Streaming Data Management and Time Series Analysis project

Samir Doghmi 897358

Abstract

This project aims to predict a time series using ARIMA, UCM , and machine learning models. The data, from the United States, represents daily average closure times for requests (2007-01-04 to 2015-03-31). The goal is to forecast values for 2015-04-01 to 2015-11-07 and evaluate predictions using the Mean Absolute Error (MAE) metric. The final output will be a dataset with four columns: dates, UCM predictions, ARIMA predictions, and ML predictions.

Contents

Introduction

The project aims to predict the same time series with an ARIMA model, and UCM model and a machine learning model. The time series data represents the average number of days needed to close requests that were closed on each day (non-negative data), measured in floating point values (`ave_days`). This dataset originates from the United States. The daily time series covers the period from 2007-01-04 to 2015-03-31 (3009 days). Additionally, several binary columns were added such as Year, Month, Day, `Month_Name` and `Quarter`, in order to facilitate the analysis.
The dataset was then divided as follows:

- Train: 2007-01-04 to 2014-08-22
- Test: 2014-08-23 to 2015-03-31

This was done to maintain the same forecast horizon as the one between the complete time series and our goal.
The objective is to forecast daily values from 2015-04-01 to 2015-11-07, which spans 221 days.The predictions will be evaluated with Mean Absolute Error (MAE[3]) metric.

1. Data exploration and preprocessing

Dataset and pre-processing

Our dataset is made of 3009 records each with the following 3 features:

- `Date`: string with date in format yyyy-mm-dd;
- `weekday`: string with the name of the weekday;
- `ave_days`: floating point number representing the average number of days needed to close the requests that were closed that day, with a mean of 37.63 days and a range from a minimum of 0 days to a maximum of 792.82 days.

The first step was to verify the temporal continuity of the series by checking for the missing values.

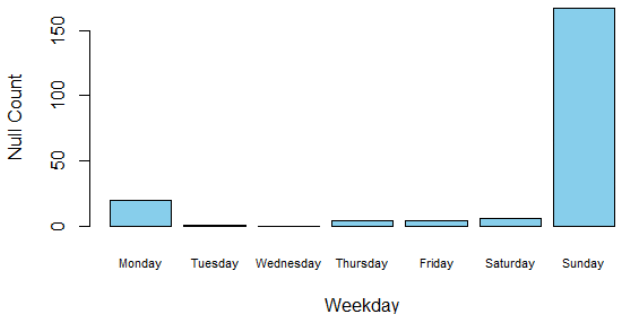

Figure 1. *Null Count for each Weekday*

We noticed that there are 202 null values, most of them appearing on Sundays.This observation can be logically justified by the fact that most companies do not operate on Sundays. We managed them by applying the function `na.locf`[7], which replaces NaN values with the last observed value in the series.

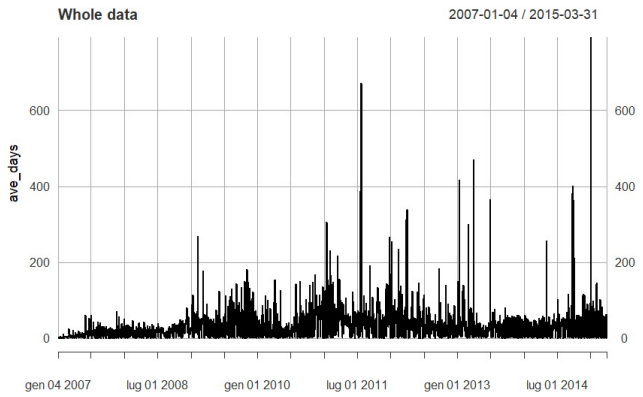

Figure 2. *overall trend of the series*

From an initial inspection of the data, we observed a substantial number of peaks in the average number of days needed to close a request.

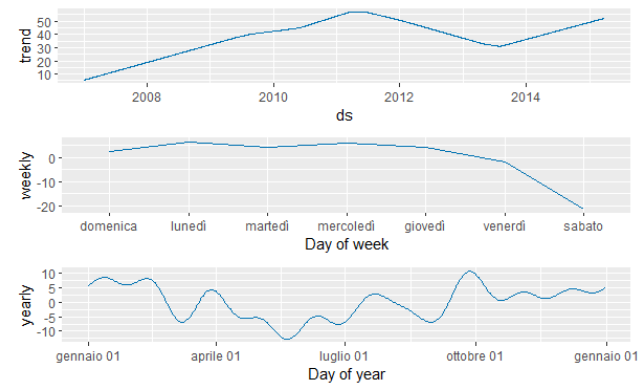

Figure 3. *Decomposition of the times series*

Using the prophet[4] library for further analysis, we noticed a generally increasing trend over the years in the average number of days required to close a request, except for the period between 2011 and mid-2013. We also noted a clear weekly seasonality in the data. However, an annual seasonality is not evident from the graph.

