CDS513: Predictive Business Analytics
Academic Session: Semester 2, 2022/2023
School of Computer Sciences, USM, Penang

Topic 2: Walmart Sales Forecasting Systems

Assignment 2

Ahmed Adel Sanhan Al-Haidary - P-COM0113/22

Table of Contents

Description of the problem

Walmart is the largest retailer in the world, employing over 2 million people and operating stores in 28 countries. Despite its global reach, approximately 70% of its sales come from the domestic market in the United States. In a remarkably short time, Walmart achieved significant milestones, reaching sales of 78 million USD within a decade, going public, and opening its first distribution centre. By 1980, its sales exceeded 1 billion USD, solidifying its position as a retail giant (Walton, 2012).

In the context of Walmart's expansive retail operations, maintaining an optimal inventory level is paramount to ensure consistent customer satisfaction while controlling costs. A delicate balance must be struck between the satisfaction of client demand and the avoidance of risks associated with over-storage. On the other hand, reducing inventory levels can lead to cost savings but may result in missed sales opportunities and diminished customer satisfaction. Precise sales forecasting plays a pivotal role in navigating this trade-off effectively, enabling the determination of appropriate inventory levels and mitigating the potential pitfalls of understocking or overstocking(Lu et al., 2012).

The impact of sales forecasting extends beyond inventory management, permeating various facets of business operations. It directly influences critical aspects such as financial planning, marketing strategies, client management, and overall business performance. As a result, enhancing the accuracy of sales forecasts has emerged as a fundamental requirement for Walmart to operate efficiently and optimize its market potential. By improving forecasting techniques, Walmart can proactively respond to customer demand, minimize inventory-related costs, and maximize its sales performance, ultimately positioning itself as an industry leader in the fiercely competitive retail landscape.

In conclusion, this paper aims to forecast the weekly sales for Walmart using the Walmart sales dataset obtained from Kaggle (UJJWAL CHOWDHURY, 2022). By employing methods such as ARIMA and machine learning, we seek to develop accurate sales prediction models. The findings of this research will contribute to enhancing inventory management strategies and optimizing business performance for Walmart.

Data Plotting

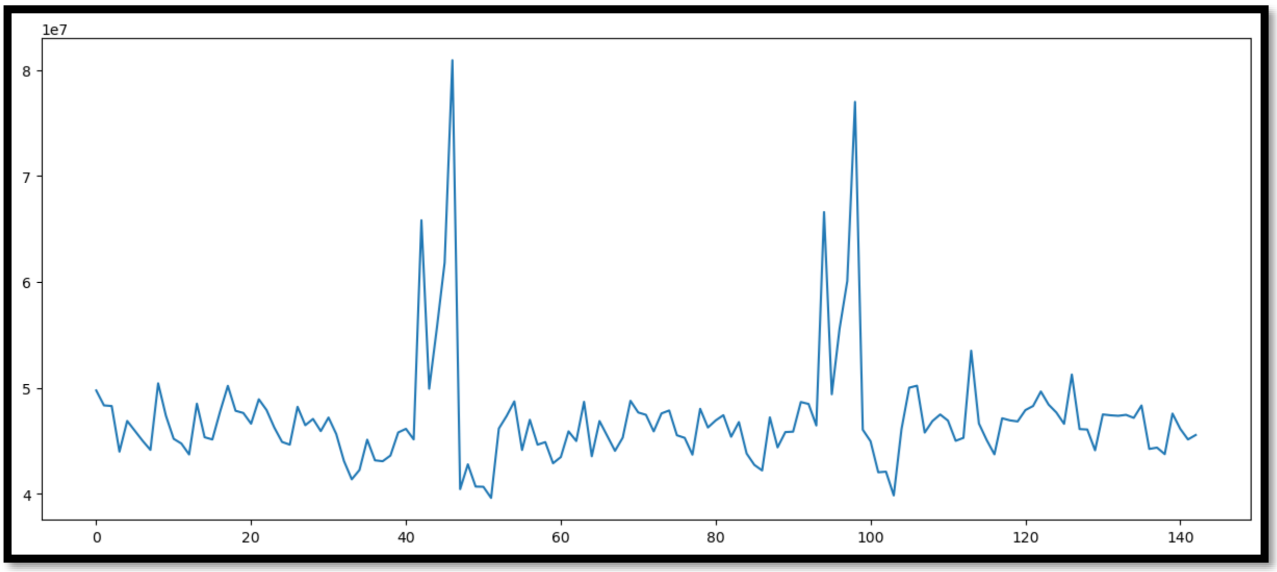
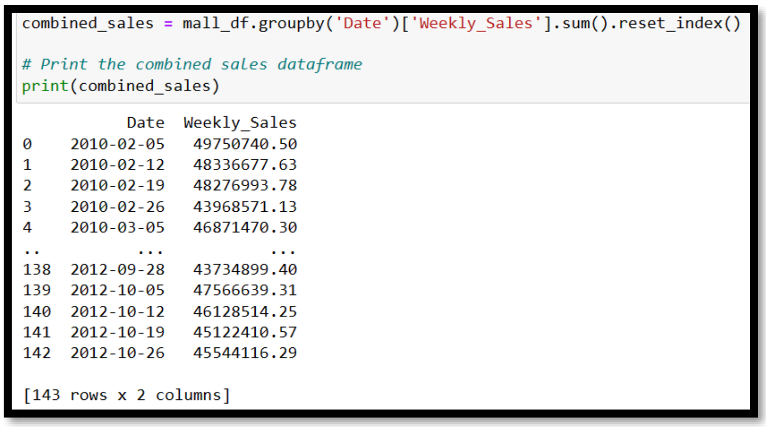


Figure1: Data Time Series Plotting.

As shown from the figure above the dataset doesn’t have a clear trend. However, being a compilation of weekly sales data, it is evident that there exists a noticeable seasonal pattern.

Data Preprocessing

Several actions have been taken to pre-process the data set. Given that the dataset contains weekly sales for 45 stores, all weekly sales had to be combined to provide an overview of the overall sales trend. This was achieved by aggregating the sales data from all stores.

