Fernando Zambrano
Dr. Jafari
Multivariate Modeling
29 January 2020

Homework 1

1. Load the time series data called AirPassangers.csv.

```
In [10]: # 1 Load the time series data
         df = pd.read_csv("AirPassangers.csv")

In [11]: df
```

Out[11]:

	Month	#Passengers
0	1949-01	112
1	1949-02	118
2	1949-03	132
3	1949-04	129
4	1949-05	121
...	...	...
139	1960-08	606
140	1960-09	508
141	1960-10	461
142	1960-11	390
143	1960-12	432

144 rows × 2 columns

2. This data relates to the air passengers between 1949-1960.
3. Write a Python code to display the first 5 rows of the data.

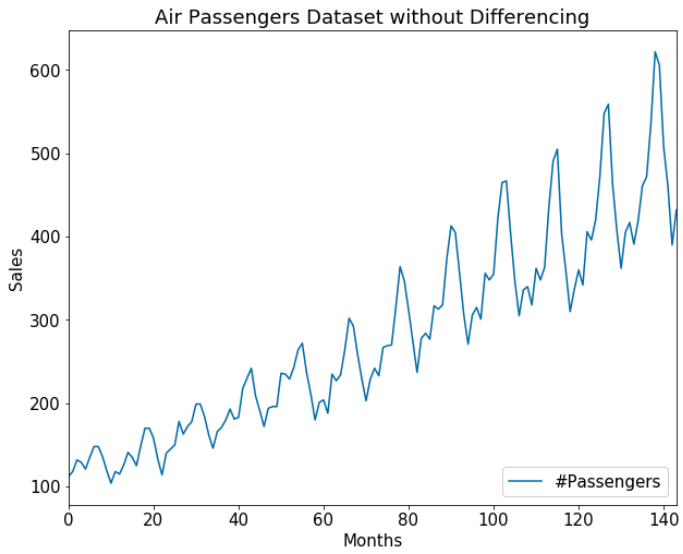
```
In [243]: # 2 Data covers 1949-1960
          # 3 Display the first 5 rows of the data
          df.head(5)
```

Out[243]:

	Month	#Passengers
0	1949-01	112
1	1949-02	118
2	1949-03	132
3	1949-04	129
4	1949-05	121

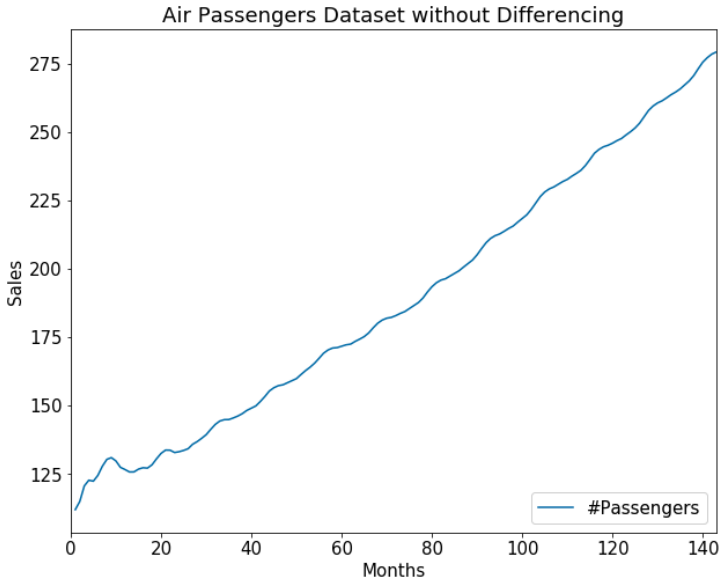
4. Plot entire dataset.

```
ut[244]: Text(0.5, 1.0, 'Air Passengers Dataset without Differencing')
```



