

Oil Price Prediction

Team 6

Karthik

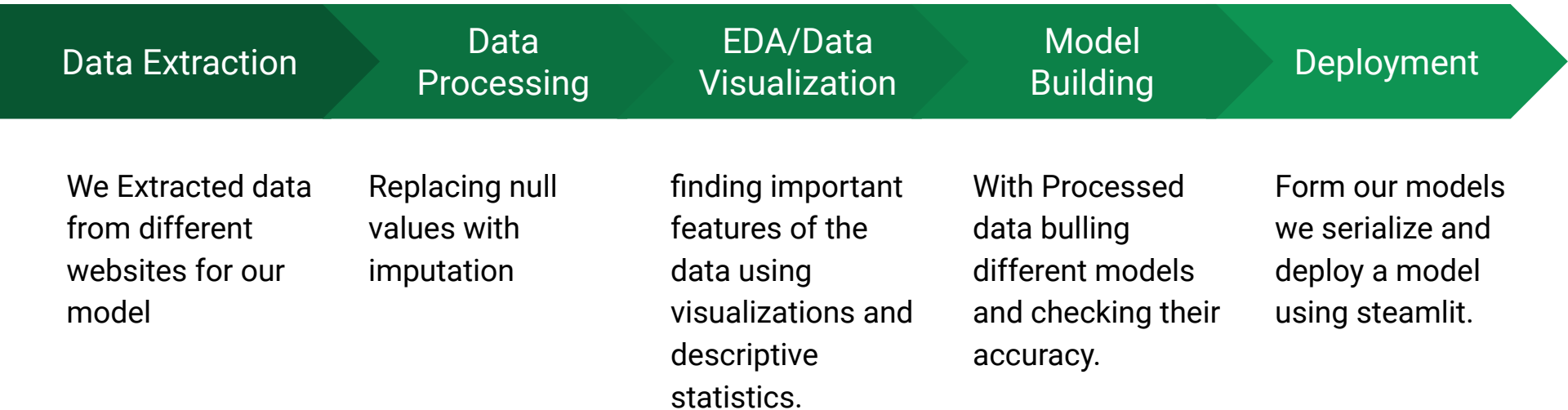
07/02/2023

Business Problem:

Objective:

Oil is a product that goes completely in a different direction for a single market event as the oil prices are rarely based on real-time data, instead, it is driven by externalities making our attempt to forecast it even more challenging. As the economy will be highly affected by oil prices, our model will help to understand the pattern in prices to help the customers and businesses to make smart decisions.

Project Architecture / Project Flow



Exploratory Data Analysis (EDA) and Feature Engineering

Data set details

The Data Set has two columns

1. Date
2. Price

Total rows are 9345

There are 6 null values in our data we have replaced it with mean of the data.

	Date	Price
0	02-01-1986	25.56
1	03-01-1986	26.00
2	06-01-1986	26.53
3	07-01-1986	25.85
4	08-01-1986	25.87
...
9340	15-12-2022	76.11
9341	16-12-2022	74.29
9342	19-12-2022	75.19
9343	20-12-2022	76.09
9344	21-12-2022	78.29
9345 rows × 2 columns		

Exploratory Data Analysis (EDA)

Descriptive Statistics of the data

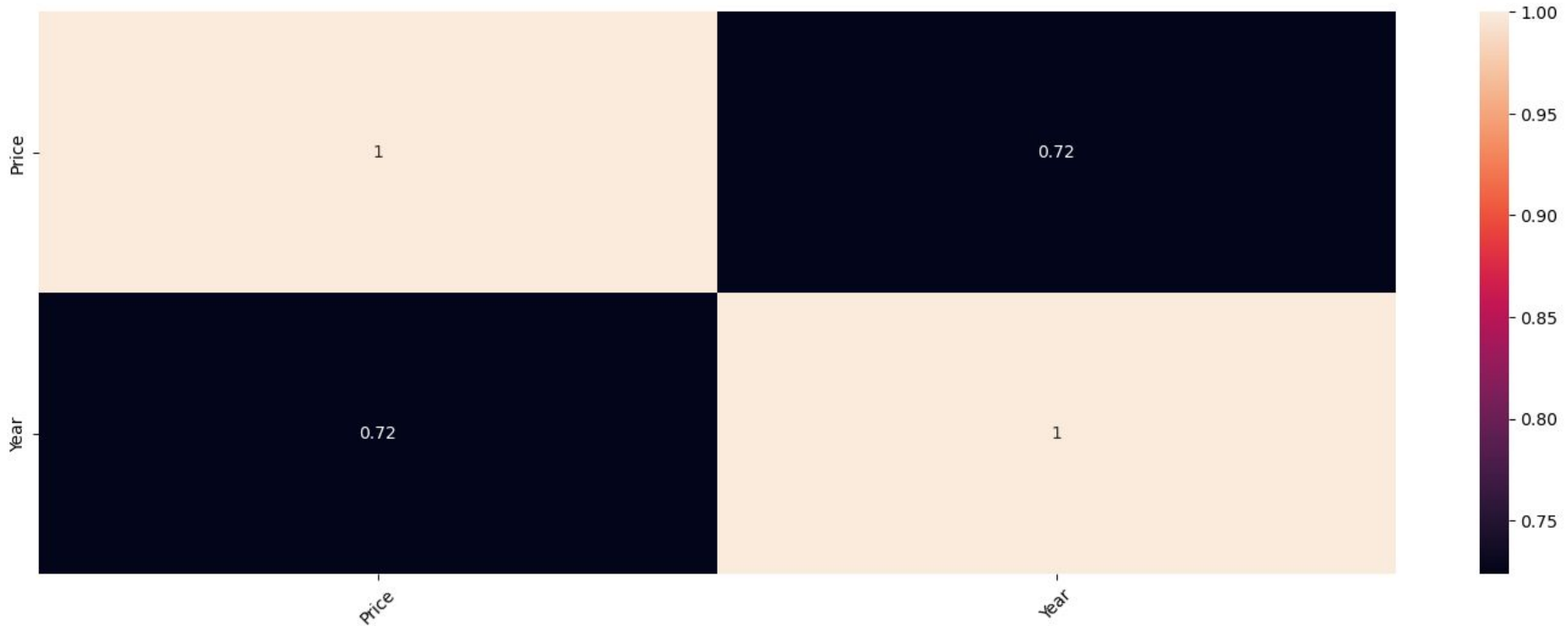
Here are the statistical measures for price column in the data.

There are 6 null values in our data we have replaced it with mean of the data.(Imputation)

Price	
count	9339.000000
mean	46.028511
std	29.515461
min	9.060000
25%	20.000000
50%	36.060000
75%	66.910000
max	145.310000

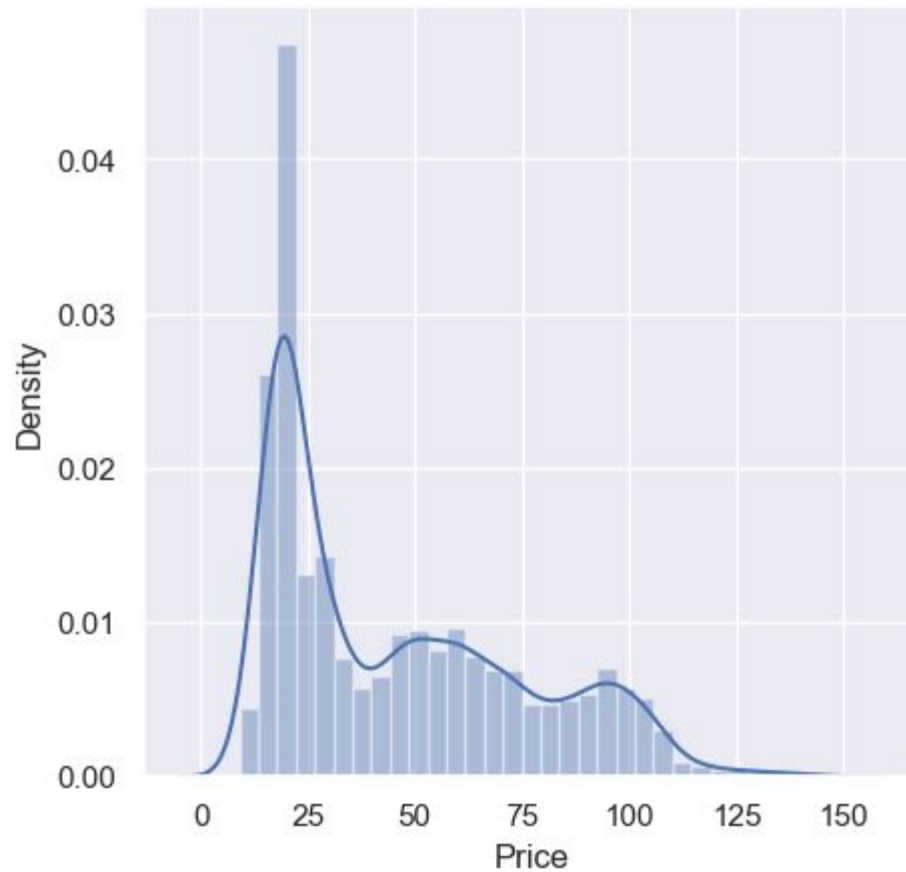
Exploratory Data Analysis (EDA)

Correlation Matrix



There is a good correlation between price and year.

Exploratory Data Analysis (EDA)



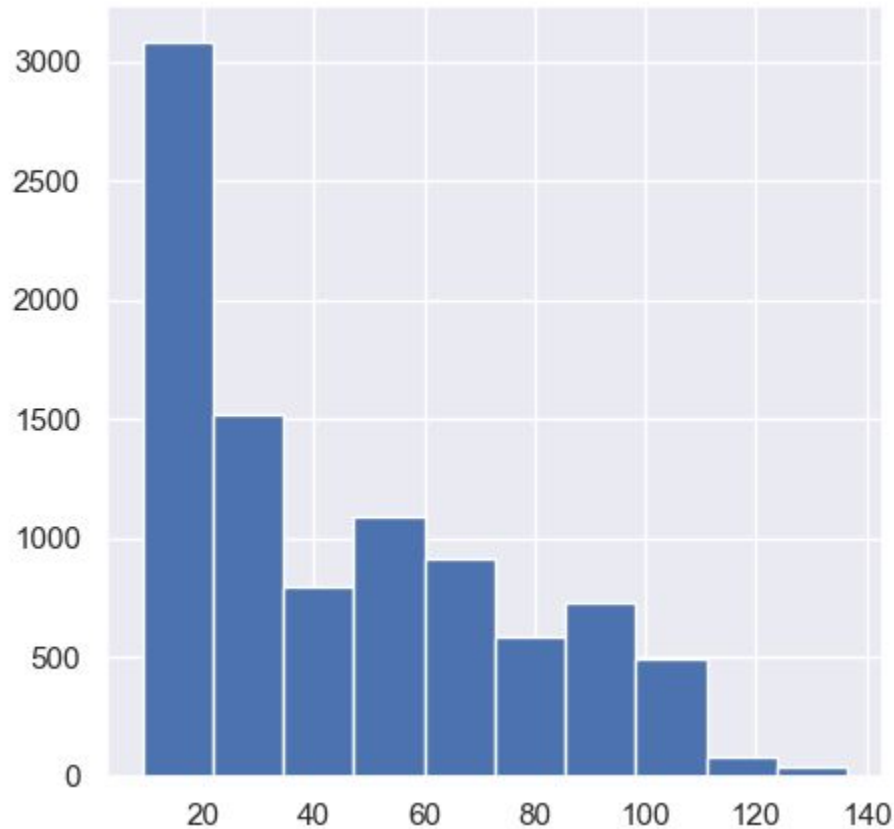
1. +ve skew/Right Skew Distribution



Which means the data is lower bound.

So the price is lower bound and rarely reaches high prices.

Exploratory Data Analysis (EDA)

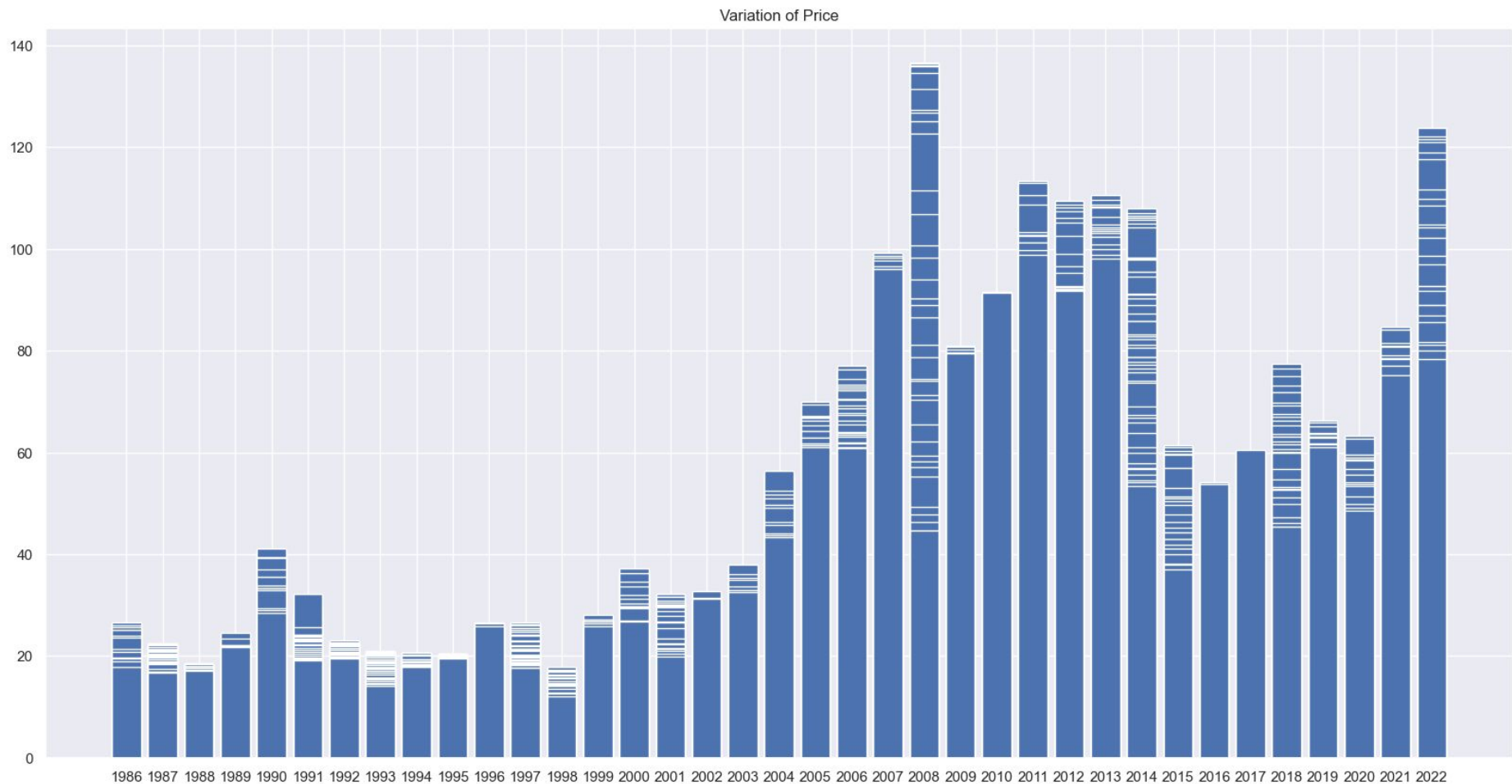


Price histogram

Which means the data is lower bound.

