BUDAPESTI GAZDASÁGI EGYETEM
ALKALMAZOTT TUDOMÁNYOK EGYETEME

PÉNZÜGYI ÉS SZÁMVITELI KAR

BUDAPESTI BUSINESS UNIVERSITY

FACULTY OF FINANCE AND

ACCOUNTANCY

THESIS

András Ecsédi

Full-time course

Business Informatics

Business data analyst specialization

1149 Budapest, Buzogány utca 10-12.
Telefon: (+36-1) 469-6600
Fax: (+36-1) 469-6610
www.uni-bge.hu

BUDAPESTI GAZDASÁGI EGYETEM
ALKALMAZOTT TUDOMÁNYOK EGYETEME

PÉNZÜGYI ÉS SZÁMVITELI KAR

2024

BUDAPEST BUSINESS UNIVERSITY

FACULTY OF FINANCE AND ACCOUNTANCY

Ship on the horizon: predicting piracy and ARAS incidents in the Straits of Malacca and Singapore with machine learning

Internal Advisor: Dr. Endre Kovács

Exterior Consultant József Csicsman

András Ecsédi

Full-time study

Business Informatics

Business data analyst specialization

2024

1149 Budapest, Buzogány utca 10-12.
Telefon: (+36-1) 469-6600
Fax: (+36-1) 469-6610
www.uni-bge.hu

Table of contents

Introduction

The Straits of Malacca and Singapore (SOMS) are a narrow, 805 km long channel of water surrounded by Singapore, Indonesia and Malaysia(Paterson, 2023) with an essential role in global trade: 40% of all globally traded goods pass through this territory (Paterson, 2023).

In light of this information, the number - 63 in 2023 (ReCAAP ISC, 2024) – and increasing tendency – only 55 happened in 2022 (ReCAAP ISC, 2024) – of piracy and ARAS (Armed robbery at sea) incidents is worrying.

The purpose of this paper is to predict these incidents: to this end I will run several machine learning algorithms on datasets gathered from global databanks, regional organizations and departments of statistcs.

Using these models, I will attempt to draw several conclusions: is there a relationship between the weather and the frequency of incidents? Which part of a month, what day of the week and which period of the day sees the most attacks? How accurately can the number of incidents in a month be predicted? My paper will attempt to answer these questions, among others.

As Chapter 4 will reveal, this isn't a recent problem and as such, solutions and organizations enacting them have emerged and the degree of regional cooperation and information sharing has increased over the past decades.

I hope my paper and its results will help the regional organizations aiming to fight piracy and ARAS incidents organize their resources more efficiently. This is extremely important, since these organizations do not possess enough resources to go on a sufficient number of patrols: they have a limited number of ships, limited fuel, and limited time and number of workers. These circumstances make the optimization of resource allocation exceedingly important.

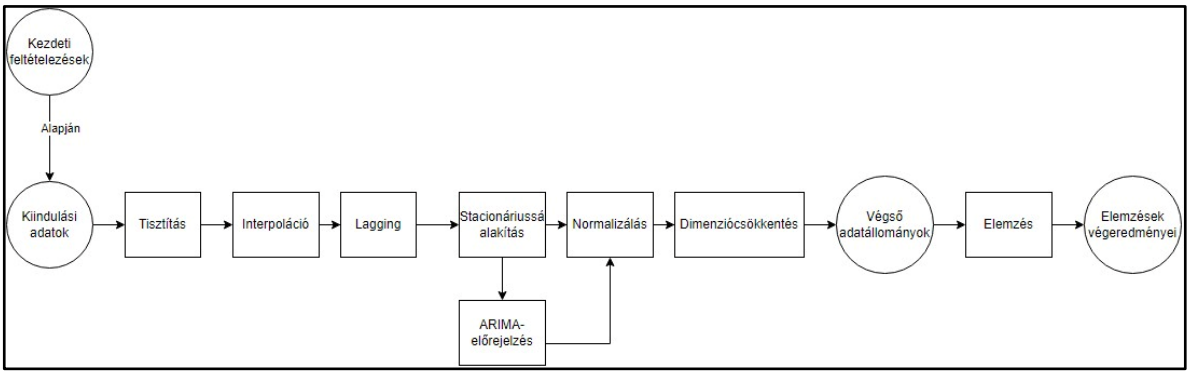
Another thing making this paper relevance is the fact that while the topic is suitable to be viewed through the lens of machine learning, it has remained relatively unexplored from this perspective thus far.

The paper's structure

Having finished the introduction, I will start by elaborating on the study's goals, then I'll move on to summarizing the literature on the topic, with special attention to the proposed causes of piracy and ARAS incidents. This will be followed by the description of the process of data

gathering and preparation, upon which I'll explain the machine learning models, their fine-tuning, their results and the evaluation of the conclusions I will draw using them. I will then make predictions for 2025 using the best-performing models.

The below diagram visualizes the data lifecycle.



Source: diagram produced by the author

Every piece of uncited data present in Chapter 7.8 is derived from the results of the models described in Chapter 5. The models, datasets and the programs executing each step of the data preparation process can be found in a GitHub repository I created (András, 2024).

The goals of the analysis

My paper has three principal goals:

a. Testing the current hypotheses on the causes of piracy and ARAS incidents. For instance, is there a relationship between the number of ARAS incidents and the political stability indices of the surrounding countries?
b. Making accurate predictions, using both classifying models (which aim to predict whether a given month is going to have "few"' or "many" attacks) and ones with numeric dependent variables (which try to predict the number of attacks a given month might have). I will then try to increase the accuracy of these models using weather data and other pieces of data (coordinates, time of the day, time of the month).
c. Predictions for 2025: based on the best-performing models during the fulfillment of the first two goals, I will run the best models on datasets expanded until the end of 2025 using ARIMA, then visualize and evaluate the results.

To sum up, the paper's goal is to test established theories and to create models capable of making accurate predictions.

Summarizing the analyzed situation

While ReCAAP, the regional organization documenting piracy and ARAS incidents possesses data from other Asian bodies of water, my analysis will concern only the Straits of Malacca and Singapore, as I've only gathered data and established hypotheses regarding this channel, which might not be relevant or helpful when trying to understand the broader Asian situation: the general trend of incidents diverges from the SOMS' trends – for instance, in 2021, the number of attacks decreased to their pre-pandemic levels everywhere but the SOMS. (Storey, 2022)

A továbbiakban a SOMS-ról, illetve a tengeri fegyveres támadások természetéről írok le releváns, az elemzés későbbi lépéseit meghatározó adatokat. In the following section, I will describe the data regarding the area and the nature of piracy and ARAS incidents that will go on to determine the analysis' subsequent steps.

A brief examination of the SOMS through the lens of geography, economics and international law

The Straits of Malacca And Singapore, an 805 km-long, 600 m-wide (at its narrowest point), 25 meters deep (at its most shallow point) body of water surrounded by Malaysia, Singapore and Indonesia (The Nippon Foundation, dátum nélk.), possesses undeniable relevance for world trade: as the corridor between the Indian and Pacific Oceans (Haire, 2021), almost half of global seaborne trade happens through this route. It holds further importance for certain countries: more than 80% of Japan's oil imports passes through the SOMS. (The Nippon Foundation, dátum nélk.). It also plays a role in connecting the other major Asian economies – Taiwan, China, India, Singapore, Vietnam, South Korea, Thailand, Indonesia, Malaysia and the Philippines.

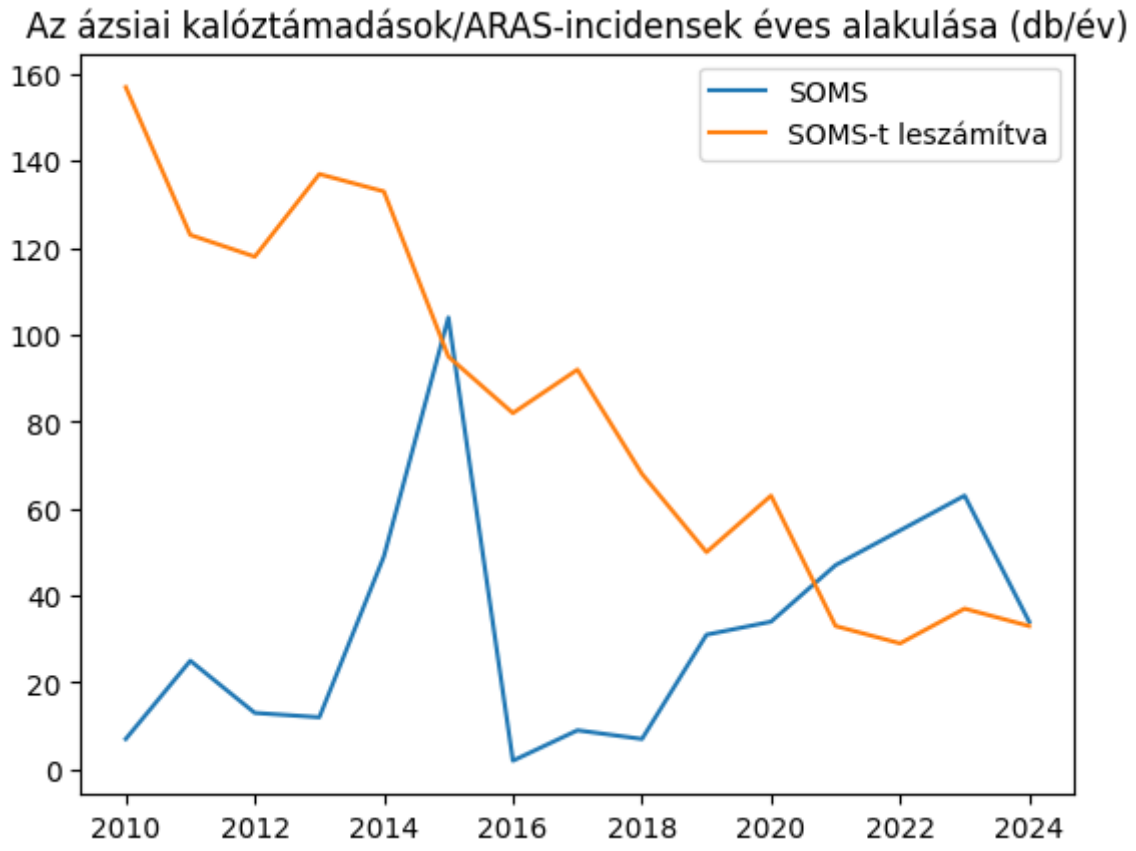
The SOMS are within the territorial waters of Singapore, Malaysia and Indonesia and as such, the responsibility jurisdiction over the Straits falls on these three countries since 1994 (The Nippon Foundation, dátum nélk.).

Piracy and ARAS incidents in the SOMS: motives, solutions and trends

Before elaborating on the incidents' motives, trends and solutions, I have to explain the difference between piracy and ARAS incidents: the further happens on international waters, the latter on the territorial waters of a given country. (ReCAAP ISC, 2024).

98% of attacks between 2019 and 2013 were ARAS incidents. (ReCAAP ISC, 2024). In the paper, I will make no distinction between piracy and ARAS incidents, as one of the starting

points of my analysis that when it comes to predicting them, there is no difference between the two.



Source: visualization (based on ReCAAP ISC' yearly reports between 2010 and 2024)

Ahogy a fentebbi, a RECAAP nevű szervezet által összeállított éves kimutatások alapján készített ábrán is látszik, az elmúlt években jelentősen megnőtt a SOMS-béli kalóztámadások száma. A most is tartó növekvő tendencia 2018-ban indult, ezelőtt két nagyobb kiugrást figyelhettünk meg, 2011-ben, illetve 2013 és 2015 közt.

As the diagram above shows, the years past have seen a significant increase in SOMS-based piracy ad ARAS incidents. The current increasing trend started in 2018, before which there were two major increases in 2011 and between 2013 and 2015.

Piracy isn't a new phenomenon in the region, records date back to as far as the 14th century (Bileta, 2022). There are several causes behind this, which I've grouped into four main categories:

a. Potential gain: as can be deduced from the information above, many ships pass throught he straits, many of which carry valuable goods.

b. Geographical characteristics: movement is slow in the SOMS' narrow, congested channel (Paterson, 2023), which makes attacking ships easier.. Furthermore, the SOMS' is home to several small islands (Paterson, 2023) and is connected to several rivers (Watson, 2013), which makes hiding and escaping easier.
c. The authorities' responsibility: the patrolling capacity of the civil and official organizations (Eg. coast guards, navies) responsible for doing so is severely limited (Storey, 2022). The situation is further exacerbated by the fact that several law enforcement officials have been caught exchanging information with pirates (Spiess, 2019). The problem also has a legal dimension: Malaysian law does not define ARAS incidents (Panneerselvam & Ramkumar, 2023) an many have accused Indonesia of having more lenient punishments for piracy than the region's other countries (Panneerselvam & Ramkumar, 2023).
d. Economic, social and political reasons. The established hypothesis is that there is a link between the region's economic stability and the frequency of pirate attacks: the political instability and recession born out of the 1997 Asian Financial Crisis gave way to an increase in incidents of piracy at the end of the 1990s, with many poor coastal Indonesians and Malaysians turning to piracy, doing which was made easier by political instability, especially in Indonesia (Raymond, 2009). Since most pirates used to be/are poor coastal citizens, the fishing industry's performance significantly affects the frequency of pirate attacks (Paterson, 2023).

Since the data I have is economic, social and political in nature (mostly economic), that's what my paper will focus on: I will attempt to prove the link between piracy and the aforementioned theorized reasons and if successfully, I will try to make precise predictions.

The most important of the responses to piracy is the foundation of ReCAAP (Regional Cooperation Agreement on Combating Piracy and Armed Robbery against Ships in Asia), a multilateral organization with 21 members (10 in 2006). The organization's goal is deepening regional cooperation, which is the reason behind the establishment of ReCAAP ISC (Information Sharing Centre), which documents, analyzes and publishes reports on incidents of piracy and ARAS (https://www.recaap.org, dátum nélk.).

Data

This chapter concerns the datasets that are to be processed by the models. Since I found monthly predictions based on economic, social and political trends both realistic and fruitful, I gathered this data into monthly datasets. I also made a separate dataset containing information regarding the incidents themselves.

The first dataset contains data for four countries: Malaysia, Indonesia, Singapore and Thailand, since these are the countries surrounding the Straits of Malacca. My preliminary intuition based on the literature on the topic was that Thailand's effect on the region is smaller than that of the other three countries (after all, it wasn't really mentioned) so I went on to use both datasets containing Thai data, and ones without them.

Gathering Data

Incident Data

I've gathered incident data from two sources: ReCAAP ISC and OpenMeteo.

ReCAAP ISC's incident reports between 2010 and 2021 contain tables of incident data in PDF format, which is not ideal for data analysis, but they were my best option. I gathered incident data from ReCAAP's yearly reports between 2010 and 2021 (www.recaap.org, dátum nélk.),and from the 2022 (www.recaap.org), (recaap.org) and 2024 (up to 22 September) (www.recaap.org) lists of incidents that can be found on the organization's homepage.

Extracting the PDFs from the tables proved challenging: I used Adobe Acrobat's free online PDF-to-Excel program to do so (www.adobe.com, dátum nélk.): I uploaded the pages of the reports that contained the tables and transformed them into Excel tables. The process was not perfect: many of the tables columns were divided vertically and horizontally in a way they shouldn't have been, the severity of the incidents didn't make it into the table when marked only by color as it was the case in some years and the program's text analysis left a lot to be desired (it frequently interpreted 0s as O's and vice versa).

I could only give a manual solution to the first two problems: I divided or merged cells by hand in order to get a well-structured table. A method that went on to help me a lot was transforming .xlsx files into CSV-files, then back to .xlsx files, thereby dropping empty cells. (This solution wasn't always viable, as the program couldn't handle the dotted-line separation present in one of the tables).

The tables I got in the steps above contained several cells that were divided into many vertically, however, Python's Pandas (pandas.pydata.org, dátum nélk.) library (a library made for data analysis and data manipulation) proved eligible to solve this: I placed the tables into dataframes and by identifying its vertical lines (done with the help of the "CAT 1-4" column) I managed to create tables that followed the vertical and horizontal structure of those found in the PDFs.

My solutions to the other problem will be detailed in chapter 5.2.1.

I've made some changes to the original tables: the 'date' and 'time' variables originally shared a column, as did 'latitude' and 'longitude' (the latter contained other data as well). I divided them into separate columns. I then went on to format the date uniformly throughout the years and converted the latitude-longitude format to decimal degrees (this was necessary for making API calls later) with the aid of Pandas.

What follows is the list of variables I established:

- Severity: ranging between CAT 1 and CAT 4 CAT 1-4 with CAT 1 (the assailants are armed with either knives or firearms and the crew was subjected to a form of violence, fi. kidnapping, injury or death) being the most and CAT 4 being the least severe (the assailants weren't armed and they either took nothing or only something of little value). A possible value of this variable is "Attempted", which means the severity can't be evaluated, since the attempt was unsuccessful.
- Type(s) of ship(s): the type(s) of the attacked ship(s). Fi. Oil tanker, General Cargo Ship.
- Date: the date of the attack in YYYY/MM/DD format.
- Time: The time at the beginning of the attack (if the attack spanned several hours, the endpoint is also present).
- Longitude
- Latitude
- Territory: the territory where the attack happened. Fi. SOMS, South China Sea, Singapore, etc. This variable was necessary for filtering out the incidents outside the SOMS.

I gathered the weather data with the API of OpenMeteo, an organization providing access to global weather data: using the API client provided by Python's openmeteo_requests library, I wrote a program that gathered weather data by coordinates, date and time (in case of incidents

lasting more than an hour, the starting time). Due to the specific way to API worked, I first gathered daily, then hourly data.

Out of the variables found on OpenMeteo's site (open-meteo.com, dátum nélk.), I gathered the ones that might help a potential attacker decide whether to go through with an attack or not. The variables are as follows:

- Cloud_cover_low: Low clouds and fog up to a height of 3 km, as a percentage of the territory.
- Temperature_2m: The temperature 2m above ground, in Celsius degrees.
- Wind_speed_10m: Wind speed 10 m above ground in km/h.
- Wind_speed_100m: Wind speed 100 m above ground in km/h.
- Weather_code: The given day's most severe weather state, based on WMO codes (WMO Code Table 4677, dátum nélk.). The codes found in my paper and their descriptions are detailed in the table below.

Code	Description
0	No cloud formation.
1	Clouds are generally dissolving or are becoming less developed.
2	State of sky generally unchanged.
3	Slight cloud formation.
51	Drizzle, not freezing, continuous, slight at time of observation.
53	Drizzle, not freezing, continuous, moderate at time of observation.
55	Drizzle, not freezing, continuous, heavy at time of observation.
61	Rain, not freezing, continuous, slight at time of observation.
63	Rain, not freezing, continuous, moderate at time of observation.
65	Rain, not freezing, continuous, heavy at time of observation.

- Wind_gusts_10m: The maximum speed of wind gusts 10 m above ground measured in the past hour in km/h.
- Rain: Rain in the past hour, measured in mm.

Connecting the two datasets described above yielded the dataset containing all incident data, in which incidents were uniquely identified by their ordinal numbers. The dataset has some empty

cells: in some cases, the date, time and the coordinates of the incidents were not disclosed in the original reports, as such, the weather data based on these variables are also missing.

I filtered the datasets so that only SOMS-and Singapore-based incidents would be kept. While the latter incidents didn't happen in the SOMS, they were close enough for me to think that it would make sense to group them in. I did the filtering with the "Territory" variable. The final dataset contains 492 incident descriptions.

Country data

I gathered the data from the countries in the region (Malaysia, Indonesia, Singapore and Thailand) from several main sources: the countries' departments of statistics, the Worldbank's DataBank, FRED (the St. Louis Federal Reserve's economic database), the IMF's DataMapper platform and the OECD Data Explorer platform.

Since gathering this data and creating a unified dataset based on them did not pose a technological challenge and the variables that would go on to be relevant will be described later on in the paper, the detailed description of this data can be found in the appendix.

Preparing the data for the models

Data Cleaning

As mentioned during the section about data gathering, creating Excel tables based on ReCAAP's yearly reports was not without its challenges. Below I shall detail the cleaning process of the variables that made it into the final dataset.

The biggest problem lay in character recognition: there was many an instance of 0s interpreted as Os or 1s interpreted as 7s or Ls, and vice versa. These primarily posed a difficulty when it came to the 'date', 'time', 'latitude', 'longitude' and 'territory' variables. For instance, in the case of the latitude-longitude variables, the "°" symbol was often misconstrued as an "O" or a "O".