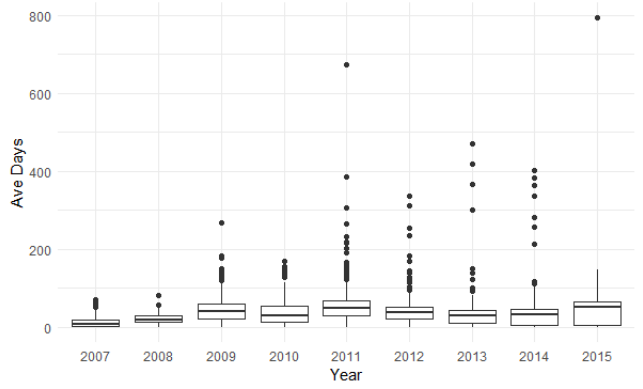

Figure 4. *Outliers by years*

It was clearly visible from the graph that there were significant outliers in the average number of days it takes to close requests each year. In particular, 2011 was the year with the highest number of outliers. To solve these issues, we decided to adjust the values by computing the interquartile range (IQR). Specifically, we replaced the values exceeding the maximum threshold (1.5 times the IQR above the third quartile) with the value of the threshold itself. This approach helps to mitigate the impact of extreme outliers on our analyses and predictions.

For deeper analysis, as the records contain zeros, the logarithm transformation will result in -Inf values for these entries. To overcome this drawback, we added a small constant to all values before performing the logarithm.

Stationarity Analysis

After the pre-processing steps, we conducted the Augmented Dickey-Fuller (ADF) test to assess the stationarity of the time series.The ADF test indicated that the time series is stationary (p-value = 0.01).

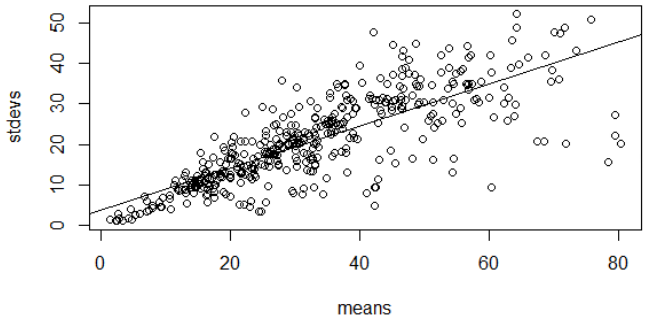

Figure 5.

However, the mean and variance change over time showing a linear growth trend, so we considered implementing a logarithmic transformation to stabilise the variance.

2. ARIMA

ARIMA[1] is a statistical model commonly used in time series forecasting because it combines the ability to capture relationships among past data and to handle the presence of seasonality and trends in time series.ARIMA utilizes an autoregressive component (AR) to capture the relationship between a value in the series and its past values, a moving average component (MA) to handle the presence of random forecasting errors, and an integrated differencing component (I) to address the presence of non-stationary trends in the series.

We experimented with several models by examining the model coefficients and analyzing the ACF[5] and PACF[6] of the residuals. A specific test set was built for each model to incorporate all the regressors used for the predictions.

Model	Mae
(2,1,1)(1,0,1)[7]	19.86
(3,0,1)(1,0,1)[7] with dummies	16.41
(2,0,1)(1, 1, 2)[7] with sinusoids	19.38

By comparing the results using mean absolute error, we determined that the ARIMA (3,0,1)(1,0,1)[7] model performed the best. Given that we are working with daily data, we included five dummy variables: one for Sundays, one for Saturdays, one for weekday holidays, one for Sunday holidays, and one for Saturday holidays. While the holiday-specific dummies are not statistically significant, we chose to include them for completeness.

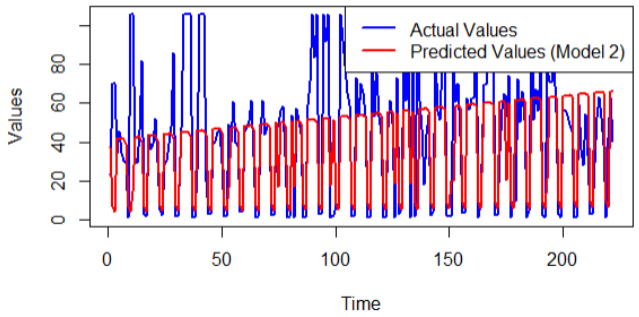

Figure 6. *Forecast Regression with ARIMA (3,0,1)(1,0,1)[7]*

In the figure above we can see the difference between the prediction of the best ARIMA model and the original value on the test set.