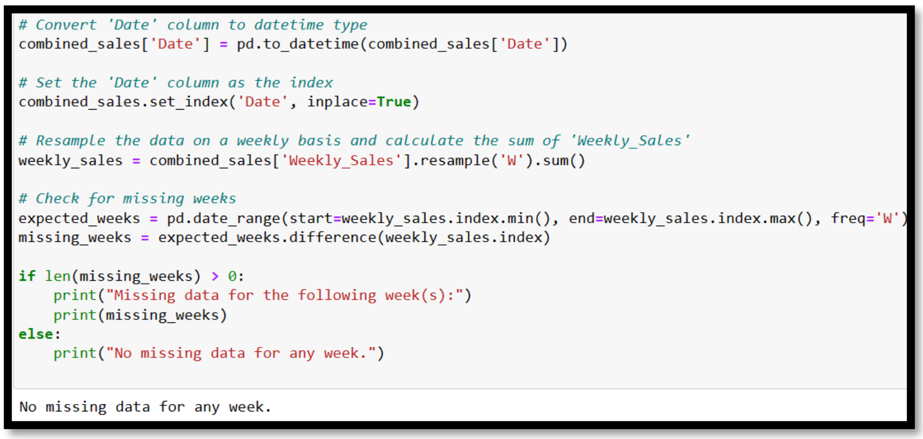


```
combined_sales = mall_df.groupby('Date')['Weekly_Sales'].sum().reset_index()

# Print the combined sales dataframe
print(combined_sales)
```

```
           Date  Weekly_Sales
0    2010-02-05   49750740.50
1    2010-02-12   48336677.63
2    2010-02-19   48276993.78
3    2010-02-26   43968571.13
4    2010-03-05   46871470.30
..          ...           ...
138  2012-09-28   43734899.40
139  2012-10-05   47566639.31
140  2012-10-12   46128514.25
141  2012-10-19   45122410.57
142  2012-10-26   45544116.29

[143 rows x 2 columns]
```

Figure 2: Aggregation of The Sales

Next, it was crucial to address any missing values in the dataset. To ensure accurate forecasting of the time series, a continuous dataset without any missing values was required. Accordingly, a thorough check of the missing values was carried out. As shown in the following figure, we found that there were no missing values. So we don't have to act in this regard.



```
# Convert 'Date' column to datetime type
combined_sales['Date'] = pd.to_datetime(combined_sales['Date'])

# Set the 'Date' column as the index
combined_sales.set_index('Date', inplace=True)

# Resample the data on a weekly basis and calculate the sum of 'Weekly_Sales'
weekly_sales = combined_sales['Weekly_Sales'].resample('W').sum()

# Check for missing weeks
expected_weeks = pd.date_range(start=weekly_sales.index.min(), end=weekly_sales.index.max(), freq='W')
missing_weeks = expected_weeks.difference(weekly_sales.index)

if len(missing_weeks) > 0:
    print("Missing data for the following week(s):")
    print(missing_weeks)
else:
    print("No missing data for any week.")
```

```
No missing data for any week.
```

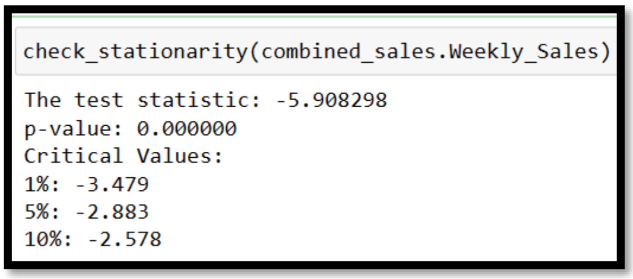
Figure 3: Checking Missing Values of The Data

Data Stationarity

The assumption of stationarity is essential for many time series models, including statistical models like ARIMA and machine learning models, to provide reliable results. Therefore, it was necessary to verify the stationarity of the dataset.

To accomplish this, the Dickey-Fuller (ADF) test was performed. The ADF test calculates a test statistic that is compared to critical values to determine the significance of the results. If the test statistic is below the critical value, there is evidence to reject the null hypothesis and conclude that the time series is stationary. On the other hand, if the test statistic is higher than the critical value, there is insufficient evidence to reject the null hypothesis, indicating that the time series is non-stationary.

From the conducted ADF test, the obtained p-value was found to be 0, which is less than the commonly used significance level of 0.05. This result indicates that the dataset is stationary. However, as it was a prerequisite for this assignment to perform a stationary transformation, the natural logarithm (log) was applied to the dataset. This logarithmic transformation stabilizes the data further and helps achieve stationarity.



```
check_stationarity(combined_sales.Weekly_Sales)
```

```
The test statistic: -5.908298
p-value: 0.000000
Critical Values:
1%: -3.479
5%: -2.883
10%: -2.578
```

Figure 4: ADF Result

The figure below illustrates the dataset after the log transformation, showing a more level and stabilized pattern with no clear trend. This visual observation aligns with the concept of a stationary dataset. Subsequently, another ADF test was conducted on the transformed dataset, and the resulting p-value was once again found to be 0, reaffirming the stationary nature of the dataset. Therefore, no further transformations were deemed necessary.

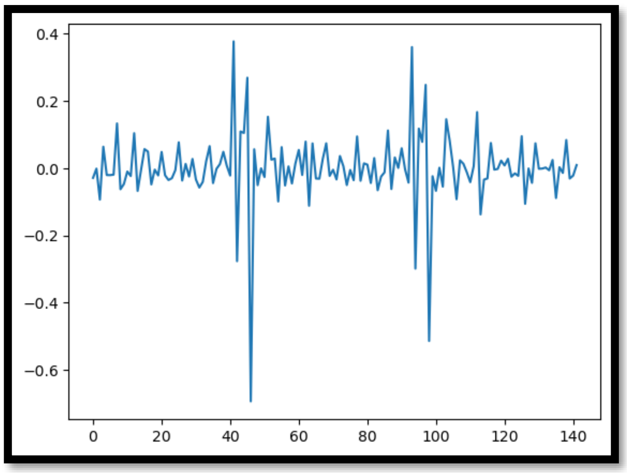
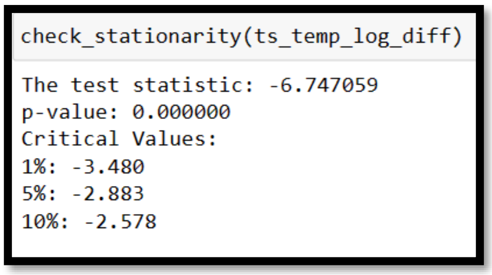


Figure 5: Stationary Data Time Series After Performing Log Different.



```
check_stationarity(ts_temp_log_diff)

The test statistic: -6.747059
p-value: 0.000000
Critical Values:
1%: -3.480
5%: -2.883
10%: -2.578
```

Figure 6: ADF Result For The Data After Performing Log.

By ensuring stationarity in the dataset through appropriate preprocessing steps, such as aggregating sales data, handling missing values, and verifying stationarity using statistical tests, we create a solid foundation for accurate analysis and modeling of the time series data.

Modelling

ARIMA

ARIMA (Autoregressive Integrated Moving Average) modelling is a widely used approach for time series forecasting. The ACF(q) and PACF(p) play a crucial role in determining the appropriate parameters for an ARIMA model, making them essential in the context of ARIMA modelling.