So the price is lower bound and rarely reaches high prices.

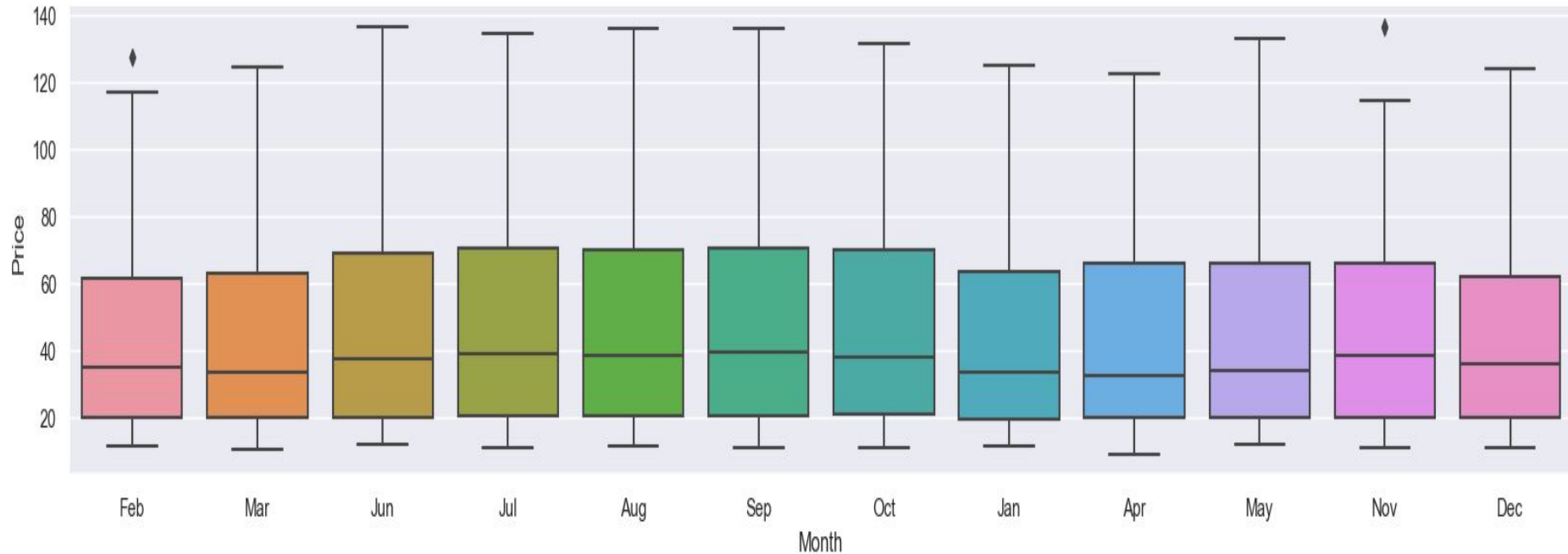
Exploratory Data Analysis (EDA)



Price distribution each year and not so often we see high prices. So the price is lower bound

Exploratory Data Analysis (EDA)

Price vs Month



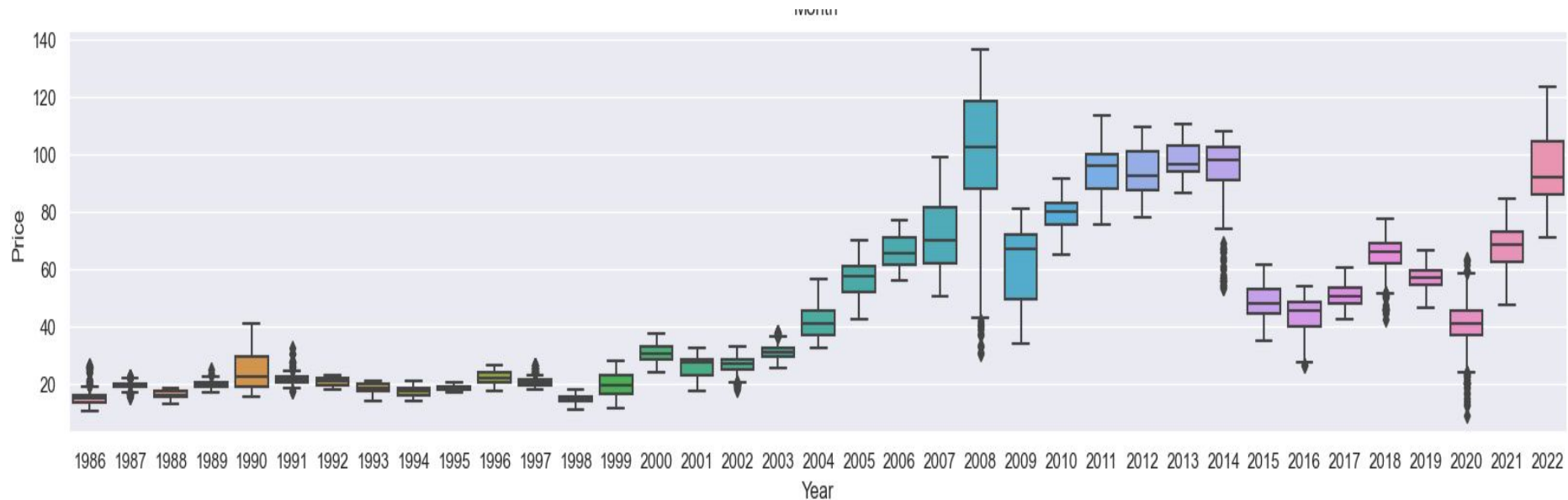
We can see the price distribution is pretty much same irrespective of month



So Month is not a good feature to predict price

Exploratory Data Analysis (EDA)

Price vs Year



We can see the price is increasing with each year holistically



So year is a good feature to predict price and even correlation coefficient is also high

Time Series Decomposition

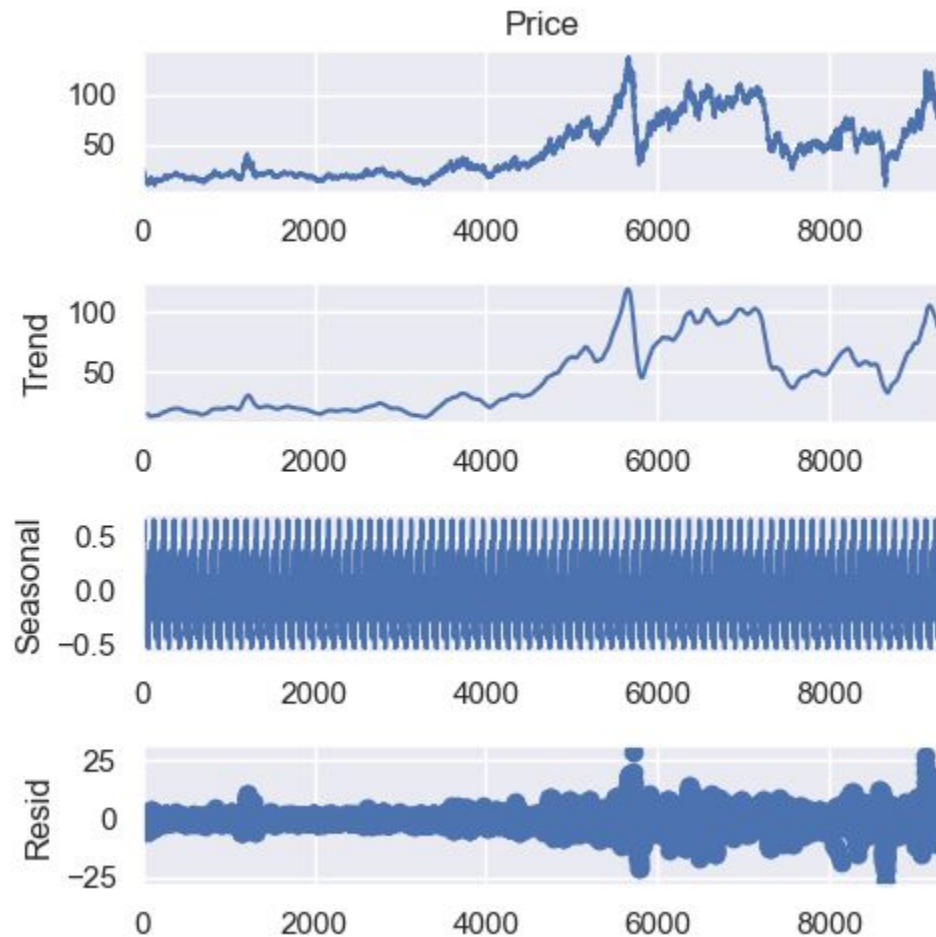
Time Series Decomposition

Additive:

Using time series decomposition we split the data into combination of :

Observed: Trend + Seasonal + Residual

Exploratory Data Analysis (EDA)



Not a Prefect upward trend

Overall upward trend

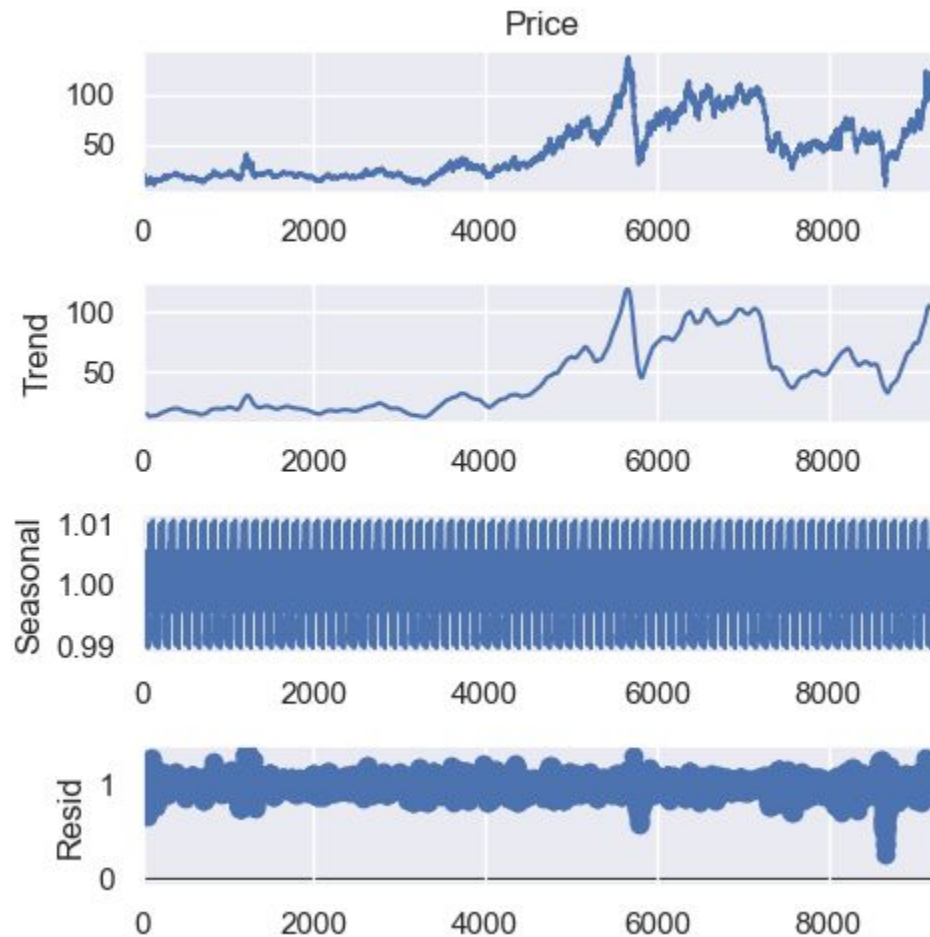
Time Series Decomposition

Multiplicative:

Using time series decomposition we split the data into combination of :