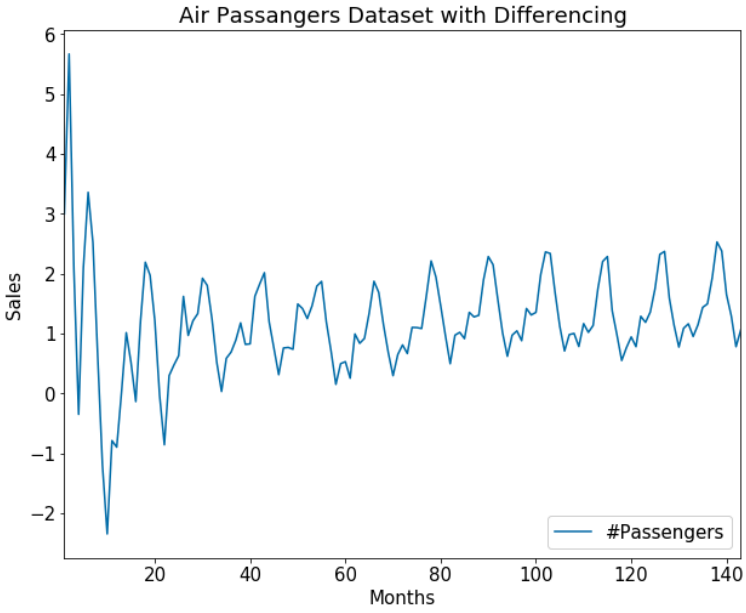
The data demonstrates a strong increasing trend and strong seasonality, but with some weak signs of cyclic behavior. The data has an increasing trend because there is long-term increase of overall sales over time. The data is seasonal because there is spike in sales during summer (July) winter (December) seasons followed with dramatic declines in spring and fall months. The frequency in these spikes occur in fixed frequencies. The data has weak to no cyclic patterns.

5. Is the data stationary or non-stationary? Justify answer. Calculate the average over the entire dataset and show the average plot.

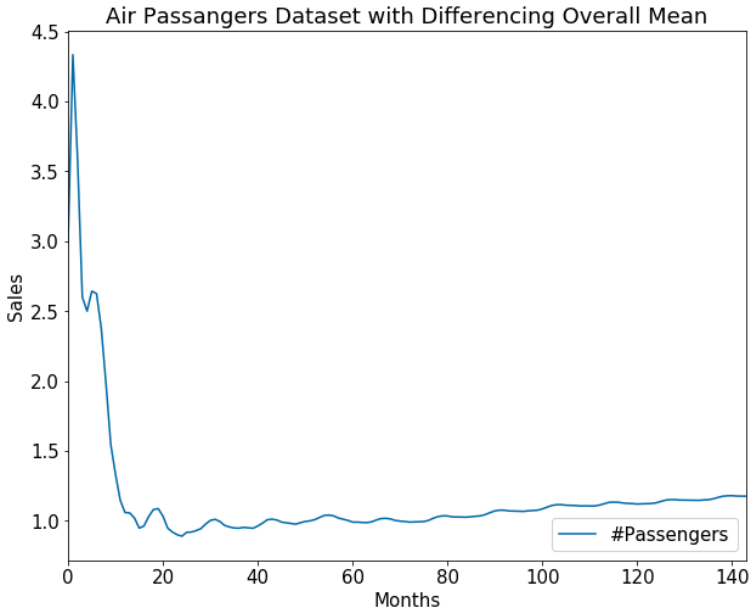


The data is non-stationary. Since there is a clear increasing trend in the mean over time, then this means the data is non-stationary. By performing an ADF test to objectively show that data is either stationary or non-stationary by comparing the resulting p-value with the critical values threshold. Since the p-value (0.991880) is much higher than the critical value threshold (0.01 – 0.05), we fail to reject the null hypothesis that the data is non-stationary. The ADF test support the visual interpretation of the data.

6. If the data is non-stationary, then write a python code that will detrend it by the 1st order difference. Plot the detrended data set.



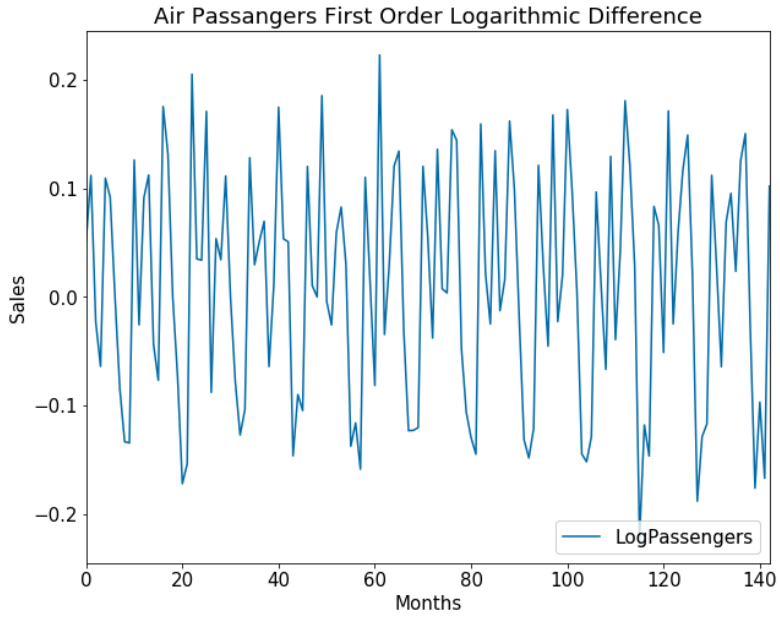
7. Is the detrended dataset stationary? Justify Answer. Calculate average over the entire dataset and show the average plot.



