|  |  |
| --- | --- |
| Name Of The Student | Kanak Sahu |
| Internship Project Topic | **“RIO-125: Forecasting System - Project Demand of Products at a Retail Outlet Based on Historical Data.”** |
| Name of the Organization | TCS iON |
| Name of the Industry Mentor | Himalaya Ashish |
| Name of the Institute | Symbiosis University of Applied Sciences |

|  |  |  |
| --- | --- | --- |
| Date | Day # | Hours Spent |
| 12/03/21 | Friday(Day-11) | 4 |
| Activities done during the day: Code The statsmodels library provides a suite of functions for working with time series data.  import numpy as np import pandas as pd from matplotlib import pyplot as plt from statsmodels.tsa.stattools import adfuller from statsmodels.tsa.seasonal import seasonal\_decompose from statsmodels.tsa.arima\_model import ARIMA from pandas.plotting import register\_matplotlib\_converters register\_matplotlib\_converters()  We’ll be working with a dataset that contains the number of airplane passengers on a given day.  df = pd.read\_csv('air\_passengers.csv', parse\_dates = ['Month'], index\_col = ['Month'])df.head()plt.xlabel('Date') plt.ylabel('Number of air passengers') plt.plot(df)  Image for post  Image for post  Image for post  Image for post  As mentioned previously, before we can build a model, we must ensure that the time series is stationary. There are two primary way to determine whether a given time series is stationary.   * **Rolling** **Statistics**: Plot the rolling mean and rolling standard deviation. The time series is stationary if they remain constant with time (with the naked eye look to see if the lines are straight and parallel to the x-axis). * **Augmented Dickey-Fuller Test**: The time series is considered stationary if the p-value is low (according to the null hypothesis) and the critical values at 1%, 5%, 10% confidence intervals are as close as possible to the ADF Statistics   For those who don’t understand the difference between average and rolling average, a 10-day rolling average would average out the closing prices for the first 10 days as the first data point. The next data point would drop the earliest price, add the price on day 11 and take the average, and so on as shown below.  Image for post  Image for post  rolling\_mean = df.rolling(window = 12).mean() rolling\_std = df.rolling(window = 12).std()plt.plot(df, color = 'blue', label = 'Original') plt.plot(rolling\_mean, color = 'red', label = 'Rolling Mean') plt.plot(rolling\_std, color = 'black', label = 'Rolling Std') plt.legend(loc = 'best') plt.title('Rolling Mean & Rolling Standard Deviation') plt.show()  Image for post  Image for post  As you can see, the rolling mean and rolling standard deviation increase with time. Therefore, we can conclude that the time series is not stationary.  result = adfuller(df['Passengers'])print('ADF Statistic: {}'.format(result[0])) print('p-value: {}'.format(result[1])) print('Critical Values:') for key, value in result[4].items():  print('\t{}: {}'.format(key, value))  Image for post  Image for post  The ADF Statistic is far from the critical values and the p-value is greater than the threshold (0.05). Thus, we can conclude that the time series is not stationary.  Taking the log of the dependent variable is as simple way of lowering the rate at which rolling mean increases.  df\_log = np.log(df) plt.plot(df\_log)  Image for post  Image for post  Let’s create a function to run the two tests which determine whether a given time series is stationary.  def get\_stationarity(timeseries):    # rolling statistics  rolling\_mean = timeseries.rolling(window=12).mean()  rolling\_std = timeseries.rolling(window=12).std()    # rolling statistics plot  original = plt.plot(timeseries, color='blue', label='Original')  mean = plt.plot(rolling\_mean, color='red', label='Rolling Mean')  std = plt.plot(rolling\_std, color='black', label='Rolling Std')  plt.legend(loc='best')  plt.title('Rolling Mean & Standard Deviation')  plt.show(block=False)    # Dickey–Fuller test:  result = adfuller(timeseries['Passengers'])  print('ADF Statistic: {}'.format(result[0]))  print('p-value: {}'.format(result[1]))  print('Critical Values:')  for key, value in result[4].items():  print('\t{}: {}'.format(key, value))  There are multiple transformations that we can apply to a time series to render it stationary. For instance, we subtract the rolling mean.  rolling\_mean = df\_log.rolling(window=12).mean() df\_log\_minus\_mean = df\_log - rolling\_mean df\_log\_minus\_mean.dropna(inplace=True)get\_stationarity(df\_log\_minus\_mean)  Image for post  Image for post  As we can see, after subtracting the mean, the rolling mean and standard deviation are approximately horizontal. The p-value is below the threshold of 0.05 and the ADF Statistic is close to the critical values. Therefore, the time series is stationary.  Applying exponential decay is another way of transforming a time series such that it is stationary.  rolling\_mean\_exp\_decay = df\_log.ewm(halflife=12, min\_periods=0, adjust=True).mean() df\_log\_exp\_decay = df\_log - rolling\_mean\_exp\_decay df\_log\_exp\_decay.dropna(inplace=True)get\_stationarity(df\_log\_exp\_decay)  Image for post  Image for post  Exponential decay performed worse than subtracting the rolling mean. However, it is still more stationary than the original.  Let’s try one more method to determine whether an even better solution exists. When applying time shifting, we subtract every the point by the one that preceded it.  **null, (x1−x0), (x2−x1), (x3−x2), (x4−x3), …, (xn−xn−1)**  df\_log\_shift = df\_log - df\_log.shift() df\_log\_shift.dropna(inplace=True)get\_stationarity(df\_log\_shift)  Image for post  Image for post  Time shifting performed worse than subtracting the rolling mean. However, it is still more stationary than the original. | | |