Observed: Trend x Seasonal x Residual

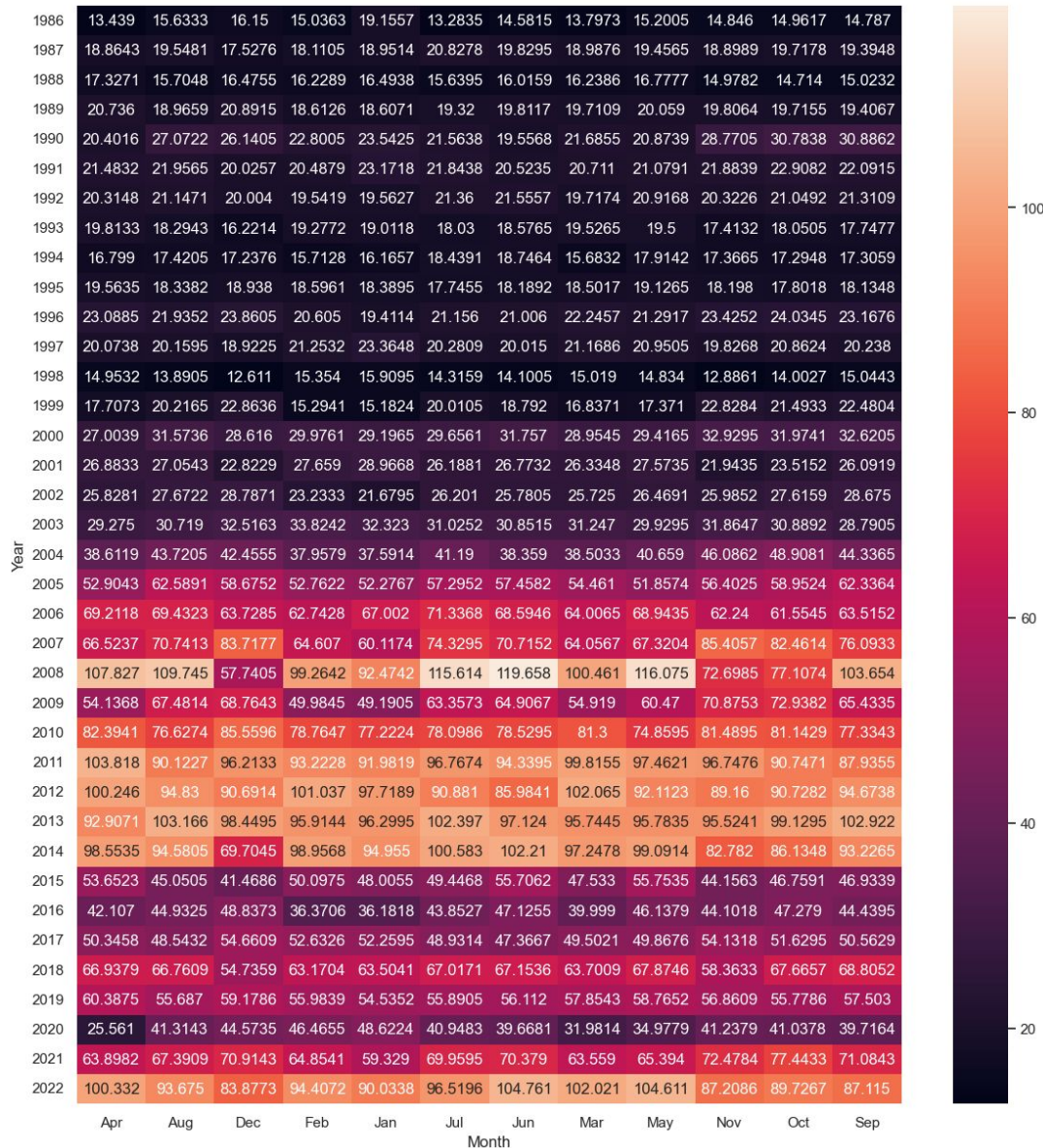
Exploratory Data Analysis (EDA)



Not a Prefect upward trend

Overall upward trend

Exploratory Data Analysis (EDA)



Exploratory Data Analysis (EDA)



Price vs date of our data

- ❖ Holistically Price increases with date
- ❖ Atomistically price fluctuates a lot.
- ❖ Year has high correlation but it doesn't change rapidly

Challenges faced?

1. Data Extraction from different websites
1. Not able to find great features for model as there is only one feature and its sub features didn't gave any high insights

Next Meeting Agenda

1. Stationary data of the data (Applying transformations to convert non-stationary data to stationary data)
2. Autocorrelation plot (ACF and PACF Plot)
3. Model Building

Differencing



Non - Stationary Data	Stationary Data
Mean $\rightarrow \mu \neq 0$	Mean $\rightarrow \mu = 0$
Variance $\rightarrow v \neq 0$	Variance $\rightarrow v = 0$
There is a trend	There is no trend

Adfuller test is used to check the stationarity of the data

```
#Null hypothesis = non-stationary; Alternate hypothesis = Stationary
#if p-value > 0.05 - accept null hypothesis(data is non stationary) else accept alternate hypothesis(data is stationary)
def check_adfuller(df3):
    result = adfuller(df3,autolag='AIC')
    print('Test statistic:',result[0])
    print('p-value:',result[1])
    print('Critical Values:',result[4])
def check_mean_std(data3):
    df3_log = np.log(df3)
    moving_avg = df3_log.rolling(12).mean()
    moving_std = df3_log.rolling(12).std()
    plt.figure(figsize=(30,15))
    orig = plt.plot(data3,color = 'red',label = 'Original')
    mean = plt.plot(moving_avg, color = 'black',label= 'Rolling Mean')
    std = plt.plot(moving_std,color = 'green',label= 'Rolling Std')
    plt.xlabel("Year")
    plt.ylabel("Price")
    plt.title('Rolling Mean & Standard Deviation')
    plt.legend()
    plt.show()
check_mean_std(df3)
check_adfuller(df3.Price)
```

Test statistic: -2.331804763291683
p-value: 0.1619153449554605
Critical Values: {'1%': -3.431053874661101, '5%': -2.861851068123899, '10%':
-2.566935576484507}

Here, the p value is greater than 0.05 (Critical value). So, Accept Null hypothesis i.e., The data is **non stationary**.

Applying Mathematical transformation on the data in order to convert it into stationary data

1. Linear Transform
2. Square root Transform
3. Log Transform
4. Exponential Transform
5. Cube root Transform
6. First order differencing Transform

Calculating the ADF Statistic for different transforms

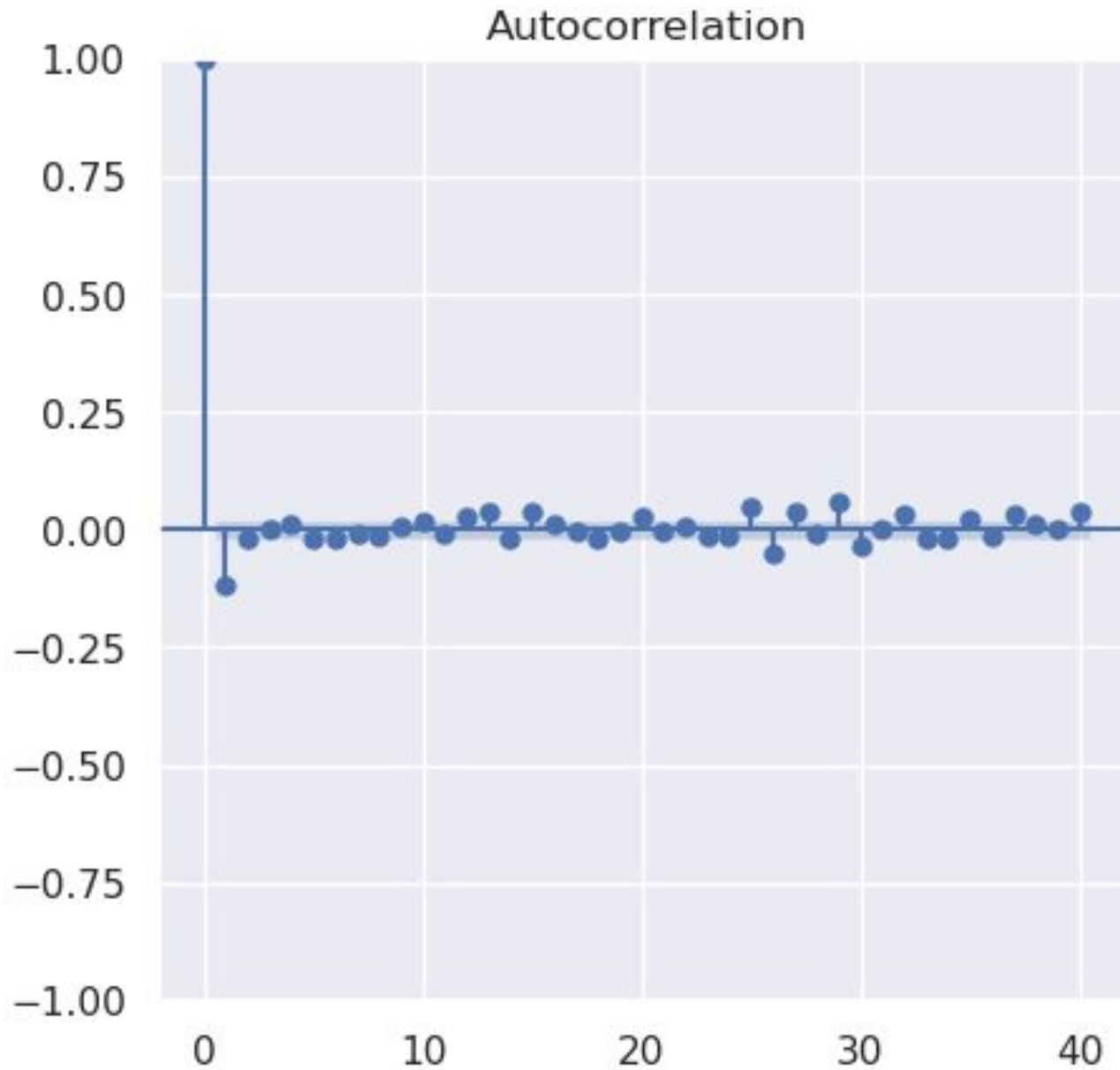
Transform Name	Critical Value (0.05)	ADF Statistic
Linear	-2.862	-2.331805
Square root	-2.862	-2.076627
Log	-2.862	-2.051004
Exponential	-2.862	-14.630666
Cube root	-2.862	-2.043514
First Order Differencing	-2.862	-13.864850

Adfuller Statistic value is less than the critical value for Exponential and First Order Differencing.

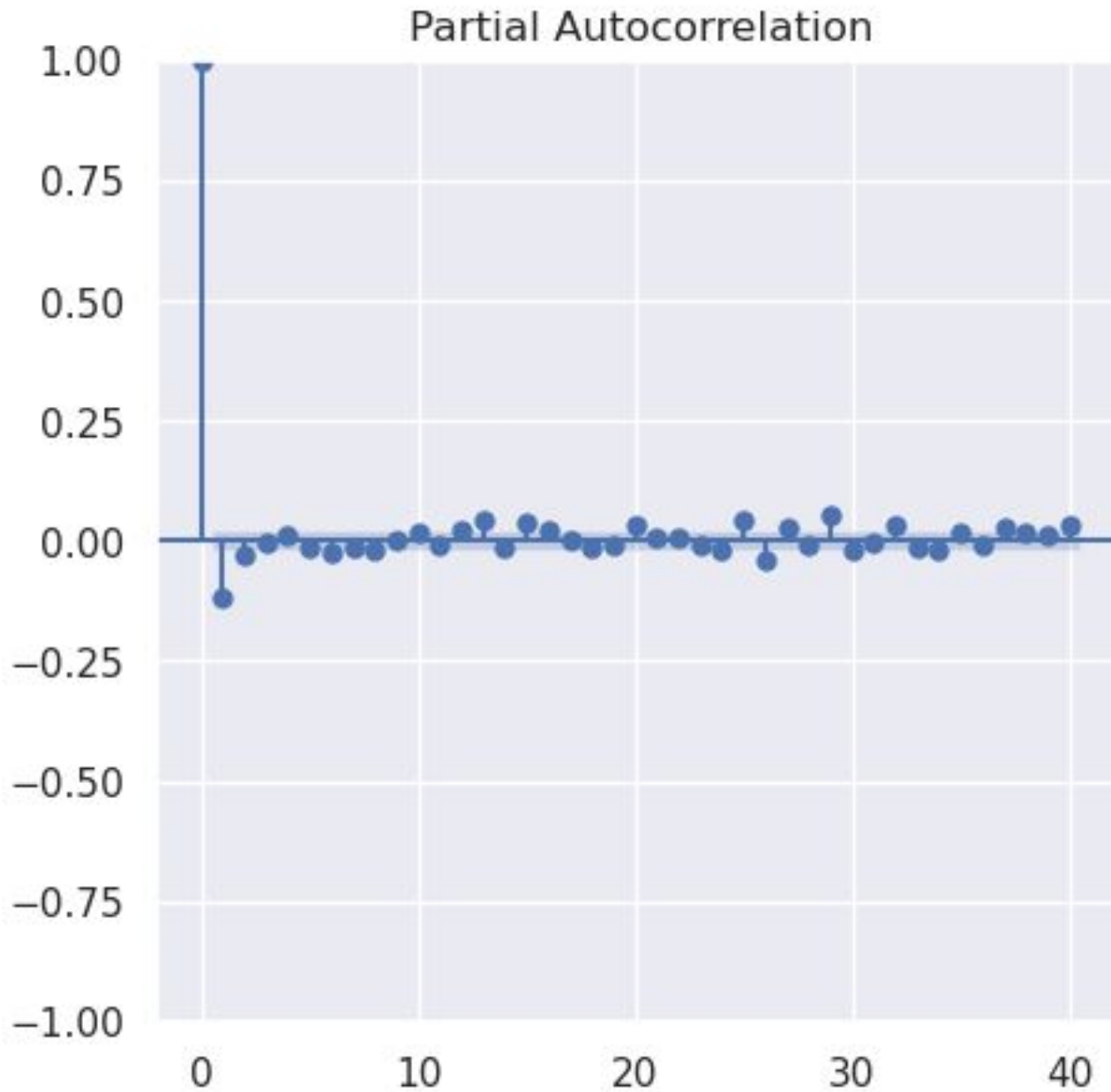
Adfuller Statistic value is less than the critical value for
Exponential and First Order Differencing.

