,0.40,0.40.plot(forecast˙index[[rgb]0.40,0.40,0.401:], forecast˙holt, la-
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13Holt[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13s
Linear Trend Forecast[rgb]0.73,0.13,0.13”, linestyle[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13-
-[rgb]0.73,0.13,0.13’) plt[rgb]0.40,0.40,0.40.xlabel([rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13’)
plt[rgb]0.40,0.40,0.40.ylabel([rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Price[rgb]0.73,0.13,0.13’)
plt[rgb]0.40,0.40,0.40.title([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13Holt[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0
Linear Trend Forecast for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13”)
plt[rgb]0.40,0.40,0.40.legend() plt[rgb]0.40,0.40,0.40.grid([rgb]0.00,0.50,0.00**True**) plt[rgb]0.40,0.40,0.40.
redtight˙layout() [rgb]0.00,0.50,0.00**if** filename˙prefix: filename [rgb]0.40,0.40,0.40=
[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13”[rgb]0.64,0.35,0.47−filename˙prefix[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13’[rgb]0.64,0
redpng[rgb]0.73,0.13,0.13” [rgb]0.00,0.50,0.00**else**: filename [rgb]0.40,0.40,0.40=
[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13”[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13’holt˙forecast.
redpng[rgb]0.73,0.13,0.13” plt[rgb]0.40,0.40,0.40.savefig(filename) [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,
Linear Trend forecast plot for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13 saved to
[rgb]0.64,0.35,0.47−filename[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13”) plt[rgb]0.40,0.40,0.40.close()
[rgb]0.00,0.50,0.00**try**: fit˙hw˙add [rgb]0.40,0.40,0.40= ExponentialSmoothing(train,
seasonal˙periods[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.40365, trend[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13
red seasonal[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13add[rgb]0.73,0.13,0.13’,
initialization˙method[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13estimated[rgb]0.73,0.13,0.13”)[rgb]0.40,0.4
redfit() forecast˙hw˙add [rgb]0.40,0.40,0.40= fit˙hw˙add[rgb]0.40,0.40,0.40.
redforecast([rgb]0.00,0.50,0.00len(forecast˙index) [rgb]0.40,0.40,0.40- [rgb]0.40,0.40,0.401)
plt[rgb]0.40,0.40,0.40.figure(figsize[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.4012,
red [rgb]0.40,0.40,0.406)) plt[rgb]0.40,0.40,0.40.plot(train˙dates, train,
red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Training
Data[rgb]0.73,0.13,0.13’) plt[rgb]0.40,0.40,0.40.plot(test˙dates, test, la-
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Testing Data[rgb]0.73,0.13,0.13’)
plt[rgb]0.40,0.40,0.40.plot(forecast˙index[[rgb]0.40,0.40,0.401:], forecast˙hw˙add,
red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13”[rgb]0.73,0.13,0.13Holt-
Winters[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13 Additive Forecast[rgb]0.73,0.13,0.13”,
red linestyle[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13--[rgb]0.73,0.13,0.13’)
plt[rgb]0.40,0.40,0.40.xlabel([rgb]0.73,0.13,0.13’[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13’)

```
plt[rgb]0.40,0.40,0.40.ylabel([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Price[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.title([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Holt-
Winters[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 Additive Forecast for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,
plt[rgb]0.40,0.40,0.40.legend() plt[rgb]0.40,0.40,0.40.grid([rgb]0.00,0.50,0.00True) plt[rgb]0.40,0.40,0.40.
    redtight˙layout() [rgb]0.00,0.50,0.00if filename˙prefix: filename [rgb]0.40,0.40,0.40=
[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.64,0.35,0.47−filename˙prefix[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13˙[rgb]0.64,0
    redpng[rgb]0.73,0.13,0.13" [rgb]0.00,0.50,0.00else: filename [rgb]0.40,0.40,0.40=
[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13˙holt˙winters˙add˙for
    redpng[rgb]0.73,0.13,0.13" plt[rgb]0.40,0.40,0.40.savefig(filename) [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,
Winters[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 Additive forecast plot for
[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13 saved to
[rgb]0.64,0.35,0.47−filename[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13") plt[rgb]0.40,0.40,0.40.close()
    fit˙hw˙mult [rgb]0.40,0.40,0.40= ExponentialSmoothing(train, seasonal˙periods[rgb]0.40,0.40,0.40=[rgb]0.40,0.40,0.4030
    red trend[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13mul[rgb]0.73,0.13,0.13',
    red seasonal[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13mul[rgb]0.73,0.13,0.13',
initialization˙method[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13estimated[rgb]0.73,0.13,0.13")[rgb]0.40,0.40
    redfit() forecast˙hw˙mult [rgb]0.40,0.40,0.40= fit˙hw˙mult[rgb]0.40,0.40,0.40.
    redforecast([rgb]0.00,0.50,0.00len(forecast˙index) [rgb]0.40,0.40,0.40- [rgb]0.40,0.40,0.401)
    plt[rgb]0.40,0.40,0.40.figure(figsize[rgb]0.40,0.40,0.40=([rgb]0.40,0.40,0.4012,
    red [rgb]0.40,0.40,0.406)) plt[rgb]0.40,0.40,0.40.plot(train˙dates, train,
    red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Training
Data[rgb]0.73,0.13,0.13') plt[rgb]0.40,0.40,0.40.plot(test˙dates, test, la-
bel[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Testing Data[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.plot(forecast˙index[[rgb]0.40,0.40,0.401:], forecast˙hw˙mult,
    red label[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Holt-
Winters[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 Multiplicative Forecast[rgb]0.73,0.13,0.13",
    red linestyle[rgb]0.40,0.40,0.40=[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13--[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.xlabel([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Date[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.ylabel([rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13Price[rgb]0.73,0.13,0.13')
plt[rgb]0.40,0.40,0.40.title([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Holt-
Winters[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 Multiplicative Forecast for
[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13") plt[rgb]0.40,0.40,0.40.