I applied a hybrid solution to the difficulties; solving the problem in Excel and Python. There were special errors that only occurred once or twice, which I corrected in Excel, as it wasn't worth writing a program for these. Sometimes I even applied column-wide changes in Excel (Fi. when it came to the coordinates).

The "date", "time", "latitude" and "longitude" variables weren't supposed to contain non-numeric characters (with a few exceptions, fi. "hrs"), so I filtered out the ones that contained

them and corrected them in Excel manually (as both “1” and “7” were often processed as “L”, so correcting them was only possible after cross-referencing them with the original reports.

I also faced challenges when extracting the type(s) of the attacked ship(s): the first such problem was the variable being in the same cell with other values (ship name, ID, flag) which were not relevant to my research, and the separation between them could not be recognized algorithmically.

I applied a multi-step solution to this problem: in the 2022-24 incident reports, the ship type appeared in a column of its own. I gathered these in a list, then iterated through the 2010-2021 data, where if a cell contained any of the values in a list, I added it to a new column.

Another challenge was the problem of substrings: the word “tanker” was present both in the ship types “tanker” and “chemical tanker”. I solved this by ordering the list of ship types by length (decreasingly), so when I iterated through the data, It would only add the appropriate one to the new column (this solution was appropriate, since there were no cells which contained both “tanker” and “chemical tanker” as different types of ships).

As a result of this process, I managed to identify the ship type in most cases. I corrected the remaining cases (fi. the ones where the PDF-to-table program added an unnecessary extra space in between two words or ones where one of the aforementioned problems appeared) using a hybrid method: I corrected them in the Excel table manually, then I iterated through the dataframe based on the refreshed, corrected table again.

Interpolation

Most pieces of data concerning countries was not monthly, but rather yearly, quarterly (in some cases, biyearly or triennial). However, since my goal was for my models to make monthly predictions, I had to transform them into monthly timeseries. I did this by interpolation.

The point of interpolation is to connect two data points with a curve (fi. linear, polynomial), thereby estimating the values between them (byjus.com, dátum nélk.). Python’ Pandas library gives users the means to do this: when creating the datasets containing country data, I transformed them into a monthly format and filled the empty spaces with the pandas.DataFrame.interpolate() method, using both “time” and “linear” interpolation. Both of them connect the data points with a linear curve, however, the latter takes into account factors like the length of months that the former doesn’t. For this reason, I only subjected the data made with time-based interpolation to analysis.

Lagging

While the incident data I've gathered starts in 2010, my other dataset (the one containing country data) has variables going as far back as 2008. The reason behind is that I felt like there could be a link between the 2008 Financial Crisis and the number of attacks in 2010, which I didn't want to leave unexamined.

I used lagging (GeeksForGeeks, 2024) to find a link: I iterated through my dataset's columns and where the first non-null value was in 2008, I created lagged columns that pushed the original column's values forward by 24 months (so the value of a column in January 2008 went on to be a value in January 2010 in the lagged column).

I divided my datasets into two based on how the co-existence of lagged and non-lagged columns (fi. lagged GDP and original GDP) was handled: one group kept both of them, the other only kept the lagged ones.

Stationarity

One of the essential conditions of regression models is that the timeseries they're processing is stationary (its properties don't change with time) (Faridi). While not all of my models are regression models (fi. there are decision trees), I made the data stationary for two key reasons: firstly because the expanded dataset mentioned in the description of the paper's third goal was made with ARIMA, a requirement of which is stationarity) and secondly because decision tree-based models struggle with trend extrapolation, to which stationarity could prove to be a cure.

I went through the following process to make the timeseries stationary: I used the pmarima.arima.util library's ndifss method to determine how many times each column had to be differentiated to be considered stationary (Gupta, 2024), then I did so the appropriate amount of times. While the differentiated datasets contained variables which used to be discrete (fi. number of crimes in Indonesian provinces), interpolation them made them continuous and as such, they could be made stationary.

Normalization

To handle potential outliers, I normalized the data; transforming it so that it would fit into certain boundaries. I did so using three methods:

- Min-Max scaling: values are transformed so that they fit into a

range of 0 to 1. This is done by substracting a given column's minimum value from each value, then dividing the result by the result of a substraction between the maximum and minimum values of the column. (Nalcin, 2022).

- Robust scaling: this scaling method removes the median and scales values based on the interquartile range. Its name is derived from its resilience in the face of outliers. (Nalcin, 2022)
- Standardizing: the mean of the column is subtracted from each of its values, then the result of the operation is divided with the column's standard deviation. (Nalcin, 2022)

Python's scikit-learn library provided me with the means to do normalize the data all three ways.

I've also tried filtering outliers based on zscore-normalization. (Bobbitt, 2021).

The reason I didn't end up using this method was that since the datasets had over 100 dimensions, even if a given column contained only a few outliers, they piled up, leaving me with a much smaller, entirely useless dataset.

Dimensionality reduction and feature selection

The aforementioned process left me with datasets containing hundreds of variables. I had several reasons to reduce their dimensionality: firstly, as a means of efficiency (faster runtime, smaller memory requirements) and secondly, to handle redundant and irrelevant features.

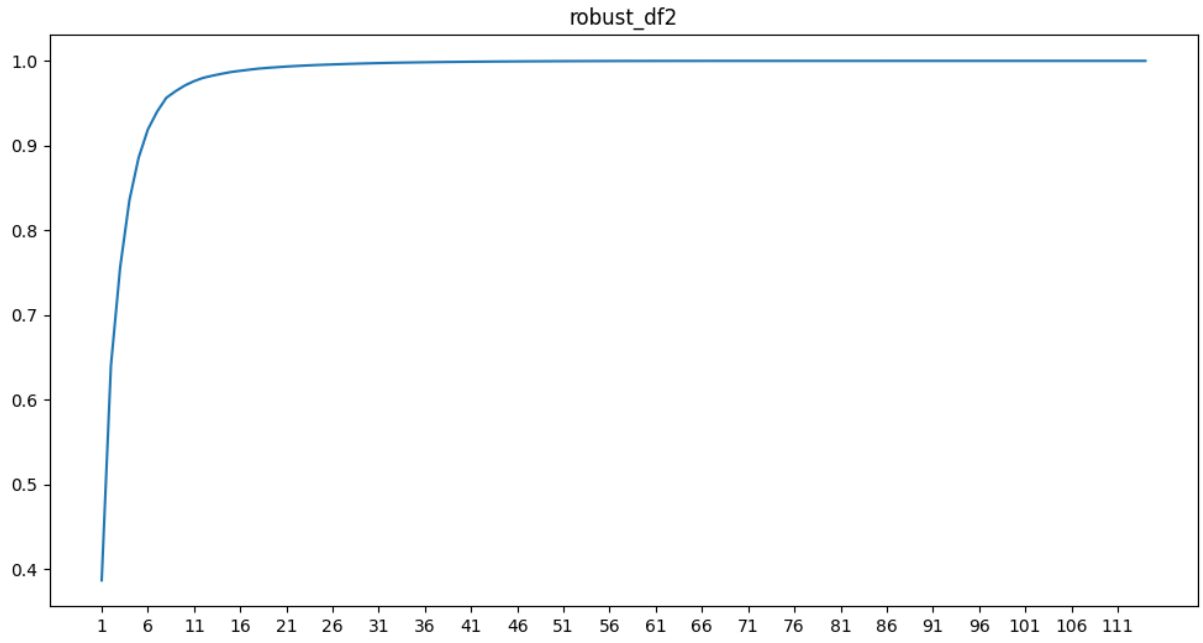
I reduced the datasets' dimensionality using two methods: PCA and AutoEncoders. This chapter is dedicated to the theoretical background and the application of these methods.

Principal component Analysis

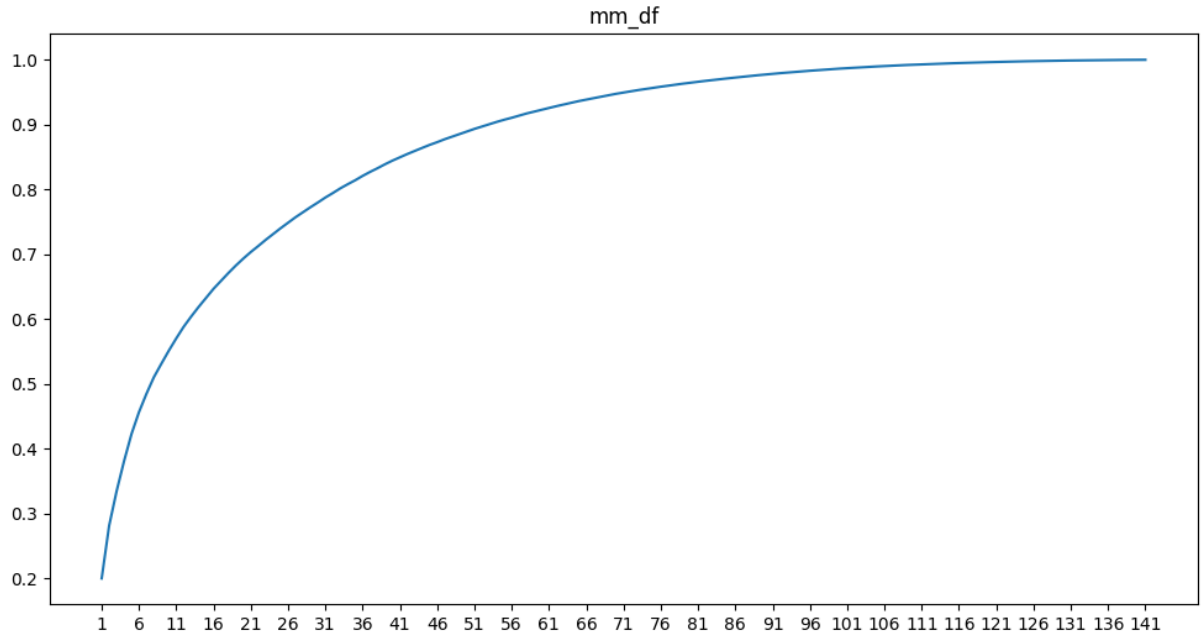
PCA (Principal Component Analysis) is a linear dimensionality reduction technique, the point of which is to reduce the information content of a dataset by organizing it into the original variables' linear combinations that correlate with each other the least. (IBM, 2023).

I determined the ideal number of principal components using the elbow method (Mangale, 2020): I ran the PCA-algorithm on a given dataset with the number of target dimensions being in a range of 0 and the maximum possible dimensions (the smaller value of the number of columns and rows the dataset has), then visualized the variance explained by the model, given the number of dimensions.

Two examples of results:



The result of the elbow method on the dataset containing Malaysian, Indonesian and Singaporean data, normalized with robust scaling and keeping only the lagged variables in case of conflict



The result of the elbow method on the dataset containing Malaysian, Indonesian and Singaporean data, normalized with min-max scaling and keeping both the lagged and non-lagged variables in case of conflict

Based on the plots, I attempted to find the elbow point, the point where the variance explained by the model starts decreasing sharply. Where there was no observable elbow point, I chose the smallest number of dimensions which still retained 90% of the explained variance. The elbow point of each dataset can be found in the table below.

	Contains thai, data, only lagged	Contains thai, data, both lagged and original	Doesn't contain thai, data, only lagged	Doesn't contain thai, data, both lagged and original
Min-Max Scaling	25	55	21	55
Robust Scaling	8	8	8	8
Standard Scaling	26	65	26	65

AutoEncoder-based dimensionality reduction

I used the ideal number of dimensions found using the elbow method for another, neural net-based dimensionality reduction method.

When using AutoEncoders for dimensionality reduction, a neural net encodes data into a lower-dimension space and then decodes that, thus rebuilding the original input. (Rajan, 2021). In this instance, the latter step is to test and fine-tune the encoding. (Riswanto, 2023): by comparing the original dataset and the one resulting from the decoding step (fi. based on mean squared error), the dimensionality reduction process can be improved.

My model had 64 dense layers (they're dense because each neuron of each layer is connected the every neuron of the preceding layer (dátum nélk.). Its activation function is LeakyReLu.

LeakyReLu is similar to ReLu. In the case of the latter, a positive input becomes the output, while a negative input makes the output zero. A possible problem emerging from this is the "dying ReLu" phenomenon, which originates from ReLu only receiving negative values, in which case the resulting null value makes backpropagation all but impossible. LeakyReLu solves this by handling negative values differently: instead of zeroing out the output, it multiplies the input by a small number (in the case of my code, 0.1), making the product the output (Olamendy, 2023).

In order to regulate the encoding process, I introduced a Dropout hyperparameter of 0.1, the point of which is to set 10% of each layer's neurons to 0, in order to prevent overfitting. Similarly to the encoding process, decoding was done using LeakyRelu (with the alpha hyperparameter being 0.1 here, as well) and a 64-layer dense neural net.

The AutoEncoder, which relied on the results of the coding-encoding process, used the "adam" method to optimize the gradient descent. The „adam" method is a composite of two gradient boosting methods (GeeksforGeeks, 2024): the "momentum" method, which calculates with the gradients' exponentially weighted average in order to reach the minima faster, and the RMSP method, which uses the gradients' exponential moving average.

The AutoEncoder calculates loss based on mean squared error. When training the model, I split the data into training and test data in an 80-20 rate. As a result of the usage of the "shuffle" hyperparameter, the split happened randomly, making overfitting harder.

The maximum number of epochs was 50, so the model could only run 50 times on the training data. Since I introduced early stopping (with a patience time of 5 epochs and validation loss as the variable based on which the decision to stop is made), this number could be even lower.

Based on the AutoEncoders' results, I encoded the datasets into the ideal number of dimensions.

Three main difficulties arose: the AutoEncoders couldn't handle the datasets normalized with robust scaling, resulting in datasets of only null values. However I tuned the hyperparameters, I couldn't solve this, so I only ran the AutoEncoders on datasets normalized with the other two methods.

The number of epochs posed another problem: originally there were 100, which resulted in null values for many a dataset. Eventually I solved this by decreasing this number to 50. I split it in half, because the model with the original number of epochs did well in datasets twice the size of the current ones (the original datasets that contained columns made with both methods of interpolation).

When running the program, certain runs had the second problem repeat even with 50 epochs. This was solved by a conditional rerun (if the resulting dataset was full of null values, the AutoEncoders restarted).

Final Datasets

The table below shows the datasets which I went on to process with the models listed in Chapter 6. All in all, 32 such datasets were created.

	Without Thai data		With Thai Data	
	Only lagged	Both lagged and original	Only lagged	Both lagged and original
PCA	3	3	3	3
AE	2	2	2	2
-	3	3	3	3

In all cases, the first month contained in a dataset is March 2010 and the final one is December 2022. The reason behind this is that while a decent chunk of variables has data up to 2024, many of them only last until December 2022. While I could have made predictions using ARIMA (This will happen in Chapter 8), considering the paper's first goal I first ran these models on data that hadn't been artificially extended.

Models

The first half of this section is dedicated to the explanation of the models' theoretical backgrounds, hyperparameters and optimizations thereof. It will be followed by the models' evaluations and the interpretation of the best-performing models' results.

Models with a numeric target variable

I shall now explain the workings of those of my models that have a numeric target variable. In all of these cases, the target variable was the same; the number of monthly pirate attacks. The independent variables were the ones described in the previous chapter.

Linear Regression

Linear regression presumes a linear relationship between dependent and independent variables (Kanade, 2023). When running this model, I split the dataset two different ways (70-30 and 80-20 percent split). I processed every one of the datasets visible in Chapter 5.2.6.1.'s table with this model.

Linear Regression With RFE

Similarly to dimension reduction, RFE (Recursive Feature Elimination) is a method of decreasing the size and complexity of datasets. It can be applied in conjunction with another model (in this case, linear regression).

It chooses to keep a given number of the original dataset's independent variables within the bounds of an iterative process: it first processes the dataset with every independent variable, then, upon each iteration, then eliminates the least important independent variable (determined by the regression coefficient; the degree of change in the dependent variable caused by the given independent variable) until only a predetermined number of independent variables is left. (Analytics Vidhya, 2024).

The number of independent variables to be kept is determined by the user before running the program. I set this number to be 1/10th of the number of independent variables in the dataset, in accordance with the "one in ten" rule (Chowdhury & Turin, 2020).

When running this model, I split the dataset two different ways (70-30 and 80-20 percent split). I processed every one of the datasets visible in the bottom row of Chapter 5.2.6.1.'s table with this model (the datasets with no prior dimension reduction techniques applied to them).

```
def linreg(df, inc, ts):
    for x in df.columns:
        if x=="inc":
            df.drop(columns=x, inplace=True)
    for x in df.columns:
        if ".1" in str(x):
            df.drop(columns=x, inplace=True)
    y=inc['incidents_per_month']
    df=df
    if 'Unnamed: 0' in df.columns:
        df.index=df['Unnamed: 0']
        df.drop(columns= "Unnamed: 0", inplace=True)
    x=df
    x_train, x_test, y_train, y_test = train_test_split(
            x, y, test_size = ts, random_state = 0)
    lm = LinearRegression()
    lm.fit(x_train, y_train)
    pred = lm.predict(x_test)
    mse = mean_squared_error(y_test, pred)
```

Source: The model's Python-based realization in my program

Ridge regression

Ridge regression is a regularized version of regression, created with the aim of managing overfitting (Shelar, 2023) by keeping its weights low. This is achieved by the introduction of a penalty function based on the sum of the square of the parameters (Kuknyó, Üzleti Elemzések Módszertana 3. Gyakorlat: Regularizált Modellek, 2024, old.: 15).

Its two most important hyperparameters are the degree of the polynomial and "alpha", the coefficient determining the degree of regularization. Another hyperparameter of note is "solver" (Cholesky in the case of my code) (scikit-learn.org, dátum nélk.).

The model's optimization was done based on the two aforementioned hyperparameters with the grid search method: the MSE (Mean Squared Error) achieved by the model was evaluated with different values of the two hyperparameters. The features of the best-performing model (the one with the lowest MSE) were saved and its results were finalized. In my code, the "alpha" hyperparameter's possible values were 0.01, 0.1 and 1, while the possible degrees of the polynomial were 2, 3 or 4. When running this model, I split the dataset two different ways (70-30 and 80-20 percent split). I processed every one of the datasets visible in the bottom row of Chapter 5.2.6.1.'s table with this model (the datasets with no prior dimension reduction techniques applied to them).

```python
def ridge1(df, inc, ps, rs, ts):
    for x in df.columns:
        if x=="inc":
            df.drop(columns=x, inplace=True)

    for x in df.columns:
        if ".1" in str(x):
            df.drop(columns=x, inplace=True)
    legjemese=0
    for p in ps:
        for r in rs:
            if 'Unnamed: 0' in df.columns:
                df.index=df['Unnamed: 0']
                df.drop(columns= "Unnamed: 0", inplace=True)
            poly_deg = p
            ridge_alpha = r

            reg = Ridge(alpha=ridge_alpha, solver="cholesky", random_state=42)
            model = Pipeline([
                ("poly_features", PolynomialFeatures(degree=poly_deg, include_bias=False)),
                ("regul_reg", reg)
            ])
            x=df
            y=inc['incidents_per_month']
            x_train, x_test, y_train, y_test = train_test_split(
            x, y, test_size = ts, random_state = 0)

            model.fit(x_train, y_train)
            pred = model.predict(x_test)
            trainpred=model.predict(x_train)
            mse = mean_squared_error(y_test, pred)
            if legjemese==0 or mse<legjemese:
                legjemese=mse
                legjpred=pred
                legjtrain=trainpred
                legjp=p
                legjr=r
```

Source: The model's Python-based realization in my program

Ridge regression with early stopping

Since ridge regression was prone to overfitting (as I evaluated the model's performance based on MSE, which favored accuracy over generalization ability, I introduced early stopping: each version of the model (a version being defined as the model with given values of its hyperparameters) had one hundred epochs to run and if its performance (measured by MSE) didn't improve after 10 epochs, the program stopped, saving the results of the best-performing model up to that point.