For our Model Building We have went with
First Order Differencing Transform

Autocorrelation Plot



Partial Autocorrelation Plot



Models

1. Arima Model
2. Exponential Model
3. StatsForecastAuto Arima
4. FFT Model
5. Prophet
6. RNN
7. Long Short Term Memory (LSTM) Mode
8. RNN (GRU)

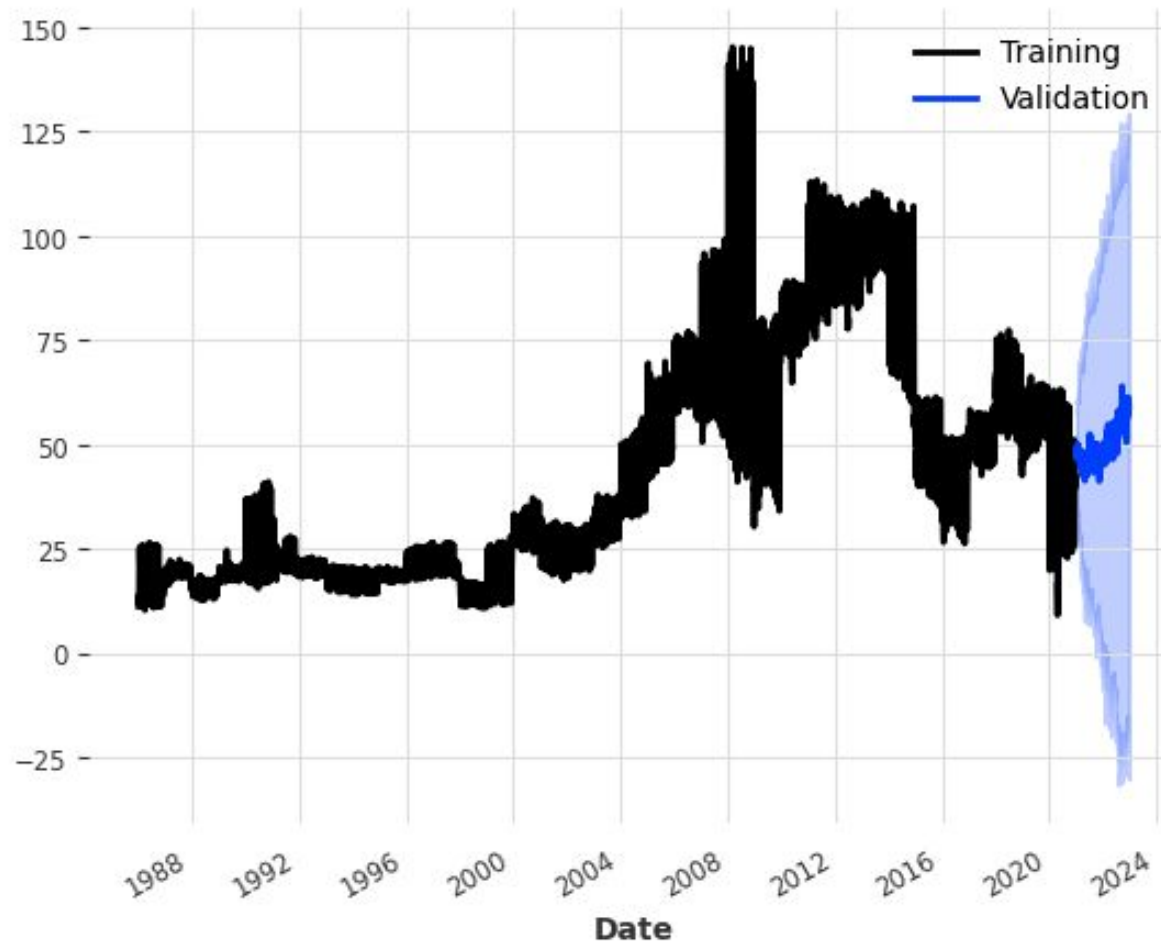
1. Arima Model

```
# from darts.models Arima building
model1 = ARIMA()
model1.fit(train)
prediction1 = model1.predict(len(val), num_samples=100)
```

	Price_s0	Price_s1	Price_s2	Price_s3	Price_s4	Price_s5	Price_s6	Price_s7	Price_s8	Price_s9	...	Price_s90	Price_s91	Price_s92	Price_s93	Price_s94	Price_s95	Price_s96
Date																		
2020-12-31	48.506504	48.506504	48.506504	48.506504	48.506504	48.506504	48.506504	48.506504	48.506504	48.506504	...	48.506504	48.506504	48.506504	48.506504	48.506504	48.506504	48.506504
2021-01-01	46.616036	52.523138	42.861924	46.265127	47.779011	48.941749	45.442813	51.259295	45.704675	52.138079	...	42.235647	49.335235	52.348617	45.336268	45.324803	47.041483	45.413076
2021-01-02	47.123306	53.095567	48.290239	43.466150	44.075189	46.766842	46.887481	55.355503	42.754585	53.243836	...	38.959707	51.285592	54.844945	43.202486	50.252727	42.889458	47.141494
2021-01-03	45.953599	48.838863	51.408382	46.613706	43.875886	43.215710	42.387063	54.696528	36.632393	55.942276	...	36.744988	51.427145	53.425769	38.233300	51.735396	38.990878	46.090911
2021-01-04	40.149270	46.541320	51.068099	50.599705	36.409612	46.158326	41.645559	54.178771	35.683654	54.405963	...	38.909993	53.445507	43.769949	45.357652	52.990992	39.093659	45.951986
...
2022-12-17	72.430552	-8.020346	-0.191889	36.314016	31.436690	-40.470571	102.052349	-20.495921	10.181074	176.069377	...	146.703166	75.400543	105.871653	27.125146	69.004635	46.278136	72.472188
2022-12-18	69.226698	-7.868161	6.919301	39.209073	34.531800	-33.166702	101.281880	-28.297334	11.288876	175.997002	...	146.052686	77.130131	103.759106	28.206549	72.910947	52.449357	68.760256
2022-12-19	70.026332	-9.848712	9.824674	36.090948	34.093396	-30.736523	103.639868	-29.757991	3.784253	173.552235	...	148.060156	78.742593	103.818870	27.710038	77.093135	54.380861	72.214083
2022-12-20	74.164986	-17.532018	6.328249	29.753496	33.939084	-30.562010	106.766416	-30.319649	1.074987	178.196330	...	152.054193	85.437373	102.329567	28.094511	71.563501	55.811625	73.167471
2022-12-21	74.988729	-15.414317	7.324965	35.506270	30.806620	-23.743440	106.319227	-28.435872	0.382491	177.199654	...	145.921909	87.477059	94.574500	32.522249	71.399437	56.047111	76.952495

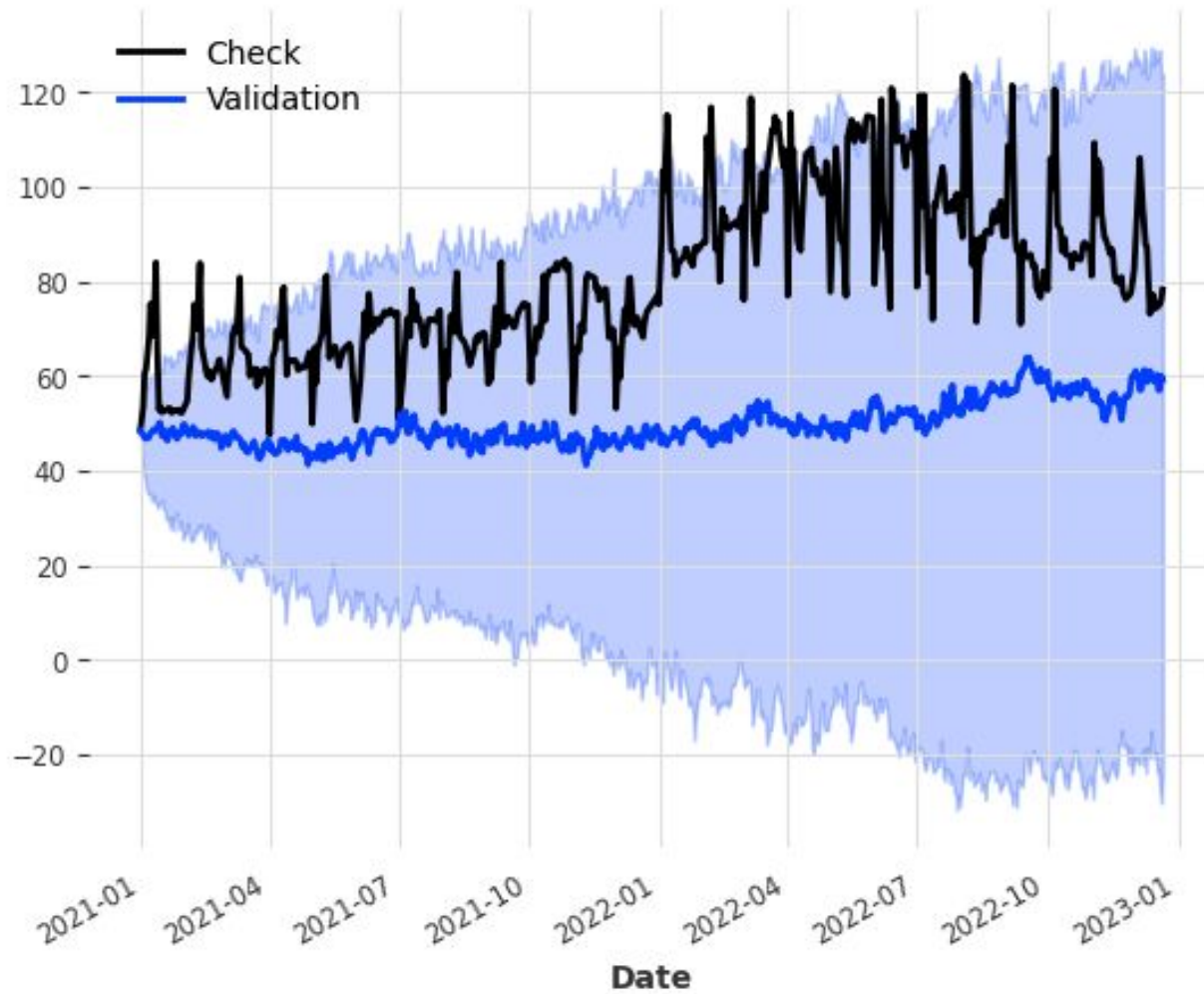
1. Arima Model

Here is our total data and prediction values in the same graph



1. Arima Model

Validation data and our prediction data



Different Evaluation Metrics

Mean absolute percentage error: 36.47105

Mean absolute scaled error : 18.23036

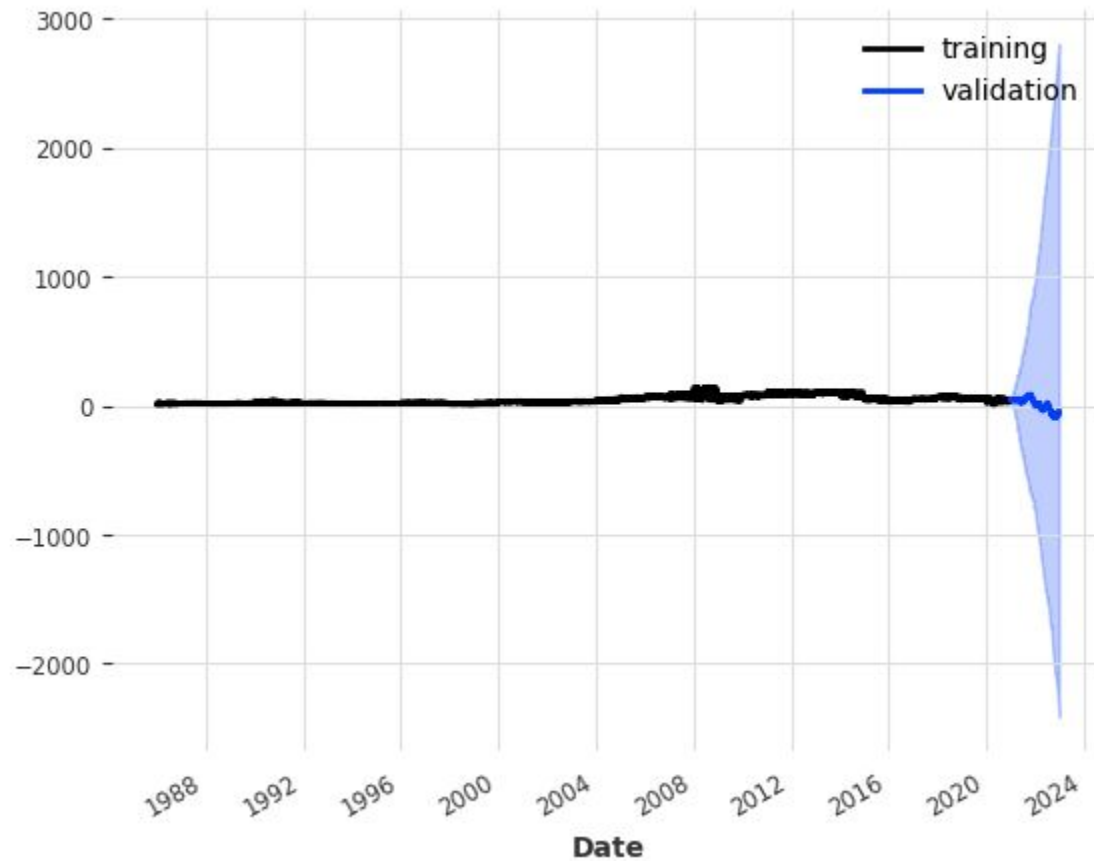
2. Exponential Model

```
# from darts.models import ExponentialSmoothing
model2 = ExponentialSmoothing()
model2.fit(train)
prediction2 = model2.predict(len(val), num_samples=100)
```

	Price_s0	Price_s1	Price_s2	Price_s3	Price_s4	Price_s5	Price_s6	Price_s7	Price_s8	Price_s9	...	Price_s90	Price_s91	Price_s92	Price_s93	Price_s94
Date																
2020-12-31	55.119747	49.864389	52.093775	56.957109	55.518575	44.556860	51.983383	47.739295	47.924782	49.904622	...	46.768984	53.032827	49.125029	52.085688	49.695655
2021-01-01	62.648645	44.780020	47.376701	61.022446	51.283485	51.989269	50.564539	44.901728	55.382568	55.719406	...	41.799607	54.269475	49.059415	47.762896	51.815493
2021-01-02	62.367513	44.420195	52.266561	64.645483	54.538547	46.524572	51.222449	42.615616	57.346531	56.245165	...	39.836400	53.287482	58.629881	44.343114	52.072973
2021-01-03	57.825761	50.675648	51.930156	62.579281	57.307675	44.717441	46.180781	37.239675	60.335086	55.910136	...	41.737367	52.607191	57.596045	37.987851	51.082478
2021-01-04	56.009609	46.599117	55.172649	64.600078	50.949396	46.130668	49.420193	37.265963	60.188446	53.219109	...	37.702112	51.562629	62.328336	42.810027	61.730105
...
2022-12-17	-1801.675865	-1060.960451	-74.788008	-1409.347471	932.118346	3223.025881	1964.678360	672.340110	388.295804	736.245811	...	-2330.166898	1352.203564	223.226065	439.025014	-337.531191
2022-12-18	-1813.386260	-1057.816832	-69.615384	-1418.756852	933.721944	3233.541316	1960.331646	672.958381	387.212258	742.164441	...	-2327.665408	1348.908752	228.855872	443.335219	-332.945084
2022-12-19	-1826.724127	-1059.347373	-65.841891	-1420.625904	935.740527	3239.129674	1955.005442	684.384882	385.628924	750.005831	...	-2333.460779	1353.633262	232.149040	445.876312	-330.148575
2022-12-20	-1837.326164	-1057.939316	-61.170612	-1420.045634	942.832040	3241.479046	1960.814349	690.121826	388.460863	753.718565	...	-2333.078689	1353.719844	237.987410	447.760961	-325.664202
2022-12-21	-1843.457902	-1066.928917	-59.417992	-1416.433710	943.909780	3246.177423	1958.138634	688.977050	389.494324	749.995733	...	-2340.340341	1357.703249	241.316933	442.254026	-329.841037