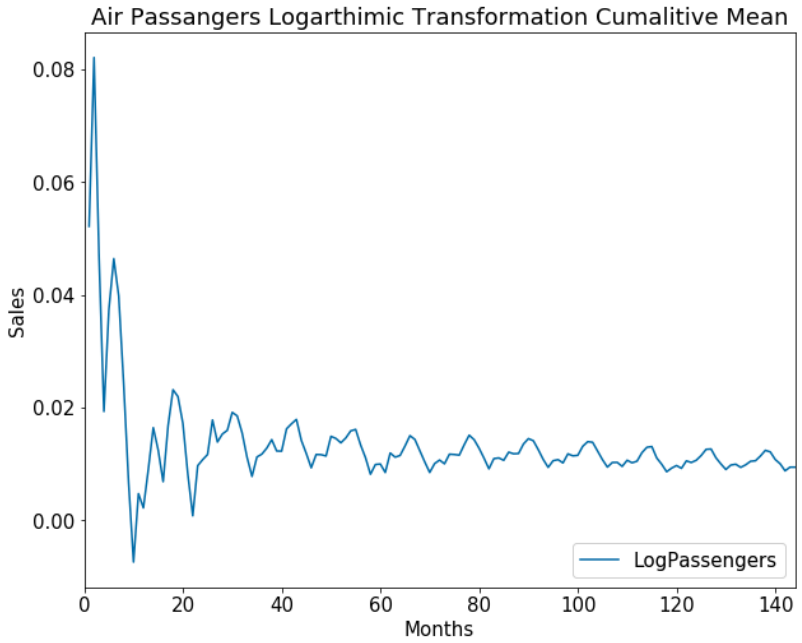
The data is stationary. Since there is no visually clear increasing trend in the average over time, then this means the data is stationary. However, to make sure the visual interpretation is correct, an ADF needs to be applied. By performing an ADF test to objectively show that data is either stationary or non-stationary by comparing the resulting p-value with the critical values threshold. Since the p-value (0.000666) is much lower than the critical value

threshold (0.01 – 0.05), we reject the null hypothesis. Therefore, the null hypothesis is rejected in favor of the alternative; that the data is stationary.

