* Scrum Master for Next Week
  + Gustavo A Hernandez
* List at least 5 things the team did well and will continue doing
  + - Working in the Python code
    - Looking forward to developing visualizations in Tableau
    - Sharing ideas
    - Developing code
    - Working with Trello
* List at least 3 things the team did poorly and how you will mitigate them next sprint
  + - Time management
    - Workspace
    - Weekly meeting
* List shout-outs to any team members for excelling in any way
  + Chelsea Miller – Taking care of Trello for this week
* What did you learn as a team this week?
  + - How to share ideas
* What did you learn as an individual this week?
  + - How to work while life is being busy

CODE

Libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from matplotlib.pylab import rcParams

rcParams['figure.figsize'] = (10,5)

import seaborn as sns

sns.set\_style('darkgrid')

from statsmodels.graphics.tsaplots import plot\_acf,plot\_pacf

from statsmodels.tsa.seasonal import seasonal\_decompose

from statsmodels.tsa.stattools import adfuller

from statsmodels.tsa.statespace.sarimax import SARIMAX

import warnings

warnings.filterwarnings('ignore')

Loading Data

coffee\_raw *=* pd.read\_csv('../Data/coffee.csv')

Data Wrangling

coffee\_raw.Date = pd.to\_datetime(coffee\_raw.Date, yearfirst=True)

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

coffee = coffee\_raw.asfreq('b', 'ffill')

Exploratory Analysis

fig,axes = plt.subplots(2,2,figsize=[15,7])

fig.suptitle('Coffee Price',size=24)

## Resampling to Daily freq (Original Data)

axes[0,0].plot(coffee.Close)

axes[0,0].set\_title("Daily",size=16)

## Resampling to Monthly freq

axes[0,1].plot(coffee.Close.resample('M').mean())

axes[0,1].set\_title("Monthly",size=16)

## Resmapling to Quarterly freq

axes[1,0].plot(coffee.Close.resample('Q').mean())

axes[1,0].set\_title('Quarterly',size=16)

## Resampling to Annualy freq

axes[1,1].plot(coffee.Close.resample('A').mean())

axes[1,1].set\_title('Annualy',size=16)

plt.tight\_layout()

plt.show()

Using statsmodels

data\_close\_price = coffee.Close.resample('Q').mean()

decompose\_result = seasonal\_decompose(data\_close\_price, model = 'additive')

## Systematic Components

trend = decompose\_result.trend

seasonal = decompose\_result.seasonal

## Non-Systematic Components

residual = decompose\_result.resid

decompose\_result.plot();

Stationarity

def plot\_rolling\_stats(series, window*=*1):

*## Calculating the Rolling Mean and Rolling Standard Deviation*

rol\_mean *=* series.rolling(window).mean()

rol\_std *=* series.rolling(window).std()

*## ploting the results along side the original data*

fig *=* plt.figure(figsize*=*(10,5))

orig *=* plt.plot(series,color*=*'blue',label*=*'Original')

mean *=* plt.plot(rol\_mean,color*=*'red',label*=*'Rolling mean')

std *=* plt.plot(rol\_std,color*=*'black',label*=*'Rolling std')

plt.title('Rolling Mean/Standard Deviation',size*=*20)

plt.legend(loc*=*'best')

plt.show(block*=*False)

def stationarity\_check(series):

print('Results of Dickey Fuller Test:')

coffee\_test *=* adfuller(series, autolag*=*'AIC')

coffee\_output *=* pd.Series(coffee\_test[0:4], index*=*['Test Statistic','p-value',

'#Lags Used','Number of Observations Used'])

*for* key,value *in* coffee\_test[4].items():

coffee\_output['Critical Value (%s)'*%*key] *=* value

print(coffee\_output)

plot\_rolling\_stats(data\_close\_price,4)

stationarity\_check(data\_close\_price)

*## Regular Differentiation*

plot\_rolling\_stats(data\_close\_price.diff()[1:],4)

stationarity\_check(data\_close\_price.diff()[1:])

Autocorrelation and Partial Correlation

fig *=* plt.figure(figsize*=*(20,5))

ax\_1 *=* fig.add\_subplot(121)

plot\_pacf(data\_close\_price,lags*=*12,zero*=*False,ax*=*ax\_1)

ax\_2 *=* fig.add\_subplot(122)

plot\_acf(data\_close\_price,lags*=*12,zero*=*False,ax*=*ax\_2)*;*

Time Series Modeling

size *=* 0.8 *## train size*

train, test *=* data\_close\_price.iloc[:int(size*\**len(data\_close\_price))], data\_close\_price.iloc[int(size*\**len(data\_close\_price)):]

SARIMAX

model *=* SARIMAX(train,order*=*(2,1,2),seasonal\_order*=*(1,1,1,4)).fit(disp*=-*1)

model.summary()

model.plot\_diagnostics(figsize*=*(20,10))

plt.show()

Predictions

predictions *=* model.get\_prediction(start*=*'2000-03-31',end*=*'2022-06-30')

conf *=* predictions.conf\_int()

test\_conf *=* conf.loc[test.index[0]:]

*## ploting results*

plt.plot(predictions.predicted\_mean[1:],color*=*'red',label*=*'predictions')

plt.plot(train,color*=*'blue',label*=*'original')

plt.plot(test,color*=*'green',label*=*'test')

plt.fill\_between(test\_conf.index, test\_conf.iloc[:,0], test\_conf.iloc[:,1], color*=*'gray', alpha*=*.2,label*=*'95% confidence')

plt.title('Original vs Predictions',size*=*20)

plt.legend(loc*=*'best')*;*

Accuracy Metrics

print(f"Mean Absolute Error: {mean\_absolute\_error(data\_close\_price[1:],predictions.predicted\_mean[1:])}")

print(f"Mean Absolute Percentage Error: {mean\_absolute\_percentage\_error(data\_close\_price[1:],predictions.predicted\_mean[1:])}")