UE20CS312 - Data Analytics

Worksheet 3b - AR and MA Model

PES University

SUNDEEP A , Dept of CSE - PES1UG20CS445

AR and MA models

Auto Regressive and Moving Average are some of the most powerful, yet simple models for time-series forecasting. They can be used individually or together as ARMA. There are many other variations as well. We will use these models to forecast time-series in this worksheet

Task

Cryptocurrency is all the rage now and it uses the very exciting technology behind blockchain. People even claim it to be revolutionary. But if you have invested in cryptocurrencies, you know how volatile these cryptocurrencies really are! People have become billionaires by investing in crypto, and others have lost all their money on crypto. The most recent instance of this volatility was seen during the Terra Luna crash. Find more info about that here and here if you are interested.

Your task is to effectively forecast the prices of **DogeCoin**, a crypto that started as a meme but now is a crypto that people actually invest in. DogeCoin prices however, are affected even by a single tweet by Elon Musk. The image below tweeted by Elon Musk shot up the prices of DogeCoin by 200%!

You have been provided with the daily prices of DogeCoin from `15-08-2021` to `15-08-2022` a period of 1 year (365 days) in the file `doge.csv`

Please download the data from this Github repo

Data Dictionary

Date - Date on which price was recorded
Price - Price of DogeCoin on a particular day

Data Ingestion and Preprocessing

- Read the file into a `Pandas` DataFrame object

In [26]:
```
import pandas as pd
df = pd.read_csv('doge.csv')

df.head()
```

Out[26]:

	Date	Price
0	2021-08-15	0.348722
1	2021-08-16	0.349838
2	2021-08-17	0.345208
3	2021-08-18	0.331844
4	2021-08-19	0.321622

Prerequisites

- Set up a new conda env or use an existing one that has `jupyter-notebook` and `ipykernel` installed (Conda envs come with these by default) Reference
- Instead, you can also use a python venv and install `ipykernel` manually (We instead suggest using conda instead for easy setup) Reference
- Install the `statsmodels` package either in your Conda environment or Python venv. Refer to the installation guide

Points

The problems in this worksheet are for a total of 10 points with each problem having a different weightage.

- Problem 0: 0.5 points
- Problem 1: 1.5 point
- Problem 2: 2 points
- Problem 3: 1 points
- Problem 4: 2 point
- Problem 5: 1 point
- Problem 6: 1 points

HINTS FOR ALL PROBLEMS:

- Consider using `inplace=True` or assign it to new DataFrame, when using pandas transformations. If none of these are done, the DataFrame will remain the same
- Search for functions in the `statsmodels` documentation

Problem 0 (0.5 point)

- Set the index of DataFrame to the `Date` column to make it a time series

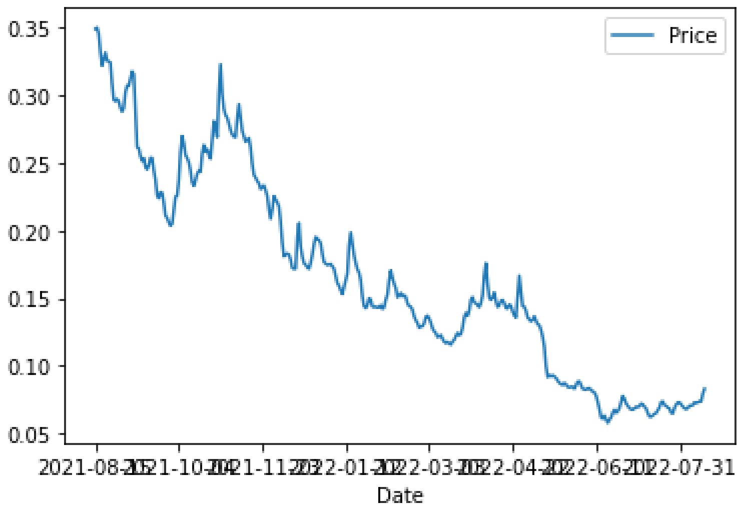
In [27]:
```
#Setting the index to date column.
df.set_index('Date', inplace=True)
print(df.head())
```

```
                Price
Date
2021-08-15  0.348722
2021-08-16  0.349838
2021-08-17  0.345208
2021-08-18  0.331844
2021-08-19  0.321622
```

