

Indian Institute of Technology Indore
Discipline of Computer Science and Engineering
Minor Project in the course “Computational Intelligence”
Spring 2022-2023

Title: Sales Forecasting Using LSTMs - SARIMA Hybrid Model

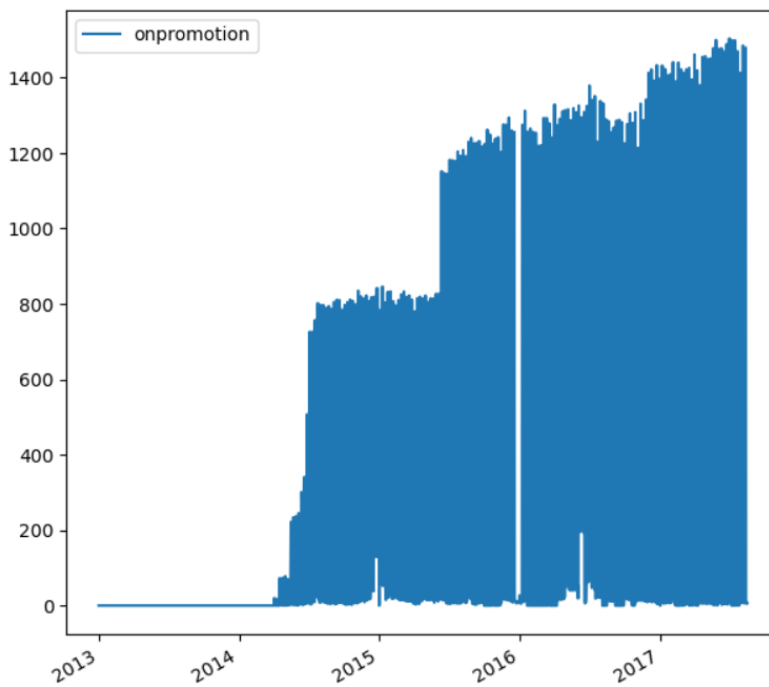
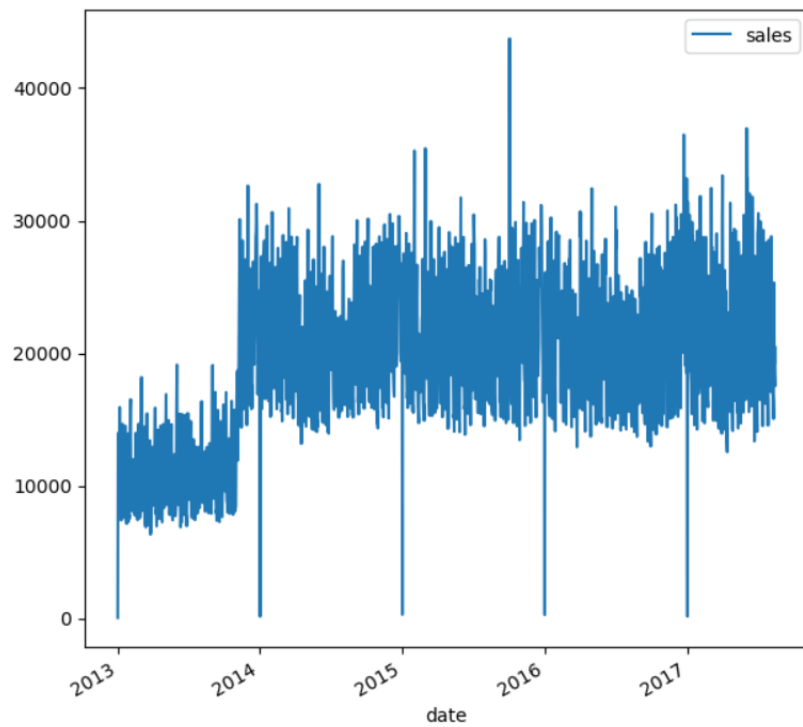
Problem Definition

