**Problem 0: recast the data into a time series**

df.set\_index('Date', inplace=True)

df

**Problem 1: comment on the pattern seen in the plot**

df.plot

**Plot the ACF and comment on the pattern**

import statsmodels.api as sm

import matplotlib.pyplot as plt

sm.graphics.tsa.plot\_acf(df.values.squeeze(), lags=50)

plt.show()

(a gradual decrease in ACF, rather than a sharp decrease/ cut off to 0 is another sign of nonstationarity)

**Problem 2: Comment on the result of the ADF test**

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))

from statsmodels.tsa.stattools import adfuller

adf\_result = adfuller(df)

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))

transformed\_df = df.diff().dropna()

trans\_adf = adfuller(transformed\_df)

print\_adf\_results(trans\_adf)

transformed\_df.plot()

**Problem 3: Plot the partial autocorrelation**

import statsmodels.api as sm

import matplotlib.pyplot as plt

sm.graphics.tsa.plot\_acf(transformed\_df, lags=50)

plt.show()

import statsmodels.api as sm

import matplotlib.pyplot as plt

sm.graphics.tsa.plot\_pacf(transformed\_df, lags=50)

plt.show()

**What are the ARMA and ARIMA parameters and why?**

**Problem 4: AR(p)**

#Add columns for lags

arima\_df = transformed\_df.copy()

#Set the value of p to an appropriate constant

#p=??

for i in range(1,p+1):

arima\_df[f'Price\_t-{i}'] = arima\_df['Price'].shift(i)

arima\_df.dropna(inplace=True)

arima\_df

#Separate data into x\_train and y\_train for linear regression

X\_train = arima\_df[['Price\_t-1', 'Price\_t-2']].values

y\_train = arima\_df['Price'].values

#run a linear regression model

from sklearn.linear\_model import LinearRegression

lr = LinearRegression()

lr.fit(X\_train, y\_train)

lr.coef\_

arima\_df['AR\_Prediction'] = X\_train.dot(lr.coef\_.T) + lr.intercept\_

arima\_df

arima\_df.plot(y=['AR\_Prediction', 'Price'])

**#fill in appropriate values for p, d and q below**

import statsmodels.api as sm

model = sm.tsa.arima.ARIMA(df, order=(p,d,q))

res = model.fit()

res.summary()

**Problem 5: Plenty of hints on the Q paper – please try this on your own!**

**Problem 6: Evaluate the arima model using the Ljung box test**

import statsmodels.api as sm

sm.stats.acorr\_ljungbox(res.resid, lags=[10], return\_df=True)