Problem 1 (1.5 point)

- Plot the time-series. Analyze the stationarity from the time-series. Provide reasoning for stationarity/non-stationarity based on visual inspection of time-series plot (0.5 point)

In [3]:
```
import matplotlib.pyplot as plt
df.plot()
plt.show()
```

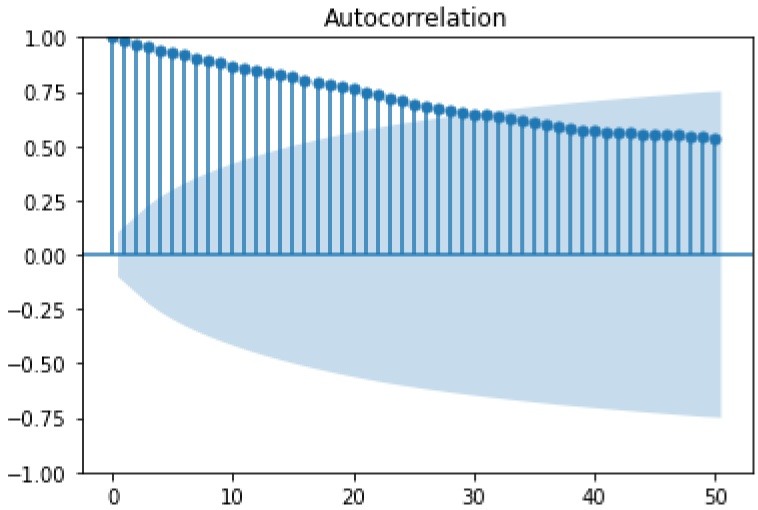


Analysis:

The data is Non-stationary as we can clearly observe the trend component. On a general Note, the price is decreasing with time.

- Plot the ACF plot of the Time series (upto 50 lags). Analyze the stationarity from ACF plot and provide reasoning (Hint: look at functions in `statsmodels` package) (1 Point)

In [4]:
```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(df.values.squeeze(),lags=50)
plt.show()
```



Analysis:

From the ACF plot we can say that the time-series plot is non-stationary as the ACF plot is decreasing slowly.

Problem 2 (2 point)

- Run Augmented Dickey Fuller Test. Analyze whether the time-series is stationary, based on ADF results (1 point)

Hint: Use the `print_adf_results` function below to print the results of the ADF function cleanly after obtaining results from the library function. Pass the results from library function to `print_adf_results` function

In [5]:
```
from statsmodels.tsa.stattools import adfuller

def print_adf_results(adf_result):
    print('ADF Statistic: %f' % adf_result[0])
    print('p-value: %f' % adf_result[1])
    print('Critical Values:')
    for key, value in adf_result[4].items():
        print('\t%s: %.3f' % (key, value))

adf_result = adfuller(df['Price'], autolag = 'AIC')
print_adf_results(adf_result)
```

```
ADF Statistic: -1.558935
p-value: 0.504182
Critical Values:
	1%: -3.449
	5%: -2.870
	10%: -2.571
```

Interpretation :

p-value is greater than 0.05 which implies we fail to reject the null hypothesis.This means the time series is non-stationary.