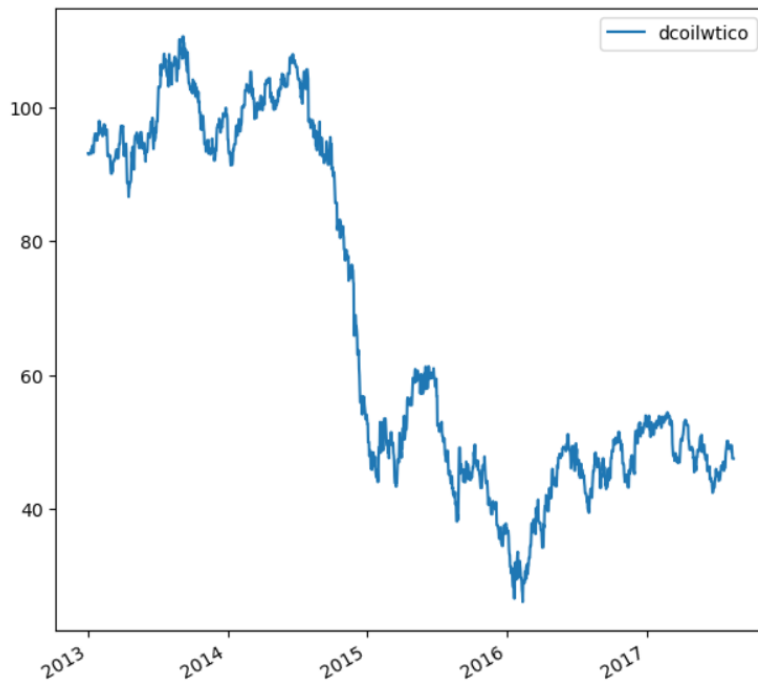
The goal is to predict the future sales of a certain product(s) based on historical sales data and other relevant factors such as marketing efforts, economic conditions and consumer behavior in order to assist in making informed business decisions, using this, businesses can better manage their inventory levels, ensuring that they have the right products in stock at the right time that is planning for future demand by identifying patterns and trends in sales data. Such a system should be able to adapt or adjust sales forecasts for seasonal trends and also special events. Using information from the sizable Ecuadorian grocery retailer Corporación Favorita, we will anticipate store sales using time-series forecasting. More precise forecasting can help grocery shops reduce food waste brought on by overstocking and improve customer happiness.

Data Collection and Preprocessing:

The dataset we shall be using is the ***Corporación Favorita Grocery Sales Forecasting*** provided by the Corporación Favorita, in Kaggle for a features prediction competition. The dataset was decided after an extensive search, to satisfy our problem statement while keeping in mind to obtain diversified multivariate data. The dataset contains the sales of a variety of products from different stores across Ecuador collected over a period of more than 4 years. The dataset also contains which category/family it belongs to, how many items belonging to a family are currently being promoted, and in separate CSV files, also the oil prices across the same time span. The oil prices help provide insight into the country's current economic situation, and provide strong indicators for economic crises like inflation as we are well aware that Ecuador is an oil-dependent country and its economic health is highly vulnerable to shocks in oil prices. We have also taken another CSV file from the dataset containing the Holiday events over the time

span, which helps us incorporate the feature of predicting the effects of holiday events over the sales of our products of interest.



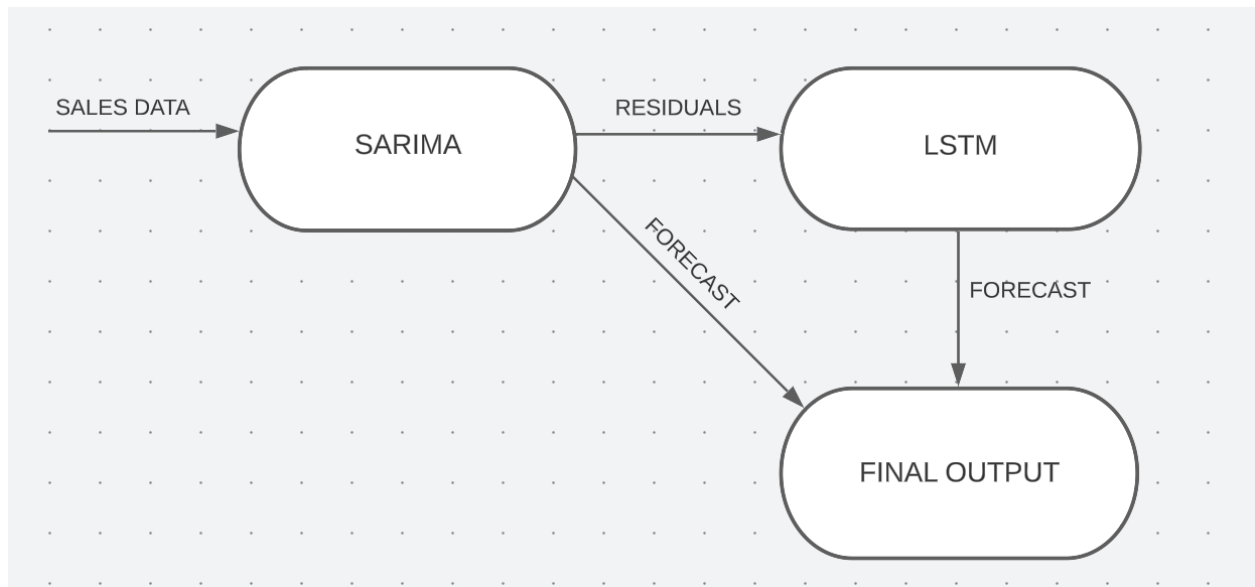


The preprocessing of the dataset was mainly handled using the Pandas Library, which is best known to be used for CSV datasets. Once the basic preprocessing steps like converting the dates into the appropriate format for easy use on our model and taking only the necessary columns required for visualization of the dataset, once completed, we proceeded to group the dataset by aggregating the sales and promotions of a particular category of groceries over all the stores. The promotions were summed up as they can be used to study the effects of promoting some groceries over time. Our next step is to append the oil prices alongside the sales dataset. The oil prices CSV file was originally missing some data on some days, for example, a strike or servicing or holiday. Once appended we modified the dataset to fill any Nan values using pre-existing library functions, using the previous day's prices, so the dataset can maintain its distribution. We then incorporate the holiday events to consider any effects caused by those dates. Finally, we remove any unnecessary columns such as the category name, which would be redundant, and also set the data frame's index to Date column. This dataset is now ready to be normalized and trained as per our model's architectural requirements.

