Faculty of Engineering and Technology

MIT-World Peace University (MIT-WPU)

Faculty of Engineering & Technology
School of Computer Engineering & Technology

<u>CERTIFICATE</u>

This is to certify that the Group-06 of Panel-D, studying in B. Tech, School of Computer Engineering & Technology, Trimester – IX, have successfully completed the Mini Project on

Airlines Passenger Count Forecasting

To my satisfaction and submitted the same during the academic year 2020 – 2021 towards the partial fulfillment of degree of Bachelor of Technology in School of Computer Engineering & Technology under Dr. Vishwanath Karad MIT-World Peace University, Pune.

Prof. Anjali Shejul

Mentor (Artificial Intelligence)

Prof. Dr. Vrushali Kulkarni

Head
School of Computer Engineering & Technology

Group Members

Tanmay Das PD 14 | Madhura Nagle PD 15 | Shriya Padhi PD 25 | Arnav Sinha PD 26

Description:

We have considered a situation where in a given year and month, our task is to predict the number of international airline passengers in units of 1000. We have considered a dataset where our data ranges from January 1949 to December 1960, or 12 years with 144 observations. The Problem Statement is an example of univariate time series forecasting. The term "univariate time series" refers to a time series that consists of single (scalar) observations recorded sequentially over equal time increments.

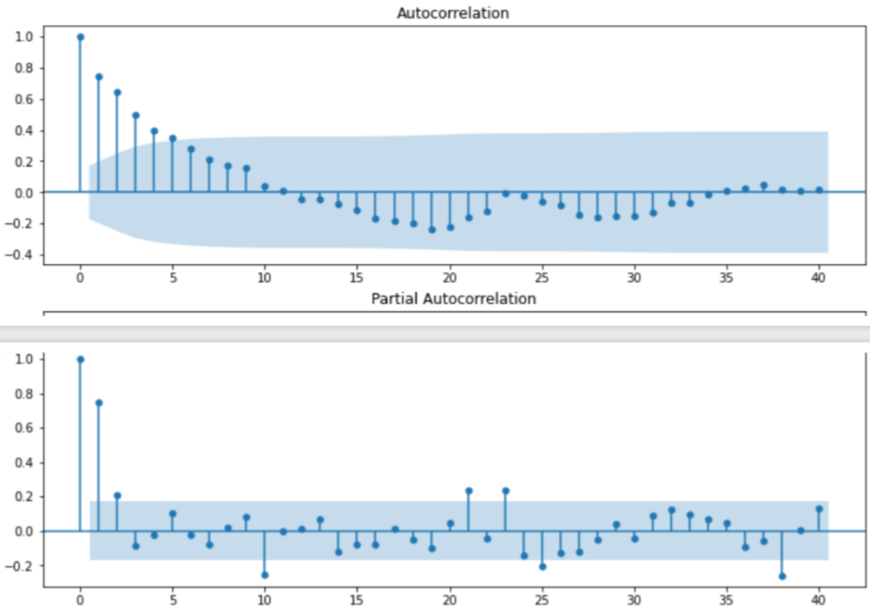
We have used Jupyter Notebook, Python language for implementing LSTM (Long short-term memory), ARIMA (Auto Regressive Integrated Moving Average), Seasonal ARIMA algorithms to predict the number of passengers travelling through a certain airline company.

Code Snippets:

```
model=Sequential() #model used is sequential
model.add(LSTM(50,return_sequences=True,input_shape=(4,1))) #hidden nodes=50, time_steps
=4, num_features=1, add lstm layer
model.add(LSTM(50)) #again adding lstm layer as it is stacked
model.add(Dense(1)) #specifying the output shape using Dense

model.compile(loss='mean_squared_error',optimizer='adam') #compile model

model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=100,batch_size=1,verbose
=1)
#model.fit: fit() is for training the model with the given inputs (and corresponding trai
ning labels).

#So here we get the loss values for every epoch (Fitting done for 100 epochs)
```

```
Epoch 1/100
88/88 [==============================] - 29s 199ms/step - loss: 0.0420 - val_loss: 0.0593
Epoch 2/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0095 - val_loss: 0.0425
Epoch 3/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0070 - val_loss: 0.0291
Epoch 4/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0047 - val_loss: 0.0330
Epoch 5/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0087 - val_loss: 0.0263
Epoch 6/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0050 - val_loss: 0.0263
Epoch 7/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0047 - val_loss: 0.0327
Epoch 8/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0062 - val_loss: 0.0232
Epoch 9/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0045 - val_loss: 0.0254
Epoch 10/100
88/88 [==============================] - 1s 7ms/step - loss: 0.0047 - val_loss: 0.0224
Epoch 11/100
88/88 [==============================] - 1s 6ms/step - loss: 0.0054 - val_loss: 0.0199
Epoch 12/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0051 - val_loss: 0.0247
Epoch 13/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0044 - val_loss: 0.0186
Epoch 14/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0043 - val_loss: 0.0157
Epoch 15/100
88/88 [==============================] - 0s 4ms/step - loss: 0.0038 - val_loss: 0.0851
Epoch 16/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0045 - val_loss: 0.0125
Epoch 17/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0030 - val_loss: 0.0101
Epoch 18/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0024 - val_loss: 0.0117
Epoch 19/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0027 - val_loss: 0.0118
Epoch 20/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0020 - val_loss: 0.0156
Epoch 21/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0033 - val_loss: 0.0068
Epoch 22/100
88/88 [==============================] - 0s 5ms/step - loss: 0.0029 - val_loss: 0.0080
Epoch 23/100
88/88 [==============================] - 1s 7ms/step - loss: 0.0017 - val_loss: 0.0239
```

```python
#passing seasonal first difference data
fig= plt.figure(figsize=(12,8))
ax1=fig.add_subplot(211)
fig=sm.graphics.tsa.plot_acf(df['Seasonal First Difference'].iloc[13:], lags=40,ax=ax1)
#autocorrelation
ax2=fig.add_subplot(212)
fig=sm.graphics.tsa.plot_pacf(df['Seasonal First Difference'].iloc[13:], lags=40,ax=ax2)
#partial autocorrelation

#Why iloc :13, This is because the first 12 values are NaN.
#lags value is taken randomly here
```




```python
model_fit.summary() #overview of the model coefficients and how well they fit, along with several other statistical measures.
```