- If not stationary, apply appropriate transformations. Run the ADF test again to show stationarity after transformation (1 Point)

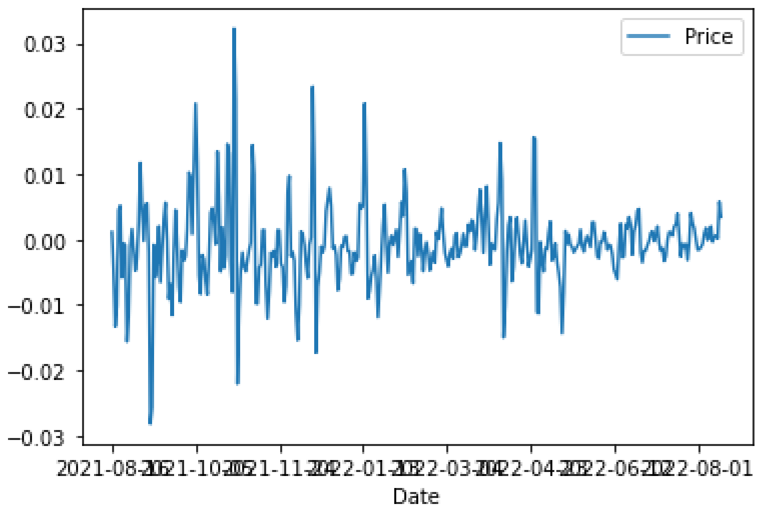
Hint: `diff` and `dropna`. Assign the DataFrame after transformation to a new DataFrame with name `transformed_df`

In [6]:
```
transformed_df = df.diff().dropna()
trans_adf = adfuller(transformed_df)
print_adf_results(trans_adf)

transformed_df.plot()
```

```
ADF Statistic: -5.593446
p-value: 0.000001
Critical Values:
        1%: -3.449
        5%: -2.870
        10%: -2.571
```

Out[6]:
```
<AxesSubplot:xlabel='Date'>
```



Interpretation :

After transformation,running the ADF test gave p-value as 0.000001 which is very less than 0.05.This means the transformed time series data is stationary.

Problem 3 (1 point)

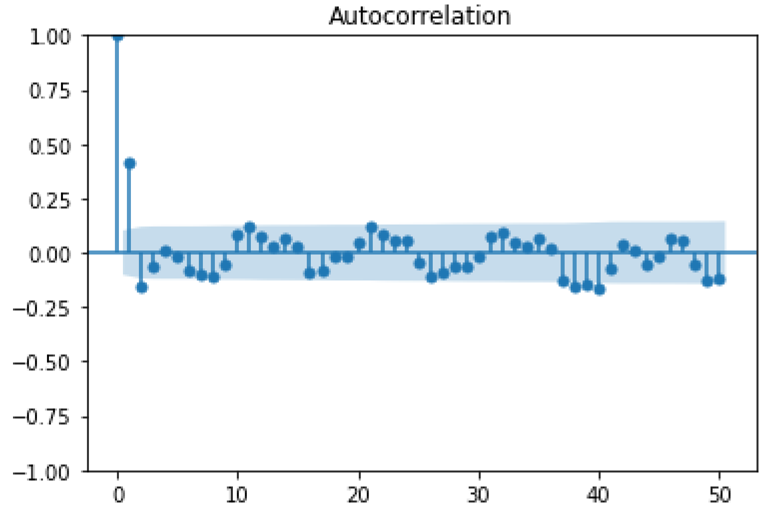
- Plot both ACF and PACF plot. From these select optimal parameters for the ARIMA(p,q) model

Hint: Negative values that are significantly outside the Confidence interval are considered significant too.
Hint: `p+q = 3`