The above steps were repeatedly performed for the different datasets we considered, so as to visualize and properly understand the distribution of the data and make a decision on whether or not to choose that particular dataset.

Algorithm:

To capture the underlying patterns and seasonality of the data, fit a SARIMA model to the time series data. Use the residual errors from the SARIMA model as input to the LSTM model. To identify any remaining patterns or relationships in the data that the SARIMA model missed, train the LSTM model on the residual errors. Combine the predictions from the SARIMA model with the predictions from the LSTM model to make the final forecast. The combination of these two models can provide more accurate forecasting than either model alone.



SARIMA

The SARIMA model works by decomposing a time series into its components: trend, seasonality, and residuals. The trend component represents the long-term changes in the data, while the seasonality component captures the periodic fluctuations that occur at fixed intervals. The residuals represent the random fluctuations in the data that cannot be explained by the trend or seasonality components.

$$\text{SARIMA} \underbrace{(p, d, q)}_{\text{non-seasonal}} \underbrace{(P, D, Q)}_{\text{seasonal}}_m$$

Where m = number of observations per year.

Reference: <https://morioh.com/p/f22da5e3137f>

The SARIMA model consists of the following components:

Autoregression (AR) component: This component models the relationship between an observation and a number of lagged observations. It assumes that the value of the time series at any given time is a linear combination of its past values. The number of lagged observations is denoted by the parameter p .

Integrated (I) component: This component is used to make the time series stationary by differencing it. It assumes that the difference between the current value and the value at the previous time step is a stationary time series. The number of differencing operations is denoted by the parameter d .

Moving Average (MA) component: This component models the relationship between the error term and a number of lagged error terms. It assumes that the error term at any given time is a linear combination of its past values. The number of lagged error terms is denoted by the parameter q .

Seasonal Autoregression (SAR) component: This component models the seasonal relationship between an observation and a number of lagged observations at fixed intervals. It assumes that the value of the time series at any given time is a linear combination of its past values at the same seasonal interval. The number of lagged seasonal observations is denoted by the parameter P .

Seasonal Moving Average (SMA) component: This component models the seasonal relationship between the error term and a number of lagged error terms at fixed intervals. It assumes that the error term at any given time is a linear combination of its past error terms at the same seasonal interval. The number of lagged seasonal error terms is denoted by the parameter Q .

The parameters p , d , and q are used to model the non-seasonal component of the time series, while the parameters P , D , and Q are used to model the seasonal component of the time series.

LSTM

The architecture of an LSTM (Long Short-Term Memory) network is based on a cell that has the ability to maintain a memory state over time and selectively update or erase this memory. The basic structure of the LSTM architecture includes three main components: the input gate, the forget gate, and the output gate. These gates control the flow of information into and out of the cell, and are modulated by activation functions that enable the LSTM to learn long-term dependencies in a sequence.

The input gate is responsible for determining how much new information to allow into the cell. It consists of a sigmoid activation function that takes as input the current input and the previous hidden state, and outputs a value between 0 and 1 that represents the amount of information to let in. This information is then combined with a candidate cell state, which is the new information that we want to store in the cell, and is computed using a tanh activation function. The resulting combination of the input and candidate cell state is then added to the previous cell state to obtain the updated cell state.