When running this model, I split the dataset two different ways (70-30 and 80-20 percent split). I processed every one of the datasets visible in the bottom row of Chapter 5.2.6.1.'s table with this model (the datasets with no prior dimension reduction techniques applied to them).

```python
def ridgerse(df, inc, ps, rs, ts):
    for x in df.columns:
        if x=="inc":
            df.drop(columns=x, inplace=True)

    for x in df.columns:
        if ".1" in str(x):
            df.drop(columns=x, inplace=True)
    legjemese="b"
    for p in ps:
        for r in rs:



            poly_deg = p
            ridge_alpha = r

            reg = Ridge(alpha=ridge_alpha, solver="cholesky", random_state=42)
            rfe = RFE(estimator=reg, n_features_to_select=int(len(df)/10))
            model = Pipeline([
                ("poly_features", PolynomialFeatures(degree=poly_deg, include_bias=False)),
                ("feature_selection", rfe),
                ("regul_reg", reg)
            ])
            x=df
            y=inc['incidents_per_month']
            print("x")
            x_train, x_test, y_train, y_test = train_test_split(
            x, y, test_size = ts, random_state = 0)
            print("y")
            model.fit(x_train, y_train)
            pred = model.predict(x_test)
            trainpred = model.predict(x_train)
            mse = mean_squared_error(y_test, pred)
            print(mse)
            if legjemese=="b" or mse<legjemese:
                legjemese=mse
                legjpred=pred
                legjtrain=trainpred
```

Source: The model's Python-based realization in my program

LASSO regression

LASSO (Least Absolute Shrinkage and Selection Operator) regression regularizes the model with the introduction of a penalty function based upon the sum of its parameters' absolute values (Kuknyó, Üzleti Elemzések Módszertana 3. Gyakorlat: Regularizált modellek, 2024, old.: 17). Its two most important parameters are the degree of the polynomial and alpha, the coefficient determining the degree of regularization.

The model's optimization was done based on the two aforementioned hyperparameters with the grid search method: the MSE (Mean Squared Error) achieved by the model was evaluated with different values of the two hyperparameters. The features of the best-performing model (the one with the lowest MSE) were saved and its results were finalized. In my code, the "alpha"

hyperparameter's possible values were 0.01, 0.1 and 1, while the possible degrees of the polynomial were 2, 3 or 4. As with the previous models, I split the dataset two different ways (70-30 and 80-20 percent split). I processed every one of the datasets visible in the bottom row of Chapter 5.2.6.1.'s table with this model (the datasets with no prior dimension reduction techniques applied to them).

```
def indiana(df, inc, ps, rs, ts):
    for x in df.columns:
        if x=="inc":
            df.drop(columns=x, inplace=True)


    for x in df.columns:
        if ".1" in str(x):
            df.drop(columns=x, inplace=True)
    legjemese=0
    for p in ps:
        for r in rs:
            if 'Unnamed: 0' in df.columns:
                df.index=df['Unnamed: 0']
                df.drop(columns= "Unnamed: 0", inplace=True)

            poly_deg = p
            lasso_alpha = r

            reg = Lasso(alpha=lasso_alpha)
            model = Pipeline([
                ("poly_features", PolynomialFeatures(degree=poly_deg, include_bias=False)),
                ("regul_reg", reg)
            ])
            x=df
            y=inc['incidents_per_month']
            x_train, x_test, y_train, y_test = train_test_split(
            x, y, test_size = ts, random_state = 0)

            model.fit(x_train, y_train)
            pred = model.predict(x_test)
            mse = mean_squared_error(y_test, pred)
            if legjemese==0 or mse<legjemese:
                legjemese=mse
                legjpred=pred
                legjtrain=model.predict(x_train)
```

Source: The model's Python-based realization in my program

LASSO regression with early stopping

In order to improve my LASSO regression model's generalization ability, I introduced early stopping: each version of the model (a version being defined as the model with given values of its hyperparameters) had one hundred epochs to run and if its performance (measured by MSE) didn't improve after 10 epochs, the program stopped, saving the results of the best-performing model up to that point.

When running this model, I split the dataset two different ways (70-30 and 80-20 percent split). I processed every one of the datasets visible in the bottom row of Chapter 5.2.6.1.'s table with this model (the datasets with no prior dimension reduction techniques applied to them).

```
def lassoes(df, inc, ps, rs, ts):
    for x in df.columns:
        if x=="inc":
            df.drop(columns=x, inplace=True)
    legjlegjemese=float('inf')
    for p in ps:
        for r in rs:
            if 'Unnamed: 0' in df.columns:
                df.index=df['Unnamed: 0']
                df.drop(columns= "Unnamed: 0", inplace=True)
            reg = Lasso(alpha=r, warm_start=True, max_iter=1, random_state=42)
            model = Pipeline([
                ("poly_features", PolynomialFeatures(degree=p, include_bias=False)),
                ("regul_reg", reg)
            ])
            x=df
            y=inc['incidents_per_month']
            x_train, x_test, y_train, y_test = train_test_split(
            x, y, test_size =ts, random_state = 0)
            legjemese = float('inf')
            patience=10
            noimp = 0
            for epoch in range(1000):
                model.fit(x_train, y_train)
                pred = model.predict(x_test)
                mse = mean_squared_error(y_test, pred)

                if mse < legjemese:
                    legjemese = mse
                    noimp = 0
                else:
                    noimp += 1
                if noimp >= patience:
                    break
            if legjemese < legjlegjemese:
                legjlegjemese = legjemese
                bestepoch=epoch
                bestmodel=clone(model)
                legjpred=pred
                legjtrain=model.predict(x_train)
                legjp=p
                legjr=r
```

Source: The model's Python-based realization in my program

Regression with gradient boosting

Gradient boosting is an ensemble machine learning method built on iteratively combining the results of several weaker models into a better model (Verma, 2023). This is achieved with the

aid of gradient descent: in each iteration, the algorithm improves the model's parameters (based on the results of the previous iteration) in order to minimize residuals (Verma, 2023).

I built the model using the GradientBoostingRegressor found in Python's scikit-ensemble library (scikit-learn.org, dátum nélk.). I used its predetermined hyperparameters: a learning speed of 0.1, a loss function optimized based on squared error and one hundred boosting stages to perform.

When running this model, I split the dataset two different ways (70-30 and 80-20 percent split).

```
def graddescent(df, inc, ts):
    for x in df.columns:
        if x=="inc":
            df.drop(columns=x, inplace=True)

    for x in df.columns:
        if ".1" in str(x):
            df.drop(columns=x, inplace=True)

    y=inc['incidents_per_month']
    df=df
    if 'Unnamed: 0' in df.columns:
        df.index=df['Unnamed: 0']
        df.drop(columns= "Unnamed: 0", inplace=True)
    x=df
    x_train, x_test, y_train, y_test = train_test_split(
            x, y, test_size = ts, random_state = 0)

    model = GradientBoostingRegressor(max_depth=2)
    model.fit(x_train,y_train)
    pred=model.predict(x_test)
    mse=mean_squared_error(y_test, pred)
    print(mse)
```

Source: The model's Python-based realization in my program

Classification models

Preparatory steps

Classification models need categories into which data can be divided. To this end, I created two categories based on whether the number of incidents in a given month were above or below the average number of attacks (I chose their average over their median because combined with the months with zero attacks, it wouldn't have represented the data well). The models described in the following section try to divide the data into these two categories based on the independent variables used in the previous models.

Logistic regression

Logistic regression is a method that achieves binary classification using probability estimates: the model makes an estimation on whether a given variable belongs to the positive class – if the estimation exceeds a threshold value, it does (Kuknyó, Üzleti Elemzések Módszertana 2. Előadás: Osztályozás, 2024, old.: 26). The estimation is achieved with the sigmoid function: a linear prediction is produced, then it gets inserted into the logistic function (Kuknyó, Üzleti Elemzések Módszertana 2. Előadás: Osztályozás, 2024, old.: 27).

I split the analyzed datasets into training and test data in an 80-20 percent split. I originally intended to do a 70-30 split, as well, however, seeing the accuracy of the predictions made with the 80-20 percent split, I opted to forgo this option, as the increase in generalization skills wouldn't have been worth the decrease in accuracy. I processed all 32 datasets with this model.

```
def logreg(df, clmet):
    if 'Unnamed: 0' in df.columns:
        df.index=df['Unnamed: 0']
        df.drop(columns= "Unnamed: 0", inplace=True)

    for x in df.columns:
        if ".1" in str(x):
            df.drop(columns=x, inplace=True)


    df["inc"]=clmet
    df.dropna(inplace=True)
    inc=df["inc"]
    df.drop(columns="inc", inplace=True)

    X_train, X_test, y_train, y_test = train_test_split(df, clmet, test_size=0.2, random_state=42)

    lr = LogisticRegression()
    model = lr.fit(X_train, np.array(y_train))
    pred = model.predict(X_test)
    og = np.array(y_test)
    og1 = np.array(y_train)
    trpred= model.predict(X_train)
```

Source: The model's Python-based realization in my program

Ridge classifiaction

Similarly to Ridge regression, Ridge classification combats overfitting by introducing a penalty function based on the sum of the parameters' squared values. As with Ridge regression, I iterated through the possible values of "alpha" (0.01, 0.1 and 1) to find the model with the best accuracy.

Beyond "alpha", "max_iter", "solver" and "tol" were also important hyperparameters.

- max_iter: The maximum possible number of iterations executed by the solver before stopping. In my case, this amounted to 1000.
- solver: In my case, “auto”, meaning it chooses the optimizer solver (Fe. Cholesky) automatically.
- tol: Convergence tolerance, 1e-3 in my case. If the difference in weights between two iterations is lower than this value, the solver stops early.

I processed all 32 datasets with this model. In all cases, I applied an 80-20 train test split.

```
def rdigecl(df, inc):
    if 'Unnamed: 0' in df.columns:
            df.index=df['Unnamed: 0']
            df.drop(columns= "Unnamed: 0", inplace=True)

    for x in df.columns:
        if ".1" in str(x):
            df.drop(columns=x, inplace=True)

    df["inc"]=inc
    df.dropna(inplace=True)
    inc=df["inc"]
    df.drop(columns="inc", inplace=True)

    X = df
    Y = inc

    x_train, x_val, y_train, y_val = train_test_split(X, Y, test_size=0.2, random_state=42)
    aa=[0.01, 0.1, 1]
    legjacc=0
    for x in aa:

        alpha = x
        max_iter = 1000
        solver = 'auto'
        tol = 1e-3
        rc = RidgeClassifier(
        alpha=alpha, max_iter=max_iter, solver=solver, tol=tol)
        rc.fit(x_train, y_train)
        pred=rc.predict(x_val)
        trpred=rc.predict(x_train)

        accuracy = accuracy_score(y_val, pred)
        if legjacc==0 or legjacc<accuracy:
            legjacc=accuracy
            legjpred=trpred
```

Source: The model’s Python-based realization in my program

Logistic regression with RFE

As with the models with a numeric target variable, RFE came in handy with the classification models. I combined it with logistic regression and used it to process all datasets with no prior

dimension reduction techniques applied to them. I applied an 80-20 train-test split to the analyzed data.

```
def rfelogreg(df, inc):
    if 'Unnamed: 0' in df.columns:
            df.index=df['Unnamed: 0']
            df.drop(columns= "Unnamed: 0", inplace=True)
    for x in df.columns:
        if ".1" in str(x):
            df.drop(columns=x, inplace=True)
    df["inc"]=inc
    df.dropna(inplace=True)
    inc=df["inc"]
    df.drop(columns="inc", inplace=True)
    x = df
    y = inc
    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
    lr = LogisticRegression()
    rfe = RFE(estimator=lr, n_features_to_select=int(len(df)/10))
    rfe.fit(X_train, y_train)
    sel= x.columns[rfe.support_]
    X_trainsel = X_train[sel]
    X_testsel= X_test[sel]
    lr.fit(X_trainsel, y_train)
    pred = lr.predict(X_testsel)
    trpred=lr.predict(X_trainsel)
    accuracy = accuracy_score(y_test, pred)
```

Source: The model's Python-based realization in my program

SVM

SVMs (Support Vector Machines) seek the widest route between a dataset's two classes. The decision boundary, beset by margins on both sides, lies in the middle of this space (Kuknyó, Üzleti Elemzések Módszertana 6. Előadás: Tartó vektor gépek, 2024, old.: 4).

We differentiate between models with linear, polynomial (models that transform data into a higher-dimensional space, in order for the determination of a linear decision boundary becoming possible (Kuknyó, Üzleti Elemzések Módszertana 6. Előadás: Tartó vektor gépek, 2024, old.: 18)) and RBF (models that determine the decision boundary based on feature similarity (Kuknyó, Üzleti Elemzések Módszertana 6. Előadás: Tartó vektor gépek, 2024, old.: 21)) kernels.

The two most important hyperparameters of SVMs are "gamma" and "C". "gamma" determines the reach of a single training example - the two are inversely proportional (Kuknyó, Üzleti Elemzések Módszertana 6. Előadás: Tartó vektor gépek, 2024, old.: 21). Its correct determination is important for avoiding over-and underfitting (avicksaha, 2024), especially when it comes to RBF-based models (avicksaha, 2024).

The C hyperparameter determines the margin's size: we differentiate between soft margin and hard margin classification, with the former tolerating data points on the margin. A tradeoff of a higher value of C is worse generalization ability (Kuknyó, Üzleti Elemzések Módszertana 6. Előadás: Tartó vektor gépek, 2024, old.: 9).

I aimed to find an optimal model by utilizing the "grid search" method: the "gamma" hyperparameter's possible values were 0.1, 1, 10, 100 and 1000, while C's possible values ranged from 1 to 10. I determined the best model based on the accuracy achieved on test data.

I processed all dimensionally reduced datasets with all three kernels. As with the previous classification models, I applied an 80-20 train-test split.

```
def gridsearchsvm(df, inc, kern):
    hulk=[0.1, 1, 10, 100, 1000]
    legj=0
    for g in hulk:
        for c in range(1, 10):
            if 'Unnamed: 0' in df.columns:
                df.index=df['Unnamed: 0']
                df.drop(columns= "Unnamed: 0", inplace=True)
            for x in df.columns:
                if ".1" in str(x):
                    df.drop(columns=x, inplace=True)

            df["inc"]=inc
            df.dropna(inplace=True)
            inc=df["inc"]
            df.drop(columns="inc", inplace=True)
            x = df
            y = inc

            X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
            svm = SVC(kernel=kern, C=c, gamma=g, random_state=0)
            svm.fit(X_train, y_train)
            sctrain=svm.score(X_train,y_train)
            sctest=svm.score(X_test,y_test)
            if sctrain>legj:
                legj=sctest
                legj0=sctrain
```

Source: The model's Python-based realization in my program

Decision tree-based classification

In decision tree-based classification, the samples in the dataset are classified based on the values of their variables by answering the questions at the nodes (Kuknyó, Üzleti Elemzések Módszertana 4. Gyakorlat: Döntési fák, 2024, old.: 6).

We differentiate between three types of node: the root (a node which only has an output), the inner node (one with both an input and an output) and the leaf, which only has an input (Kuknyó, Üzleti Elemzések Módszertana 4. Gyakorlat: Döntési fák, 2024, old.: 7).

We can also classify decision tree-based models by whether the classification at the nodes is based on the Gini-index or entropy: the former estimates a node's impurity, (Thakar & Tahsildar, 2022), the latter its uncertainty.

Entropy is calculated based on the classes' proportions relative to one another: the more equal the rates of classes, the higher it is. If a class has more samples relative to the other classes, entropy is lower. Gini-index, which measures the probability of a variable's misclassification (Thakar & Tahsildar, 2022), correlates with entropy. These make determining the ideal split at a node possible: depending on which method one chooses, the decision tree strives to find the split resulting in either the purest or most certain classification (Thakar & Tahsildar, 2022).

I used both Gini-index- and entropy-based decision trees and applied an 80-20 train-test split on the datasets. Every node had to get an input of at least 5 samples in order to be eligible to lead to an output and the tree's depth was maximized in "5". These two hyperparameters weren't determined using the "grid search" method, but by regular experimentation (I also ran the model with values of 1, 3 and 10).

I processed all 32 datasets with these models.

```
def ginitree(df, inc):
    if 'Unnamed: 0' in df.columns:
        df.index=df['Unnamed: 0']
        df.drop(columns= "Unnamed: 0", inplace=True)
    for x in df.columns:
        if ".1" in str(x):
            df.drop(columns=x, inplace=True)
    df["inc"]=inc
    df.dropna(inplace=True)
    inc=df["inc"]
    df.drop(columns="inc", inplace=True)
    x = df
    y = inc

    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)


    treegini= DecisionTreeClassifier(criterion="gini", random_state=42, max_depth=5, min_samples_leaf=5)


    treegini.fit(X_train, y_train)
    pred=treegini.predict(X_test)
    trpred=treegini.predict(X_train)
```

Source: The Gini index-based decision tree model's Python-based realization in my program

```python
def onodrim(df, inc):
    #entropy tree
    if 'Unnamed: 0' in df.columns:
        df.index=df['Unnamed: 0']
        df.drop(columns= "Unnamed: 0", inplace=True)
    for x in df.columns:
        if ".1" in str(x):
            df.drop(columns=x, inplace=True)
    df["inc"]=inc
    df.dropna(inplace=True)
    inc=df["inc"]
    df.drop(columns="inc", inplace=True)
    x = df
    y = inc

    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)


    ent= DecisionTreeClassifier(criterion="entropy", random_state=42, max_depth=5, min_samples_leaf=5)


    ent.fit(X_train, y_train)
    pred=ent.predict(X_test)
    trpred=ent.predict(X_train)
```

Source: The entropy-based decision tree model's Python-based realization in my program

GBDT

GBDT (Gradient-Boosted Decision Tree) is an ensemble machine learning method, which works similarly to the model described in Chapter 4.1.7. The difference between the two is that this one optimizes a classification model, rather than a regression model, with the help of gradient boosting.

The model I utilized used 200 estimators, had a learning rate of 0.1 and its resulting tree had a maximum depth of 3. I applied an 80-20 train-test split on the data and processed all 32 datasets.

```python
from sklearn.ensemble import GradientBoostingClassifier

def gbc(df, inc):
    if 'Unnamed: 0' in df.columns:
        df.index=df['Unnamed: 0']
        df.drop(columns= "Unnamed: 0", inplace=True)

    for x in df.columns:
        if ".1" in str(x):
            df.drop(columns=x, inplace=True)

    df["inc"]=inc
    df.dropna(inplace=True)
    inc=df["inc"]
    df.drop(columns="inc", inplace=True)
    x = df
    y = inc

    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

    gbclass=GradientBoostingClassifier(n_estimators=200, learning_rate=0.01,
    max_depth=3, random_state=0)
    gbclass.fit(X_train, y_train)
```

Source: the model's Python-based realization in my program

Gaussian Naive Bayes

The Gaussian Naive Bayes model is a probabilistic classification model based on the core assumptions (Martins, 2023) that every independent variable can predict the target variable independently of the other, and that every class follows a Gaussian distribution.

I processed all 32 datasets with this model. An 80-20 train-test split was applied to all datasets.

```python
from sklearn.naive_bayes import GaussianNB

def nb(df, inc):
    if 'Unnamed: 0' in df.columns:
        df.index=df['Unnamed: 0']
        df.drop(columns= "Unnamed: 0", inplace=True)
    for x in df.columns:
        if ".1" in str(x):
            df.drop(columns=x, inplace=True)
    df["inc"]=inc
    df.dropna(inplace=True)
    inc=df["inc"]
    df.drop(columns="inc", inplace=True)
    x = df
    y = inc

    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

    gauss = GaussianNB()
    gauss.fit(X_train, y_train)
    pred = gauss.predict(X_test)
    trpred=gauss.predict(X_train)
    print(accuracy_score(y_test, pred)*100)
```

Source: the model's Python-based realization in my program

Sub-monthly forecasts

The previous models set out to predict incidents on a monthly basis. In this section, I will lay out the methods used to make sub-monthly predictions.

- Dividing months into intervals: I divided each month into intervals, the first one ranging from the 1st to the 10th, the second one from the 11th to the 20th, and the third one containing the rest of its days. By counting the attacks based on these group I will attempt to estimate which interval is an incident the most likely to happen in.
- Weekly predictions: I will visualize the incidents grouped by the day of the week.
- Daily predictions: kimutatást készítek arról, az adott napon belül reggel, délelőtt, délután és este milyen valószínűséggel történik támadás.
- Based on weather data: I will utilize the weather data gathered with the OpenMeteo API to find a relationship between weather and the likelihood of incidents.
- Analysis based on the types of ships: I will group incidents based on ship types.
- Analysis based on coordinates: I will subject the incidents' coordinates to clustering in the hopes of helping organize more efficient patrolling routes.

Model evaluation and results

General evaluation criteria

When evaluating the models described in the previous chapter, there are criteria that apply independently of whether the subject is a classification or a regression model. My primary goal is selecting a model, which demonstrates adequate generalization ability and good accuracy (or, in the case of models with a numeric variable, a sufficiently low MSE). As such, it's neither under-nor overfitted.

