Assignment-3: Time Series

(CSL 7580 Advanced AI)

Report

Assignment	**03**
Roll No.	**B20CS018**
Name	**Harshita Gupta**

PART 1

Introduction:
In this report, we will explain the code for ARIMA (Autoregressive Integrated Moving Average) Forecast function. The ARIMA model is a widely used time series forecasting technique that combines three key components - Autoregression, Integration, and Moving Average.

Functionality:
The ARIMA_Forecast function takes five parameters: input_series, P, D, Q, and prediction_count.

The input_series parameter represents the time series data to be analyzed and forecasted. The P, D, and Q parameters represent the order of the autoregressive, integrated, and moving average components of the ARIMA model. Finally, the prediction_count parameter specifies the number of time steps ahead to forecast.

The function first converts the input_series list into a numpy array and initializes the predicted_series array with the same values as the input_series. The coefficients phi and theta are initialized to zeros.

The function then iterates over the number of predictions specified by the prediction_count parameter. For each iteration, it calculates the difference series and the autoregressive and moving average terms based on the values of P, D, and Q.

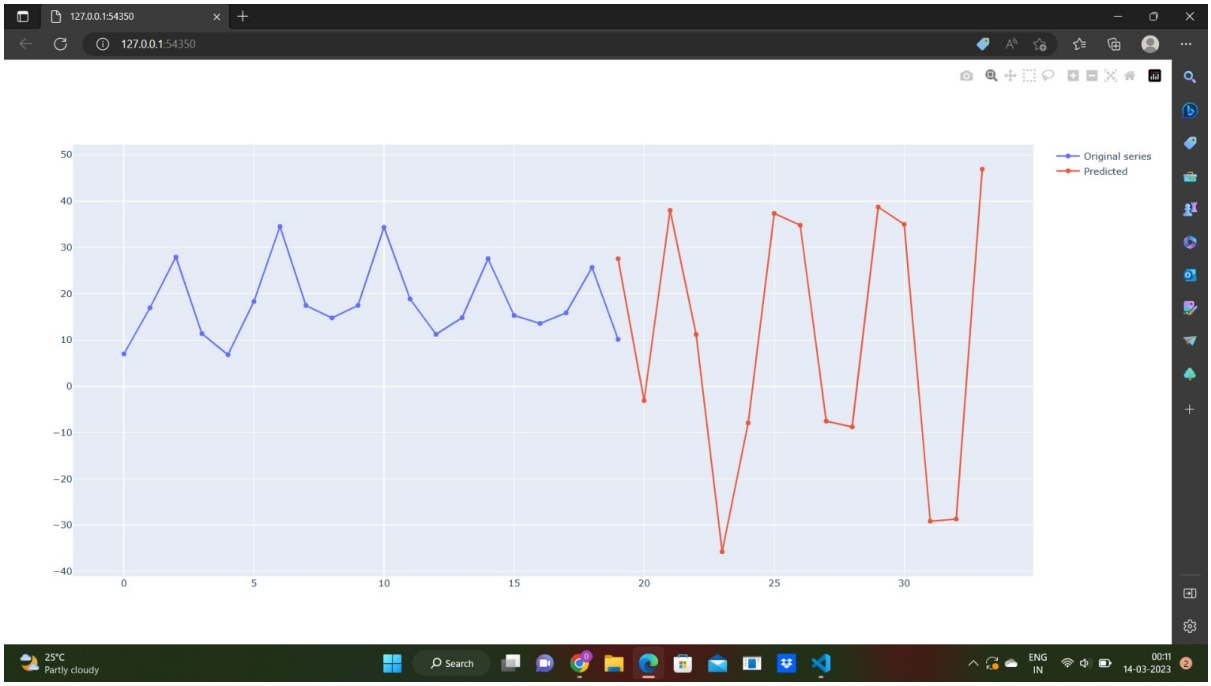
The autoregressive terms are calculated using the correlation coefficient between the predicted_series and the previous values of the series. The moving average terms are calculated using the correlation coefficient between the different series and the previous values of the predicted_series.

The next term in the series is then calculated as the sum of the autoregressive and moving average terms, and this value is added to the predicted_series array.

Finally, the function returns the new terms in the predicted sequence for the specified prediction_count.

Conclusion:

In conclusion, the ARIMA_Forecast function is a simple implementation of the ARIMA time series forecasting technique. It takes the input_series data and uses it to forecast future values based on the autoregressive, integrated, and moving average components of the ARIMA model. The function could be improved by incorporating more advanced techniques for optimizing the model parameters and for checking stationarity.



PART 2

Introduction:

The given code is an implementation of the Holt-Winters forecasting method. This method is used to predict future values of a time series, taking into account its trend, seasonality, and level components. The implementation takes a list of input values, alpha, beta, gamma, seasonality, and prediction_count as input parameters and returns a list of predicted values.

Function Signature:

```
def HoltWinter_Forecast(input: list, alpha: float, beta: float, gamma: float, seasonality: int, prediction_count: int) -> list
```

Parameters:

- `input`: A list of input values representing the time series.
- `alpha`: A float value between 0 and 1 representing the smoothing parameter for the level component.
- `beta`: A float value between 0 and 1 representing the smoothing parameter for the trend component.
- `gamma`: A float value between 0 and 1 representing the smoothing parameter for the seasonal component.
- `seasonality`: An integer value representing the number of observations in each season.
- `prediction_count`: An integer value representing the number of periods for which to predict future values.