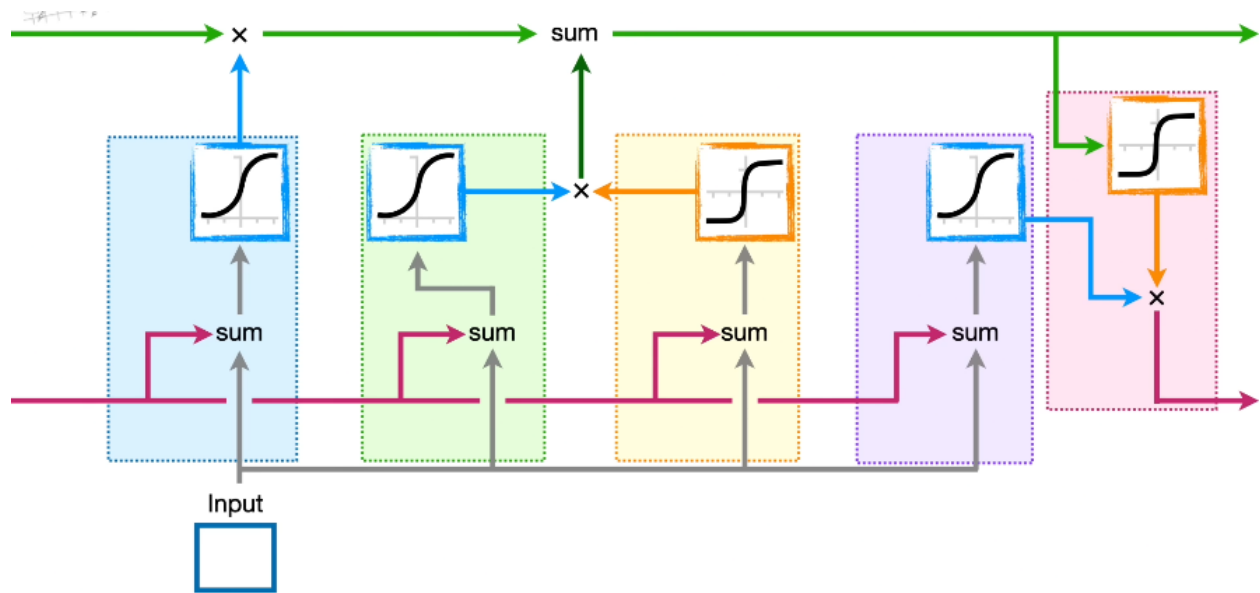
The forget gate is responsible for determining how much old information to erase from the cell. It also consists of a sigmoid activation function that takes as input the current input and the previous hidden state, and outputs a value between 0 and 1 that represents the amount of information to forget. This value is then multiplied element-wise with the previous cell state, erasing the parts of the cell state that are deemed unimportant.

The output gate is responsible for determining how much of the updated cell state to output as the new hidden state. It consists of a sigmoid activation function that takes as input the current input and the previous hidden state, and outputs a value between 0 and 1 that represents the amount of information to output. This value is then multiplied element-wise with the tanh of the updated cell state, producing the new hidden state.

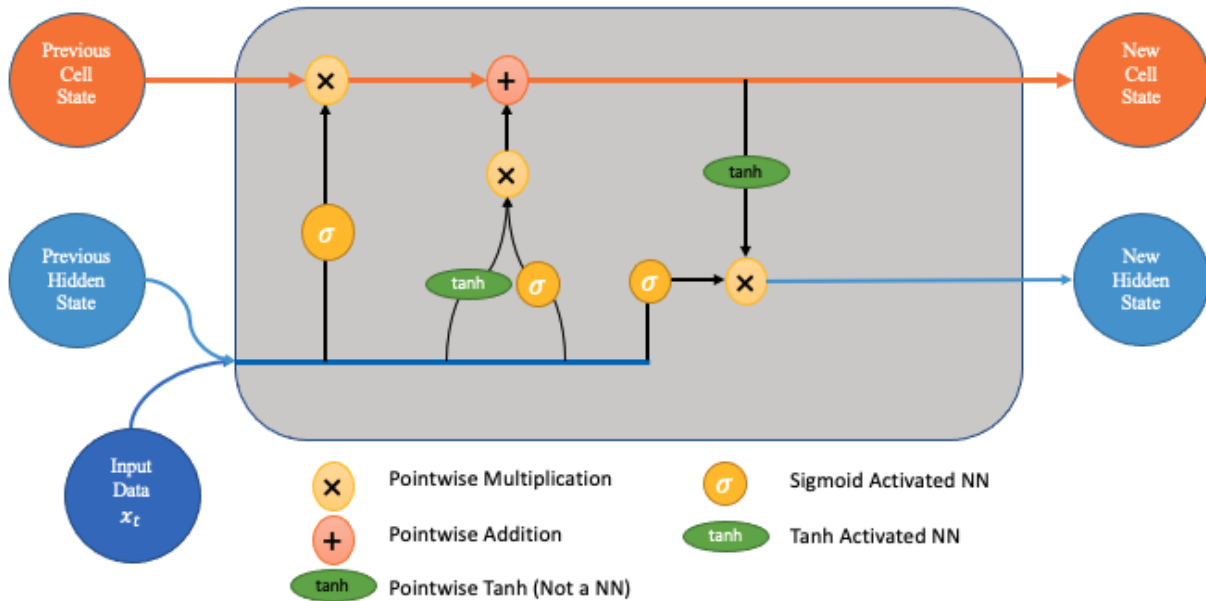
In addition to these three gates, the LSTM architecture also includes a peephole connection, which allows the cell state to directly influence the input and forget gates. This enables the

LSTM to more selectively control the flow of information into and out of the cell and can improve the performance of the network.

The LSTM architecture is well-suited to a wide range of sequence modeling tasks, as it allows the network to selectively remember or forget information over long sequences, making it a powerful tool for tasks such as natural language processing, speech recognition, and time series prediction.