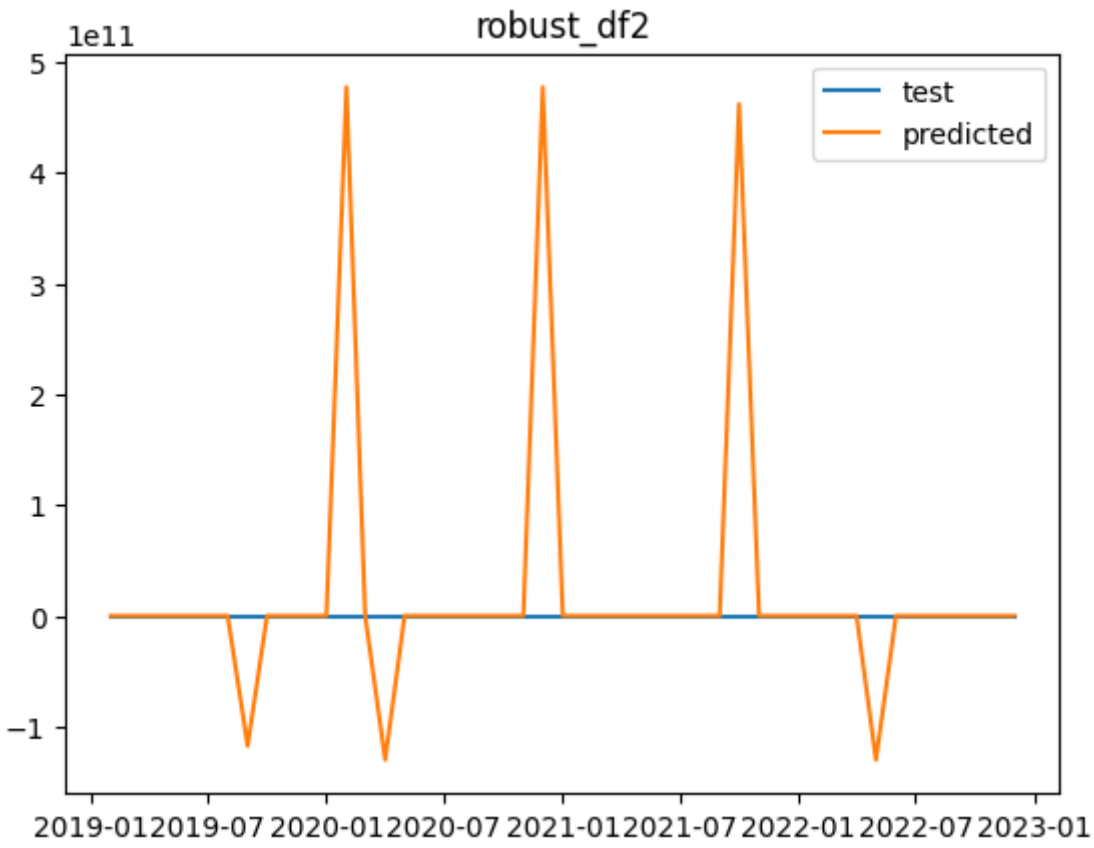
Another criterion is for the model to rely on a relatively small number of original variables (not the ones produced by dimensionality reduction techniques). The rationale behind this is to make the model easily maintainable and to avoid the worst-case scenario of having to communicate with the statistical departments of four different countries in order to receive the data needed to tune the model. Of course, this criterion only applies if the first criterion is fulfilled.

Evaluating models with a numeric target variable

In light of the general criteria, I strived to find two main models: one that ran on dimensionally reduced datasets and achieved better accuracy and one that ran on datasets whose size was decreased with RFE. Good generalization ability was a criterion in both cases. I was open to the best model and the model combined with RFE being one and the same (it would have been ideal), but when evaluating the models, it became clear that this was not the case.

Evaluation was a two-step process: first I evaluated the models based on their MSE (I had to take into consideration the fact that the models were scaled differently, so MSE could only be used to measure the accuracy of models running on data scaled similarly), then I visualized the results of the promising models and chose one that met the aforementioned requirements.

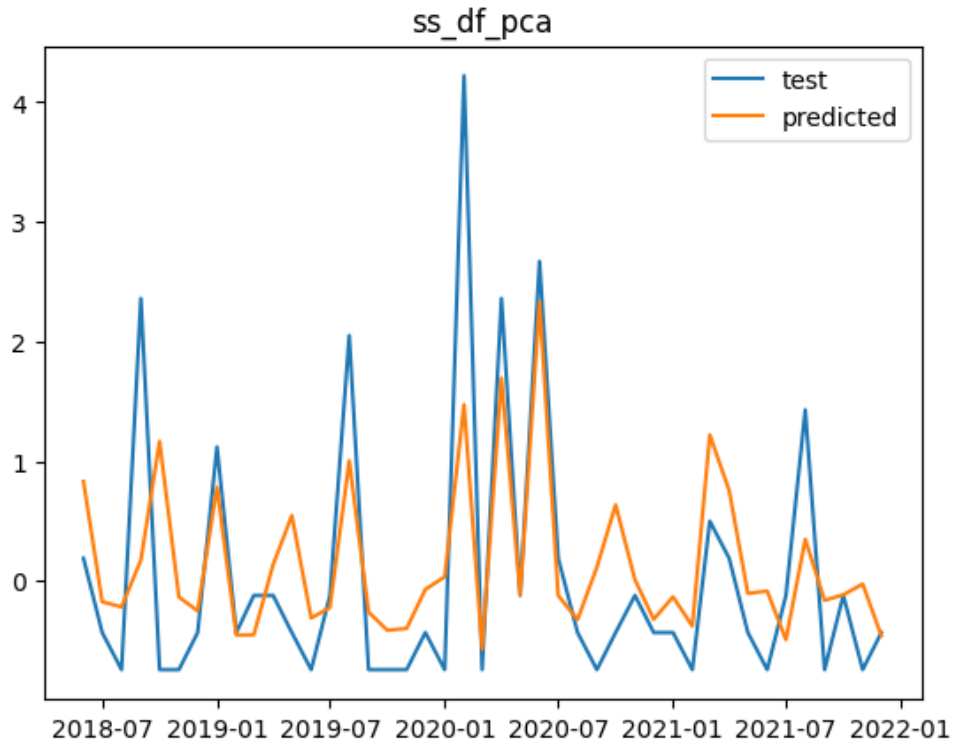
I had to pay particular attention to scaling: models with promising MSEs often turned out to be subpar, because the scaling method did away with the target variable's outliers and as such, the achieved MSE was only seemingly adequate.



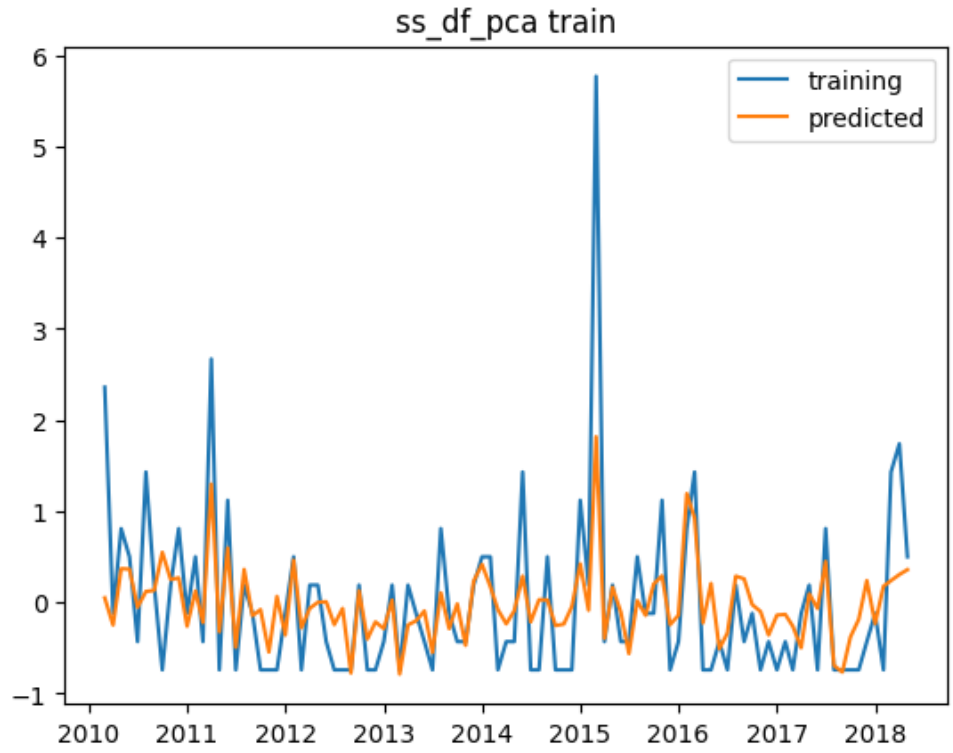
Source: visualization based on the results of the models running on the datasets (see: Chapter 5.2.6.1)

The above visualization shows the results of the linear regression model running on a robustly scaled dataset by comparing the target variable's predicted and actual values. The problem wasn't unique to robustly scaled datasets, examples of the issue could be found regardless of scaling method.

In light of the information laid out above, an adequately performing model untouched by the issues caused by scaling was the early stopping-enhanced LASSO regression model with a polynomial degree of 2 and an "alpha" of 1, which achieved an MSE of 0.607 on the test portion of a dataset containing the data of all four countries, scaled with min-max scaling, dimensionally reduced with the PCA method, keeping both lagged and non-lagged variables in case of collision and divided into training and test data at a 70-30 rate.



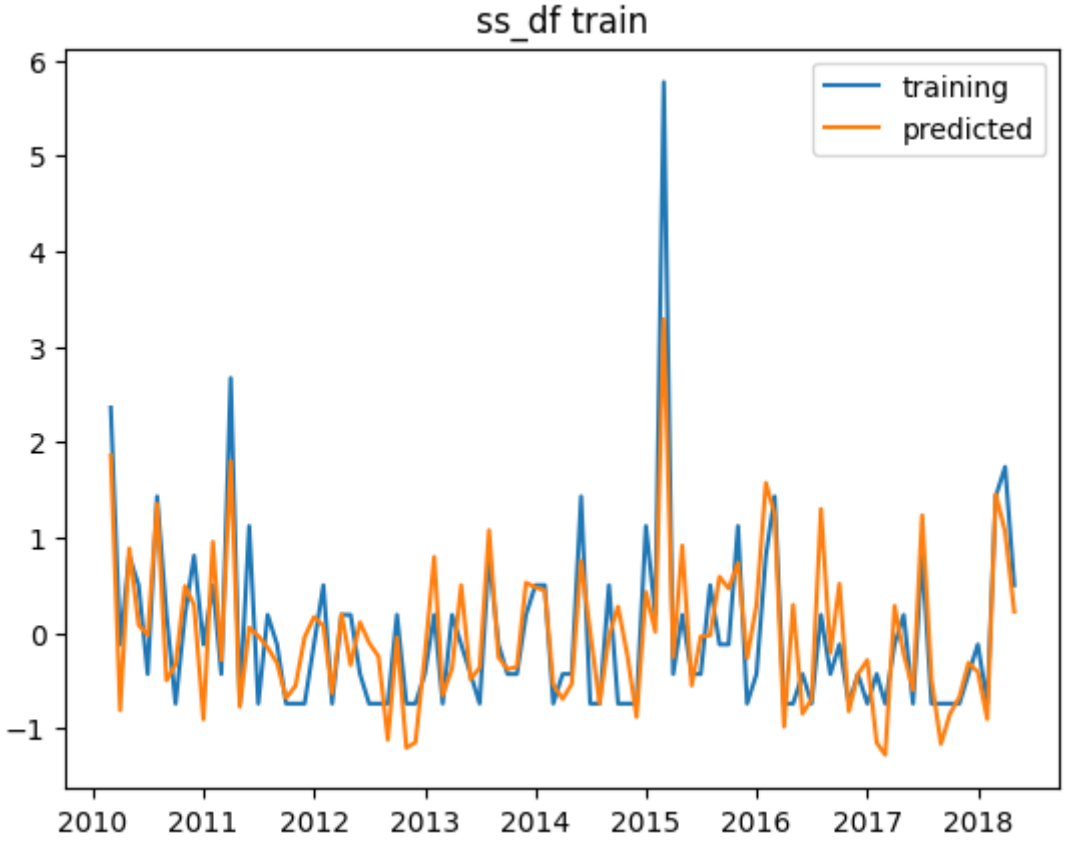
Source: visualization based on the results of the models running on the datasets (see: Chapter 5.2.6.1)



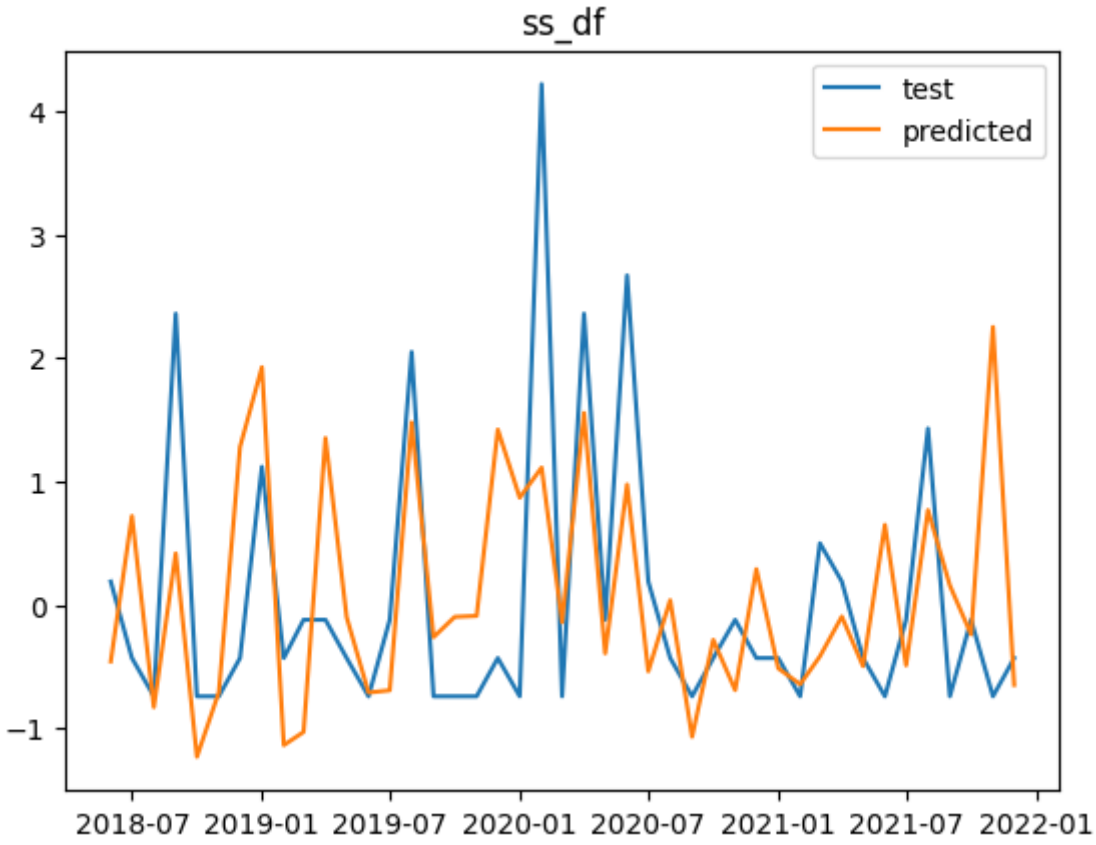
Source: visualization based on the results of the models running on the datasets (see: Chapter 5.2.6.1)

The figure above visualizes the results of the model on the training and test data. While it exhibits a slight tendency to over-or underestimate the number of attacks by a little, it consistently shows the direction of growth well and with few errors, which is extremely important if the model is used to organize patrolling routes in the SOMS.

The model running on fewer independent variables was a regression model combined with RFE. It achieved an MSE of 1.146 on the test portion of a standard scaled dataset that only contained the data of Singapore, Malaysia and Indonesia, was divided into training and test portions at a 70-30 rate and in case of collision, kept both the lagged and non-lagged variable. The figure below shows the model's results on both the training and test data.



Source: visualization based on the results of the models running on the datasets (see: Chapter 5.2.6.1)



Source: visualization based on the results of the models running on the datasets (see: Chapter 5.2.6.1)

While the model is somewhat less accurate than the one described before, it makes up for the loss in accuracy with its generalization ability (the visualized results were achieved with a 70-30 train-test split) and the fact that it relies on fewer independent variables and as such, is more easily maintained.

The variables the RFE kept were the following:

- Number of criminal offences in the Indonesian police jurisdictions of Juma, Nusa Tenggara Barat, Kalimantan Tengah and Kaminatan Selatan.
- Malaysia: lagged leading diffusion (See: Chapter 3.1.3.).
- Singaporean port traffic statistics (Total Cargo (1000 Tonnes)).
- The GDRP produced by agriculture, forestry and fishing in Indonesia's Bali province.
- Malaysian inflation, GDP deflator (yearly %)
- The CPI of health care and the GDP of wholesale trade in Singapore.
- Lagged unemployment in the Indonesian provinces of Bengkulu, Kaminatan Selatan, Di Yogyakarta and Sulawesi Utara.

Evaluating classification models

I evaluated the classification models based on the criteria laid out in Chapter 4.4.1. The metric by which I chose the best-performing model was accuracy, a measure of what percentage of the samples were classified correctly. Having filtered for models achieving adequate accuracy, I evaluated the selected models based on their F1 scores, the harmonic mean of accuracy and recall (developers.google.com, 2021), thereby eliminating the possibility of a high accuracy score being caused by an imbalance in class sizes.

I considered both the accuracy achieved on test data and the one achieved on training data in order to filter out cases where an adequate accuracy on test data was the result of overfitting (signified by a significantly higher accuracy score on training data, in some cases, 100%) and cases where test accuracy was higher than training accuracy – a sign of unreliability.

The best-performing model (though one marred by the threat of unmaintainability) was a Gaussian Naïve Bayes model, which achieved a test accuracy of 86,2% and a training accuracy of 87,61% on a dataset normalized with the min-max scaling method, dimensionally reduced with PCA, keeping only the lagged variables in case of a collision, containing the data of all four countries and divided into training and test data at an 80-20 rate. It achieved an F1 score of 0.8 on the test data, which is adequate, as the two classes were relatively balanced in size (61% of samples belonged to the bigger class).

The second classification model I chose was one that combined logistic regression with RFE and achieved an accuracy of 80,64% on the test data and an accuracy of 87,8% on the training data. The dataset it processed with the results above was one that didn't contain Thai data, only kept lagged data in case of collision, was standardized, not dimensionally reduced, and divided into training and test data at an 80-20 rate.

The independent variables kept by the model were the following:

- The GRDP produced by agriculture, forestry and fishing in the Indonesian provinces of Kepulauan Bangka Belitung, Jawa Tengah, Di Yogyakarta, Nusa Tenggara Barat, Kalimantan Timur, Sulawesi Selatan, Maluku and Maluku Utara.
- The number of criminal offences in the Indonesian police jurisdictions of Lampung, Metro Jaya, Nusa Tenggara Barat, Sulawesi Utara and Maluku Utara.
- The countrywide number of criminal offences in Indonesia.

On average, the best-performing classification models were the SVM-based ones: their average test accuracy was 74,38%. They were followed by models with RFE, the average test accuracy of which was 71,43%. However, it must be noted that the SVMs' average training accuracy was over 96,99% and as such, their generalization ability was much worse. With an average test accuracy of 71,12%, decision tree-based models came in third.

I also ran the models using classes that were not divided along the mean of monthly attacks, but – in order to handle the possible size difference of the classes – by the midpoint of samples ordered increasingly by the number of monthly attacks. The best-performing model, a linear SVM model achieved an F1-score of 78,79% on test and 80% on training data and an accuracy of 77,4% on test data and 81,15% on training data. It achieved these results on a dataset that had all four countries' data, kept only the lagged variables in case of a collision, was dimensionally reduced with PCA and normalized with the min-max scaling method.

Interpreting the results

This chapter will be dedicated to the interpretation of the models' and other statistical tests results, paying particular attention to the paper's goals.