The ACF helps identify the autoregressive (AR) component of the ARIMA model. Significant positive ACF values at certain lags suggest that past observations have a direct influence on the current observation. This indicates the presence of autocorrelation, which motivates the inclusion of the AR component in the model. By examining the ACF plot and considering the significant lags, one can determine the order of the autoregressive term.

On the other hand, the PACF helps determine the moving average (MA) component of the ARIMA model. Significant positive PACF values at certain lags indicate that the residual errors from previous observations have a direct influence on the current observation. This suggests the presence of moving average effects, which warrant the inclusion of the MA component. By analysing the PACF plot and considering the significant lags, one can determine the order of the moving average term.

The appropriate selection of AR and MA orders is crucial for building an accurate ARIMA model. The ACF and PACF plots provide valuable insights into the lag structure and correlations in the time series, enabling the identification of significant lags and guiding the selection of ARIMA parameters. By understanding the importance of ACF and PACF in ARIMA modelling, one can effectively determine the orders of the autoregressive and moving average components, resulting in a well-fitted model that captures the underlying patterns and improves forecasting accuracy.

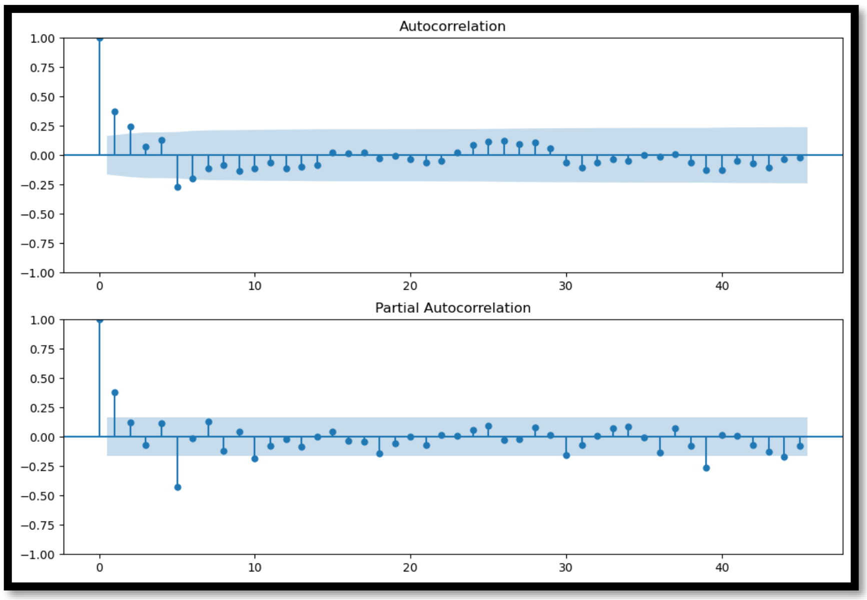
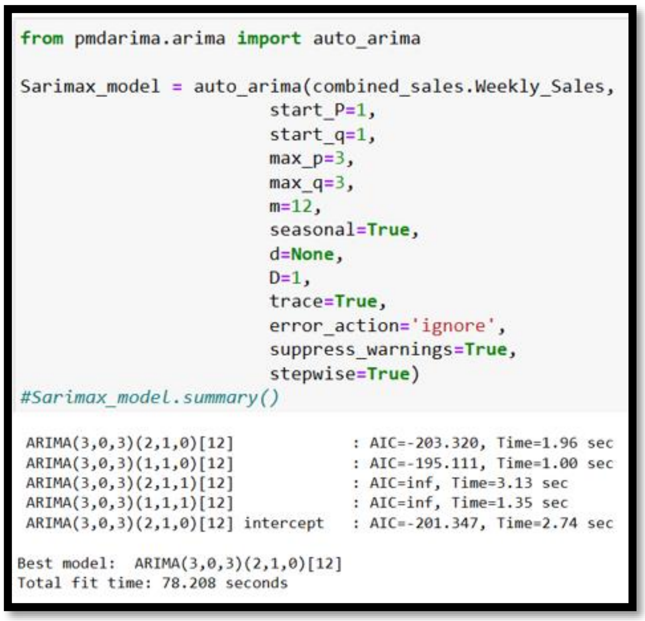


Figure 7: Autocorrelation and Partial Autocorrelation

Since our dataset contains seasonal data we used pmdarima.arima module to automatically select the best ARIMA model for forecasting using the auto_arima function. The auto_arima function performs a stepwise search to minimize the Akaike Information Criterion (AIC), a measure of the model's goodness of fit. The code input time series data and sets various parameters for the ARIMA model search. These parameters include the starting values for the autoregressive and moving average orders (start_P and start_q), the maximum values for these orders (max_p and max_q), the seasonal period (m), and the differencing order (D).

The output of the code provides a list of ARIMA models that were evaluated during the search process, along with their respective AIC values and the time taken for model fitting. The model that has the lowest AIC is considered the best model. In this case, the best model is identified as ARIMA(3,0,3)(2,1,0)[12], indicating that the model has an autoregressive order of 3, a differencing order of 0, a moving average order of 3, and a seasonal component with a period of 12.

The interpretation of the result is that the identified ARIMA(3,0,3)(2,1,0)[12] model is expected to provide a good fit for the data and can be used for forecasting.

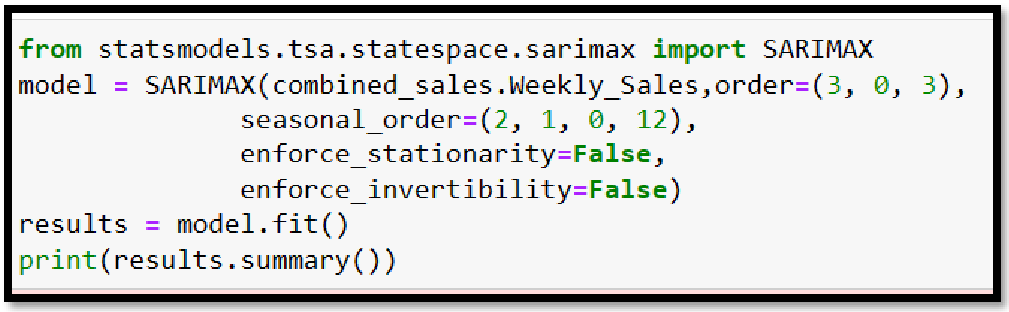


```
from pmdarima.arima import auto_arima

Sarimax_model = auto_arima(combined_sales.Weekly_Sales,
                           start_P=1,
                           start_q=1,
                           max_p=3,
                           max_q=3,
                           m=12,
                           seasonal=True,
                           d=None,
                           D=1,
                           trace=True,
                           error_action='ignore',
                           suppress_warnings=True,
                           stepwise=True)
#Sarimax_model.summary()
```

```
 ARIMA(3,0,3)(2,1,0)[12]             : AIC=-203.320, Time=1.96 sec
 ARIMA(3,0,3)(1,1,0)[12]             : AIC=-195.111, Time=1.00 sec
 ARIMA(3,0,3)(2,1,1)[12]             : AIC=inf, Time=3.13 sec
 ARIMA(3,0,3)(1,1,1)[12]             : AIC=inf, Time=1.35 sec
 ARIMA(3,0,3)(2,1,0)[12] intercept   : AIC=-201.347, Time=2.74 sec

Best model:  ARIMA(3,0,3)(2,1,0)[12]
Total fit time: 78.208 seconds
```