Reference: <https://statquest.org/long-short-term-memory-lstm-clearly-explained/>



Reference https://miro.medium.com/max/674/1*jikKbzFXCq-IYnFZankIMg.png

Performance measurement criteria:

Mean Absolute Error (MAE): This measures the average absolute difference between the predicted sales and the actual sales. A lower MAE indicates better performance.

Root Mean Squared Error (RMSE): This measures the square root of the average of the squared differences between the predicted sales and the actual sales. RMSE is useful because it puts more weight on larger errors. A lower RMSE indicates better performance.

Mean Absolute Percentage Error (MAPE): This measures the average percentage difference between the predicted sales and the actual sales. MAPE is useful for comparing the performance of models on different scales. A lower MAPE indicates better performance.

The data after the pre processing step is now to be trained using the SARIMA model, to do so we need to identify the seasonality or trends and calculate the parameters used for training the SARIMA model on our dataset, to do so, we use the ADF test as described below

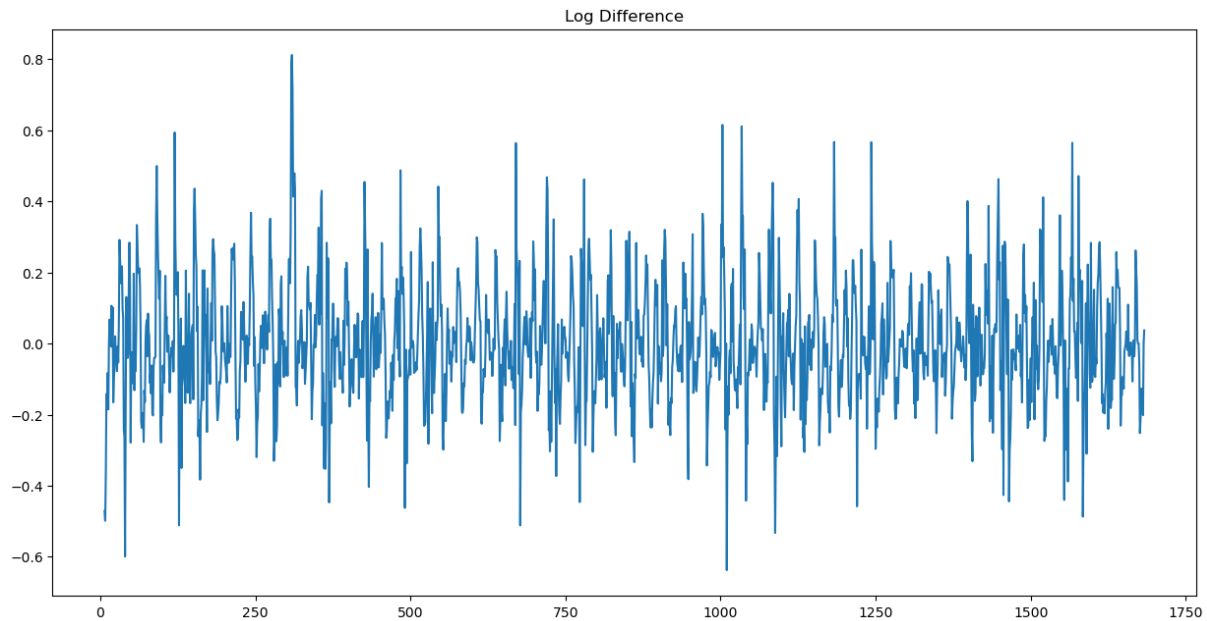
5.3.2 Augmented Dickey-Fuller test

The Augmented Dickey-Fuller test allows for higher-order autoregressive processes by including Δy_{t-p} in the model. But our test is still if $\gamma = 0$.

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \delta_2 \Delta y_{t-2} + \dots$$

The null hypothesis for both tests is that the data are non-stationary. We want to REJECT the null hypothesis for this test, so we want a p-value of less than 0.05 (or smaller).

We use the imported ADF testing libraries to test for the p-value and plot the Autocorrelation graphs The logarithmic differencing is done for the data until the ADF test is passed. The log difference for the sales data is as shown in the graph below, the ADF test is computed on this dataframe to test for the p-value (as seen above compute the new p-value, which now lies in the required range that is, less than 0.05.)