The models seem to confirm the theories explaining the incidents' prevalence with economic reasons (mentioned in Chapter 2.1.2): the second classification model I chose achieved an accuracy of 80% on test data based solely on the GRDP owed to certain Indonesian province's agriculture, forestry and fishing sectors and the number of criminal offences in certain Indonesian police jurisdictions. Without exception, the aforementioned provinces and prefectures were coastal (citypopulation.de, dátum nélk.) and most of them (all but "Metro Jaya") were rural, which are factors contributing to the presence of fishing. In light of all this, the fishing industry's performance does well as an explanatory factor behind the number of incidents.

The second model with a numeric target variable I chose (the one combining linear regression with RFE) also kept a lot of Indonesian economic and social variables, among them the number of criminal offences and the provincial GRDP owed to fishing, agriculture and forestry (which were also amongst the previously mentioned model's chosen variables) and provincial unemployment rates.

The model also kept Malaysian economic indicators. However, it must be noted that in the 2010s, Indonesia's and Malaysia's key economic indicators like the inflation rate (imf.org-2,

2024) and the rate of GDP growth (imf.org-3, 2024) tended to correlate. As such, the Malaysian inflation rate, which the model kept, can be considered both an Indonesian and a Malaysian economic indicator. However, as the model favored this indicator over several Indonesian ones, I concur that Malaysian economic indicators can also predict and/or cause the prevalence of incidents, even if not to the extent Indonesian data does.

The above data supports the theories laid out in Chapter 2.1.2.

The tendency of Indonesian variables being kept by models with RFE can't simply be explained by their excessive presence in the datasets: the RFE-based models (both the best-performing ones and the others) tended to keep them at a rate that went beyond their prevalence.

The data above seems to support the idea that the Indonesian economic, political and social situation has a greater effect on the number of incidents than other countries': however, I had way fewer Malaysian variables (I had a lot of Indonesian provincial data). As such, I assigned greater significance to the Malaysian indicators that made it past the RFE. They also support the theory that the Malaysian economic situation has an effect on the occurrence of incidents.

Singapore's role also bears some discussion: the literature doesn't suggest that the country's economic, social and political indicators have an effect on the prevalence of pirate and ARAS incidents. The Singaporean variables that made it past the RFE filter seem to support this: only three variables were chosen, one of which pertains to port traffic. This – especially considering the rate of Singaporean variables in the dataset – isn't much.

The models suggest that Thailand has a relatively small effect on the prevalence of incidents: not a lot of Thai data made it past the RFE filters. This can partially be explained by the relatively small number of Thai variables.

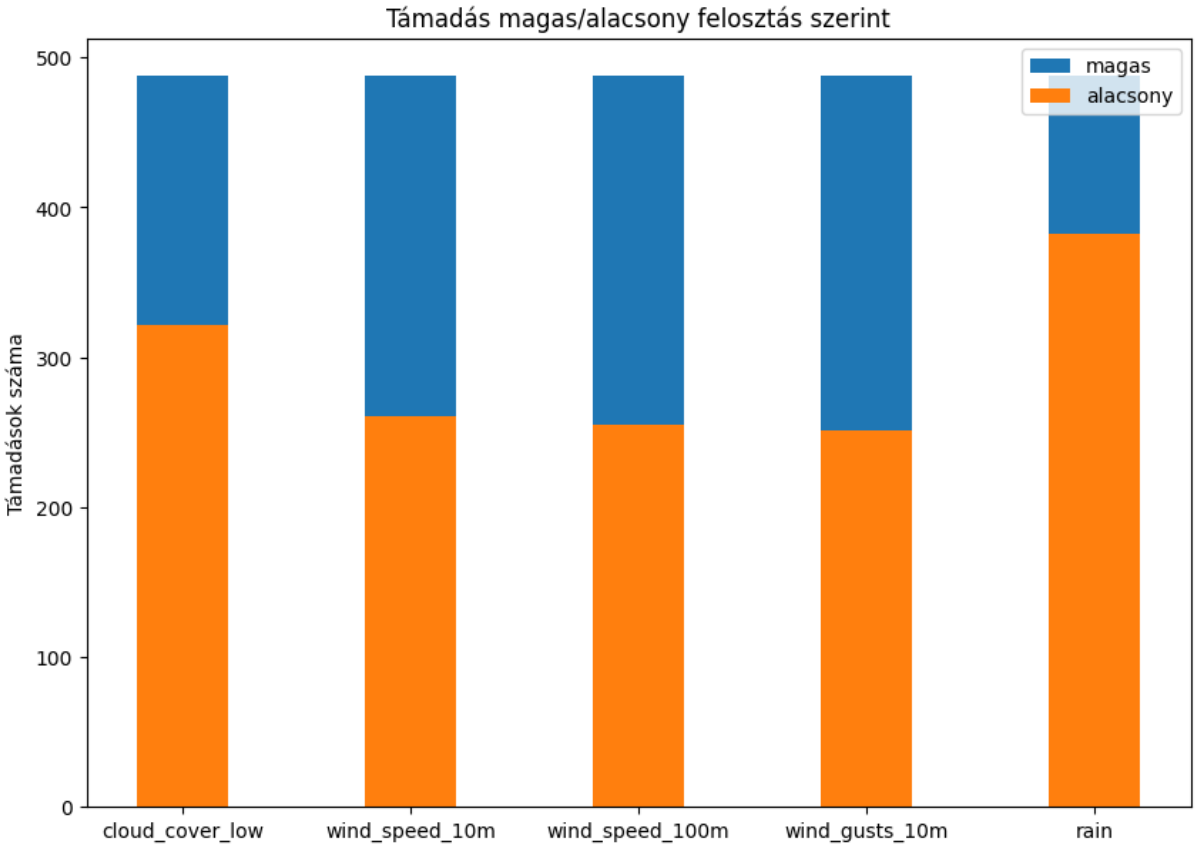
It must also be noted that some Thai variables were present in some RFE-based models after the feature elimination process concluded: a political stability indicator, the rate of GDP growth and the rate of children starting elementary school. Thus, it can be supposed that Thailand's political, social and economic situation has an effect on the rate of incidents in the SOMS, even if a relatively small one.

In conclusion, Indonesian and – to a lesser extent – Malaysian (and to an even lesser extent, Thai) variables seem to be capable of prediction pirate and ARAS incidents, which supports the longstanding theories on the causes of piracy in the SOMS.

Conclusions drawn from incident data

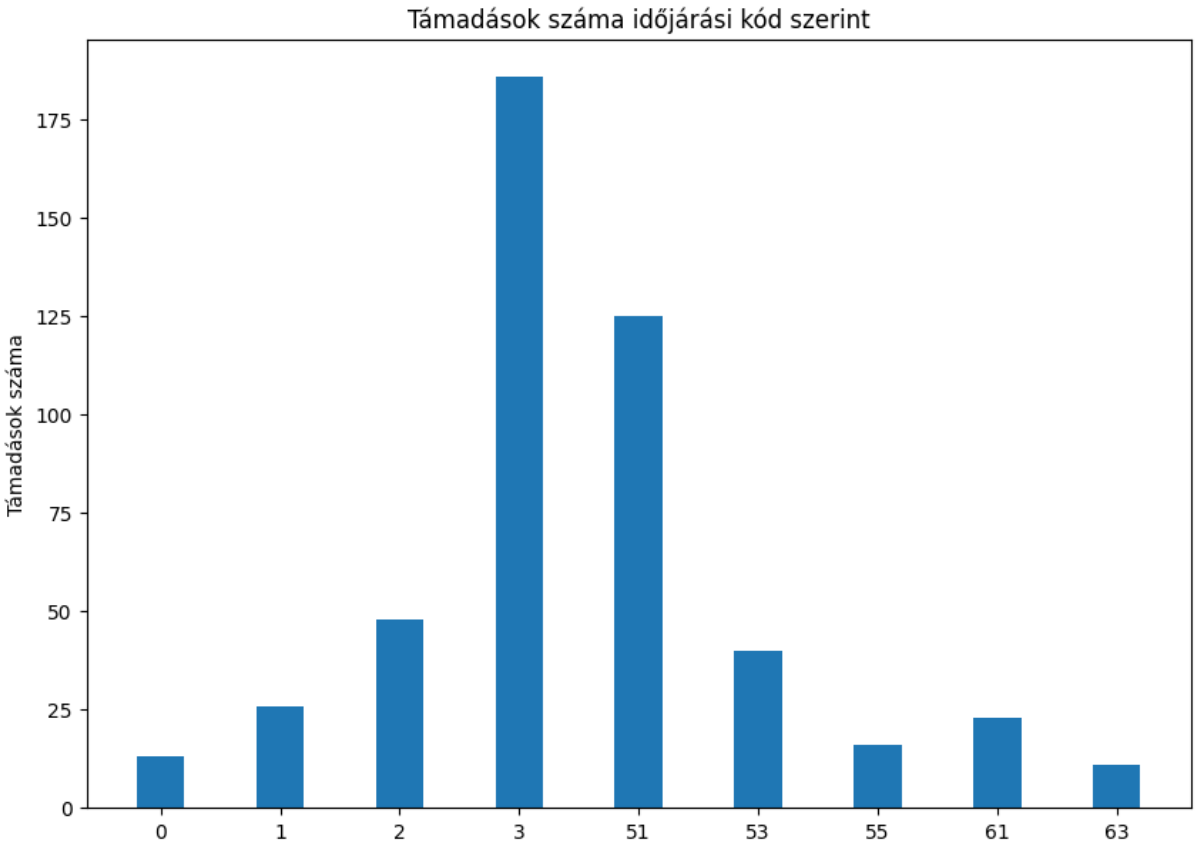
This chapter is dedicated to my analysis made with the goal of making sub-monthly predictions.

Prevalence of attacks in the context of weather data



Source: visualization (based on ReCAAP ISC' yearly data between 2010-2024 and the weather data gathered with the OpenMeteo API)

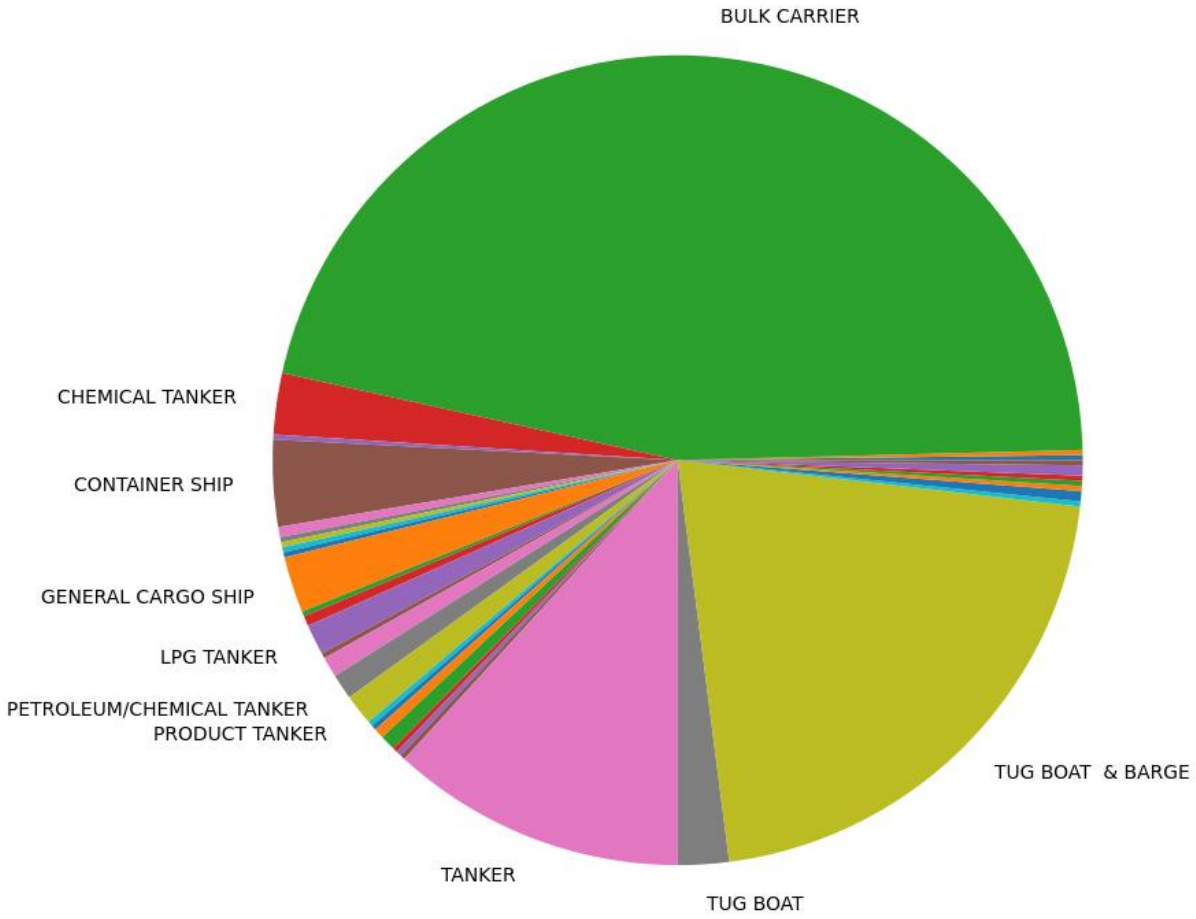
The above graph visualizes months with either a high (signified by the color blue) or low (signified by the color orange) number of attacks (the distinction was made based on the mean of the number of attacks) grouped by weather indicators. The "rain" indicator seems more important than the others: this is the only one where months with a high and months with a low number of attacks aren't in a near 50/50 split, although the relationship between the indicator and the number of attacks doesn't seem to be strong. However, it can't simply be explained by the frequency of rainy days: on average, every second day in Singapore is rainy (https://federicotartarini.github.io/, dátum nélk.). A possible explanation for the overrepresentation of non-rainy days is that attackers might not want to have to contend with the extra risk imposed by the rain.



Source: visualization (based on ReCAAP ISC' yearly data between 2010-2024 and the weather data gathered with the OpenMeteo API)

Grouping the number of attacks based on weather code seemingly produced more interesting results: codes "3" (Slight cloud formation) and "51" (Drizzle, not freezing, continuous, slight at time of observation) seem to be highly correlated with the number of attacks. However, this can be explained by their frequency: these two types of weather are usual in the SOMS, therefore no relationship can be determined between weather code and the frequency of incidents based on the above data.

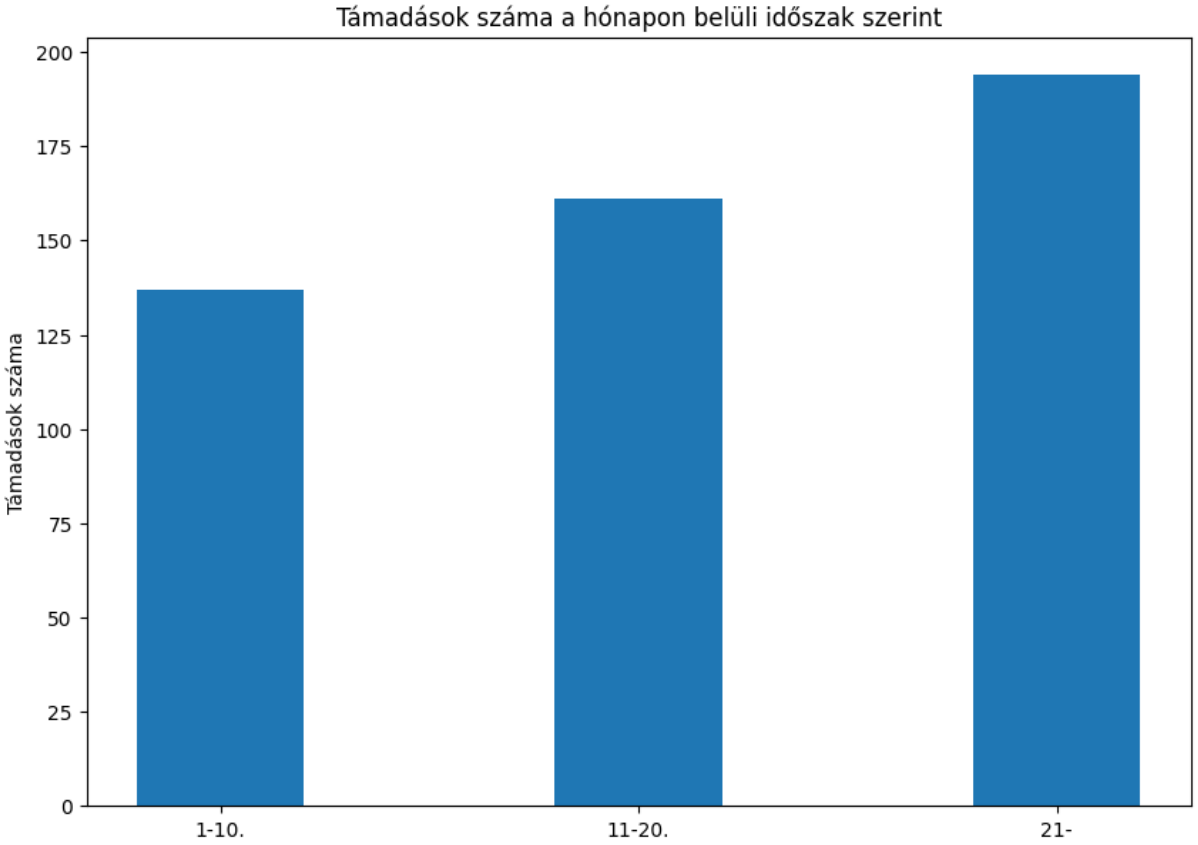
Prevalence of incidents in the context of ship type



Source: visualization (based on ReCAAP ISC' yearly reports between 2010 and 2024)

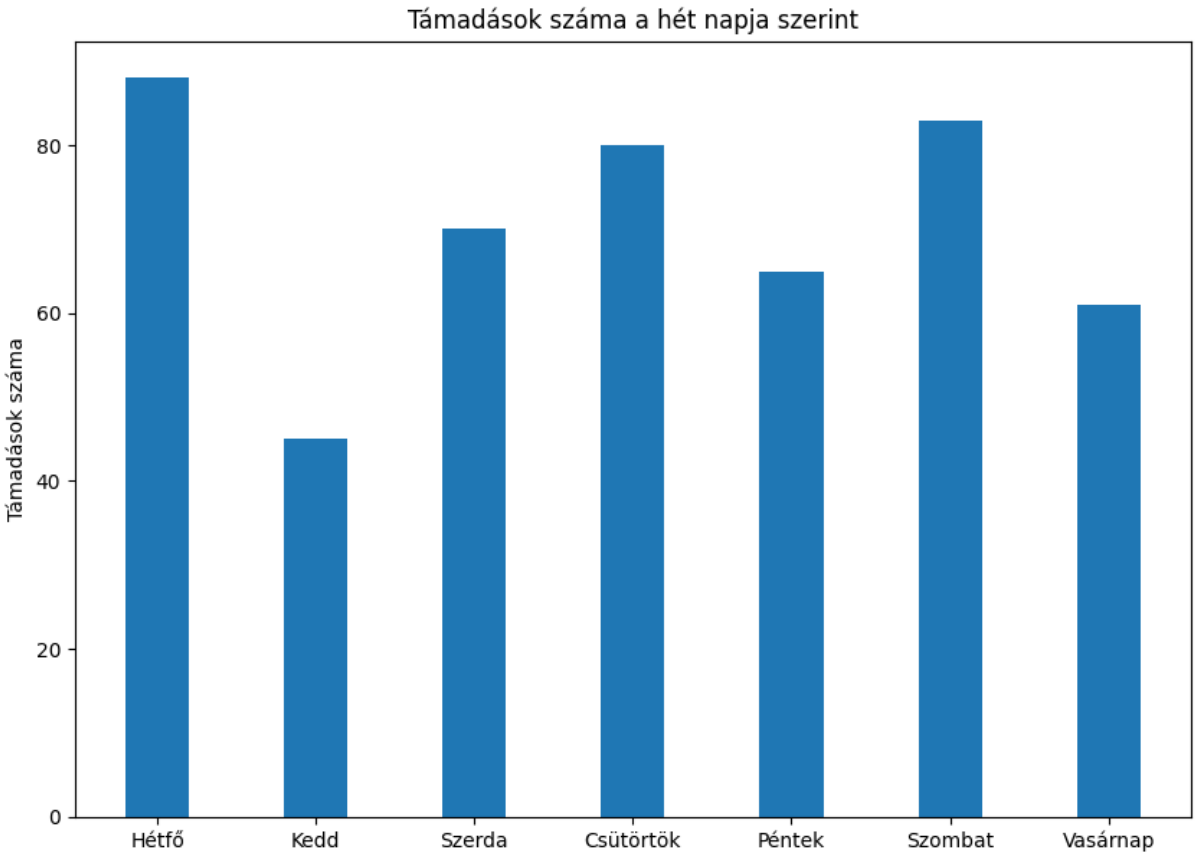
While it doesn't help predict when in a month an attack might occur, grouping attacks by the type of attacked ships still has its uses. The graph clearly shows that most attacked ships are either bulk carriers, tankers, oil tankers or tug boats and barges. An outsized portion of the attacked ships are bulk carriers.