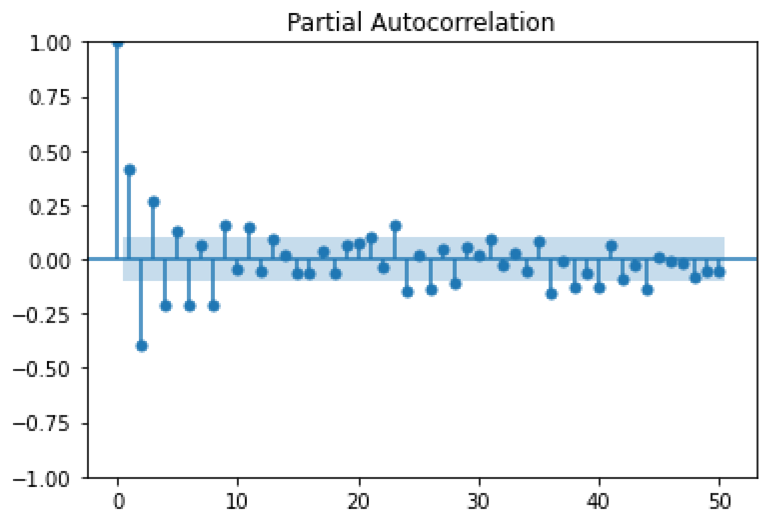
In [7]:
```
import statsmodels.graphics.tsaplots as smp

smp.plot_acf(transformed_df,lags=50)
plt.show()

smp.plot_pacf(transformed_df,lags=50)
plt.show()
```




```
C:\Users\HP\anaconda3\envs\NLP\lib\site-packages\statsmodels\graphics\tsaplots.py:348: FutureWarning: The default method 'yw' can produce PACF values outside of the [-1,1] interval. After 0.13, the default will change tounadjusted Yule-Walker ('ywm'). You can use this method now by setting method='ywm'.
  warnings.warn(
```



Analysis :

For the ARIMA(p,q) model let the value of p be 2 and let the value of q be 1 and d=1 as we are differencing the model only once(1) to become stationary.

Problem 4 (2 point)

- Write a function to forecast values using only AR(p) model (2 Points)
  Only use `pandas` functions and Linear Regression from `sklearn`. LR documentation

Hint: Create p new columns in a new DataFrame that is a copy of `transformed_df`
Each new column has lagged value of Price. `Price_t-i` (From `Price_t-1` upto `Price_t-p`)
Look at the `shift` function in pandas to create these new columns Link

In [8]:
```
from math import nan
from sklearn.linear_model import LinearRegression

### Adding columns for lagged values
arima_df = transformed_df.copy()

## AR terms
```

```python
p = 2

# Creating p new columns, for p lagged values
for i in range(1,p+1):
    arima_df[f'Price_t-{i}'] = df['Price'].shift(i)

arima_df.dropna(inplace=True)

print(arima_df.head())
```

```
               Price  Price_t-1  Price_t-2
Date
2021-08-17 -0.004630   0.349838   0.348722
2021-08-18 -0.013363   0.345208   0.349838
2021-08-19 -0.010222   0.331844   0.345208
2021-08-20  0.004498   0.321622   0.331844
2021-08-21  0.005169   0.326120   0.321622
```

Hint:

- **Seperate into `X_train` and `y_train` for linear regression**
- We know that AR(p) is linear regression with p lagged values, and we have created p new columns with the p lagged values
- `X_train` is training input that consists of the columns `Price_t-1` upto `Price_t-p` (p columns in total)
- `y_train` is the training output (truth values) of the Price, i.e the `Price` column (Only 1 column)

In [9]:
```python
X_train = arima_df[['Price_t-1', 'Price_t-2']].values
y_train = arima_df['Price'].values
```

- Set up the Linear Regression between `X_train` and `y_train` [LR documentation]

Name the `LinearRegression()` object `lr`

In [10]:
```python
# TODO: Perform Linear Regression
lr = LinearRegression()
lr.fit(X_train, y_train)
```