The ADF test before and after log differencing as shown below:

```
ad_fuller_result = adfuller(ndf['sales'])
print(f'ADF Statistic: {ad_fuller_result[0]}')
print(f'p-value: {ad_fuller_result[1]}')
```

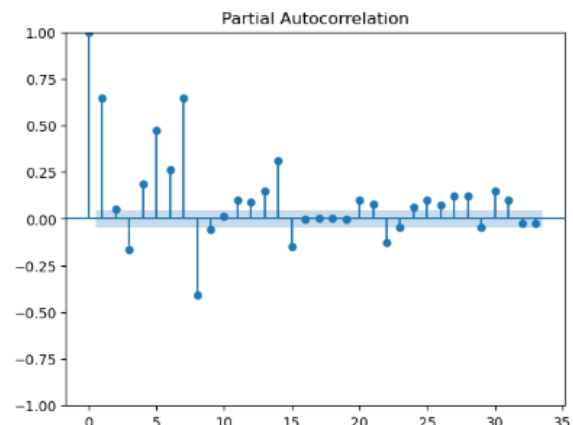
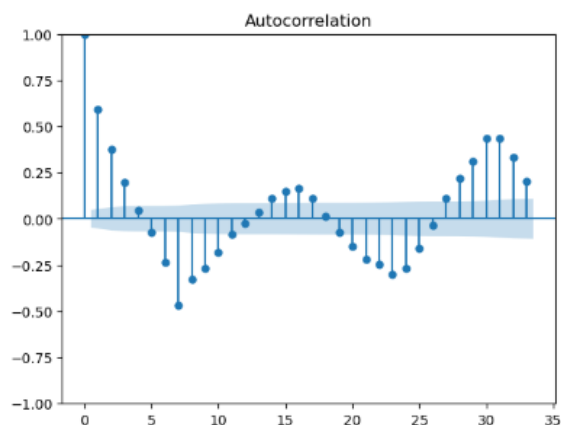
```
ADF Statistic: -2.423203535013838
p-value: 0.1352941873173153
```

After log differencing the p value is well below the required value of 0.05

```
ndf1=ndf1.dropna()
ad_fuller_result = adfuller(ndf1['sales'])
print(f'ADF Statistic: {ad_fuller_result[0]}')
print(f'p-value: {ad_fuller_result[1]}')
```

```
ADF Statistic: -15.142806996147565
p-value: 6.972022187682759e-28
```

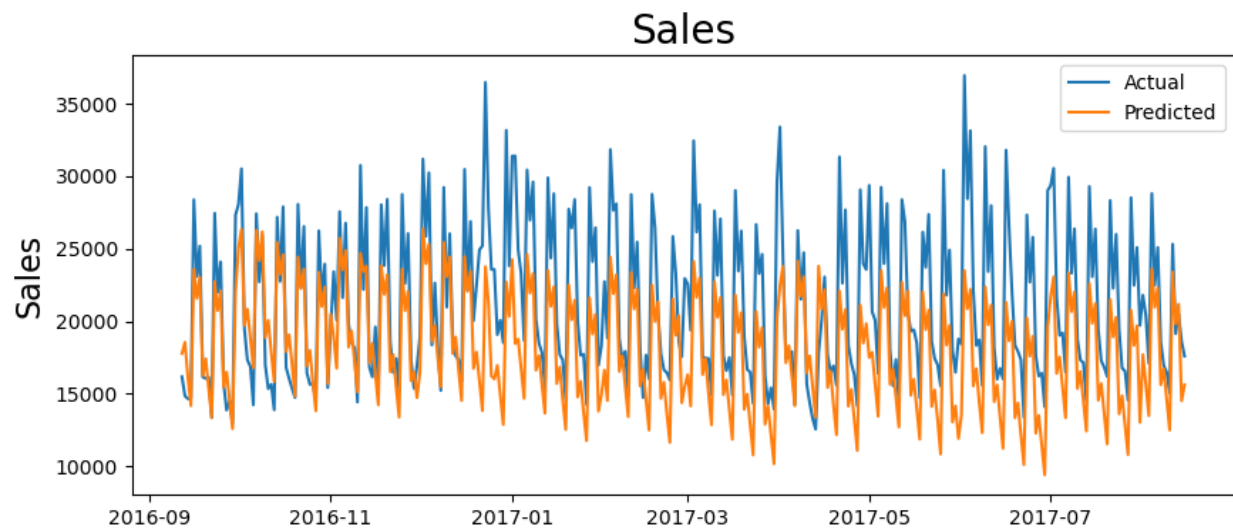
The Autocorrelation and partial Autocorrelation graphs help us calculate the exogenous parameters for the training model, we need to set the proper parameters for a given dataset without which the model may not converge.