Figure 8: Best ARIMA Result

After identifying the best ARIMA model using the previous code, we proceed to build and train a SARIMAX model for predicting and forecasting the time series. The purpose of the subsequent code is to instantiate the SARIMAX model, incorporating the optimal ARIMA parameters.



```
from statsmodels.tsa.statespace.sarimax import SARIMAX
model = SARIMAX(combined_sales.Weekly_Sales,order=(3, 0, 3),
              seasonal_order=(2, 1, 0, 12),
              enforce_stationarity=False,
              enforce_invertibility=False)
results = model.fit()
print(results.summary())
```

,

Figure 9: Performing ARIMA Model

The ARIMA model demonstrates a strong predictive ability, as evidenced by its ability to accurately capture the underlying patterns and shape of the data. This suggests that the model is well-suited for providing reliable forecasts of weekly sales. The figure below visually depicts the performance of the ARIMA model, showcasing its close alignment with the actual data points. However, it is important to note that the ARIMA model requires a learning period to accurately predict the shape of the data. It starts with initial estimations and iteratively adjusts its parameters until it converges to the true values of the data's shape. Despite this initial learning phase, the ARIMA model shows promising potential for accurately forecasting future weekly sales.

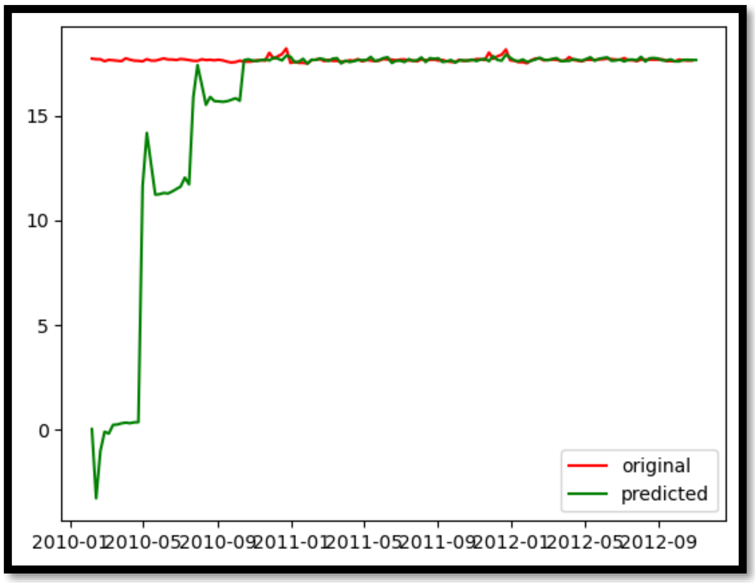


Figure 10: Actual and Predicted Result of ARIMA Model

To enhance the visual representation of the predicted data, a filtering technique was applied to the plot. This filtering process aimed to eliminate values that do not exist in the original dataset, specifically those with values less than 17. To showcase the shape and pattern of the predicted data more clearly, without including values that are not present in the actual dataset. The figure below demonstrates the resulting plot after the application of this filtering technique, providing a clearer representation of the predicted data and enabling a more accurate assessment of its alignment with the original dataset.

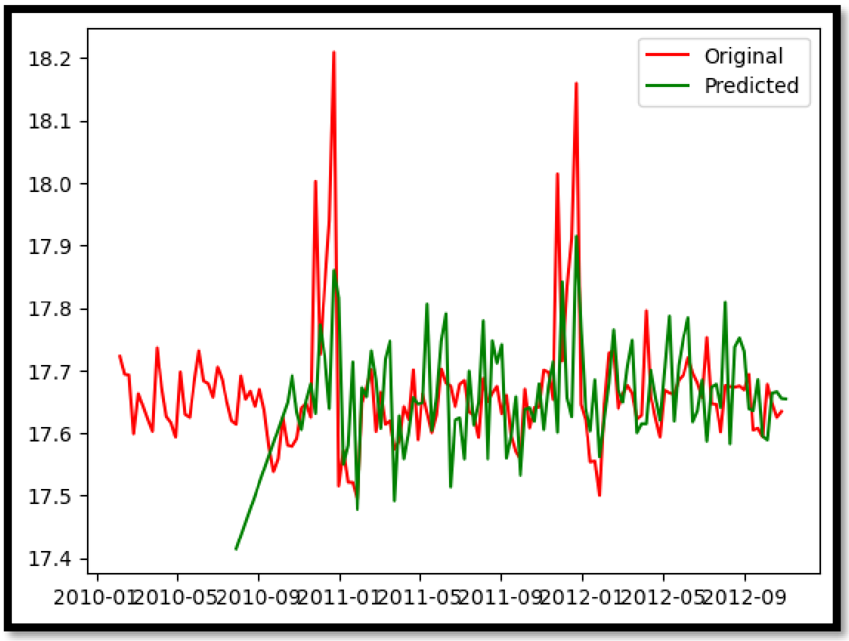


Figure 11: Actual and Predicted Result of ARIMA Model After Performing Filter.

The figure below illustrates that the forecasting results align closely with the previous shape of the dataset, particularly in the last portion of the data. This alignment indicates that the ARIMA model exhibits a high level of accuracy in predicting future sales. By capturing the underlying patterns and trends in the data, the model can provide reliable and precise forecasts. This alignment between the forecasted values and the actual sales data suggests that the ARIMA model is a valuable tool for accurately predicting sales.