Prevalence of attacks by sub-monthly period



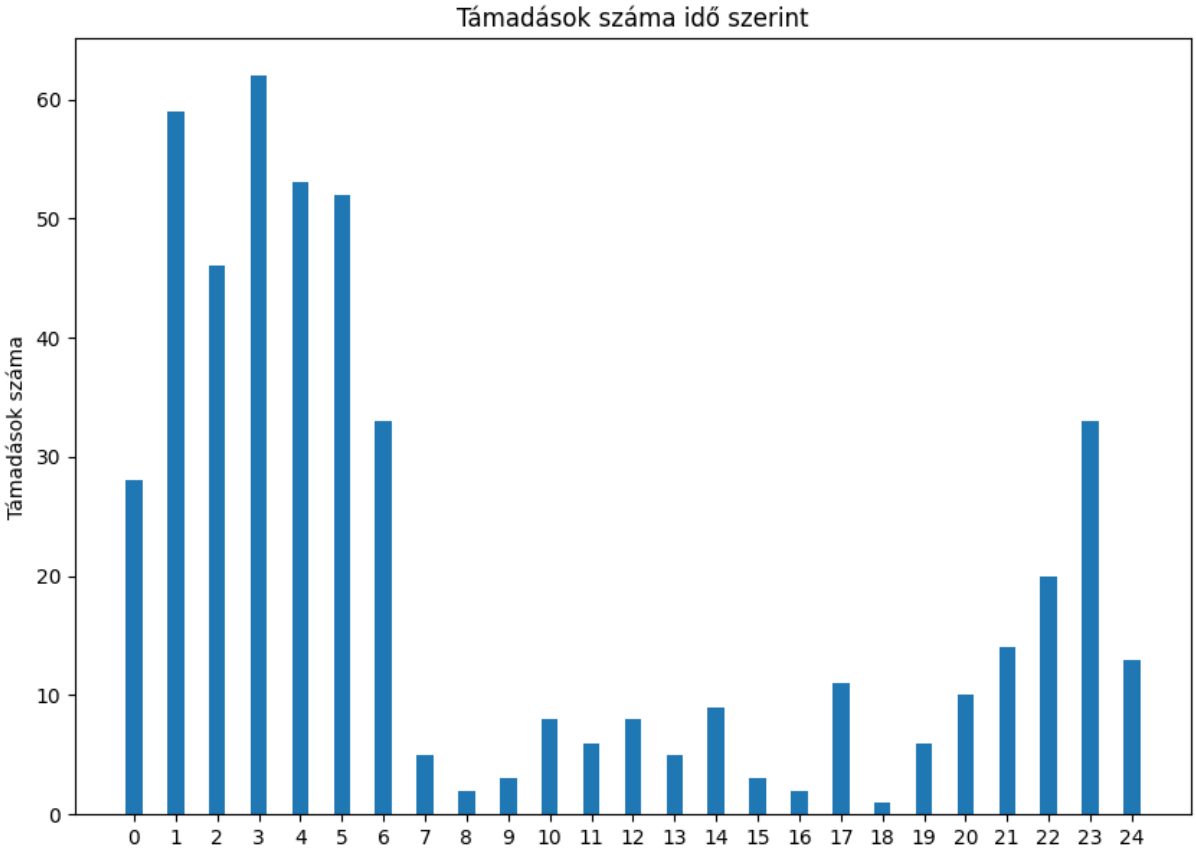
Source: visualization (based on ReCAAP ISC' yearly reports between 2010 and 2024)

A relationship can be observed between the number of attacks and the sub-monthly periods: while only 137 attacks occurred between the 1st and the 10th, 194 (over 1.416 of the first period) did in the third period ranging form the 21st until the end of the month). A possible explanation behind the trend is the Malaysian and Indonesian payday, which falls on the 7th in the former country and on the month's final working day in the latter. As attacks tend to be committed by people from a lower socioeconomic background, it's not a cockamamie assumption that as their previous payday gets further and further, someone might back a risky decision. However, the data I analyzed does not prove this relationship indisputably.



Source: visualization (based on ReCAAP ISC' yearly reports between 2010 and 2024)

The graph above visualizes the number of attacks grouped by the day of the week on which they occurred. No relationship could be ascertained between the two.



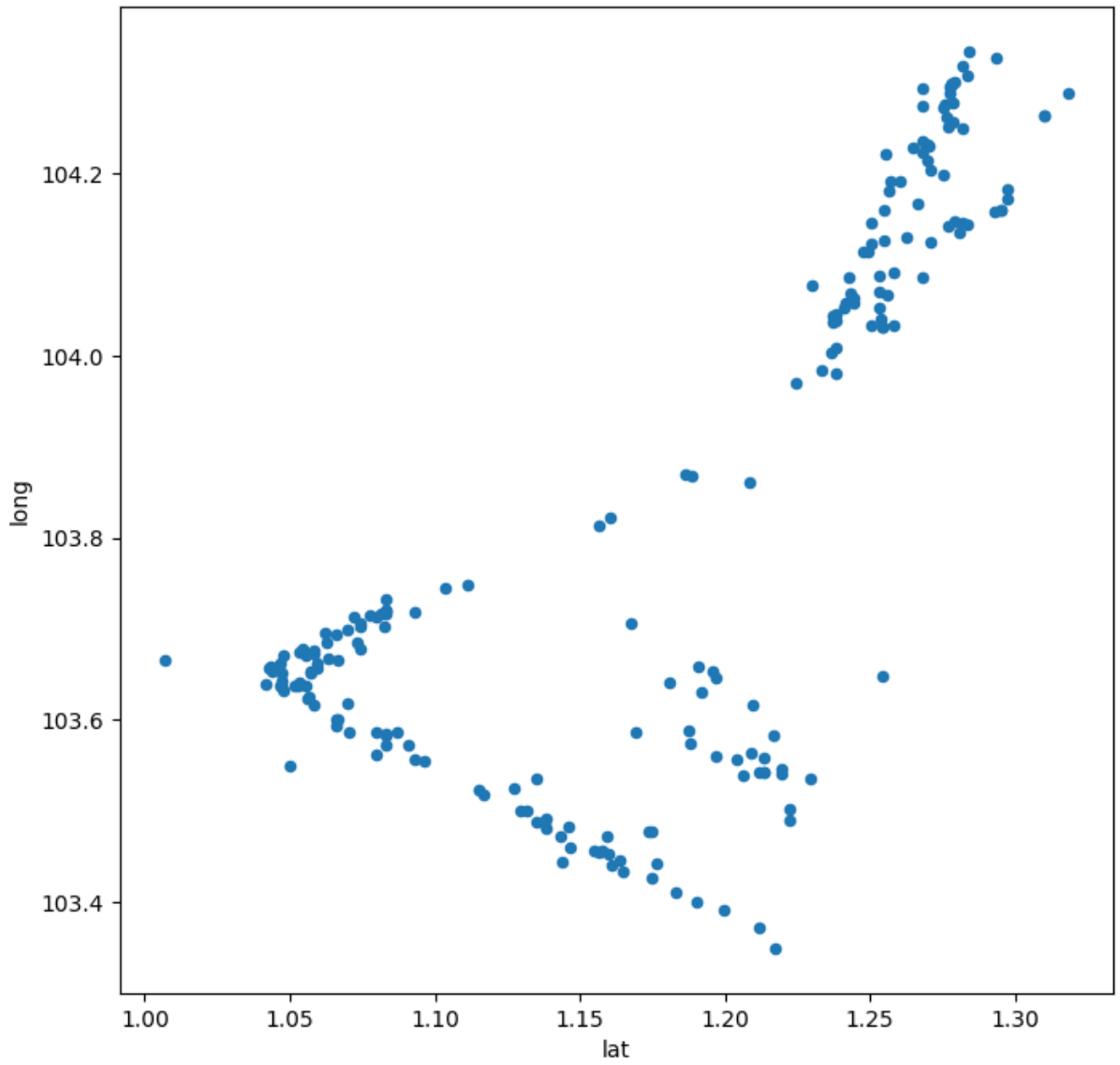
Source: visualization (based on ReCAAP ISC' yearly reports between 2010 and 2024)

The graph above shows the number of attacks grouped by the hour of the day they occurred on. A spike is noticeable between the hours of 9 PM and 6 AM, especially in the period starting at midnight. This can be explained by the darkness endemic to this period, which favors the attackers.

Incidents by coordinates

I clustered the incidents based on their coordinates. I shall explain the process in the following paragraphs.

I started off with outlier filtering, using z-score normalization, a process already mentioned in the paper. As a result of this process, I eliminated 7 of the 220 incidents, then did some manual filtering based on the scatterplot that visualized the results. Below is the resulting scatterplot.



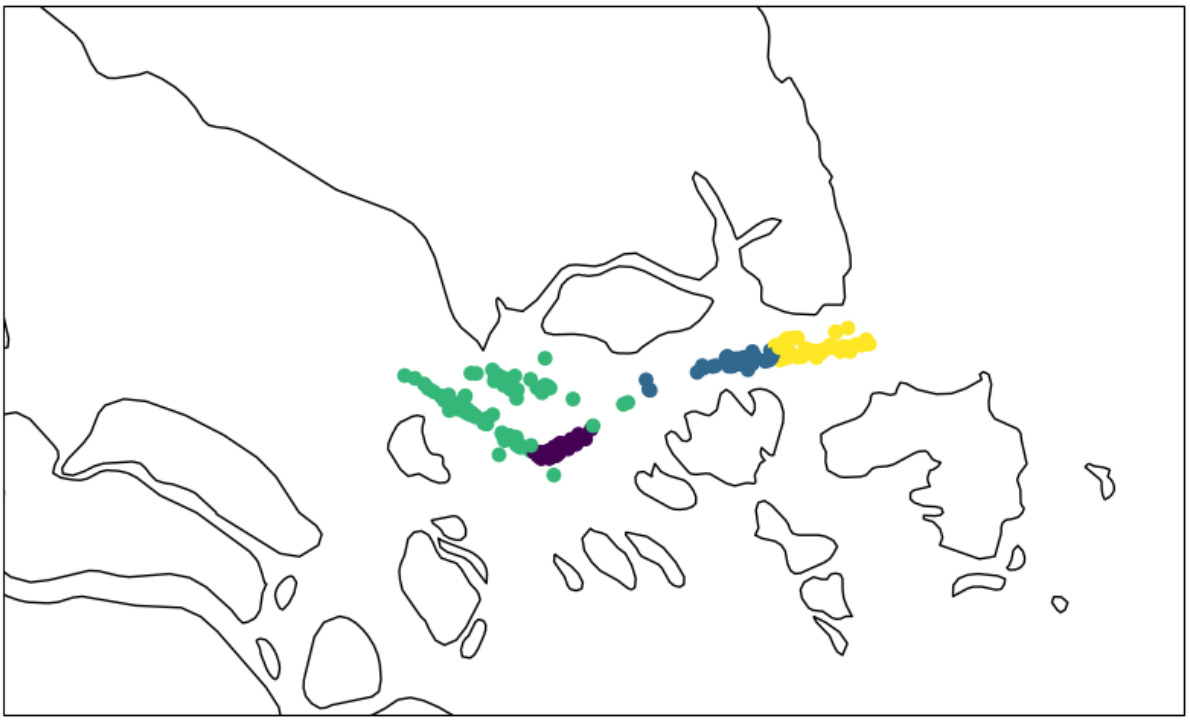
Source: visualization (based on ReCAAP ISC' yearly reports between 2010 and 2024)

Before applying a clustering algorithm, I had to find the elbow point, the number of clusters before which the variance explained the number of clusters decreases rapidly and after which it starts decreasing more slowly (ZalaRushirajsinh, 2023).

The below graph shows the inertia produced by a given number of clusters (ranging from 1 to 10). While the elbow point was 2, in light of the graph above, I found 2 clusters inadequate to describe the situation and be a basis to patrolling route planning, so I ran the clustering algorithm with 4 clusters.

Source: visualization (based on ReCAAP ISC' yearly reports between 2010 and 2024)

The graph below shows the results of the clustering algorithm. The clusters are signified by their colors. A map of the SOMS was inserted into the background using Python's "cartopy" library.



Source: visualization (based on ReCAAP ISC' yearly reports between 2010 and 2024)

Predictions for 2025

Predicting the independent variables

I extended the independent variables until the end of 2025 using the ARIMA method: I made a dataset containing the gathered data of all four countries stationary using the method described in Chapter 3.2.4. Afterwards I applied the ARIMA method with the help of the ARIMA modul from Python's statsmodels.tsa.arima.model library and the pmdarima library's auto_arima modul.

Predicting the dependent variable

As the dependent variable was discrete, the ARIMA method couldn't be applied, since it's only applicable in case of continuous variables (tibco.com, dátum nélk.). I used Poisson regression instead, which is better suited for predicting variables like my target variable in case of Poisson regression, the predicted values can't go below zero, since they refer to quantities. The model also assumes that – since it's based on the Poisson distribution curve (itl.nist.gov, dátum nélk.) – lower values are more frequent.

The aforementioned predictions weren't used to make forecasts, but for the sole purpose of extending the dependent and independent variables in order to have something by which to measure the models' accuracy.

Data preparation

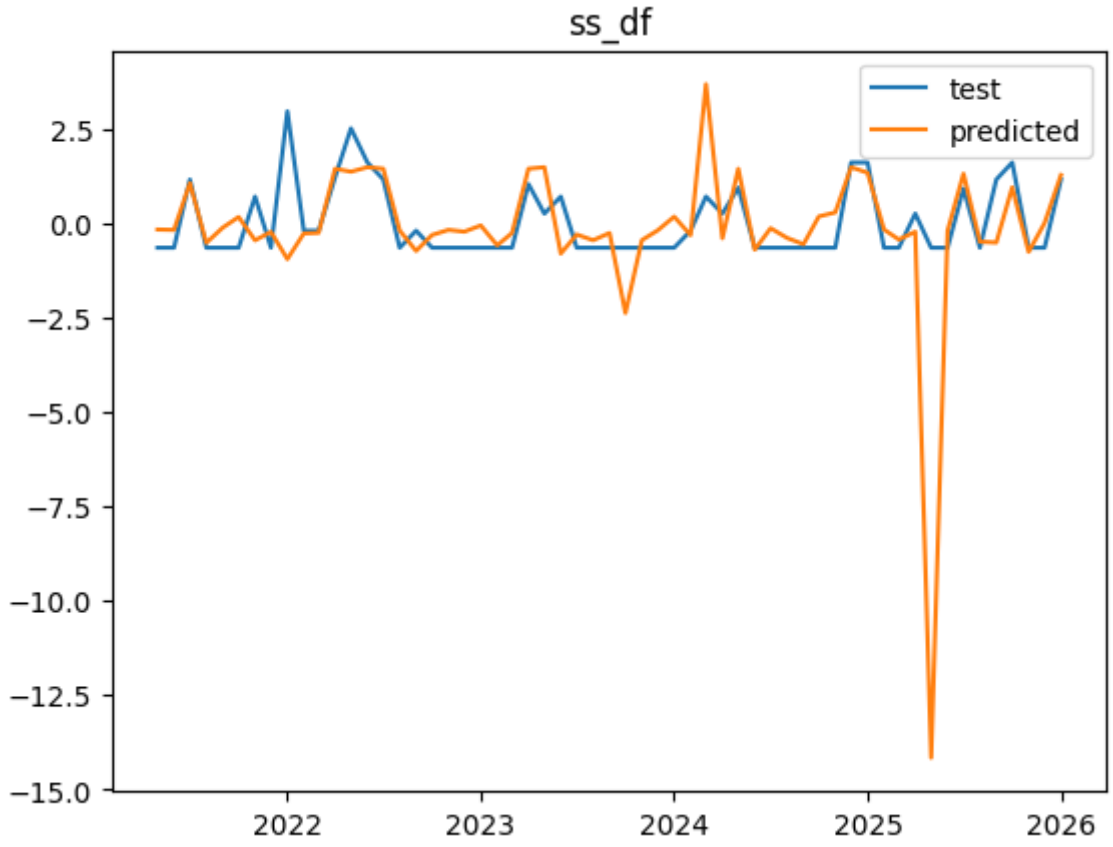
The newly created dataset underwent every preparatory step the original datasets underwent: two types of dimensionality reduction (in conjunction with the elbow method) and three types of normalization. Yet again, the datasets were divided based on how they handled collision caused by the presence of lagged and non-lagged variables: in some datasets, both were lagged and in some, the lagged ones were discarded. As was the case with the original datasets, half the datasets included Thailand-based data and the other half didn't.

Applying the selected models

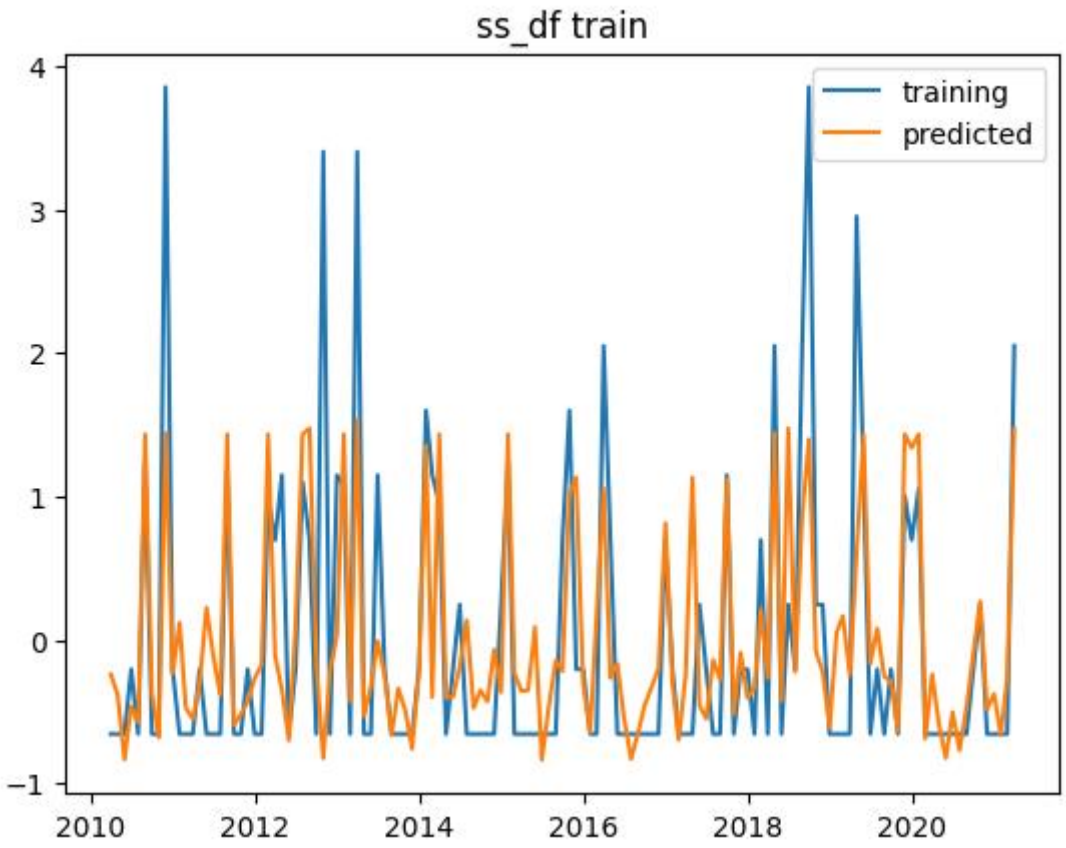
Models with a numeric target variable

The first of the two best-performing models was the one combining linear regression with RFE. As in the case of the original, I ran the model on a standardized dataset from which neither

lagged nor non-lagged variables were discarded, that did not contain data concerning Thailand and was split into training and test data at a 70-30 rate. It achieved adequate results on the extended dataset:



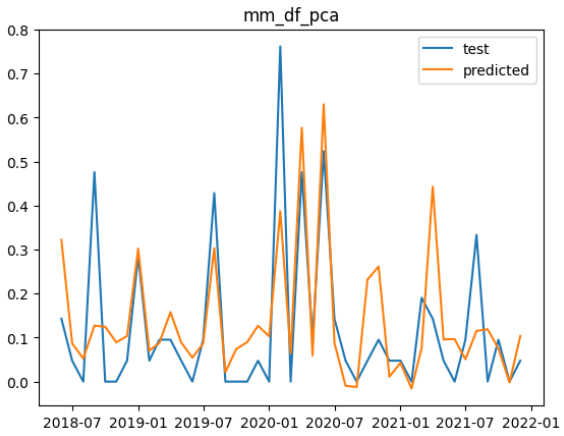
Forrás: visualization based on the models processing the datasets (See: Chapter 5.2.6.1)



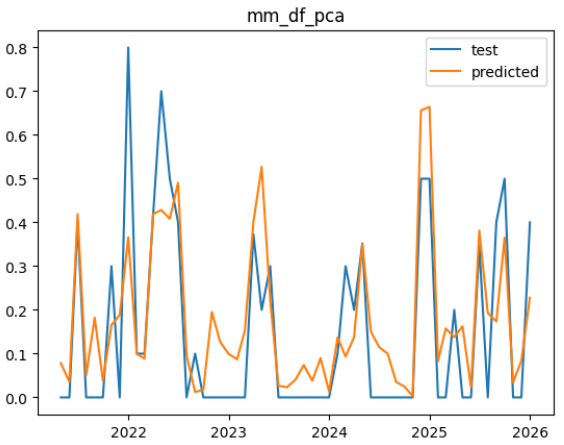
Source: visualization based on the models processing the datasets (See: Chapter 5.2.6.1)

As it can be ascertained from the figures above, the model's predicted target variable's values follow the real variable's values (or starting from October 2024, the ones predicted with Poisson regression) closely, but not closely enough to be considered overfitted. Its two major deviations from the actual values are owed to the fact that the model wasn't fed the information that the predicted values can't be negative. This issue can easily be handled by switching negative values with 0. Its MSE on the test data was 3.99, which, while seemingly high, would be much lower if not for the aforementioned, easily solvable issue.