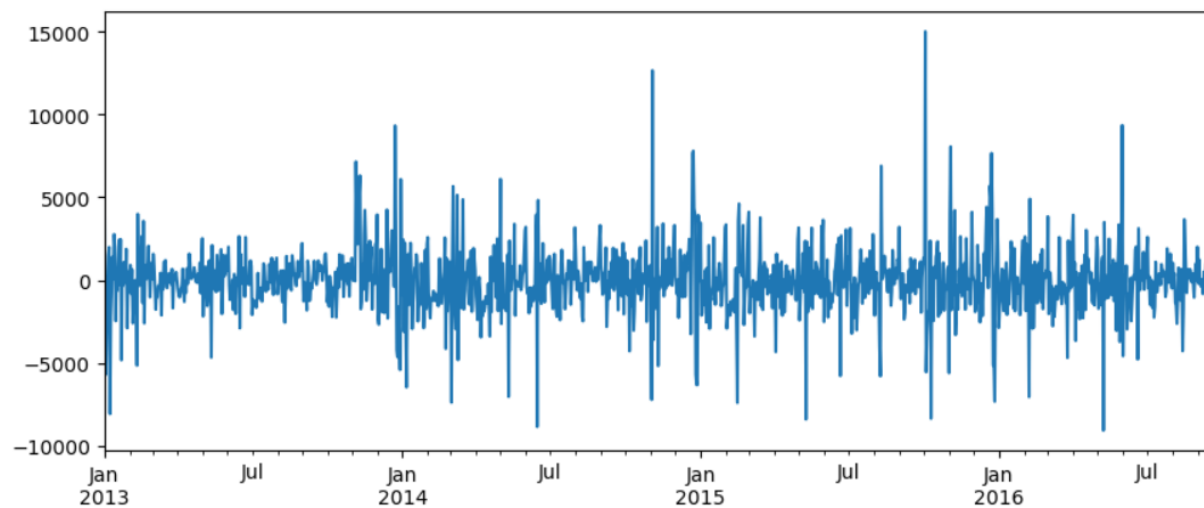
From the above graphs, we get our exogenous parameters as follows:

```
order=(1, 1, 1), seasonal_order=(3, 1, 1, 7)
```

Test data predictions using SARIMA (RMSLE : 0.214, RMSE : 4446):



Residual Plot from training data predicted using SARIMA



Now training the LSTM separately we find out noticeable deviations from the expected values. Also upon closer notice both the trend and magnitude expected is far from the predicted. So it stands to reason from the graph before that we need to include better features as in what we are about to do in our Hybrid model by including the residuals from SARIMA model as a feature in training data for LSTM, this can be done by merging the residuals into the training dataset.

The sales data after introducing the new feature ‘residuals’ merged to the original training data with the residuals obtained from training the SARIMA model gives the following new train dataframe shown below:

date	sales	onpromotion	national_holiday_type	dayofweek	quarter	month	year	dayofyear	dcoilwtico	residuals
2016-09-12	16167.302992	0	0	0	3	9	2016	256	46.28	8099.384132
2016-09-13	14840.499920	0	0	1	3	9	2016	257	44.91	-1944.143900
2016-09-14	14646.700994	1	0	2	3	9	2016	258	43.62	-5683.314011
2016-09-15	14658.210000	0	0	3	3	9	2016	259	43.85	-2748.576346
2016-09-16	28385.163000	1288	0	4	3	9	2016	260	43.04	-769.669652
...
2017-08-11	25318.297990	1480	5	4	3	8	2017	223	48.81	2170.790102
2017-08-12	19134.510058	6	0	5	3	8	2017	224	47.59	3954.637335
2017-08-13	20509.265004	10	0	6	3	8	2017	225	47.59	-2356.973909
2017-08-14	18597.508060	7	0	0	3	8	2017	226	47.59	-2666.321012
2017-08-15	17586.709986	6	0	1	3	8	2017	227	47.57	-1171.015497

We now supply this data to our LSTM Model after applying the StandardScalar transforms

The use of LSTMs for time-series data is widely known, as such several different architectures can be tried out, depending on the dataset. We shall be using the pre-defined Keras LSTM libraries to define the architecture of the LSTM for training on the dataset. LSTMs use tanh activation which makes the model sensitive to magnitudes, so proper preprocessing steps are to be followed, so that the model can converge on the data.

```
residuals1=pd.DataFrame(residuals)
residuals1=residuals1.reset_index()
residuals1.columns=['date', 'sales']
s=s.reset_index()
s.rename(columns={'index': 'date'}, inplace=True)
s['residuals']=residuals1['sales']
```

```
residuals_pred=pd.DataFrame(residuals)
residuals_pred=residuals_pred.reset_index()
residuals_pred.columns=['date', 'sales']
s1=s1.reset_index()
s1.rename(columns={'index': 'date'}, inplace=True)
s1['residuals']=residuals_pred['sales']
```

The Hybrid Model's LSTM architecture has been decided through a series of trial and error, based on the requirements of the dataset being used, among the experimented architectures, the best is as follows: We have used a total of 3 LSTM layers including dropout layers (to avoid overfitting).

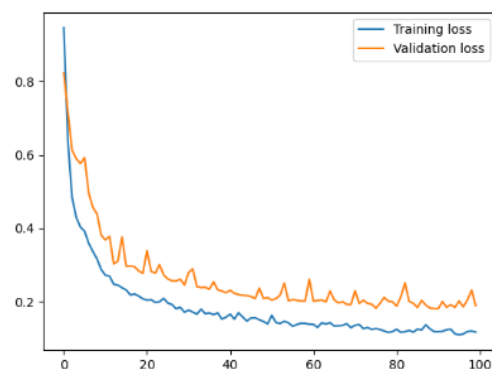
Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 7, 64)	19200
lstm_1 (LSTM)	(None, 7, 32)	12416
dropout (Dropout)	(None, 7, 32)	0
lstm_2 (LSTM)	(None, 16)	3136
dropout_1 (Dropout)	(None, 16)	0
dense (Dense)	(None, 1)	17
=====		
Total params: 34,769		
Trainable params: 34,769		
Non-trainable params: 0		

As noticeable, the output shape from the LSTM layers is 7 x 64 and so on, the 7 here indicates our parameter we set earlier where we decided to use 7 days data from the past to predict one day in the future. We now simply have to train the LSTM model with some parameters.