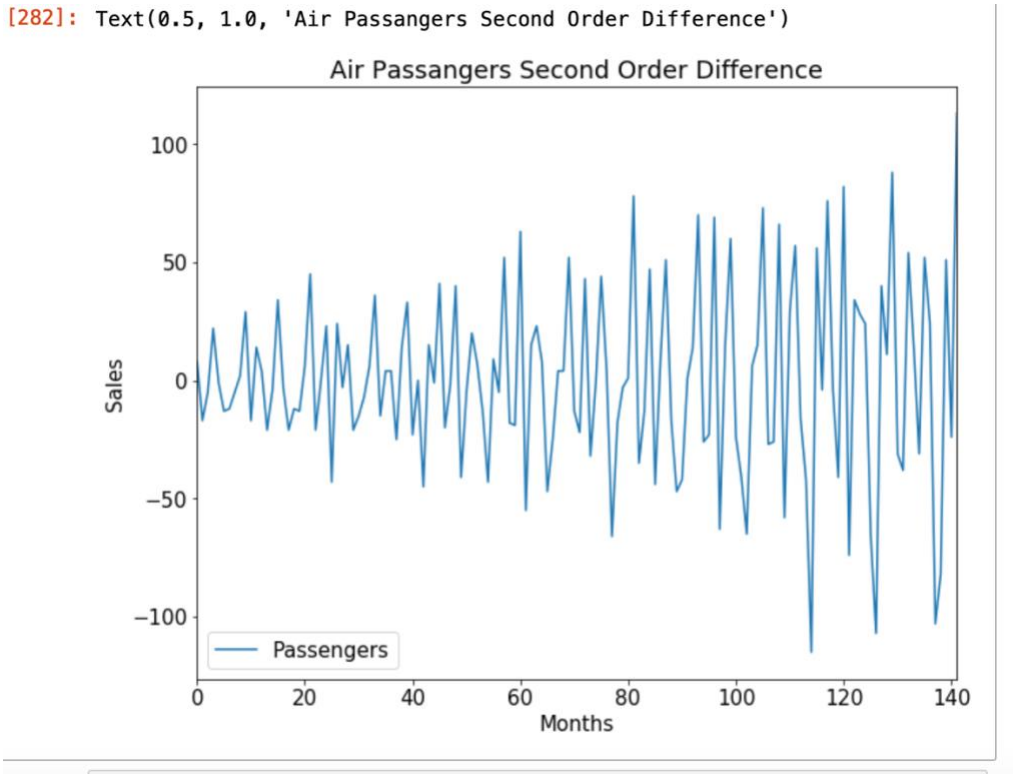
8. Using the logarithmic transformation method, and differencing method, detrend the data. Use NumPy library and convert air passenger numbers in logarithmic scale. Take the difference between two adjacent observations. Plot results.



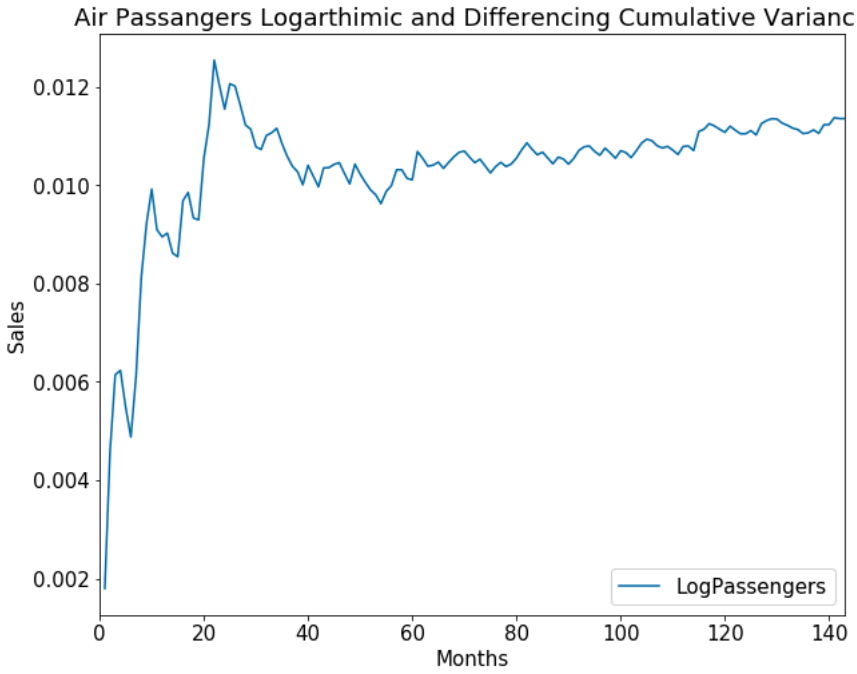
9. Is the transformed data now stationary? Justify answer. Calculate average over the entire dataset and show average plot. Calculate variance and show the transformation converts the non-stationary data into stationary.



The data is non-stationary. Visually, there is a lack of an increasing trend in the average over time which would make the data stationary. However, visual interpretation is subjective. Hence, by performing an ADF test to objectively show that data is either stationary or non-stationary by comparing the resulting p-value with the critical values threshold. Since the p-value (0.071121) is slightly larger than the critical value threshold (0.01 – 0.05), we fail to reject the null hypothesis. This confirms that the data is non-stationary.



However, using the second order differencing of the data, the data does become stationary. The resulting p-value performing an ADF test on the second order differencing set was 0.0000. This means we can reject the null hypothesis in favor of the alternative confirming the data as stationary.