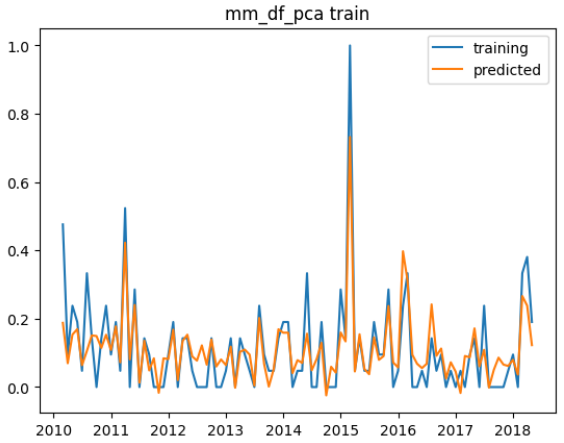
The second model chosen in the chapter dedicated to evaluating models was the early stopping-enhanced LASSO regression, which I used to process a dataset normalized with min-max scaling and dimensionally reduced with PCA, which kept both lagged and non-lagged variables in the event of a collision, contained data pertaining to Thailand and was divided into training and test data at a 70-30 rate. Its results are illustrated in the following figure. The model's "alpha" hyperparameter was 0.1 in case of both the original and the extended dataset, while the polynomial degree was 2 in case of the original dataset and 3 in case of the extended one.



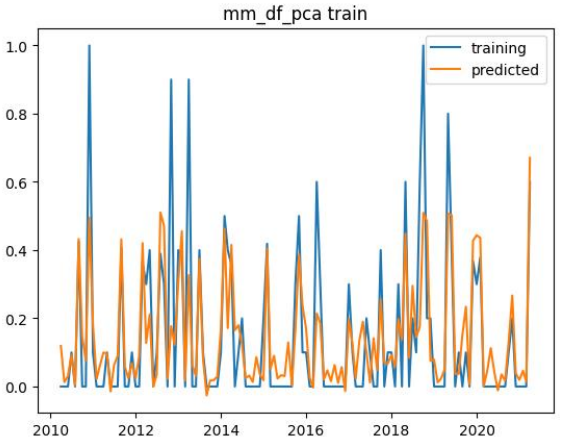
Original test data (MSE: 0.6399)



Test data extended until December 2025 (MSE: 0.2772)



Original training data



Training data extended until December 2025

Source: visualization based on the models processing the datasets (See: Chapter 5.2.6.1)

Classification models

The best-performing classification models selected based on their performance on the original datasets achieved adequate results on the extended datasets, too: the Gaussian Naïve Bayes model achieved an accuracy of 86,84% on the test data and an accuracy of 89,47% on the training data of the extended dataset that underwent PCA and min-max scaling, contained data pertaining to all four countries and was divided into training and test data at an 80-20 rate. It achieved an F1-score of 0,8 on the test data and an F1 score of 0,84 on the training data. As such, its performance can't solely be explained by the imbalance between class sizes.

The other selected classification model, the one combining logistic regression with RFE, achieved an accuracy of 92,11% on the test data and an accuracy of 92,76% on the training data. Its corresponding F1 scores were 0,86 and 0,874.

As these models performed well on both the original and the extended datasets, I did not try to find a model that might have performed even better on the extended datasets. The best-performing model predicted May and November to belong to the category of months with a high number of attacks.

Beyond creating categories based on the mean of the number of monthly incidents, I also ran the models with categories based on whether they were greater than or smaller than or equal to three. They performed well: the Gaussian Naïve Bayes model running on a standardized, non-dimensionally reduced dataset containing the data of all four countries and divided into training and test data at an 80-20 rate achieved 92% test accuracy and 93% training accuracy. Its F1 score was 0,84 on the test data and 0,85 on the training data.

While I do not consider these results as significant as the ones achieved with the classes divided along the mean of the number of monthly incidents, they were useful in evaluating whether my models were able to recognize rarer occurrences (out of the 190 months present in the extended dataset, only 40 contained more than 3 attacks).

Notes

As the dependent and independent variables were extended using Poisson regression and ARIMA, the accuracy scores and other metrics based on the extended data must be taken with a grain of salt.

Summary and future opportunities, challenges

This chapter is dedicated to summarizing the process and its results described in the paper.

Based on the literature on the topic, I compiled a dataset containing data pertaining to the economic, social and political data of four countries (Thailand, Indonesia, Malaysia and Singapore), and another one built around incident data.

Cleaning and filtering the data played an important role in the creation of the datasets, as did the other preparatory steps I executed (introducing lagged variables, interpolation, dimensionality reduction).

I processed the data using two types of models: classification models (logistic regression, ridge classification, GBDT, decision trees, SVMs and Gaussian Naïve Bayes models) and models with a numeric target variable (linear regression, ridge regression, LASSO regression, Gradient boosted regression). I tuned these models in a number of ways (feature selection, hyperparameter-optimization, early stopping) in order to achieve adequate accuracy and avoid overfitting, thereby losing generalization ability.

My analysis showed a link – previously alluded to in the literature – between the frequency of piracy and ARAS incidents in the SOMS and the surrounding countries' social (Indonesian crime rates) and economic (the most important which was the GRDP owed to the fishing, agricultural and forestry sectors of some Indonesian coastal provinces) indicators. Indonesia has been shown to play an outside role (even having accounted for their overwhelming presence in the dataset), but several Malaysian and Thai variables also proved suited to predict incidents (even if not alone).

I processed the datasets using several models. While many of them achieved adequate results on both the original ones and the ones I extended until December 2025 with the help of ARIMA and Poisson regression), there were models that performed better than most. Out of the classification models relying on classes divided upon the mean of the number of monthly incidents, the Gaussian Naïve Bayes model and the RFE-enhanced logistic regression model did exceedingly well. I managed to use these models to make adequate predictions for a given month. On average, SVM-based models performed well, too.

High-performing non-classification models could be discovered, too: the early stopping-enhanced LASSO regression model did extremely well on both the original datasets, and the ones extended until December 2025.

My attempts at making sub-monthly predictions bore meager results: I established strong links between the frequency of attacks, the type of attacked ship(s) and the time of day, and weak ones between the frequency of attacks and the sub-monthly period. Furthermore, I used cluster analysis in an attempt to help organize patrolling routes.

Based on the results of my thesis, I feel confident in saying that machine learning models – both the ones used in my paper and others – could have a positive role to play in predicting incidents of piracy and ARAS in the Straits of Singapore and Malacca. This is not without its prerequisites, of course: in its current form, ReCAAP's data isn't suited for analysis and regional cooperation must be expanded to include the statistical departments of the countries in the region for the purpose of building sustainable models.

Bibliography

Analytics Vidhya. (2024. november 19). Recursive Feature Elimination (RFE): Working, Advantages & Examples. *Analytics Vidhya*. Forrás: https://www.analyticsvidhya.com/blog/2023/05/recursive-feature-elimination/

András, E. (2024. december 10). *github.com*. Letöltés dátuma: 2024. december 10, forrás: ecsand/szakdolgozat: https://github.com/ecsand/Szakdolgozat

avicksaha. (2024. június 10). Gamma Parameter in SVM. *GeeksforGeeks*. Forrás: https://www.geeksforgeeks.org/gamma-parameter-in-svm/

Bileta, V. (2022. november 27). The Seven Voyages of Zheng He: When China Ruled the Seas. *TheCollector*. Forrás: https://www.thecollector.com/zheng-he-seven-voyages/

Bobbitt, Z. (2021. augusztus 12). Z-Score Normalization: Definition & Examples. *Statology*. Forrás: https://www.statology.org/z-score-normalization/

byjus.com. (dátum nélk.). Letöltés dátuma: 2024. december 6, forrás: Interpolation | Definition, Formula, Methods & Uses: https://byjus.com/maths/interpolation/

Chowdhury, M. Z., & Turin, T. (2020. február 16). Variable selection strategies and its importance in clinical prediction modelling. *Fam Med Community Health*, 2. Forrás: https://pmc.ncbi.nlm.nih.gov/articles/PMC7032893/

citypopulation.de. (dátum nélk.). Letöltés dátuma: 2024. december 4, forrás: Indonesia: Administrative Division (Provinces, Regencies and Cities) with population statistics, charts and maps.: https://www.citypopulation.de/en/indonesia/admin/

data.gov.my-1. (2023. július 28). Letöltés dátuma: 2024. október 30, forrás: Poverty by State: https://data.gov.my/data-catalogue/hh_poverty_state

data.gov.my-1. (2023. július 28). Letöltés dátuma: 2024. 10 30, forrás: Poverty by State: https://data.gov.my/data-catalogue/hh_poverty_state

data.gov.my-2. (2024). Letöltés dátuma: 2024. november 6, forrás: Monthly Principal Labour Force Statistics: https://data.gov.my/data-catalogue/lfs_month

data.gov.my-2. (2024). Letöltés dátuma: 2024. 11 6, forrás: Monthly Principal Labour Force Statistics: https://data.gov.my/data-catalogue/lfs_month

data.gov.my-3. (2024). Letöltés dátuma: 2024. november 6, forrás: Malaysian Economic Indicator: https://data.gov.my/data-catalogue/economic_indicators

data.gov.my-4. (2024). Letöltés dátuma: 2024. november 6, forrás: Producer Price Index: https://data.gov.my/data-catalogue/ppi

databank.worldbank.org/-1. (2024). Letöltés dátuma: 2024. november 5, forrás: Indonesia Database for Policy and Economic Research: https://databank.worldbank.org/source/indonesia-database-for-policy-and-economic-research/preview/on#

databank.worldbank.org/-1. (2024). Forrás: Indonesia Database for Policy and Economic Research: https://databank.worldbank.org/source/indonesia-database-for-policy-and-economic-research/preview/on#

databank.worldbank.org-2. (2024). Letöltés dátuma: 2024. október 31, forrás: World Development Indicators | DataBank: https://databank.worldbank.org/reports.aspx?source=2&country=MYS#

databank.worldbank.org-3. (dátum nélk.). Letöltés dátuma: 2024. november 27, forrás: World Development Indicators | DataBank: https://databank.worldbank.org/reports.aspx?source=2&country=THA

data-explorer.oecd.org. (2024. március 8). Letöltés dátuma: 2024. december 3, forrás: OECD Data Explorer - Employment in fisheries, aquaculture and processing: https://data-explorer.oecd.org/vis?lc=en&df[ds]=dsDisseminateFinalDMZ&df[id]=DSD_FISH_EMP%40DF_FISH_EMPL&df[ag]=OECD.TAD.ARP&dq=.A...._T._T&pd=2010%2C&to[TIME_PERIOD]=false

data-explorer.oecd.org. (2024. március 8). Forrás: OECD Data Explorer - Employment in fisheries, aquaculture and processing: https://data-explorer.oecd.org/vis?lc=en&df[ds]=dsDisseminateFinalDMZ&df[id]=DSD_FISH_EMP%40DF_FISH_EMPL&df[ag]=OECD.TAD.ARP&dq=.A...._T._T&pd=2010%2C&to[TIME_PERIOD]=false

developers.google.com. (2021. Szeptember 8). Letöltés dátuma: 2024. december 4, forrás: Classification: Accuracy, recall, precision, and related metrics: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

Faridi, R. (dátum nélk.). Time Series Analysis. Letöltés dátuma: 2024. december 3, forrás: https://bookdown.org/rushad_16/TSA_Lectures_book/regression-models.html

fred.stlouisfed.org-1. (2024. április 15). Letöltés dátuma: 2024. november 2, forrás: Consumer Price Index: All Items: Total for Indonesia (CPALTT01IDM657N): https://fred.stlouisfed.org/series/CPALTT01IDM657N

fred.stlouisfed.org-10. (2024. október 15). Letöltés dátuma: 2024. november 7, forrás: World Uncertainty Index for Singapore (WUISGP): https://fred.stlouisfed.org/series/WUISGP

fred.stlouisfed.org-2. (2024. október 18). Letöltés dátuma: 2024. november 2, forrás: Nominal Gross Domestic Product for Indonesia (NGDPSAXDCIDQ): https://fred.stlouisfed.org/series/NGDPSAXDCIDQ

fred.stlouisfed.org-3. (2024. október 18). Letöltés dátuma: 2024. november 2, forrás: Nominal Gross Domestic Product for Indonesia (NGDPNSAXDCIDQ): https://fred.stlouisfed.org/series/NGDPNSAXDCIDQ

fred.stlouisfed.org-4. (2024. október 18). Letöltés dátuma: 2024. november 2, forrás: Real Gross Domestic Product for Indonesia (NGDPRSAXDCIDQ): https://fred.stlouisfed.org/series/NGDPRSAXDCIDQ

fred.stlouisfed.org-5. (2024. október 18). Letöltés dátuma: 2024. november 2, forrás: Real Gross Domestic Product for Indonesia (NGDPRNSAXDCIDQ): https://fred.stlouisfed.org/series/NGDPRNSAXDCIDQ

fred.stlouisfed.org-6. (2024. október 15). Letöltés dátuma: 2024. november 2, forrás: World Uncertainty Index for Indonesia (WUIIDN): https://fred.stlouisfed.org/series/WUIIDN

fred.stlouisfed.org-7. (2024. július 2). Letöltés dátuma: 2024. november 2, forrás: Gross Domestic Product for Malaysia (MKTGDPMYA646NWDB): https://fred.stlouisfed.org/series/MKTGDPMYA646NWDB

fred.stlouisfed.org-8. (2024. október 15). Letöltés dátuma: 2024. november 2, forrás: World Uncertainty Index for Malaysia (WUIMYS): https://fred.stlouisfed.org/series/WUIMYS

fred.stlouisfed.org-9. (2024. július 2). Letöltés dátuma: 2024. november 2, forrás: Inflation, consumer prices for Malaysia (FPCPITOTLZGMYS): https://fred.stlouisfed.org/series/FPCPITOTLZGMYS

GeeksforGeeks. (2024. március 20). What is Adam Optimizer? *GeeksforGeeks*. Forrás: https://www.geeksforgeeks.org/adam-optimizer/

GeeksForGeeks. (2024. szeptember 19). What is Lag in Time Series Forecasting. *GeeksForGeeks*. Letöltés dátuma: 2024. december 3, forrás: https://www.geeksforgeeks.org/what-is-lag-in-time-series-forecasting/

Gupta, L. (2024. március 29). Time Series Forecasting (Stationarity , Differencing , Transformations). *Medium*. Forrás: https://medium.com/@lahar091103/time-series-forecasting-stationarity-differencing-transformations-c4e2d52ddd47

Haire, P. (2021. január 25). The Currents in Singapore Strait are Extremely Complex. Here's Why. *Tidetech*. Forrás: https://www.tidetechmarinedata.com/news/the-complex-currents-in-singapore-strait

https://federicotartarini.github.io/. (dátum nélk.). Letöltés dátuma: 2024. december 5, forrás: Singapore's climate: https://federicotartarini.github.io/air-quality-weather-sg/climate-of-singapore/

https://www.recaap.org. (dátum nélk.). Letöltés dátuma: 2024. december 5, forrás: About ReCAAP Information Sharing Centre: https://www.recaap.org/about_ReCAAP-ISC

https://www.singstat.gov.sg/-1. (2024). Letöltés dátuma: 2024. november 7, forrás: Singapore Department of Statistics (DOS) | SingStat Table Builder: https://tablebuilder.singstat.gov.sg/table/TS/M651101

https://www.singstat.gov.sg/-2. (2024). Letöltés dátuma: 2024. november 7, forrás: Singapore Department of Statistics (DOS): https://tablebuilder.singstat.gov.sg/table/TS/M015651

https://www.singstat.gov.sg/-3. (2024. október 29). Letöltés dátuma: 2024. november 7, forrás: Singapore Department of Statistics (DOS): https://tablebuilder.singstat.gov.sg/table/TS/M182332

IBM. (2023. december 8). What is principal component analysis (PCA)? *IBM*. Letöltés dátuma: 2024. december 3, forrás: https://www.ibm.com/topics/principal-component-analysis

imf.org-2. (2024. október). Letöltés dátuma: 2024. december 4, forrás: World Economic Outlook (October 2024) - Inflation rate, average consumer prices: https://www.imf.org/external/datamapper/PCPIPCH@WEO/IDN/MYS

imf.org-3. (2024. október). Letöltés dátuma: 2024. december 4, forrás: World Economic Outlook (October 2024) - Real GDP growth: https://www.imf.org/external/datamapper/NGDP_RPCH@WEO/IDN/MYS

itl.nist.gov. (dátum nélk.). Letöltés dátuma: 2024. december 9, forrás: 1.3.6.6.19. Poisson Distribution: https://www.itl.nist.gov/div898/handbook/eda/section3/eda366j.htm

Kanade, V. (2023. április 3). What Is Linear Regression? Types, Equation, Examples, and Best Practices for 2022. *Spiceworks*. Forrás: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-linear-regression/

keras.io. (dátum nélk.). Letöltés dátuma: 2024. december 3, forrás: Dense layer: https://keras.io/api/layers/core_layers/dense/

Kuknyó, D. (2024). Üzleti Elemzések Módszertana 3. Gyakorlat: Regularizált modellek. 17. Forrás: https://github.com/basictask/Elemzesmodszertan/blob/main/3_regularization/doc/3_regularization.pdf

Kuknyó, D. (2024). Üzleti Elemzések Módszertana 2. Előadás: Osztályozás. Letöltés dátuma: 2024. december 3, forrás: https://github.com/basictask/Elemzesmodszertan/blob/main/2_classification/doc/2_classification.pdf

Kuknyó, D. (2024). Üzleti Elemzések Módszertana 3. Gyakorlat: Regularizált Modellek. Forrás: https://github.com/basictask/Elemzesmodszertan/blob/main/3_regularization/doc/3_regularization.pdf

Kuknyó, D. (2024). Üzleti Elemzések Módszertana 3. Gyakorlat: Regularizált Modellek. 17. Letöltés dátuma: 2024. december 3, forrás: https://github.com/basictask/Elemzesmodszertan/blob/main/3_regularization/doc/3_regularization.pdf

Kuknyó, D. (2024). Üzleti Elemzések Módszertana 4. Gyakorlat: Döntési fák. Letöltés dátuma: 2024. december 3, forrás: https://github.com/basictask/Elemzesmodszertan/blob/main/4_decision_trees/doc/4_decision_trees.pdf

Kuknyó, D. (2024). Üzleti Elemzések Módszertana 6. Előadás: Tartó vektor gépek. Letöltés dátuma: 2024. december 3, forrás: https://github.com/basictask/Elemzesmodszertan/blob/main/6_svm/doc/6_svm.pdf