We are using the Mean Squared Error as the metric to compile and train our model, using the built-in Adam Optimizer for 100 epochs, with a batch size of 64 and a validation split of 0.2. The variation of training loss and validation loss is as shown in the graph to the right. Epochs, batch size and architecture were adjusted so as to not underfit or overfit the data.

```
# model fitting
history = model.fit(trainX, trainY,
                    epochs=100,
                    batch_size=64,
                    validation_split=0.2,
                    verbose=1)
```

Final Loss loss: 0.1129 - val_loss: 0.2035
At 100 Epochs



The training converges within 100 epochs as noticeable from the above graph.

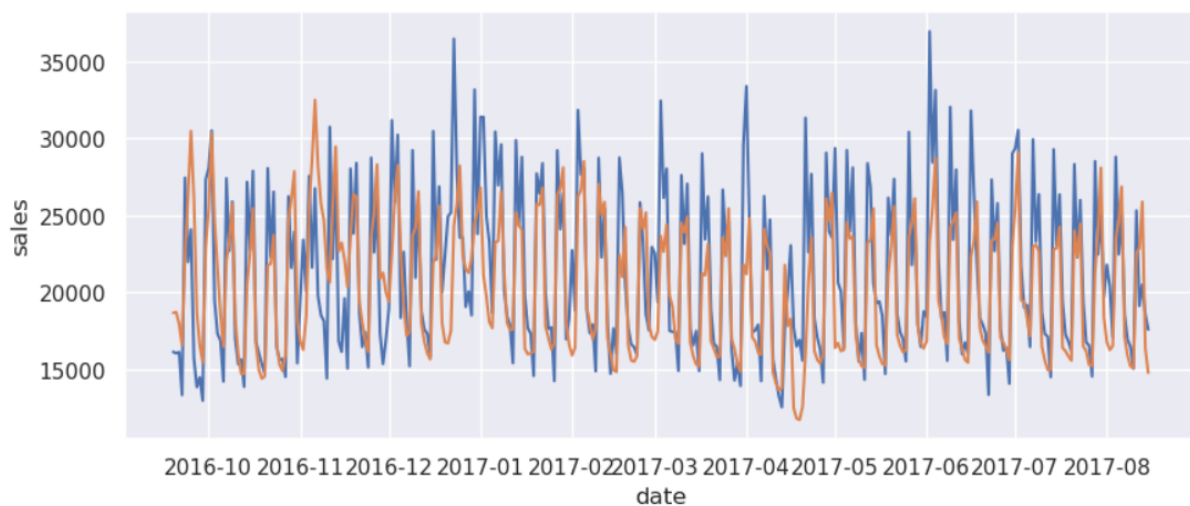
The sales data trained on our Hybrid LSTM - SARIMA Model, gives the following Root Mean Squared Error, for its predictions on the last 330 days (test data) :

```
lstmresidualsquared=np.square(lstmresiduals)
rmseror=np.sqrt(np.sum(lstmresidualsquared)/331)
rmseror
```

3804.769014253311

As seen, the RMS Error is much better than both the SARIMA and LSTM models trained independently. This confirms our hypothesis that the Hybrid Model is better performing than the independent models. Let's now check the plot on test data.

Test data predictions using LSTM with residuals (RMSLE : 0.168, RMSE : 3804)


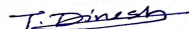


We have incorporated better trend and closer prediction values to that expected from test data.

References:

1. Fattah J, Ezzine L, Aman Z, El Moussami H, Lachhab A. Forecasting of demand using ARIMA model. International Journal of Engineering Business Management. 2018;10. doi:10.1177/1847979018808673
2. Woo, Gerald, et al. "ETSformer: Exponential Smoothing Transformers for Time-series Forecasting." arXiv preprint arXiv:2202.01381 (2022).
3. Zunic, Emir, et al. "Comparison Analysis of Facebook's Prophet, Amazon's DeepAR+ and CNN-QR Algorithms for Successful Real-World Sales Forecasting." *arXiv preprint arXiv:2105.00694* (2021).
4. Vallés-Pérez, Iván, et al. "Approaching sales forecasting using recurrent neural networks and transformers." Expert Systems with Applications 201 (2022): 116993.

Team Members:

Byri Sahas Reddy	200001017	
Tadi Dinesh Kumar Reddy	200001075	

**Under the supervision of
Dr. Aruna Tiwari
Professor, CSE.**