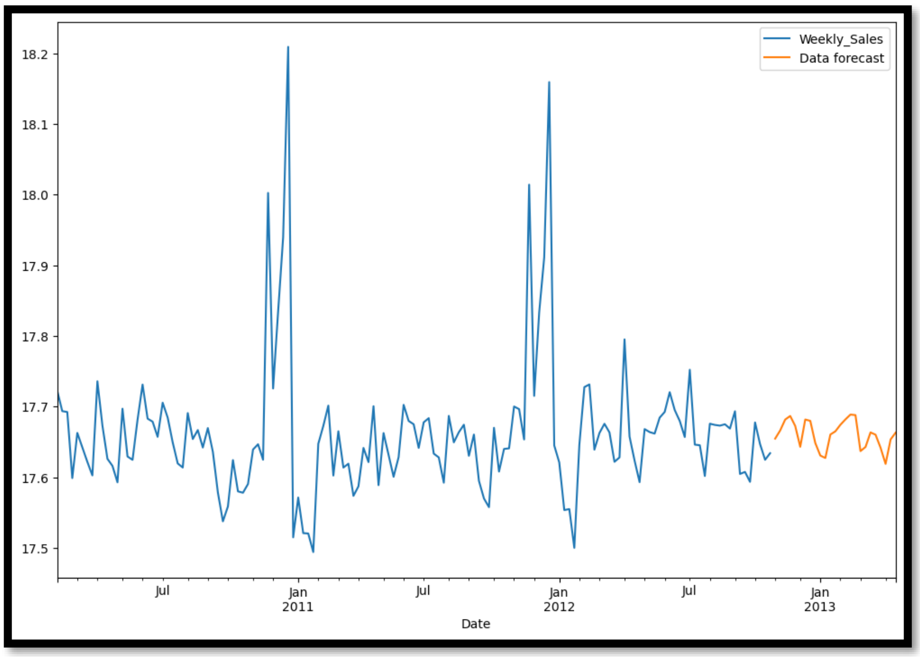
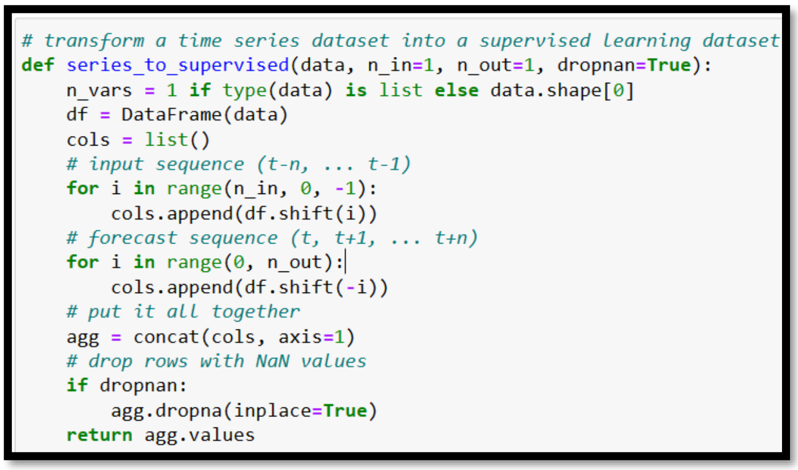


Figure 12: ARIMA Forecasted Results.

Machine Learning models

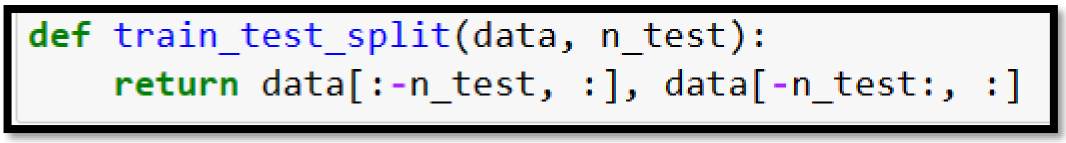
Before performing the supervised learning, we have to transform the raw time series data into a suitable format that is understandable by the machine. The following code is invaluable for preparing time series data for supervised learning tasks, enabling the application of various machine learning algorithms for accurate forecasting or other time-dependent predictions. By converting the dataset into a supervised learning format, it allows for the utilization of powerful modelling techniques that rely on labelled input-output pairs.



```
# transform a time series dataset into a supervised learning dataset
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[0]
    df = DataFrame(data)
    cols = list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
    # put it all together
    agg = concat(cols, axis=1)
    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg.values
```

Figure 13: Transform Dataset Into Supervised Dataset Function.

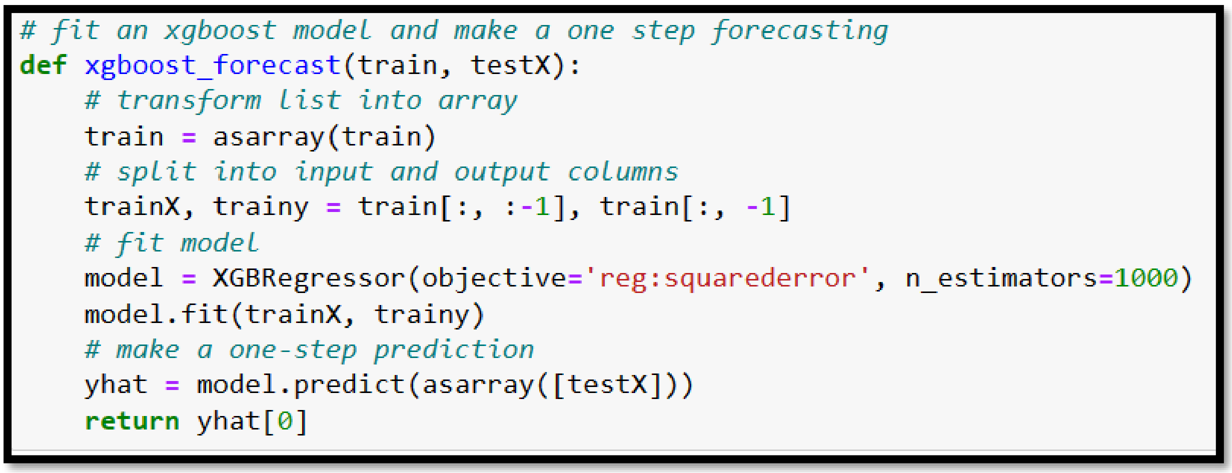
After that we have to split the dataset into training and testing datasets, this will play a crucial role in the evaluation and validation of machine learning models using univariate datasets. By partitioning the data in this manner, the function ensures that the model learns from a substantial portion of the dataset while being evaluated on unseen data, which is crucial for assessing its generalization capabilities.



```
def train_test_split(data, n_test):
    return data[:-n_test, :], data[-n_test:, :]
```

Figure 14: Performing the Transformation Function.