Visually analyzing the cumulative variance of the data, there is a lack of an increasing trend over time which would make the data stationary. However, as the ADF test demonstrated earlier, this data still does not pass the stationary test.

APPENDIX:

PYTHON CODE

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
import numpy as np
import os

def ADF_Cal(x):
    result = adfuller(x)
    print("ADF Statistic: %f" %result[0])
    print("p-value: %f" %result[1])
    print("Critical Values:")
    for key, value in result[4].items():
        print('\t%s: %3f' % (key,value))

print(os.listdir())

# 1 Load the time series data
df = pd.read_csv("AirPassangers.csv")

# 2 Data covers 1949-1960
# 3 Display the first 5 rows of the dat

print(df.head(5))

# 4 Plot the entire dataset
# Is there a trend, seasonality, cyclic
# Add label to horizonatal and vertical axis (month vs sales)
# # Add Title "Air passangers Dataset without differencing"
# Add a legend
```

```
ax = df.plot(kind='line',figsize=(10,8), fontsize=15)
plt.legend(loc='lower right', fontsize=15)
ax.set_xlabel('Months', fontsize=15)
ax.set_ylabel('Sales', fontsize=15)
ax.set_title('Air Passangers Dataset without Differencing',fontsize=18)
plt.show()

# 5 is the data stationary or non-stationary?
# Justify answer.
# Calculate the average over the entire data set and show the average plot

print("Summary Stats:",df.describe())
sales_means = []
for i in range(0,144):
    sales_means.append(df.head(i).mean())
# Convert list into dataframe
df_sales_mean = pd.DataFrame(sales_means)

# Plot average data
ax = df_sales_mean.plot(kind='line',figsize=(10,8), fontsize=15)
plt.legend(loc='lower right', fontsize=15)
ax.set_xlabel('Months', fontsize=15)
ax.set_ylabel('Sales', fontsize=15)
ax.set_title('Air Passengers Dataset Cumulative Average without Differencing',fontsize=18)
plt.show()

# Since there is a clear increasing trend in the mean over time, then this means the data is non-stationary
# Perform ADF test to objectively show that data is either sationary or non-stationary

ADF_Cal(df["#Passengers"])

# 6 If the data is non-stationary then write a python code that will detrend it by the 1st order difference transformation.
# Plot the detrended data set.

# Create a new list with just passenger information
sales_means1 = []
for i in range(0,145):
```

```
    sales_means1.append(df["#Passengers"].head(i).mean())

# Convert passenger list into a numpy array
sm_diff_arr = np.array(sales_means1)
# apply first order differncing on the passenger array
sm_diff = np.diff(sm_diff_arr)
print("Input array  : ", sm_diff_arr)
print("First order difference  : ", sm_diff)

# Convert first order differencing array into a pandas dataframe
df_sm_diff = pd.DataFrame(sm_diff)
# Rename columns
df_sm_diff = df_sm_diff.rename(columns={0: "#Passengers"})
# drop first observation since it is missing due to differencing
df_sm_diff = df_sm_diff.drop([0])

# Plot the first order difference of passenger data
ax = df_sm_diff.plot(kind='line',figsize=(10,8), fontsize=15)
plt.legend(loc='lower right', fontsize=15)
ax.set_xlabel('Months', fontsize=15)
ax.set_ylabel('Sales', fontsize=15)
ax.set_title('Air Passengers Dataset with Differencing',fontsize=18)
plt.show()

# 7 Is the detrended dataset stationary? Justify Answer.

# Since there is no clear increasing trend in the mean over time, then this means the data is stationary.
# Perform ADF test to objectively show that data is either sationary or non-stationary

ADF_Cal(df_sm_diff["#Passengers"])

# Calculate average over the entire dataet and show the average plot
# Change range from 1,145 since we dropped the first obsevation using differencing
sales_means_diff = []
for i in range(1,145):
    sales_means_diff.append(df_sm_diff["#Passengers"].head(i).mean())
```

```
df_sales_cdm = pd.DataFrame(sales_means_diff)
# Rename columns
df_sales_cdm = df_sales_cdm.rename(columns={0: "#Passengers"})

# Plot the first order difference overall mean of passenger data
ax = df_sales_cdm.plot(kind='line',figsize=(10,8), fontsize=15)
plt.legend(loc='lower right', fontsize=15)
ax.set_xlabel('Months', fontsize=15)
ax.set_ylabel('Sales', fontsize=15)
ax.set_title('Air Passengers Dataset Cumulative Average with Differencing',fontsize=18)
plt.show()