Out[10]:
```
LinearRegression()
```

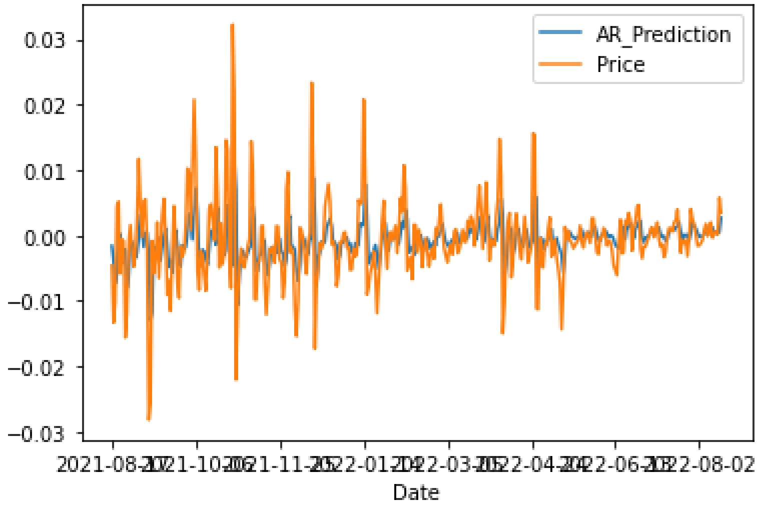
In [11]:
```python
lr.coef_
```

Out[11]:
```
array([ 0.39922266, -0.40813018])
```

In [28]:
```python
# Adding new column with predictions using the LR coefficients. The LR Coefficients are
arima_df['AR_Prediction'] = X_train.dot(lr.coef_.T) + lr.intercept_
```

In [29]:
```python
arima_df.plot(y=['AR_Prediction', 'Price'])
```

Out[29]:
```
<AxesSubplot:xlabel='Date'>
```



Once you get predicitons like this using AR you would have to, undifference the predictions (which are differenced), but we will not deal with that here. For some hints on how to undifference the data to get actual predictions look here

Problem 5 (1 Point)

Phew! Just handling AR(2) manually required us to difference, apply regression, undifference. Let's make all of this much easier with a simple library function

- **Use the ARIMA function using parameters picked to forecast values (1 point)**

Hint: Look at ARIMA function the `statstmodels`. Pass the `p,d,q` inferred from the previous tasks
We **DO NOT** need to pass the `transformed_df` to the ARIMA function.
Pass the orirginal `df` as input to ARIMA function, with the `d` value inferred when Transforming the df to make it stationary
The ARIMA function automatically performs the differencing based on the `d` value passed
Store the `.fit()` results in an object named `res`

In [14]:

```
## TODO: Use ARIMA function
import statsmodels.api as sm
model = sm.tsa.ARIMA(df['Price'], order=(2,1,1))
res = model.fit()
res.summary()
```

```
C:\Users\HP\anaconda3\envs\NLP\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471:
ValueWarning: No frequency information was provided, so inferred frequency D will be use
d.
  self._init_dates(dates, freq)
C:\Users\HP\anaconda3\envs\NLP\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471:
ValueWarning: No frequency information was provided, so inferred frequency D will be use
d.
  self._init_dates(dates, freq)
C:\Users\HP\anaconda3\envs\NLP\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471:
ValueWarning: No frequency information was provided, so inferred frequency D will be use
d.
  self._init_dates(dates, freq)
```

Out[14]:

SARIMAX Results

Dep. Variable:	Price	**No. Observations:**	366
Model:	ARIMA(2, 1, 1)	**Log Likelihood**	1431.397
Date:	Sat, 08 Oct 2022	**AIC**	-2854.795

Time:	22:47:31	**BIC**	-2839.195
Sample:	08-15-2021	**HQIC**	-2848.595
	- 08-15-2022		
Covariance Type:	opg		