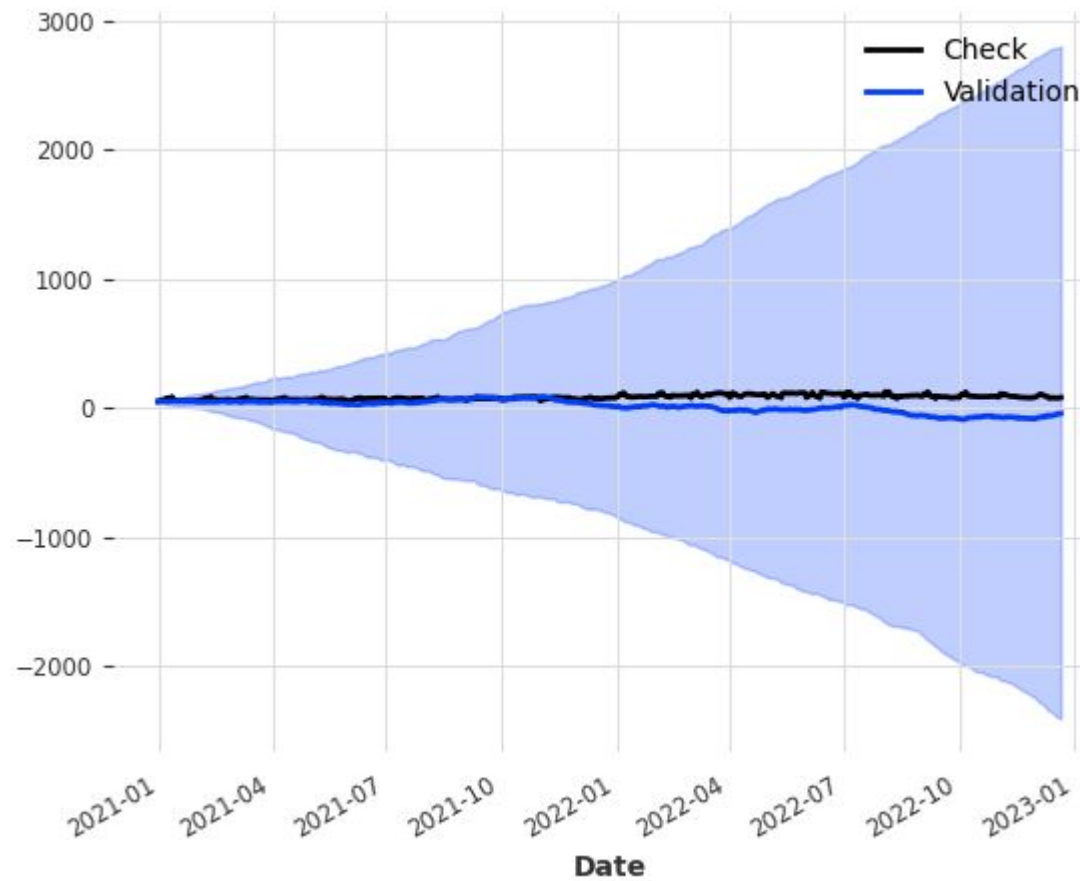
2. Exponential Model

Here is our total data and prediction values in the same graph



2. Exponential Model

Validation data and our prediction data



Different Evaluation Metrics

Mean absolute percentage error: 80.89652

Mean absolute scaled error : 41.73528

3. StatsForecastAuto Arima

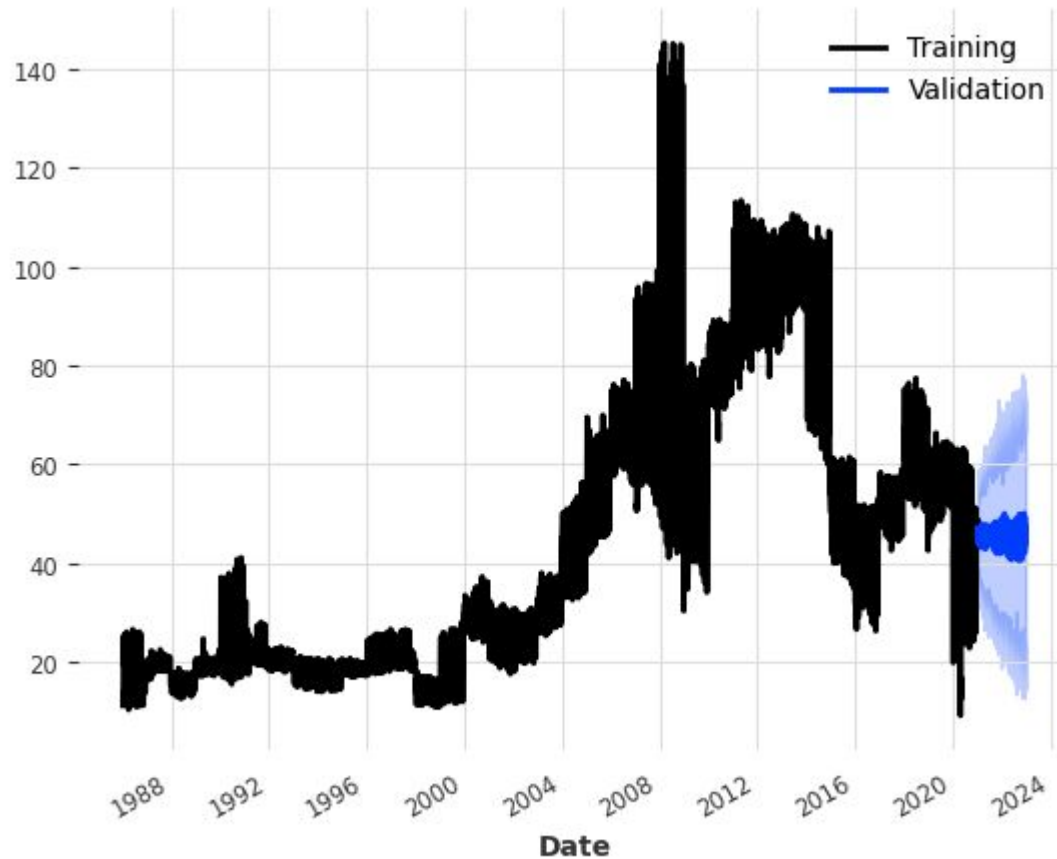
```
# from darts.models Arima building
model3 = StatsForecastAutoARIMA()
model3.fit(train)
prediction3 = model3.predict(len(val), num_samples=100)
```

	Price_s0	Price_s1	Price_s2	Price_s3	Price_s4	Price_s5	Price_s6	Price_s7	Price_s8	Price_s9	...	Price_s90	Price_s91	Price_s92	Price_s93	Price_s94	Price_s95	Price_s96
Date																		
2020-12-31	43.384053	48.665910	48.230778	49.180727	41.820229	47.598662	58.246728	45.919740	53.571084	49.522449	...	47.258122	46.133265	46.695717	44.524815	50.609547	47.001360	46.585637
2021-01-01	43.213323	44.684382	49.594950	49.324298	45.255522	52.152288	44.548743	43.552646	43.317357	48.381866	...	42.107323	41.073059	53.431044	49.334169	53.320143	49.316706	43.855540
2021-01-02	47.566443	38.755246	45.419906	39.389342	54.668983	33.176384	54.730700	48.587122	49.681081	44.416812	...	45.578643	40.021011	56.090379	43.559045	50.135801	52.380564	49.322348
2021-01-03	46.497177	47.557150	46.917766	43.118213	47.568573	34.087130	48.196828	44.806659	45.374472	38.363781	...	38.700078	44.947641	40.557864	45.353540	39.386299	46.840814	50.030386
2021-01-04	47.394980	38.086451	39.252301	41.771137	50.760711	46.542741	39.699126	44.069041	56.625845	39.294585	...	48.891590	33.404947	41.205840	44.147808	40.417693	39.149604	50.802606
...
2022-12-17	50.637708	39.070292	61.500290	24.776342	54.245568	67.524916	31.338011	9.237270	44.650418	59.643974	...	81.130241	72.939170	23.813333	27.227680	53.713365	35.377669	45.360946
2022-12-18	60.764063	39.354271	26.748660	54.135630	57.315399	48.199293	50.960564	44.493452	38.745265	19.638027	...	45.045272	50.767193	26.397680	61.135332	32.097407	84.792825	47.453558
2022-12-19	29.327403	47.245851	48.881396	50.326088	11.456532	52.456586	30.111689	39.075071	77.010079	62.956951	...	60.005473	54.576810	41.009024	45.403691	45.269793	63.397028	77.164362
2022-12-20	72.602148	37.239058	17.624676	89.749296	76.299636	56.819661	66.940481	66.380676	30.345689	53.019227	...	45.703444	50.296270	43.884399	62.631758	30.042305	56.872077	50.580871
2022-12-21	34.616416	15.010784	35.567248	52.178623	50.846071	50.014808	68.821698	55.895470	26.511419	61.499407	...	54.712513	46.382699	70.832929	31.273572	32.127650	36.934486	50.296445

721 rows × 100 columns

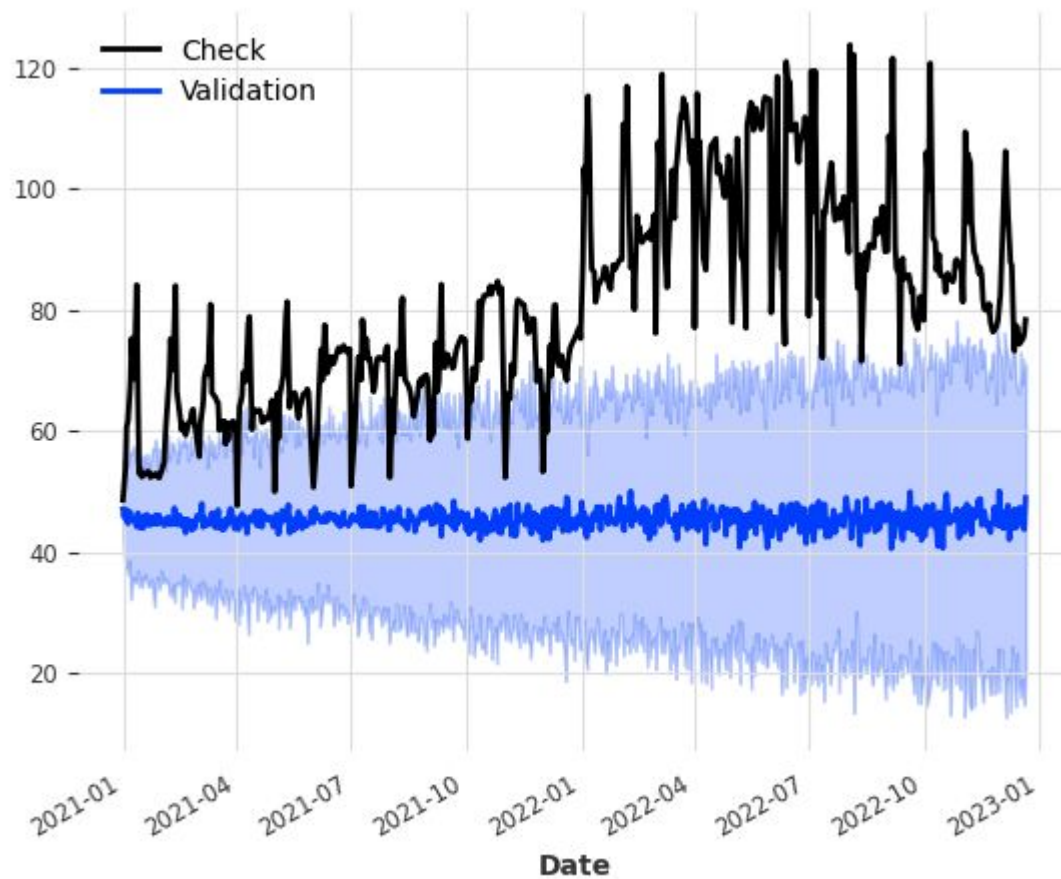
3. StatsForecastAuto Arima

Here is our total data and prediction values in the same graph



3. StatsForecastAuto Arima

Validation data and our prediction data



Different Evaluation Metrics

Mean absolute percentage error: 41.59005

Mean absolute scaled error : 20.80369

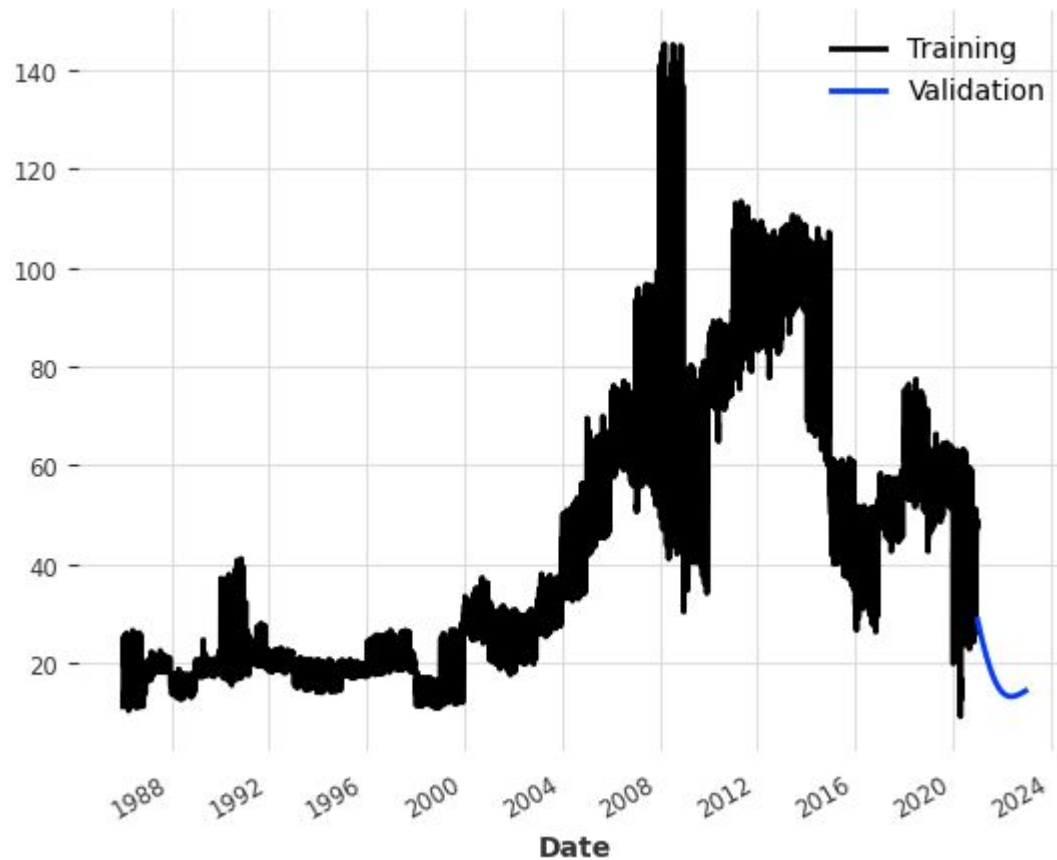
4. FFT Model

```
# from darts.models Arima building
model4 = FFT()
model4.fit(train)
prediction4 = model4.predict(len(val))
```

component	Price
Date	
2020-12-31	28.686897
2021-01-01	28.626326
2021-01-02	28.565814
2021-01-03	28.505362
2021-01-04	28.444970
...	...
2022-12-17	14.257850
2022-12-18	14.266248
2022-12-19	14.274657
2022-12-20	14.283078
2022-12-21	14.291510