Mangale, S. (2020. augusztus 28). Scree Plot. *Medium*. Forrás: https://sanchitamangale12.medium.com/scree-plot-733ed72c8608

Martins, C. (2023. november 2). Gaussian Naive Bayes Explained With Scikit-Learn. *Built In*. Forrás: https://builtin.com/artificial-intelligence/gaussian-naive-bayes

Nalcin, S. (2022. október 11). StandardScaler vs. MinMaxScaler vs. RobustScaler: Which one to use for your next ML project? *Medium*. Forrás: https://medium.com/@onersarpnalcin/standardscaler-vs-minmaxscaler-vs-robustscaler-which-one-to-use-for-your-next-ml-project-ae5b44f571b9

Olamendy, J. C. (2023. december 4). Understanding ReLU, LeakyReLU, and PReLU: A Comprehensive Guide. *Medium*. Forrás: https://medium.com/@juanc.olamendy/understanding-relu-leakyrelu-and-prelu-a-comprehensive-guide-20f2775d3d64

open.dosm.gov.my. (dátum nélk.). Letöltés dátuma: 2024. 11 6, forrás: CPI by Strata & Division (2-digit): https://open.dosm.gov.my/data-catalogue/cpi_strata

OpenDOSM. (dátum nélk.). Letöltés dátuma: 2024. 11 6, forrás: CPI by Strata & Division (2-digit): https://open.dosm.gov.my/data-catalogue/cpi_strata

open-meteo.com. (dátum nélk.). Letöltés dátuma: 2024. december 5, forrás: Weather Forecast API: https://open-meteo.com/en/docs

pandas.pydata.org. (dátum nélk.). Letöltés dátuma: 2024. december 6, forrás: Pandas: https://pandas.pydata.org/

Panneerselvam, P., & Ramkumar, K. (2023. május 15). Piracy and Armed Robbery in Southeast Asia: The Need for a Fresh Approach. *The Diplomat*. Forrás: https://thediplomat.com/2023/05/piracy-and-armed-robbery-in-southeast-asia-the-need-for-a-fresh-approach/

Paterson, S. (2023. július 27.). Dire straits: Malacca, Singapore and the future of the global economy. *Asia Scotland Institute*. Forrás: https://asiascot.com/articles/straits-of-malacca

Rajan, S. (2021. október 26). Dimensionality Reduction using AutoEncoders in Python. *Analytics Vidhyia*. Forrás: https://www.analyticsvidhya.com/blog/2021/06/dimensionality-reduction-using-autoencoders-in-python/

Raymond, C. Z. (2009). PIRACY AND ARMED ROBBERY IN THE MALACCA STRAIT: A Problem Solved? *Naval War College Review*, 2. Letöltés dátuma: 2024. 12 3, forrás: https://www.jstor.org/stable/26397033

ReCAAP ISC. (2024). *ReCAAP ISC Annual Report 2023.* RecAAP ISC. Forrás: https://www.recaap.org/resources/ck/files/reports/annual/ReCAAP%20ISC%20Annual%20Report%202023.pdf: https://www.recaap.org/resources/ck/files/reports/annual/ReCAAP%20ISC%20Annual%20Report%202023.pdf

recaap.org. (dátum nélk.). Letöltés dátuma: 2024. december 5, forrás: ReCAAP ISC: https://www.recaap.org/resources/ck/files/Number%20of%20Incidents/2023/List%20of%20Incidents%20for%202023.pdf

Riswanto, U. (2023. május 18). What Is An Autoencoder? Why It's So Important For Dimensionality Reduction. *Medium*. Forrás: https://ujangriswanto08.medium.com/what-is-an-autoencoder-why-its-so-important-for-dimensionality-reduction-9ba723b34c2b

scikit-learn.org. (dátum nélk.). Letöltés dátuma: 2024. december 3, forrás: Ridge: https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.Ridge.html

scikit-learn.org. (dátum nélk.). Letöltés dátuma: 2024. december 6, forrás: sklearn.ensemble — scikit-learn 1.5.2 documentation: https://scikit-learn.org/1.5/api/sklearn.ensemble.html

Shelar, V. (2023. szeptember 10). *"Ridge Regression : Empowering Predictions with Ridge Regression"*. Forrás: medium.com: https://medium.com/@vishalshelar328/ridge-regression-empowering-predictions-with-ridge-regression-58f1075e548a

singstat.gov.sg-4. (2024). Letöltés dátuma: 2024. november 7, forrás: (DOS) | SingStat Table Builder – Consumer Price Index (CPI), 2019 As Base Year: https://tablebuilder.singstat.gov.sg/table/TS/M212881

Spiess, R. (2019. július 16). A pirate's paradise. *Southeast Asia Globe*. Letöltés dátuma: 2024. 12 3, forrás: https://southeastasiaglobe.com/how-corruption-is-fuelling-modern-day-piracy/

Storey, I. (2022. augusztus 15). Piracy and the Pandemic: Maritime Crime in Southeast Asia, 2020-2022. *Fulcrum*. Forrás: https://fulcrum.sg/piracy-and-the-pandemic-maritime-crime-in-southeast-asia-2020-2022/

Thakar, C., & Tahsildar, S. (2022. november 22). Gini Index: Decision Tree, Formula, Calculator, Gini Coefficient in Machine Learning. *Quantitative Finance & Algo Trading Blog by QuantInsti*. Forrás: https://blog.quantinsti.com/gini-index/

The Nippon Foundation. (dátum nélk.). *Safety in the Straits of Malacca and Singapore*. Letöltés dátuma: 2024. december 3, forrás: https://www.nippon-foundation.or.jp: https://www.nippon-foundation.or.jp/en/what/projects/safe_passage

tibco.com. (dátum nélk.). Letöltés dátuma: 2024. december 9, forrás: ARIMA Models and Forecasting: https://docs.tibco.com/pub/stat/14.0.0/doc/html/UsersGuide/GUID-BCF27023-F83B-4327-9271-9D760B67470D.html

Verma, N. (2023. november 6). *Gradient Boosting Regression Implementation in Python for Predictive Modeling.* Letöltés dátuma: 2024. december 3, forrás: medium.com: https://medium.com/@nandiniverma78988/gradient-boosting-regression-implementation-in-python-for-predictive-modeling-437e4ece8c9e

Watson, W. H. (2013. június 13). Singapore at Heart of Counter Piracy Worldwide. *MarineLink*. Forrás: https://www.marinelink.com/news/singapore-worldwide355616

Wintergalen, E. W., Oyanedel, R., Villaseñor-Derbez, J. C., Fulton, S., & Molina, R. (2022). Opportunities and challenges for livelihood resilience in urban and rural Mexican small-scale fisheries. *Ecology and Society*. Forrás: https://ecologyandsociety.org/vol27/iss3/art46/

WMO Code Table 4677. (dátum nélk.). Letöltés dátuma: 2024. december 4, forrás: https://www.nodc.noaa.gov: https://www.nodc.noaa.gov/archive/arc0021/0002199/1.1/data/0-data/HTML/WMO-CODE/WMO4677.HTM

www.adobe.com. (dátum nélk.). Letöltés dátuma: 2024. december 6, forrás: Convert PDF to Excel: https://www.adobe.com/uk/acrobat/online/pdf-to-excel.html

www.bps.go.id-1. (2024). Letöltés dátuma: 2024. november 5, forrás: [2010 Version] Quarterly GRDP At Current Market Price by Industrial Origin in Province All Over Indonesia (Billion Rupiah), 2010-2024 - Statistical Data: https://www.bps.go.id/en/statistics-table/1/MjIwNSMx/-2010-version--quarterly-grdp-at-current-market-price-by-industrial-origin-in-province-all-over-indonesia--billion-rupiah---2010-2024.html

www.bps.go.id-1. (2024). Forrás: [2010 Version] Quarterly GRDP At Current Market Price by Industrial Origin in Province All Over Indonesia (Billion Rupiah), 2010-2024 - Statistical Data: https://www.bps.go.id/en/statistics-table/1/MjIwNSMx/-2010-version--quarterly-grdp-at-current-market-price-by-industrial-origin-in-province-all-over-indonesia--billion-rupiah---2010-2024.html

www.bps.go.id-2. (2023). Letöltés dátuma: 2024. november 5, forrás: Number Of Crime According To Police Territorial Jurisdiction - Statistical Data: https://www.bps.go.id/en/statistics-table/2/MTAxIzI=/number-of-crime-according-to-police-territorial-jurisdiction.html

www.bps.go.id-2. (2023). Forrás: Number Of Crime According To Police Territorial Jurisdiction - Statistical Data: https://www.bps.go.id/en/statistics-table/2/MTAxIzI=/number-of-crime-according-to-police-territorial-jurisdiction.html

www.bps.go.id-3. (dátum nélk.). Letöltés dátuma: 2024. december 5, forrás: Unemployment Rate by Province (Percent), 2024: https://www.bps.go.id/en/statistics-table/2/NTQzIzI=/unemployment-rate--february-2024.html

www.bps.go.id-3. (dátum nélk.). Forrás: Unemployment Rate by Province (Percent), 2024: https://www.bps.go.id/en/statistics-table/2/NTQzIzI=/unemployment-rate--february-2024.html

www.imf.org. (2024). Letöltés dátuma: 2024. november 6, forrás: World Economic Outlook (October 2024) - GDP per capita, current prices: https://www.imf.org/external/datamapper/PPPPC@WEO/OEMDC/ADVEC/WEOWORLD

www.imf.org. (2024. november 6). Forrás: World Economic Outlook (October 2024) - GDP per capita, current prices: https://www.imf.org/external/datamapper/PPPPC@WEO/OEMDC/ADVEC/WEOWORLD

www.macrotrends.net. (dátum nélk.). Letöltés dátuma: 2024. november 9, forrás: Singapore Crime Rate & Statistics 1990-2024: https://www.macrotrends.net/global-metrics/countries/sgp/singapore/crime-rate-statistics

www.recaap.org. (dátum nélk.). Letöltés dátuma: 2024. december 5, forrás: Reports: https://www.recaap.org/reports

www.recaap.org. (dátum nélk.). Letöltés dátuma: 2024. szeptember 22, forrás: ReCAAP ISC: https://www.recaap.org/resources/ck/files/Number%20of%20Incidents/2022/List%20of%20Incidents%20for%202022.pdf

www.recaap.org. (dátum nélk.). Letöltés dátuma: 2024. szeptember 24, forrás: https://www.recaap.org/resources/ck/files/Number%20of%20Incidents/2024/List%20of%20Incidents%20for%202024.pdf

ZalaRushirajsinh. (2023. november 4). The Elbow Method: Finding the Optimal Number of Clusters. *Medium*. Forrás: https://medium.com/@zalarushirajsinh07/the-elbow-method-finding-the-optimal-number-of-clusters-d297f5aeb189

Glossary: describing country data

Data pertaining to Indonesia

A továbbiakban az Indonéziára vonatkozó adatokat fogom ismertetni. Ezek négy fő forrásból származnak: a BPS-től (Indonéz Statisztikai Hivatal), a St. Louis-i Központi Bank által fenntartott FRED (Federal Reserve Economic Data) adatbázisból, a Világbank DataBank oldaláról, illetve az OECD Data Explorer oldaláról.

Below I will describe the data pertaining to Indonesia. These hail from four main sources: BPS (The Indonesian Department of Statistics), the St. Lous Federal Reserve's FRED (Federal Reserve Economic Data) database, the World Bank and the OECD Data Explorer platform.

The variables pertaining to Thailand are the following:

- yearly and quarterly GRDP by province and industry between 2010 and 2024, in billion Indonesian rupees (www.bps.go.id-1, 2024). I filtered for "Agriculture, Forestry and Fishing" and eliminated yearly data.
- Yearly number of criminal offences by police territorial jurisdictions between 2008 and 2022 (www.bps.go.id-2, 2023). Since the data could be downloaded in 2-3 year batches from the website, I turned these batches into dataframes and concatenated them.
- Biannually measured unemployment by province between 2006 and 2024, as a percentage (www.bps.go.id-3, dátum nélk.). Data pertaining to yearly periods could be downloaded from the site. I turned these batches into dataframes and concatenated them.
- Monthly CPI: the growth rate of the previous period. The timeseries contained data from February 1968 to March 2024 (fred.stlouisfed.org-1, 2024).
- Quarterly nominal GRDP, seasonally adjusted, in million Indonesian rupees between January 1990 and March 2024 (fred.stlouisfed.org-2, 2024).
- Quarterly nominal GRDP in million Indonesian rupees between January 2008 and March 2024 (fred.stlouisfed.org-3, 2024).
- Quarterly real GRDP, seasonally adjusted, in million Indonesian rupees between January 2000 and March 2024 (fred.stlouisfed.org-4, 2024).
- Quarterly real GRDP in million Indonesian rupees between January 2008 and March 2024 (fred.stlouisfed.org-5, 2024).
- Uncertainty index (fred.stlouisfed.org-6, 2024): The World Uncertainty Index's quarterly data for Indonesia between January 1952 and July 2024. It counts the

appearances of the word “uncertainty” in the quarterly country reports of the Economist Intelligence Unit.

- The yearly pieces of data gathered from the World Bank’s Databank platform (databank.worldbank.org/-1, 2024) are the following:
    - Population, total
    - GNI, Atlas method (current US$)
    - GDP (current US$)
    - Inflation, GDP deflator (annual %)
    - Agriculture, forestry, and fishing, value added (% of GDP)
    - Adjusted net national income (current US$)
    - Adjusted net national income per capita (current US$)
    - Adjusted net national income per capita (constant 2015 US$)
    - Adjusted net national income per capita (annual % growth)
    - Poverty headcount ratio at $2.15 a day (2017 PPP) (% of population)
- The number of people working in aquaculture and fish processing (data-explorer.oecd.org, 2024).

The above variables underwent interpolation (see: Chapter 3.2.2) and were compiled into a monthly dataset.

Data pertaining to Malaysia

Below I will describe the pieces of data pertaining to Malaysia. They hail from the following sources: the data.gov.my site (Malaysia’s official open data portal), OpenDOSM (the data portal of the Malaysian Department of Statistics), the FRED database, the World Bank’s DataBank portal, the OECD Data Explorer site and the IMF’s (International Monetary Fund) DataMapper platform.

The pieces of data making it into the dataset are the following

- Yearly GDP in current US dollars, not seasonally adjusted, for the period between 1960 and 2023 (fred.stlouisfed.org-7, 2024).
- Uncertainty Index (fred.stlouisfed.org-8, 2024): The World Uncertainty Index’s quarterly data for Malaysia between January 1952 and July 2024. It counts the appearances of the word “uncertainty” in the quarterly country reports of the Economist Intelligence Unit.

- Yearly inflation as a percent, not seasonally adjusted, for the period between 1960 and 2023 (fred.stlouisfed.org-9, 2024).
- GDP per capita based on PPP (www.imf.org, 2024).
- Poverty rate by state between 1972 and 2022, measured every 2 or 3 years (data.gov.my-1, 2023): its three variables are absolute poverty (the percent of household with a monthly income putting them below the poverty line), hardcore poverty (the percent of households with a monthly income putting them below the food poverty line) and relative poverty (the percent of households whose monthly income doesn't reach half of the given state's median income).
- Monthly labor market statistics between January 2010 and September 2024 (data.gov.my-2, 2024): monthly data on the size of the labor force in 1000 people), the number of employed people (in 1000 people), the number of unemployed people (in 1000 people), the number of people outside the labor market (in 1000 people), the rate of labor force participation (in %) and the rate of employment in the total population (%).
- Malaysian economic indicators (data.gov.my-3, 2024): monthly economic indicators between January 1999 and September 2024. Its variables are the Leading Index (it measures the expectations for the coming month), the Coincident Index (the indicator of monthly economic performance), the Lagging Index (an index validating the first two indices) and the Leading Index (Diffusion) and the Coincident Index (Diffusion). The latter two act as complementors of the first two indices: their values range from 0 to 100. A value of 100 indicates that all components are on a rise, while a value of 0 indicates that they're decreasing.
- Producer Price Index (data.gov.my-4, 2024): ranging from January 2010 to September 2024, this indicator measures the producer price index for five sectors (agriculture, mining, electricity, water supply, manufacture) and their aggregation. It compares each year to 2010.
- CPI by rural/urban divide and 13 groups of products and services for the period between January 2010 and September 2024 (OpenDOSM, dátum nélk.):it compares each year to 2010. I filtered for the rural data, as the fishing industry is more prevalent there. (Wintergalen, Oyanedel, Villaseñor-Derbez, Fulton, & Molina, 2022)
- The yearly data extracted from the World Bank's DataBank is as follows (databank.worldbank.org-2, 2024):

    - Population, total
    - GNI, Atlas method (current US$)
    - GDP (current US$)
    - Inflation, GDP deflator (annual %)
    - Agriculture, forestry, and fishing, value added (% of GDP)
- The number of people working in fishing, aquaculture and fish processing (data-explorer.oecd.org, 2024).

The data above underwent interpolation, then I compiled them into a monthly dataset.

Data pertaining to Singapore

This section is dedicated to the description of data pertaining to Singapore. These hail from the following sources: SingStat (the Singaporean Department of Statistics' site), FRED and the MacroTrends site.

The variables are the following:

- Singaporean crime rate between 1990 and 2021 (www.macrotrends.net, dátum nélk.): while the data source claims to have data until 2024, the last year it contains information for is 2021.
- Uncertainty Index (fred.stlouisfed.org-10, 2024) between January 1952 and July 2024. I filtered out the results before 2008.
- Monthly sea cargo and shipping (https://www.singstat.gov.sg/-1, 2024) between January 1995 and September 2024. It contains the following statistics:
    - Vessel Arrivals (Number)
    - Vessel Arrivals - Shipping Tonnage (Thousand Gross Tonnes)
    - Total Cargo (Thousand Tonnes)
    - Cargo (General) (Thousand Tonnes)
    - Cargo (Bulk) (Thousand Tonnes)
    - Cargo (Oil-In-Bulk) (Thousand Tonnes)
    - Cargo (General & Non-Oil In Bulk) (Thousand Tonnes)
    - Total Container Throughput (Thousand Twenty-Foot Equivalent Units)
    - Bunker Sales (Thousand Tonnes)
    - Singapore Registry Of Ships (End Of Period) - Number (Number)
    - Singapore Registry Of Ships (End Of Period) - '000 GT (Thousand Gross Tonnes)

- Quarterly GDP at current prices, aggregated and pertaining to sectors (https://www.singstat.gov.sg/-2, 2024): the timeseries ranges from the first quarter of 1975 to the second quarter of 2024.
- Unemployment rate aged 15 years and over (https://www.singstat.gov.sg/-3, 2024): Yearly, seasonally adjusted statistics for the period between 1992 and 2024. It shows the state of unemployment for July of every year.
- CPI between January 1961 and September 2024 (singstat.gov.sg-4, 2024): the index uses 2019 as a base year and contains both aggregated data and data for products (eg. bread) and product groups.

The data above underwent interpolation, then I compiled them into a monthly dataset.

Data pertaining to Thailand

I downloaded the data Pertaining to Thailand from the World Bank's DataBank (databank.worldbank.org-3).

- Population, total
- Population growth (annual%)
- Population density (people per sq. km of land area)
- Poverty headcount ratio at national poverty lines (% of population)
- Poverty headcount ratio at $2.15 a day (2017 PPP) (% of population)
- GNI, Atlas method (current US$)
- GNI per capita, Atlas method (current US$)
- GNI, PPP (current international $)
- GNI per capita, PPP (current international $)
- Income share held by lowest 20%
- Life expectancy at birth, total (years)
- Mortality rate, under-5 (per 1,000 live births)
- School enrollment, primary (% gross)
- School enrollment, secondary (% gross)
- School enrollment, primary and secondary (gross), gender parity index (GPI)
- Annual freshwater withdrawals, total (% of internal resources)
- Urban population growth (annual %)
- GDP (current US$)

- GDP growth (annual %)
- Exports of goods and services (% of GDP)
- Imports of goods and services (% of GDP)
- Revenue, excluding grants (% of GDP)
- Domestic credit provided by financial sector (% of GDP)
- Merchandise trade (% of GDP)
- Net barter terms of trade index (2015 = 100)
- External debt stocks, total (DOD, current US$)
- Total debt service (% of exports of goods, services and primary income)
- Political Stability and Absence of Violence/Terrorism: Estimate

The data above underwent interpolation, then I compiled them into a monthly dataset.