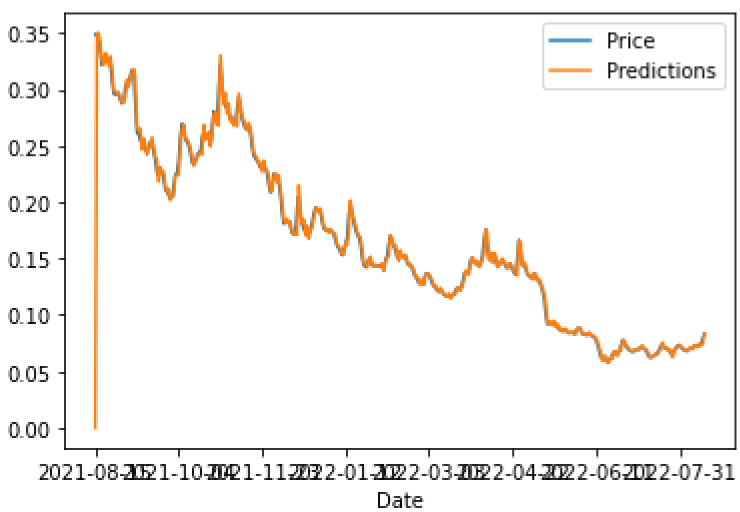
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.0436	0.058	0.757	0.449	-0.069	0.157
ar.L2	-0.2682	0.043	-6.279	0.000	-0.352	-0.184
ma.L1	0.7783	0.041	18.928	0.000	0.698	0.859
sigma2	2.284e-05	7.19e-07	31.765	0.000	2.14e-05	2.42e-05

Ljung-Box (L1) (Q):	0.14	**Jarque-Bera (JB):**	1884.22
Prob(Q):	0.71	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.16	**Skew:**	1.38
Prob(H) (two-sided):	0.00	**Kurtosis:**	13.78

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

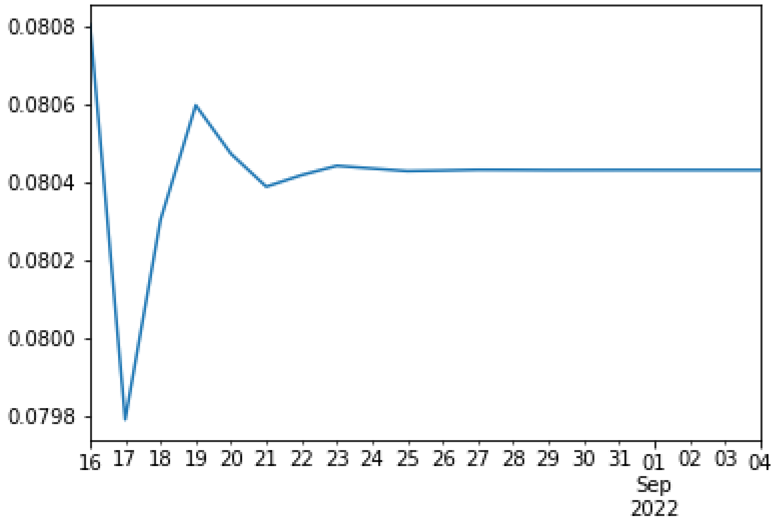
In [15]:
```
# Making predictions and plotting
df['Predictions'] = res.predict(0, len(df)-1)
df.plot()
```

Out[15]: <AxesSubplot:xlabel='Date'>



In [16]:
```
# Forecast for 20 future dates after training data ends
res.forecast(20).plot()
```

Out[16]: <AxesSubplot:>



Problem 6 (1 point)

- Evaluate the ARIMA model using Ljung Box test. Based on p-value infer if the Model shows lack of fit

Hint: Pass the `res.resid` (Residuals of the ARIMA model) as input the Ljung-Box Text.
Pass `lags=[10]`. Set `return_df=True` For inference, refer back to the Null and Alternate Hypotheses of Ljung-Box test. (If p value high, Null Hypothesis is significant)

In [17]:
```
import statsmodels.api as sm
sm.stats.acorr_ljungbox(res.resid, lags=[10], return_df=True)
```

Out[17]:

	lb_stat	lb_pvalue
10	0.855819	0.999916

Analysis :

We fail to reject null hypothesis because p>0.05 which implies the null-hypothesis is significant. So we can conclude that the model does not show lack of fit.