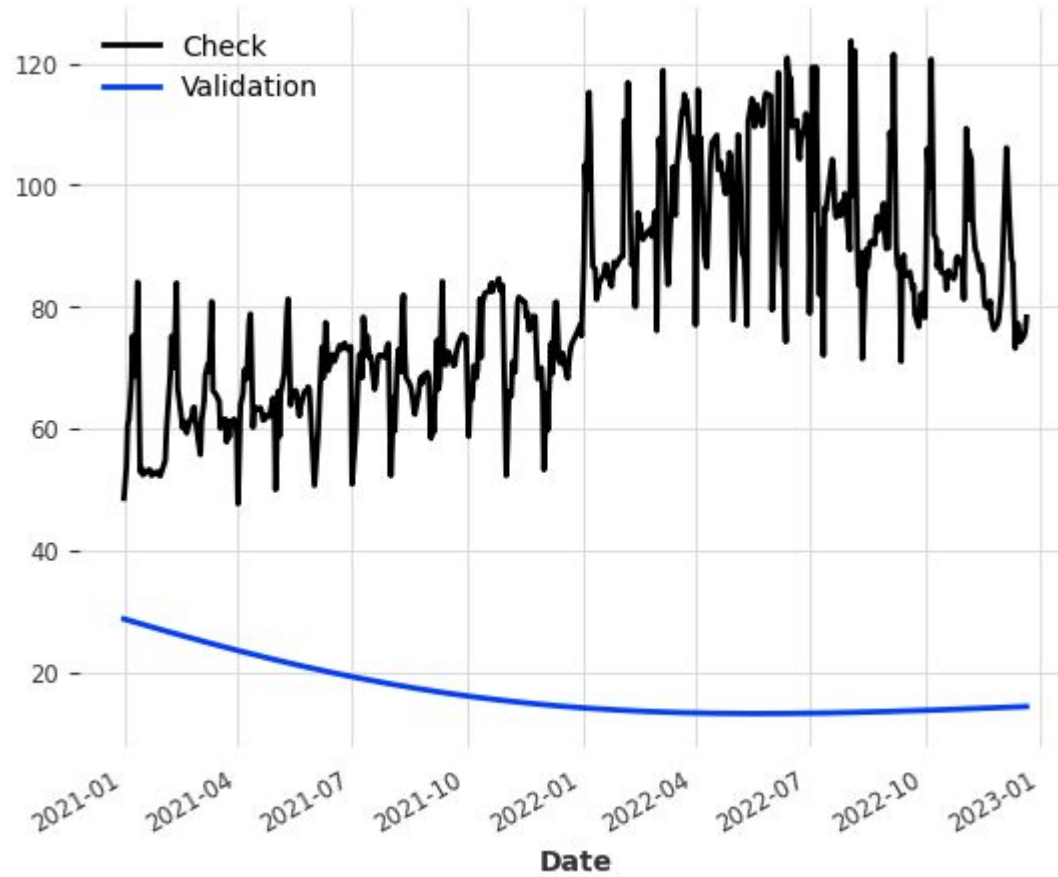
4. FFT Model

Here is our total data and prediction values in the same graph



4. FFT Model

Validation data and our prediction data



Different Evaluation Metrics

Mean absolute percentage error: 77.46323

Mean absolute scaled error : 37.468001

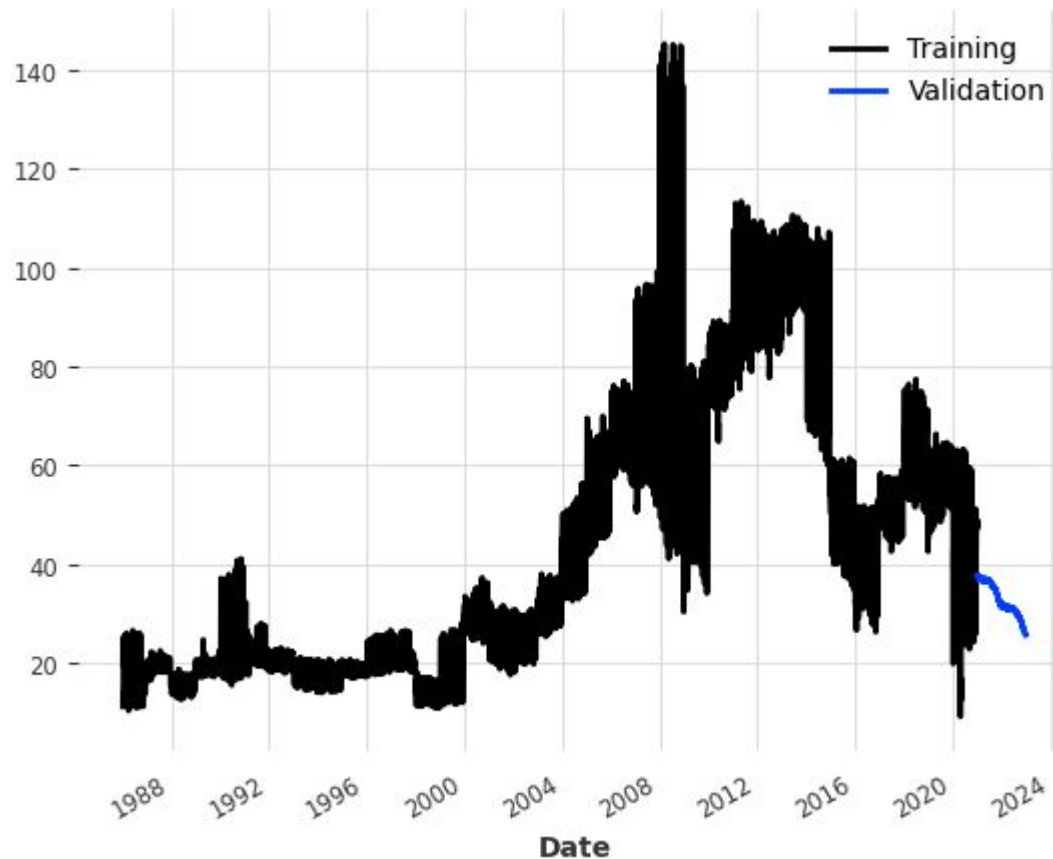
5. Prophet

```
# from darts.models Arima building
model5 = Prophet()
model5.fit(train)
prediction5 = model5.predict(len(val))
```

component	Price
Date	
2020-12-31	37.438148
2021-01-01	37.418363
2021-01-02	37.427820
2021-01-03	37.511397
2021-01-04	37.642012
...	...
2022-12-17	25.646738
2022-12-18	25.664690
2022-12-19	25.742870
2022-12-20	25.579294
2022-12-21	25.597313

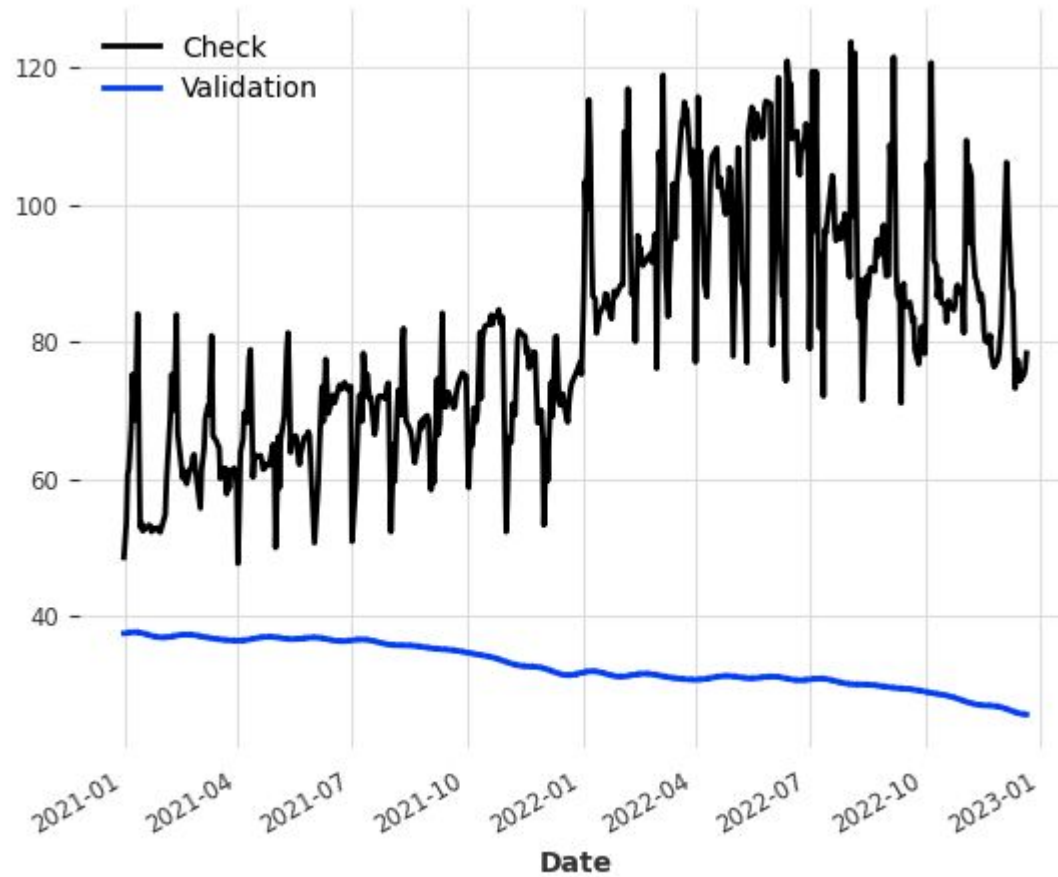
5. Prophet

Here is our total data and prediction values in the same graph



5. Prophet

Validation data and our prediction data



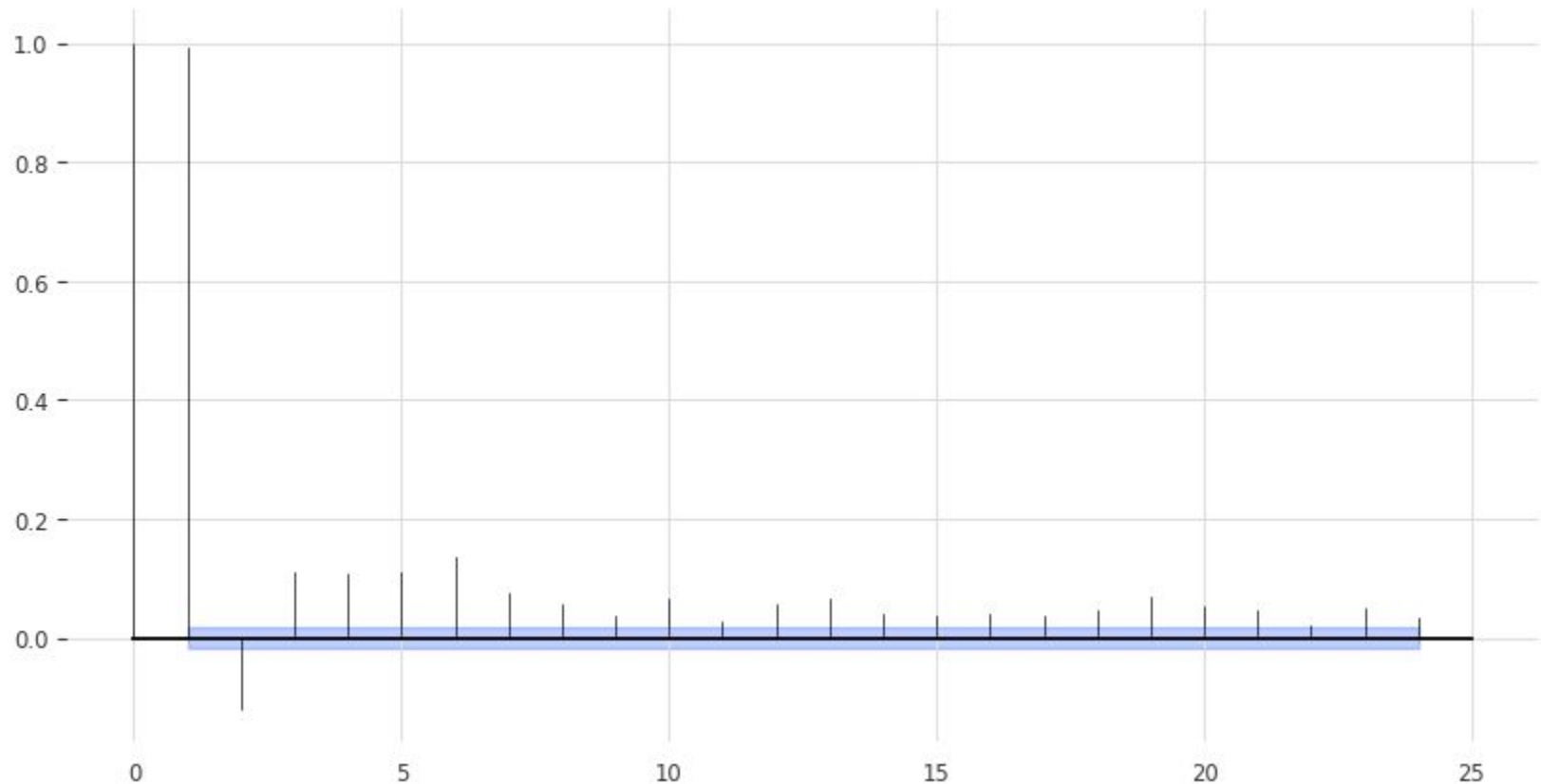
Different Evaluation Metrics

Mean absolute percentage error: 57.14686

Mean absolute scaled error : 28.13417

6. RNN

So based on the pack graph we know 24 days data has impact on the prediction so for the RNN model better to go with 24 days data as chunk length.