Out[73]:

ARIMA Model Results

Dep. Variable:	D.#Passengers	**No. Observations:**	143
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	-697.073
Method:	css-mle	**S.D. of innovations**	31.338
Date:	Fri, 11 Jun 2021	**AIC**	1402.145
Time:	23:57:43	**BIC**	1413.997
Sample:	02-01-1949	**HQIC**	1406.961
	- 12-01-1960		

	coef	std err	z	P>|z|	[0.025	0.975]
const	2.6112	0.228	11.435	0.000	2.164	3.059
ar.L1.D.#Passengers	0.7400	0.058	12.778	0.000	0.626	0.854
ma.L1.D.#Passengers	-1.0000	0.019	-53.425	0.000	-1.037	-0.963

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.3513	+0.0000j	1.3513	0.0000
MA.1	1.0000	+0.0000j	1.0000	0.0000

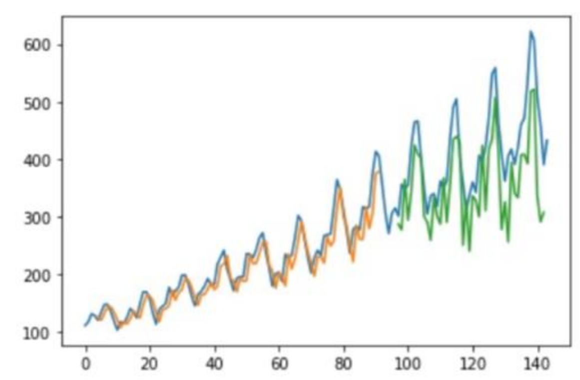
Parameters: Month and passengers

Significant Observation:

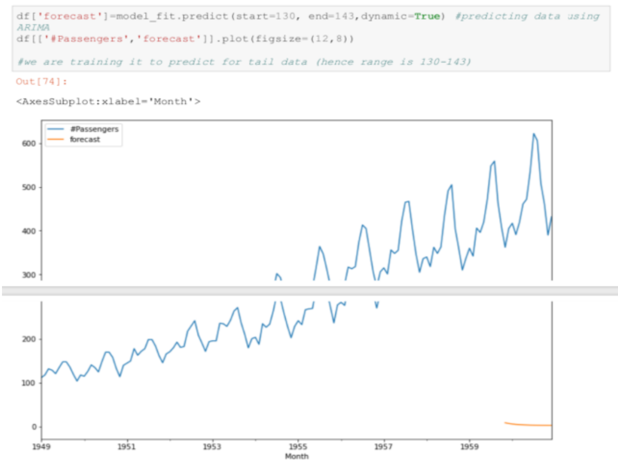
LSTM and ARIMA do not give as accurate results as Seasonal ARIMA model while predicting the number of passengers. ARIMA does not work for seasonal data. A seasonal ARIMA model is formed by including additional seasonal terms in the ARIMA models. The seasonal part of the model consists of terms that are similar to the non-seasonal components of the model, but involve backshifts of the seasonal period. Backshift notation is particularly useful when combining differences. Say quarterly (4 months), annually (12 months). Hence it works better.

Results:

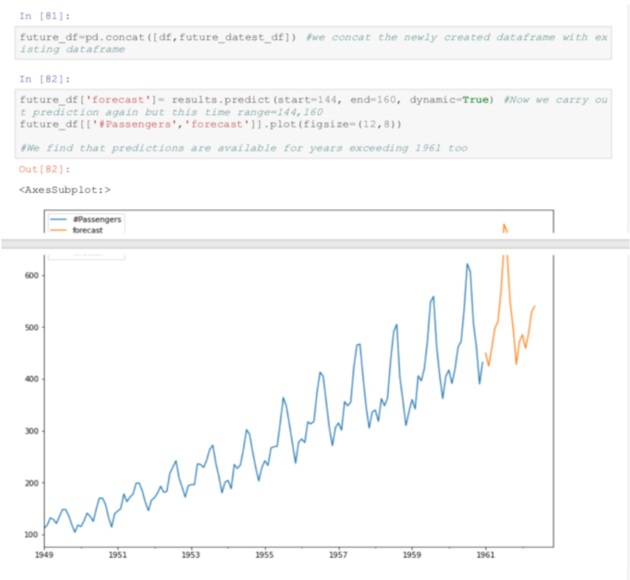
LST



ARIMA Model



Seasonal ARIMA model



Conclusion:

Thus, we have implemented LSTM, ARIMA and Seasonal ARIMA algorithms to predict the number of passengers travelling through a certain airline. Season ARIMA model gives us the most accurate results.