Return Value:

- A list of predicted values for the given `prediction_count`.

Implementation:

Initializing Model Components

The implementation initializes the level, trend, and seasonal components of the model using the given input series. The level is calculated as the average of the first `seasonality` observations. The trend is calculated as the difference between the first and `seasonality`-th observations divided by `seasonality`. The seasonal components are calculated as the difference between each observation and the level.

Generating Predictions

The implementation iterates over the input series and predicts future values for each time step. The forecast for each time step is generated as the sum of the level, trend, and the corresponding seasonal component. The predicted values are stored in a list.

Updating Model Components

After generating each forecast, the implementation updates the level, trend, and seasonal components of the model. For each observation in the input series, the level is updated as a weighted average of the observation and the previous level. The trend is updated as a weighted average of the difference between the current level and the average of the previous `seasonality` observations and the previous trend. The seasonal component is updated as a weighted average of the difference

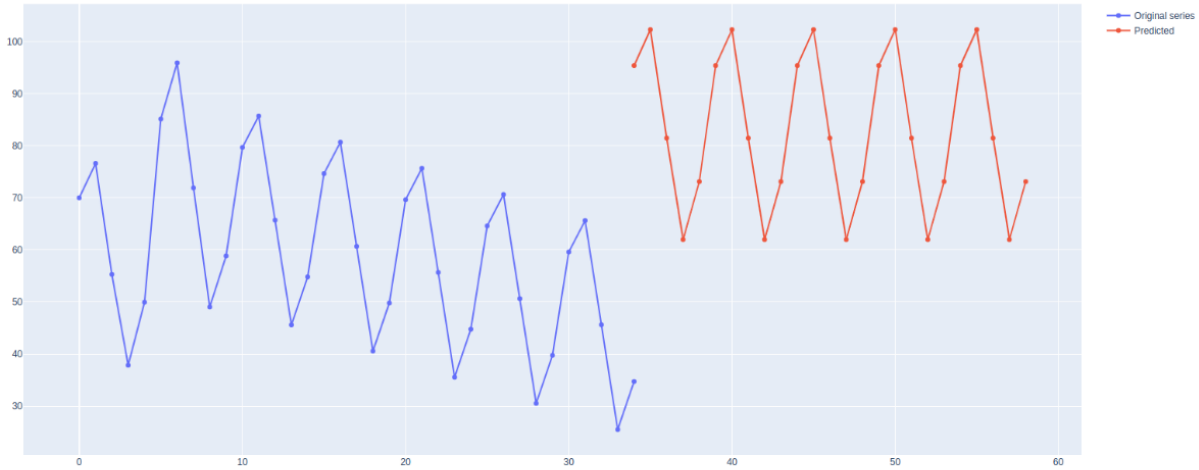
between the observation and the current level and the previous seasonal component.

Returning Predictions

The implementation returns the last `prediction_count` predicted values.

Conclusion:

The given code implements the Holt-Winters forecasting method to predict future values of a time series. It takes into account the trend, seasonality, and level components of the time series and updates them as new observations become available. The implementation is parameterized by the smoothing parameters `alpha`, `beta`, and `gamma`, as well as the seasonality and prediction count.



PART 3

Introduction

This code includes two functions that find the optimal parameters for two time series models: ARIMA and Holt-Winter's.

ARIMA_Parameters

Input

The `ARIMA_Parameters` function takes a list of a time series data as input.

Output

The function returns a tuple `(P, D, Q)` representing the optimal parameters for the ARIMA model that minimizes the AIC score.

Procedure

1. The input time series data is converted to a numpy array.
2. The best parameters are initialized as None and the best AIC score as infinity.
3. The function iterates over all possible combinations of `(P, D, Q)` where each value ranges from 0 to 2.
4. For each combination of `(P, D, Q)`, the ARIMA model is fit with the corresponding parameters using `ARIMA` function from `statsmodels` library and the model is trained using `fit()` function.
5. The AIC score is calculated using `aic` attribute of the trained model.
6. If the AIC score of the current model is lower than the best AIC score, the best parameters and best AIC score are updated with the current parameters and AIC score respectively.
7. The function returns the best parameters `(P, D, Q)` as a tuple.

HoltWinter_Parameters

Input

The `HoltWinter_Parameters` function takes a list of a time series data as input.

Output

The function returns a tuple `(Alpha, Beta, Gamma, Seasonality)` representing the optimal parameters for the Holt-Winter's model that minimizes the RMSE score.

Procedure

1. The input time series data is converted to a numpy array.
2. The best parameters are initialized as None and the best RMSE score as infinity.
3. The function iterates over all possible combinations of `(alpha, beta, gamma)` where each value ranges from 0 to 0.9 with a step of 0.1.
4. For each combination of `(alpha, beta, gamma)`, the Holt-Winter's model is fit with the corresponding parameters using `ExponentialSmoothing` function from `statsmodels` library and the model is trained using `fit()` function. The

`seasonal_periods` parameter is set to a fixed value of `seasonality` and `trend` and `seasonal` parameters are set to `'add'`.

5. Predictions are generated for the next 20 time steps using the `predict()` function of the trained model.
6. The root mean squared error (RMSE) is calculated between the predicted values and the true values from the last 20 time steps of the input time series data.
7. If the RMSE score of the current model is lower than the best RMSE score, the best parameters are updated with the current parameters and best RMSE score is updated with the current RMSE score.
8. The function returns the best parameters `(Alpha, Beta, Gamma, Seasonality)` as a tuple.