6. RNN

```
# from darts.models Arima building
model6 = RNNModel(input_chunk_length= 15 , # Input window size
    output_chunk_length=1, # Output window size
    hidden_dim=25, # Number of neurons in the hidden layer
    n_rnn_layers=3, # Number of layers in the RNN
    dropout=0.5, # Dropout rate
    batch_size=16, # Number of input-output pairs in each training batch
    n_epochs=50, # Number of training epochs
    # optimizer_kwargs={'lr': 1e-3} # Optimizer parameters
)
model6.fit(train)
prediction6 = model6.predict(len(val))
```

GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs

	Name	Type	Params
0	criterion	MSELoss	0
1	train_metrics	MetricCollection	0
2	val_metrics	MetricCollection	0
3	rnn	RNN	3.3 K
4	V	Linear	26

3.3 K Trainable params
0 Non-trainable params
3.3 K Total params
0.013 Total estimated model params size (MB)

Epoch 49: 100%| | 798/798 [00:29<00:00, 27.39it/s, loss=17.1, train_loss=9.800]

`Trainer.fit` stopped: `max_epochs=50` reached.

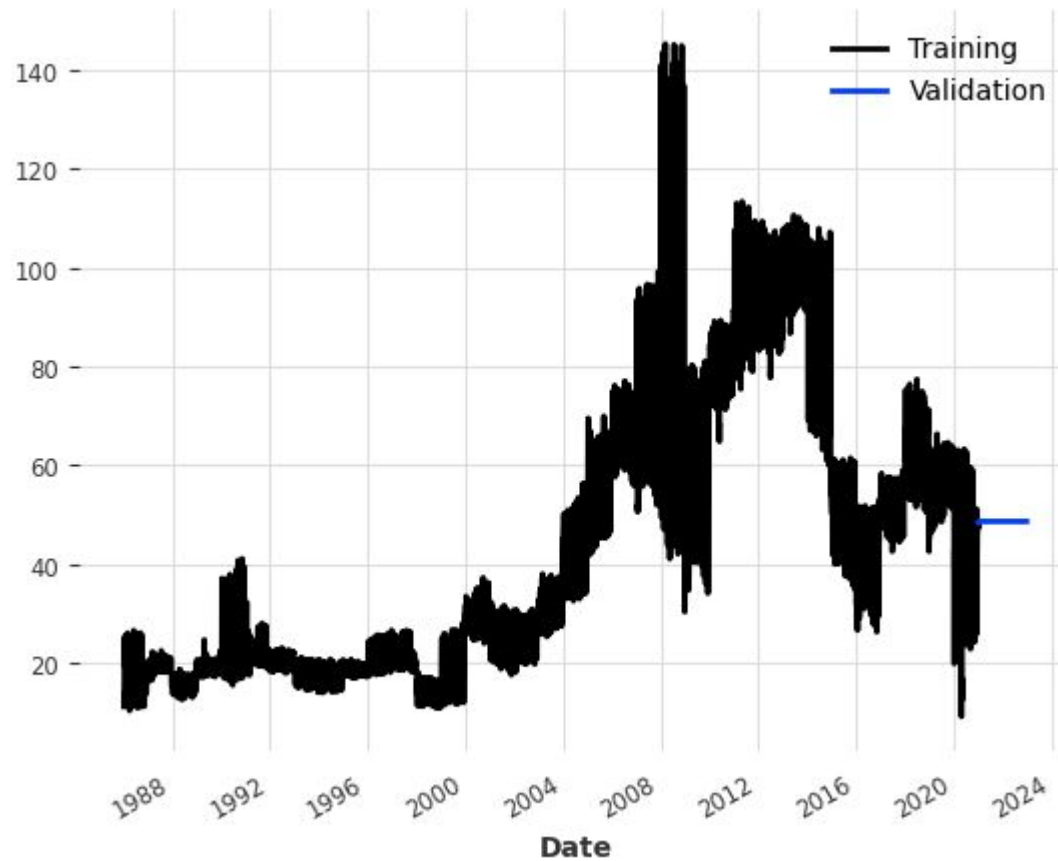
Epoch 49: 100%| | 798/798 [00:29<00:00, 27.38it/s, loss=17.1, train_loss=9.800]

GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs

Predicting DataLoader 0: 100%| | 1/1 [00:00<00:00, 4.51it/s]

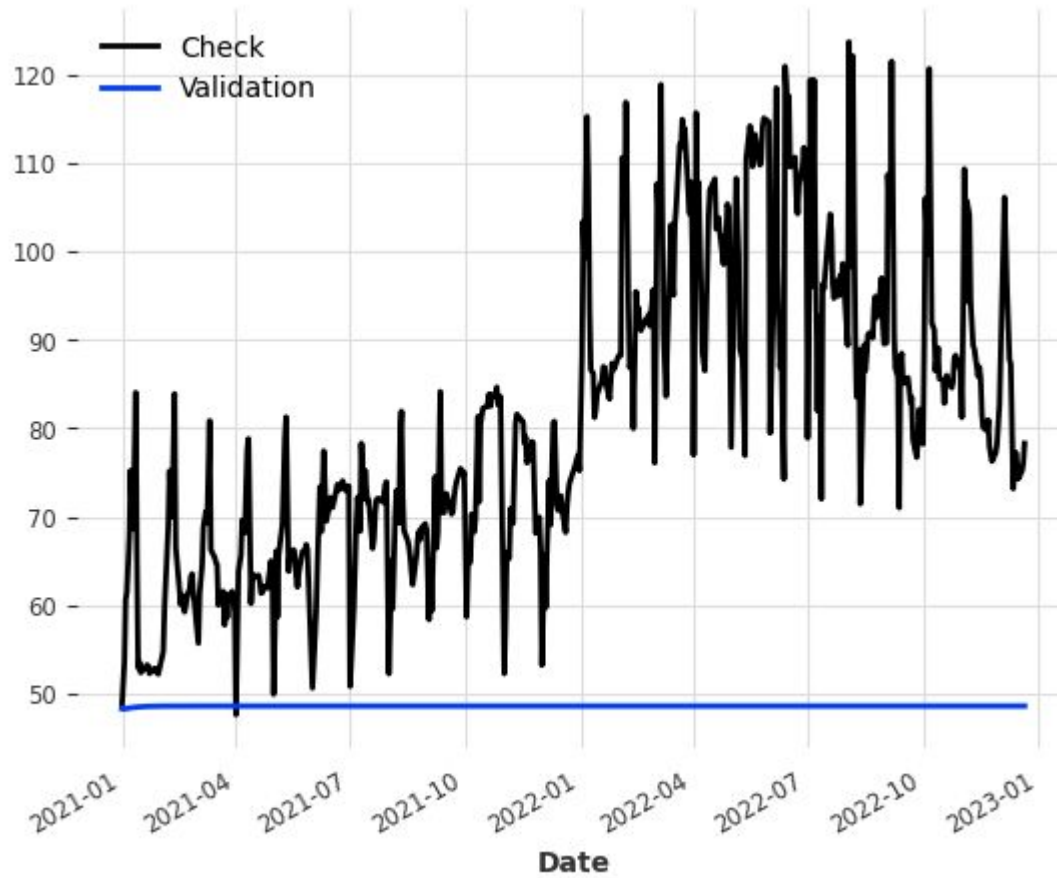
6. RNN

Here is our total data and prediction values in the same graph



6. RNN

Validation data and our prediction data



Different Evaluation Metrics

Mean absolute percentage error : 37.39202

Mean absolute scaled error : 18.90647

7. Long Short Term Memory (LSTM) Mode

```
# from darts.models.Arma building
model7 = RNNModel(model='LSTM',
    hidden_dim=25,
    dropout=0.5,
    batch_size=16,
    n_epochs=50,
    input_chunk_length= 15 , # Input window size
    output_chunk_length=1, # Output window size
    n_rnn_layers=3
)
```

```
model7.fit(train)
```

```
GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
```

	Name	Type	Params
0	criterion	MSELoss	0
1	train_metrics	MetricCollection	0
2	val_metrics	MetricCollection	0
3	rnn	LSTM	13.2 K
4	V	Linear	26

13.2 K	Trainable params
0	Non-trainable params
13.2 K	Total params
0.053	Total estimated model params size (MB)

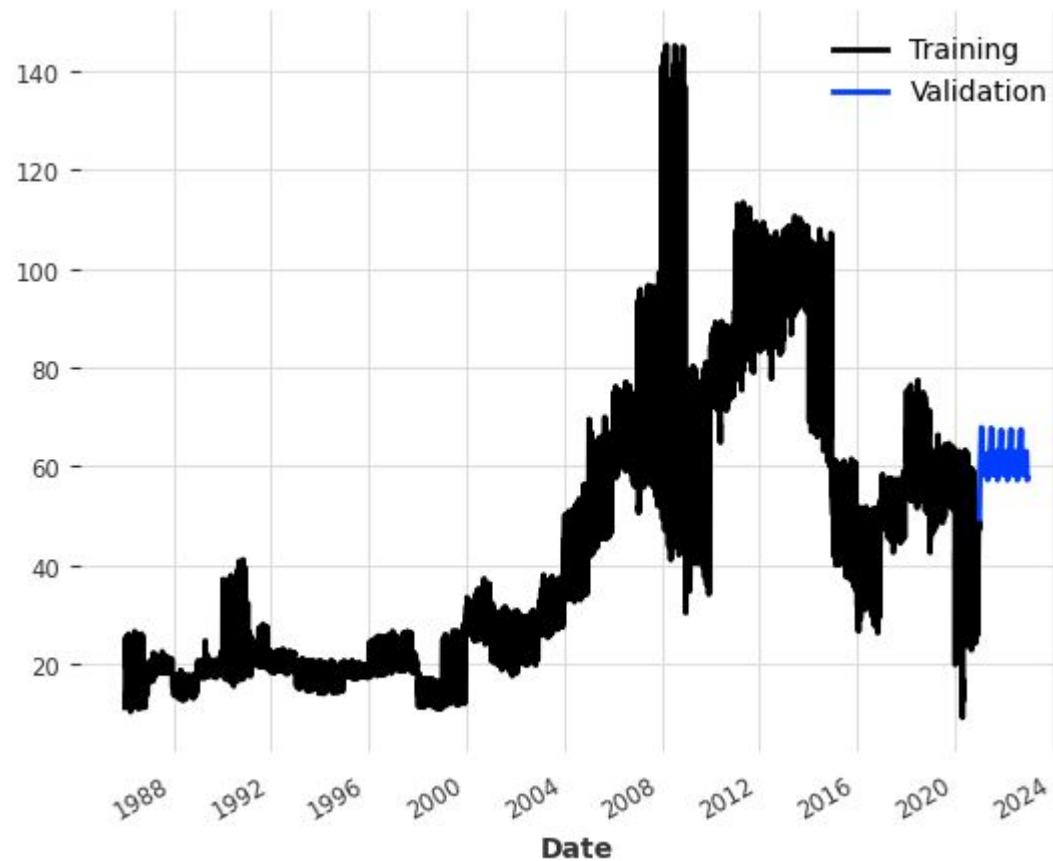
Epoch 49: 100% | ██████████ 798/798 [00:52<00:00, 15.33it/s, loss=16.1, train_loss=22.00]

```
`Trainer.fit` stopped: `max_epochs=50` reached.
```

Epoch 49: 100% | 798/798 [00:52<00:00, 15.33it/s, loss=16.1, train_loss=22.00]