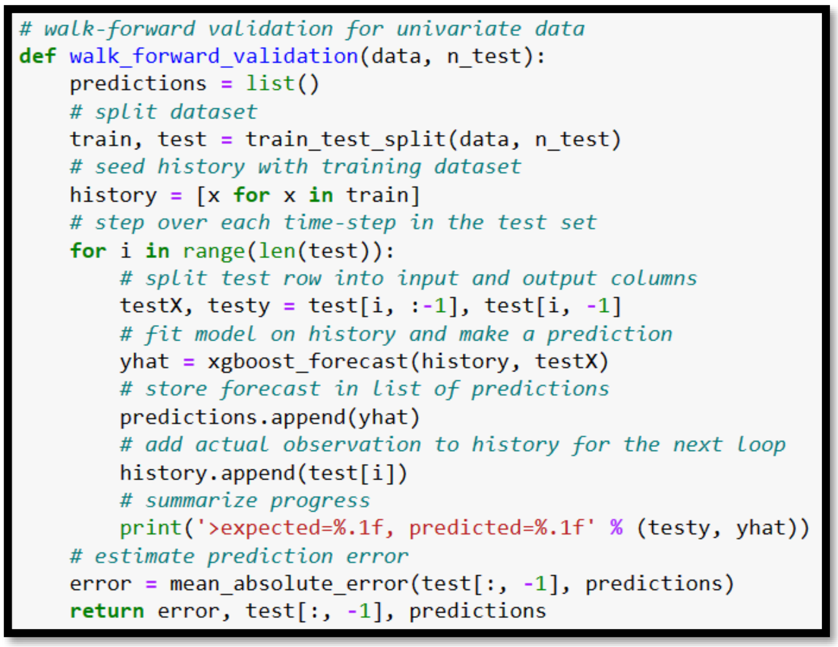
Next, we performed an XGBoost model forecasting to make one-step forecasts. XGBoost (eXtreme Gradient Boosting) is a powerful machine learning algorithm known for its excellent predictive performance and ability to handle complex datasets. It begins by converting the training dataset from a list to an array format, splitting it into input features (trainX) and the corresponding output column (trainy). The testX input vector is transformed into an array format, and then the model predicts the corresponding output (yhat) based on this input. Finally, the function returns the forecasted value (yhat[0]) as the result.



```python
# fit an xgboost model and make a one step forecasting
def xgboost_forecast(train, testX):
    # transform list into array
    train = asarray(train)
    # split into input and output columns
    trainX, trainy = train[:, :-1], train[:, -1]
    # fit model
    model = XGBRegressor(objective='reg:squarederror', n_estimators=1000)
    model.fit(trainX, trainy)
    # make a one-step prediction
    yhat = model.predict(asarray([testX]))
    return yhat[0]
```

Figure 15: XGBoost Model Forecasting Function.

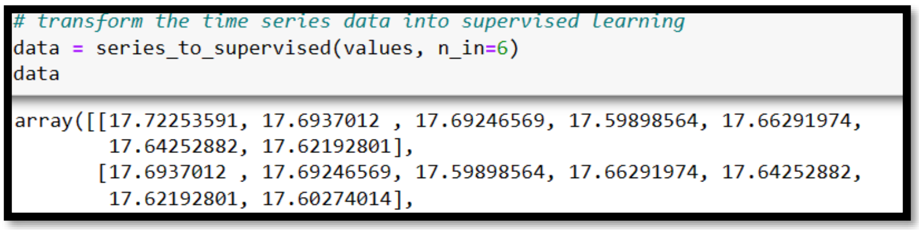
After that, to evaluate the performance of time series forecasting models we conduct a walk-forward validation. It enables the iterative assessment of the model's predictions by sequentially updating the history with each new observation and obtaining forecasted values using the XGBoost model through the xgboost_forecast function. The code facilitates the calculation of prediction errors and provides valuable insights into the accuracy of the forecasting model.



```python
# walk-forward validation for univariate data
def walk_forward_validation(data, n_test):
    predictions = list()
    # split dataset
    train, test = train_test_split(data, n_test)
    # seed history with training dataset
    history = [x for x in train]
    # step over each time-step in the test set
    for i in range(len(test)):
        # split test row into input and output columns
        testX, testy = test[i, :-1], test[i, -1]
        # fit model on history and make a prediction
        yhat = xgboost_forecast(history, testX)
        # store forecast in list of predictions
        predictions.append(yhat)
        # add actual observation to history for the next loop
        history.append(test[i])
        # summarize progress
        print('>expected=%.1f, predicted=%.1f' % (testy, yhat))
    # estimate prediction error
    error = mean_absolute_error(test[:, -1], predictions)
    return error, test[:, -1], predictions
```

Figure 16: Forward Validation Function.

Then we need to organize the data into a tabular format, where each line represents a specific time step and includes the two input characteristics and the corresponding output value. The input features are derived from the lagged observations, capturing the historical patterns leading up to the current time step. The output value represents the target variable to be predicted.



```
# transform the time series data into supervised learning
data = series_to_supervised(values, n_in=6)
data
```

```
array([[17.72253591, 17.6937012 , 17.69246569, 17.59898564, 17.66291974,
        17.64252882, 17.62192801],
       [17.6937012 , 17.69246569, 17.59898564, 17.66291974, 17.64252882,
        17.62192801, 17.60274014],
```

Figure 16: Data After Transformation into Labelled Data.

The provided figure illustrates the forecasting results obtained from the machine learning model. The model's predictions align closely with the shape of the original dataset right from the beginning. This shows that the model is capable of accurately capturing the underlying patterns and trends in the data. Furthermore, the model has a very efficient learning process because it does not take a long time to adapt and predict the structure of the dataset.

However, the shape of the predicted values looks to be sharper when compared to the actual dataset. This could imply that the model has learned details or differences in the data, resulting in noticeable alterations in its predictions. While this sharper shape may introduce some deviation from the original dataset.