# 8 Using the logarithimc transformation mehtod, and differencing method, detrend the data.
# Use numpy library and convert air passenger numbers inot logarthimic scale
# Take the difference between two adjacent observations
# Plot results

# Use Second Order Differencing
# Convert passenger list into a numpy array
first_diff = np.array(df["#Passengers"])
# apply first order differncing on the passenger array
first_diff = np.diff(first_diff)
second_diff = np.diff(first_diff)

df_second_diff = pd.DataFrame(second_diff)
# Rename columns
df_second_diff = df_second_diff.rename(columns={0: "Passengers"})
df_second_diff

# Plot the Second difference of passenger data
ax = df_second_diff.plot(kind='line',figsize=(10,8), fontsize=15)
plt.legend(loc='lower left', fontsize=15)
ax.set_xlabel('Months', fontsize=15)
ax.set_ylabel('Sales', fontsize=15)
ax.set_title('Air Passangers Second Order Difference',fontsize=18)
plt.show()
```

```
ADF_Cal(df_second_diff["Passengers"])


# Create new column taking the log(base 10) of passenger data
df["LogPassengers"] = np.log(df['#Passengers'])

# Plot the logarithmic transformation
# Logoarithmic transformation removes varrying variance
ax = df["LogPassengers"].plot(kind='line',figsize=(10,8), fontsize=15)
plt.legend(loc='lower right', fontsize=15)
ax.set_xlabel('Months', fontsize=15)
ax.set_ylabel('Sales', fontsize=15)
ax.set_title('Air Passangers Logarithmic Transformation',fontsize=18)

# Convert passenger list into a numpy array
log_sm_arr = np.array(df["LogPassengers"])
# apply first order differncing on the passenger array
log_sm_diff = np.diff(log_sm_arr)

df_log_diff = pd.DataFrame(log_sm_diff)
# Rename columns
df_log_diff = df_log_diff.rename(columns={0: "LogPassengers"})

# Plot the logarithmic difference of passenger data
ax = df_log_diff.plot(kind='line',figsize=(10,8), fontsize=15)
plt.legend(loc='lower right', fontsize=15)
ax.set_xlabel('Months', fontsize=15)
ax.set_ylabel('Sales', fontsize=15)
ax.set_title('Air Passangers First Order Logarithmic Difference',fontsize=18)
plt.show()

# 9 is the transformed Data now stationary?
# Justify answer.

# Calculate average over the entire dataset and show average plot.
log_means = []
```

```
for i in range(0,145):
    log_means.append(df_log_diff["LogPassengers"].head(i).mean())

df_log_means = pd.DataFrame(log_means)
# Rename columns
df_log_means = df_log_means.rename(columns={0: "LogPassengers"})

# Plot Log Means
ax = df_log_means.plot(kind='line',figsize=(10,8), fontsize=15)
plt.legend(loc='lower right', fontsize=15)
ax.set_xlabel('Months', fontsize=15)
ax.set_ylabel('Sales', fontsize=15)
ax.set_title('Air Passangers Logarthimic Transformation Cumalitive Mean',fontsize=18)
plt.show()

# Visually, since there is no clear increasing trend in the mean over time, then this means the data is stationary.
# Perform ADF test to objectively show that data is either sationary or non-stationary

ADF_Cal(df_log_diff["LogPassengers"])

# Calcualte variance and show transformation converts the non-stationary data into stationary
log_var = []
for i in range(1, 145):
    log_var.append(df_log_diff["LogPassengers"].head(i).var())

df_log_var = pd.DataFrame(log_var)
# Rename columns
df_log_var = df_log_var.rename(columns={0: "LogPassengers"})
df_log_var = df_log_var.drop([0])

# Plot Log Means
ax = df_log_var.plot(kind='line', figsize=(10, 8), fontsize=15)
plt.legend(loc='lower right', fontsize=15)
ax.set_xlabel('Months', fontsize=15)
ax.set_ylabel('Sales', fontsize=15)
ax.set_title('Air Passangers Logarthimic and Differencing Cumulative Variance', fontsize=18)
plt.show()
```

```
# Perform ADF test on transforemd cumalitive variance
ADF_Cal(df_log_var["LogPassengers"])
```