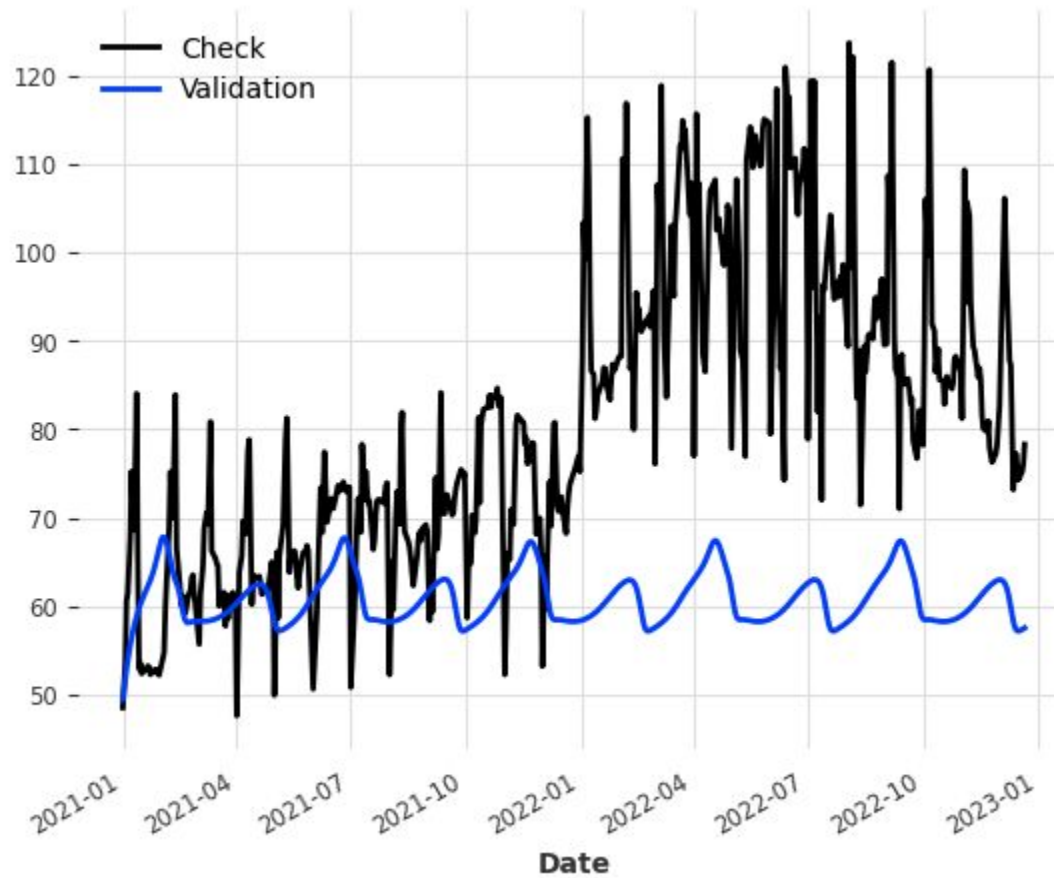
7. Long Short Term Memory (LSTM) Mode

Here is our total data and prediction values in the same graph



7. Long Short Term Memory (LSTM) Mode

Validation data and our prediction data



Different Evaluation Metrics

Mean absolute percentage error: 23.75631

Mean absolute scaled error : 12.43866

8. RNN (GRU)

```
# from darts.models Arima building
model8 = RNNModel(model='GRU',
    hidden_dim=25,
    dropout=0.5,
    batch_size=16,
    n_epochs=50,
    input_chunk_length= 15 , # Input window size
    output_chunk_length=1, # Output window size
    n_rnn_layers=3
)
```

```
model8.fit(train)
```

```
GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
```

	Name	Type	Params
0	criterion	MSELoss	0
1	train_metrics	MetricCollection	0
2	val_metrics	MetricCollection	0
3	rnn	GRU	9.9 K
4	V	Linear	26

```
9.9 K    Trainable params
0        Non-trainable params
9.9 K    Total params
0.040    Total estimated model params size (MB)
```

```
Epoch 49: 100%|██████████████████████████████████████████████████████████████████████████████| 798/798 [00:50<00:00, 15.72it/s, loss=14.8, train_loss=20.60]
```

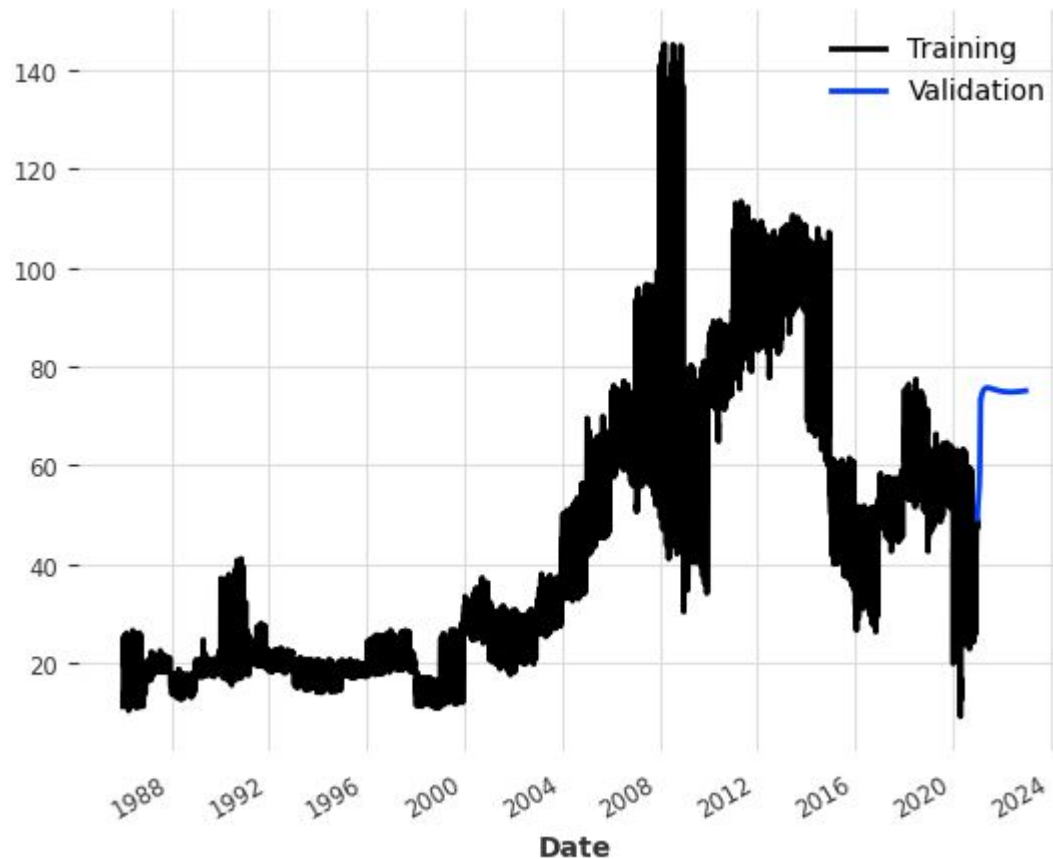
```
`Trainer.fit` stopped: `max_epochs=50` reached.
```

```
Epoch 49: 100% 798/798 [00:50<00:00, 15.72it/s, loss=14.8, train loss=20.60]
```

```
<darts.models.forecasting.rnn_model.RNNModel at 0x2cad59aec40>
```

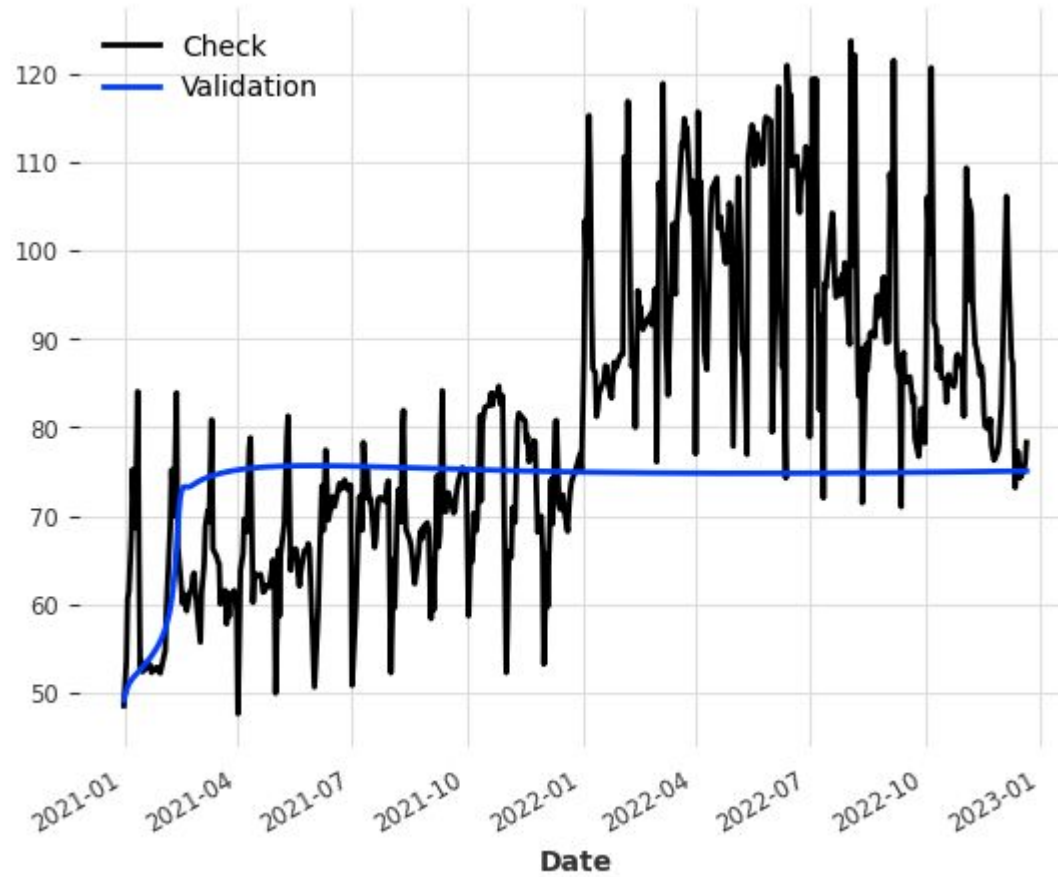
8. RNN (GRU)

Here is our total data and prediction values in the same graph



8. RNN (GRU)

Validation data and our prediction data



Different Evaluation Metrics

Mean absolute percentage error: 15.92568

Mean absolute scaled error : 8.022075

Model Name	MAPE	MASE
Arima Model	36.47105	18.23036
Exponential Model	80.89652	41.73528
StatsForecastAuto Arima	41.59005	20.80369
FFT Model	77.46323	37.468001
Prophet	57.14686	28.13417
RNN	37.39202	18.90647
Long Short Term Memory (LSTM) Mode	23.75631	12.43866
RNN (GRU)	15.92568	8.022075

Deployment

We are deploying Model Using Streamlit App

Crude oil Price Prediction

The following model app is based on facebook prophet Model where it is trained on crude oil price data from 1986-01-02 to 2020-12-30 using darts library functions. You can make predictions for any date after 2020-12-30 and the accuracy of the prediction will fall with increasing dates away from 2020-12-30

Select a date

2023/02/21

Predict

Crude oil Price Prediction

The following model app is based on facebook prophet Model where it is trained on crude oil price data from 1986-01-02 to 2020-12-30 using darts library functions. You can make predictions for any date after 2020-12-30 and the accuracy of the prediction will fall with increasing dates away from 2020-12-30

Select a date

2023/02/21

Predict



Click this to enter the date

Crude oil Price Prediction

The following model app is based on facebook prophet Model where it is trained on crude oil price data from 1986-01-02 to 2020-12-30 using darts library functions. You can make predictions for any date after 2020-12-30 and the accuracy of the prediction will fall with increasing dates away from 2020-12-30

Select a date

2023/02/21

<	February	2023	>			
Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28				

Click here to select the year

Crude oil Price Prediction

The following model app is based on facebook prophet Model where it is trained on crude oil price data from 1986-01-02 to 2020-12-30 using darts library functions. You can make predictions for any date after 2020-12-30 and the accuracy of the prediction will fall with increasing dates away from 2020-12-30

Select a date

2023/02/21

<

February

>

2023

Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28				

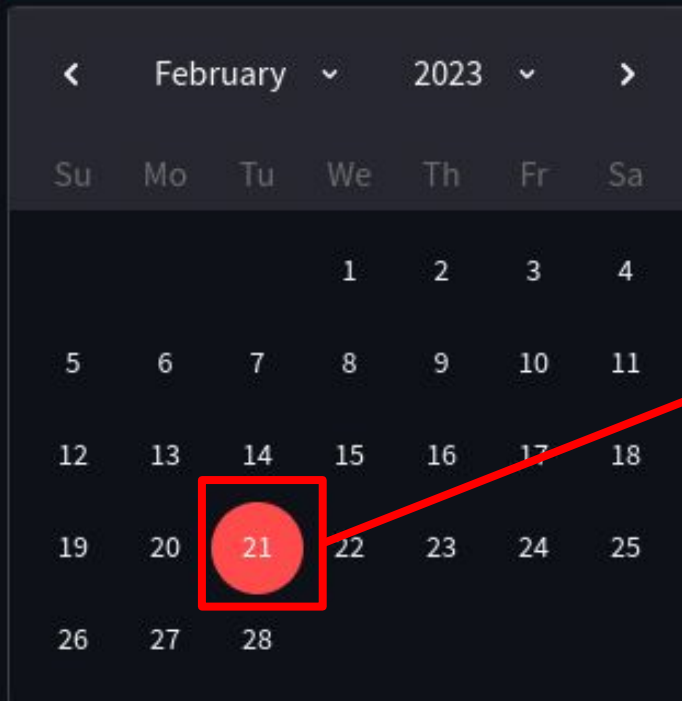
Click here to select the month

Crude oil Price Prediction

The following model app is based on facebook prophet Model where it is trained on crude oil price data from 1986-01-02 to 2020-12-30 using darts library functions. You can make predictions for any date after 2020-12-30 and the accuracy of the prediction will fall with increasing dates away from 2020-12-30

Select a date

2023/02/21



Click here to select the day of your choice


Crude oil Price Prediction

The following model app is based on facebook prophet Model where it is trained on crude oil price data from 1986-01-02 to 2020-12-30 using darts library functions. You can make predictions for any date after 2020-12-30 and the accuracy of the prediction will fall with increasing dates away from 2020-12-30

Select a date

2023/02/21

Predict



Once the date of choice is entered.
Click this to See the Prediction

Crude oil Price Prediction

The following model app is based on facebook prophet Model where it is trained on crude oil price data from 1986-01-02 to 2020-12-30 using darts library functions. You can make predictions for any date after 2020-12-30 and the accuracy of the prediction will fall with increasing dates away from 2020-12-30

Select a date

2022/02/23

Predict

The predicted price for crude oil on 2022-02-23 is given below

Date	Price
2022-02-23 00:00:00	31.4458

Here it shows the date you entered and shows the corresponding price.

The below table gives crude oil price from 2020-12-31 to 2022-02-23

Date	Price
2020-12-31 00:00:00	37.4105
2021-01-01 00:00:00	37.3909
2021-01-02 00:00:00	37.4002
2021-01-03 00:00:00	37.4837
2021-01-04 00:00:00	37.6143
2021-01-05 00:00:00	37.4889
2021-01-06 00:00:00	37.5301
2021-01-07 00:00:00	37.6314
2021-01-08 00:00:00	37.5731
2021-01-09 00:00:00	37.5389



Once you scroll down the web page you can see full dataframe giving prices from 2020-12-31 to selected date of your choice

Thank you