3. UCM

UCM[2] is another statistical model commonly used in time series forecasting because it can capture both the trend component and the seasonal component present in the data. UCM uses a decomposition-based approach to identify and treat separately the various components present in the data, such as trend and seasonality. This model is highly flexible and adaptable, and it can be used in many different contexts.
For the implementation of these models, it was it was decided to keep the same split of the time series into training and test series as has been done for the ARIMA models.

Model	Mae
dummy variables as regressors, weekly seasonal dummies and integrated random walk	16.76
dummy variables as regressors, weekly seasonal dummies and local linear trend	18.87
dummy variables as regressors, weekly seasonal dummies and annual trigonometric component and local linear trend	19.38

It can be seen from the graph that the first model performed better than the other two. The same dummies used in the ARIMA model were taken as regressors, with the exception of dum_sunday and dum_saturday, because we included the weekly seasonal dummies . Both smoothing and filtering were applied to the resulting model to improve the estimates.

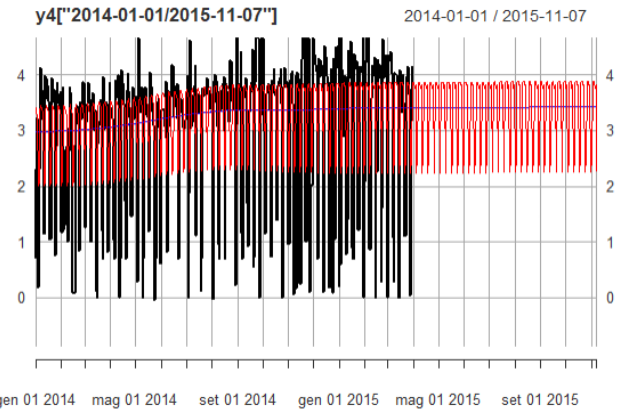

Figure 7. *Forecast Best model Ucm*

By observing the forecast graph from 2015-04 to 2015-11-07, the model seems to capture the general level and several of the periodic patterns of the data.

4. ML

For machine learning models, it is essential to have a set of highly explanatory covariates. To capture the various characteristics of each day, we split the date attribute into several components: the month, the day of the week, the day of the year and the dummy variables for the holidays. All variables were considered for one-hot encoding except for the day of the year. Additionally, as done in previous models, we handled outliers by replacing them with the third quartile of the ave_days range values.
The chosen models are the following:

- Random forest uses an ensemble of decision trees and evaluates the majority of predictions from the trees to obtain the final prediction
- XGBoost is an implementation of gradient-boosted decision trees designed for speed and performance. We adjust the model by implementing a maximum depth of 2, a learning rate of 0.2, and early stop training of 15 rounds. The model uses squared error as the loss function to optimize its predictions.
- Support vector machine works by finding the hyperplane that best separates different classes in the feature space. We employed radial kernel that is a used in to measure the similarity between two data points in a way that allows the model to handle non-linear relationships.

All the parameters were adjusted to minimize the MAE value.

Model	Mae
Random Forest	17.10
SVM	16.25
XGBoost	18.71

By comparing the results using mean absolute error, we determined that the SVM model performed the best, achieving an MAE value of 16.25.

5. Conclusions and future developments

This study aimed to predict a time series using ARIMA, UCM , and machine learning models, focusing on the average closure times for requests per day in the United States. Throughout our analysis, we observed a strong weekly seasonality in the time series data, which we managed by incorporating dummy variables. We also handled outliers and missing values.

Despite experimenting with various combinations of parameters and model configurations, our results showed that all three models, ARIMA, UCM , and ML, achieved similar Mae values around 16.50 when tested on the period from 2014-08-22 to 2015-03-31. This indicates that no single model consistently outperformed the others across all metrics.

For future developments, we recommend exploring different models such as deep neural networks ,with various parameter combinations. Additionally, better handling of NaN values and considering more regressors could provide deeper insights and enhance model performance.

References

[1] ARIMA: *Autoregressive Integrated Moving Average*

[2] UCM: *Unobserved Components Model*

[3] MAE: *mean absolute error*

[4] Library Prophet: *https://cran.r-project.org/web/packages/prophet/index.html*

[5] ACF: *Autocorrelation Function*

[6] PACF: *PArtial Autocorrelation Function*

[7] na.locf: *Last Observation Carried Forward*