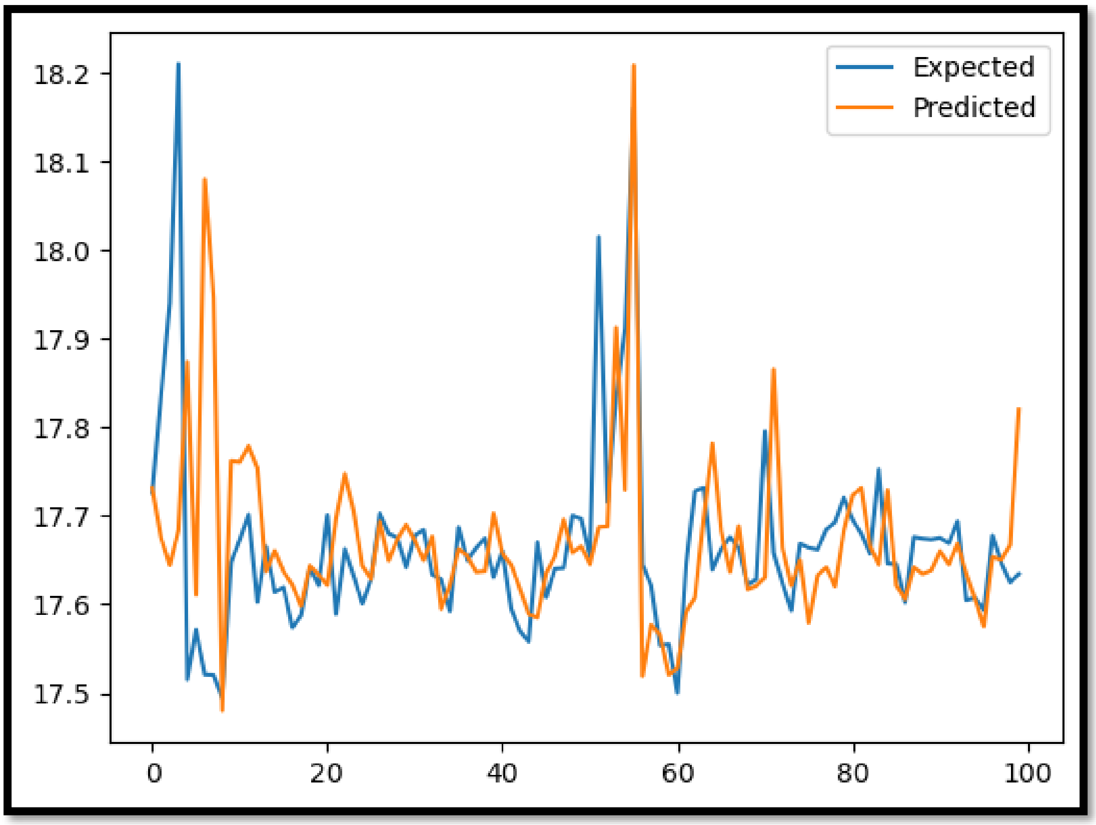


Figure 17: Actual and Predicted Result of Machine Learning Model

Upon analysing the figure below, it becomes evident that the shape of the forecasting exhibits a frequent repetition pattern. This repetition suggests that the model utilized its own predicted values as inputs for forecasting future sales. While this approach may yield satisfactory results in the short term, it raises concerns about the reliability and accuracy of the model's predictions over an extended period.

By relying on its predictions, the model might introduce a compounding effect, wherein any errors or inaccuracies in the initial predictions can amplify and propagate throughout subsequent forecasts. Consequently, this repetitive behaviour diminishes the trustworthiness of the model for making long-term predictions, as it becomes less capable of capturing and adapting to underlying changes and dynamics in the sales data.

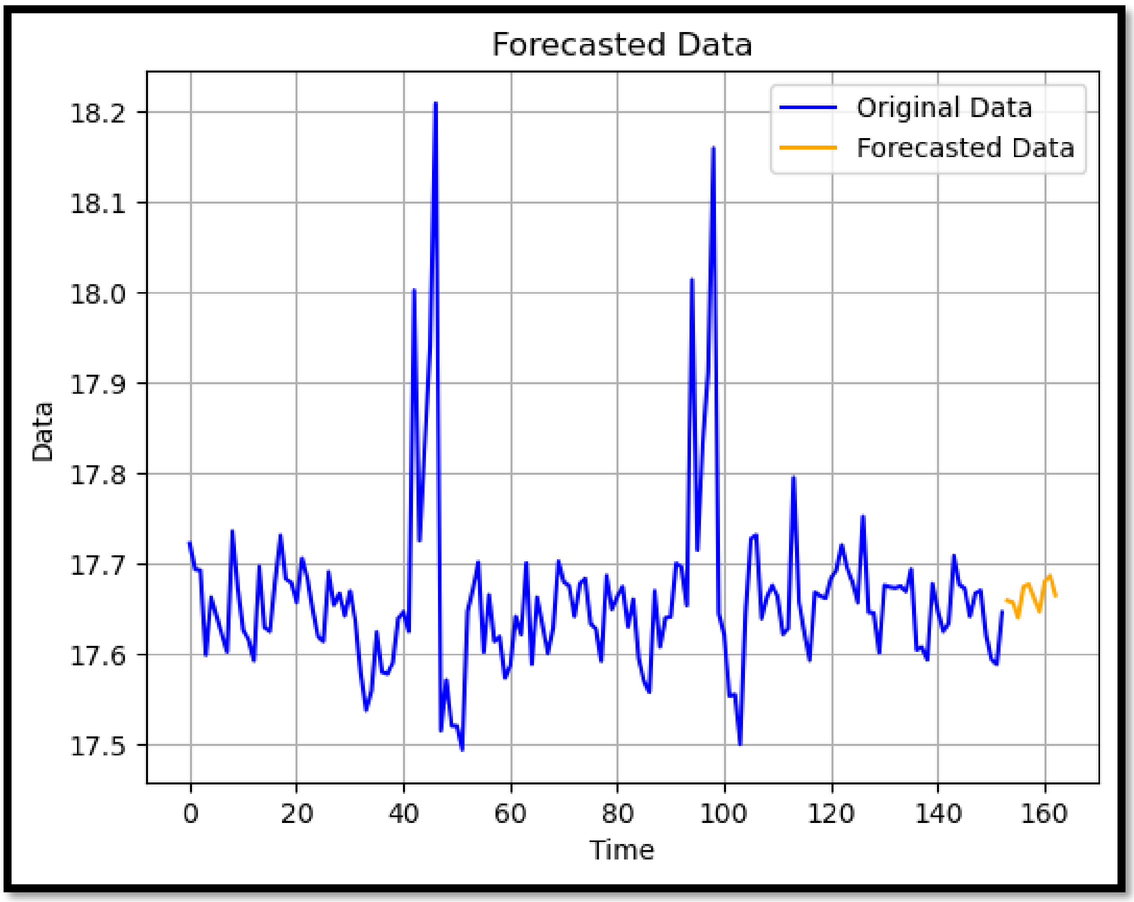
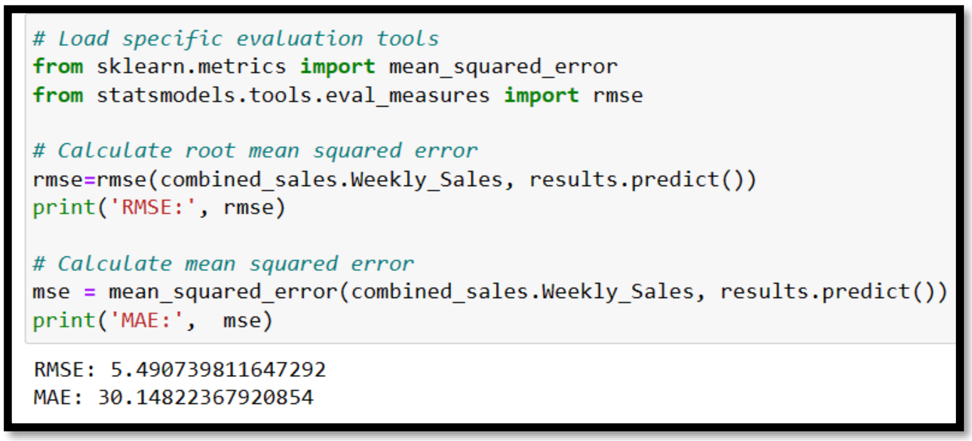


Figure 18: Machine Learning Forecasted Results.