    redlegend() plt[rgb]0.40,0.40,0.40.grid([rgb]0.00,0.50,0.00True) plt[rgb]0.40,0.40,0.40.
    redtight˙layout() [rgb]0.00,0.50,0.00if filename˙prefix: filename [rgb]0.40,0.40,0.40=
[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.64,0.35,0.47−filename˙prefix[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13˙[rgb]0.64,0
    redpng[rgb]0.73,0.13,0.13" [rgb]0.00,0.50,0.00else: filename [rgb]0.40,0.40,0.40=
[rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13˙holt˙winters˙mult˙fo
    redpng[rgb]0.73,0.13,0.13" plt[rgb]0.40,0.40,0.40.savefig(filename) [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,
Winters[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 Multiplicative forecast plot
for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13 saved to
[rgb]0.64,0.35,0.47−filename[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13") plt[rgb]0.40,0.40,0.40.close()
    [rgb]0.00,0.50,0.00except [rgb]0.80,0.25,0.22Exception [rgb]0.00,0.50,0.00as e:
    red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error
during Holt-Winters for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13:
[rgb]0.64,0.35,0.47−e[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13")
    [rgb]0.00,0.50,0.00if [rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13'
[rgb]0.67,0.13,1.00in [rgb]0.00,0.50,0.00locals() [rgb]0.67,0.13,1.00and close˙prices˙df [rgb]0.67,0.13,1.00is
[rgb]0.67,0.13,1.00not [rgb]0.00,0.50,0.00None [rgb]0.67,0.13,1.00and [rgb]0.67,0.13,1.00not
close˙prices˙df[rgb]0.40,0.40,0.40.empty: tickers [rgb]0.40,0.40,0.40= [col [rgb]0.00,0.50,0.00for col
[rgb]0.67,0.13,1.00in close˙prices˙df[rgb]0.40,0.40,0.40.columns] filename˙prefix˙es [rgb]0.40,0.40,0.40=
[rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13exponential˙smoothing[rgb]0.73,0.13,0.13" [rgb]0.00,0.50,0.00for
ticker [rgb]0.67,0.13,1.00in tickers: [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13f[rgb]0.73,0.13,0.13"[rgb]0.67,0.36,0.12"n[rgb
-- Exponential Smoothing Forecasts for [rgb]0.64,0.35,0.47−ticker[rgb]0.64,0.35,0.47″[rgb]0.73,0.13,0.13
---[rgb]0.73,0.13,0.13") forecast˙exponential˙smoothing(close˙prices˙df, ticker,
    red filename˙prefix[rgb]0.40,0.40,0.40=filename˙prefix˙es) [rgb]0.00,0.50,0.00else:
    red [rgb]0.00,0.50,0.00print([rgb]0.73,0.13,0.13"[rgb]0.73,0.13,0.13Error:
[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13close˙prices˙df[rgb]0.73,0.13,0.13'[rgb]0.73,0.13,0.13 not found.
Run data loading first.[rgb]0.73,0.13,0.13")
```

[commandchars=
{},codes*=]
— Exponential Smoothing Forecasts for GC=F — Simple Exponential Smoothing forecast plot for GC=F saved to exponential_smoothing_GC=F_ses_forecast.png Holt's Linear Trend forecast plot for GC=F saved to exponential_smoothing_GC=F_holt_forecast.png Holt-Winters' Additive forecast plot for GC=F saved to exponential_smoothing_GC=F_holt_winters_add_forecast.png Holt-Winters' Multiplicative forecast plot for GC=F saved to exponential_smoothing_GC=F_holt_winters_mult_forecast.png
— Exponential Smoothing Forecasts for PA=F — Simple Exponential Smoothing forecast plot for PA=F saved to exponential_smoothing_PA=F_ses_forecast.png Holt's Linear Trend forecast plot for PA=F saved to exponential_smoothing_PA=F_holt_forecast.png Holt-Winters' Additive forecast plot for PA=F saved to exponential_smoothing_PA=F_holt_winters_add_forecast.png Holt-Winters' Multiplicative forecast plot for PA=F saved to exponential_smoothing_PA=F_holt_winters_mult_forecast.png
— Exponential Smoothing Forecasts for PL=F — Simple Exponential Smoothing forecast plot for PL=F saved to exponential_smoothing_PL=F_ses_forecast.png Holt's Linear Trend forecast plot for PL=F saved to exponential_smoothing_PL=F_holt_forecast.png Holt-Winters' Additive forecast plot for PL=F saved to exponential_smoothing_PL=F_holt_winters_add_forecast.png Holt-Winters' Multiplicative forecast plot for PL=F saved to exponential_smoothing_PL=F_holt_winters_mult_forecast.png
— Exponential Smoothing Forecasts for SI=F — Simple Exponential Smoothing forecast plot for SI=F saved to exponential_smoothing_SI=F_ses_forecast.png Holt's Linear Trend forecast plot for SI=F saved to exponential_smoothing_SI=F_holt_forecast.png Holt-Winters' Additive forecast plot for SI=F saved to exponential_smoothing_SI=F_holt_winters_add_forecast.png Holt-Winters' Multiplicative forecast plot for SI=F saved to exponential_smoothing_SI=F_holt_winters_mult_forecast.png
[breakable, size=fbox, boxrule=1pt, pad at break*=1mm,colback=cellbackground, colframe=cellborder]
[commandchars=
{},codes*=]