Analysis and Discussion

The root mean squared error (RMSE) of the ARIMA prediction model was calculated to be 5.4907, showing the average size of the prediction errors. Furthermore, the mean absolute error (MAE) was calculated as 30.1482, indicating the average absolute difference between the predicted and actual values. These evaluation metrics help assess the accuracy and precision of the ARIMA model's predictions.

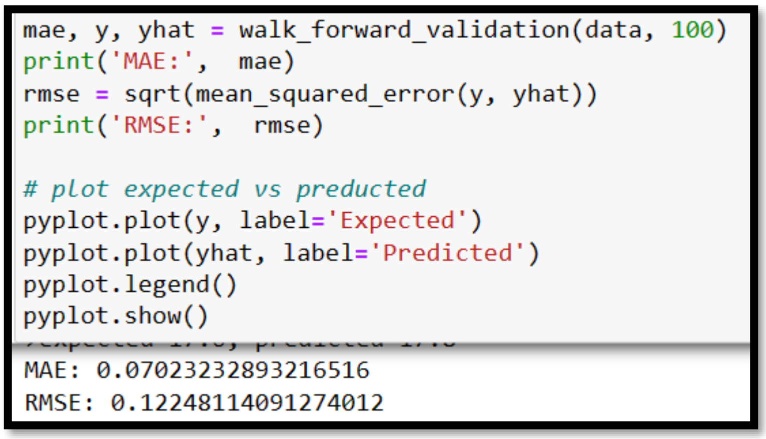


```
# Load specific evaluation tools
from sklearn.metrics import mean_squared_error
from statsmodels.tools.eval_measures import rmse

# Calculate root mean squared error
rmse=rmse(combined_sales.Weekly_Sales, results.predict())
print('RMSE:', rmse)

# Calculate mean squared error
mse = mean_squared_error(combined_sales.Weekly_Sales, results.predict())
print('MAE:',  mse)
```

```
RMSE: 5.490739811647292
MAE: 30.14822367920854
```

Figure 19: ARIMA MAE and RMSE Results.

On the other hand, the XGBoost machine learning prediction model demonstrated superior performance with a significantly lower MAE of 0.0702, indicating a smaller average absolute difference between the predicted and actual values. The root mean squared error (RMSE) for the XGBoost model was computed as 0.1225, further highlighting the model's ability to make more precise predictions with lower overall error.



```
mae, y, yhat = walk_forward_validation(data, 100)
print('MAE:',  mae)
rmse = sqrt(mean_squared_error(y, yhat))
print('RMSE:',  rmse)

# plot expected vs preducted
pyplot.plot(y, label='Expected')
pyplot.plot(yhat, label='Predicted')
pyplot.legend()
pyplot.show()
```

```
MAE: 0.07023232893216516
RMSE: 0.12248114091274012
```

Figure 20: XGBoost MAE and RMSE Results.

These evaluation results emphasize the superior predictive capabilities of the XGBoost machine learning model compared to the ARIMA model. The significantly lower MAE and RMSE values obtained from the XGBoost model indicate its ability to provide more accurate and reliable predictions for the given dataset.

All the models, including ARIMA and machine learning algorithms, demonstrated good accuracy in predicting the time series data. However, there are certain considerations to be made when comparing their suitability for Walmart's weekly sales forecasting, particularly in the context of inventory management.

The machine learning models, represented by the XGBoost algorithm in this case, exhibited superior performance in terms of lower mean absolute error (MAE) and root mean squared error (RMSE) values. These metrics indicate that the machine learning model achieved a smaller average absolute difference and overall error between the predicted and actual values, suggesting higher precision in its predictions. Furthermore, the machine learning model was able to capture the shape of the dataset right from the start, demonstrating its ability to adapt and forecast data structure efficiently.

The machine learning model's inability to provide reliable long-term forecasts is a key drawback. The model's performance decreases over time, which is a critical part of Walmart's inventory management needs. The repetitive pattern observed in the forecasting results indicates that the model relies heavily on its own predicted values as inputs for future forecasts. While this approach may yield satisfactory short-term predictions, it raises concerns about the model's reliability and accuracy for longer forecasting horizons.

In contrast, the ARIMA model has the advantage of providing accurate forecasts for longer periods of time, making it more suitable for Walmart's weekly sales problem and inventory management. The ARIMA model incorporates the autoregressive integrated moving average approach, which systematically accounts for the linear relationships, moving average effects, and differencing requirements in the data. By analysing the autocorrelation function (ACF) and partial autocorrelation function (PACF) plots, the ARIMA model identifies the appropriate orders of autoregressive (AR), moving average (MA), and differencing (I) component, enabling it to capture the underlying patterns and trends in the data.

In conclusion, while machine learning models, such as XGBoost, provide excellent accuracy and are proficient in predicting the shape of the dataset, their limitations in forecasting over extended periods hinder their suitability for Walmart's inventory management needs. On the other hand, the ARIMA model offers accurate forecasts for longer time horizons, making it more appropriate for addressing Walmart's weekly sales forecasting requirements. By leveraging the inherent characteristics of the ARIMA model, such as the identification of significant lags and capturing temporal relationships, Walmart can enhance its inventory management strategies and ensure optimal stock levels based on reliable long-term sales predictions.

References

Lu, C.-J., Lee, T.-S., & Lian, C.-M. (2012). Sales forecasting for computer wholesalers: A comparison of multivariate adaptive regression splines and artificial neural networks. *Decision Support Systems*, *54*(1), 584–596. https://doi.org/https://doi.org/10.1016/j.dss.2012.08.006

UJJWAL CHOWDHURY. (2022). *Walmart Cleaned Data*. https://www.kaggle.com/datasets/ujjwalchowdhury/walmartcleaned

Walton, S. (2012). Case Study: Walmart. *Strategic International Management*, *67*.