National University of “Kyiv-Mohyla Academy”
Information technologies

Report

Homework #3. Prepare Data & Train Model

Prepared
2nd year bachelor’s student
group 113
Programme Applied Maths
Team #6
Team Lead Honcharenko Vladyslav
Team members:
Spitkovska Vladyslava
Tyschenko Ivan
Nych Kateryna
Zasyadko Matviy

Kyiv – 2024

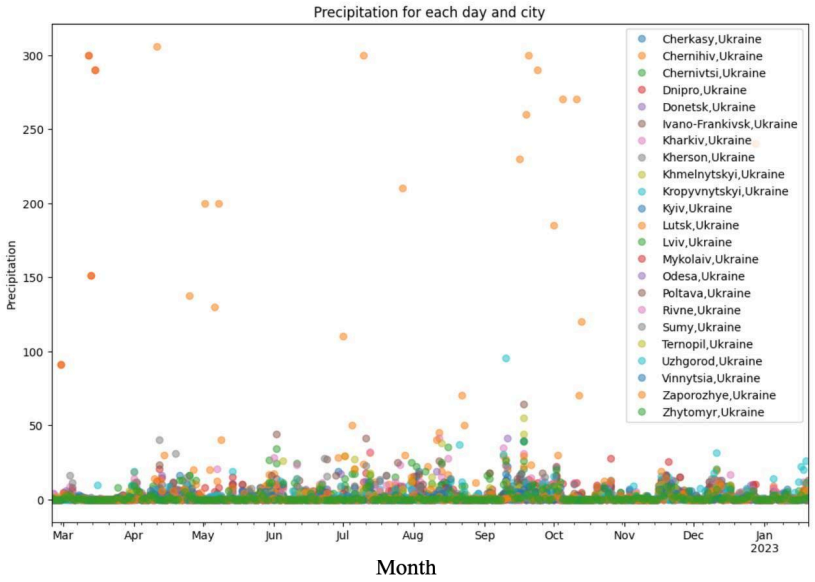
Task list:

1. Exploratory Data Analysis
2. Prepare data
3. Separate data: train & test
4. Train models
5. Evaluate models

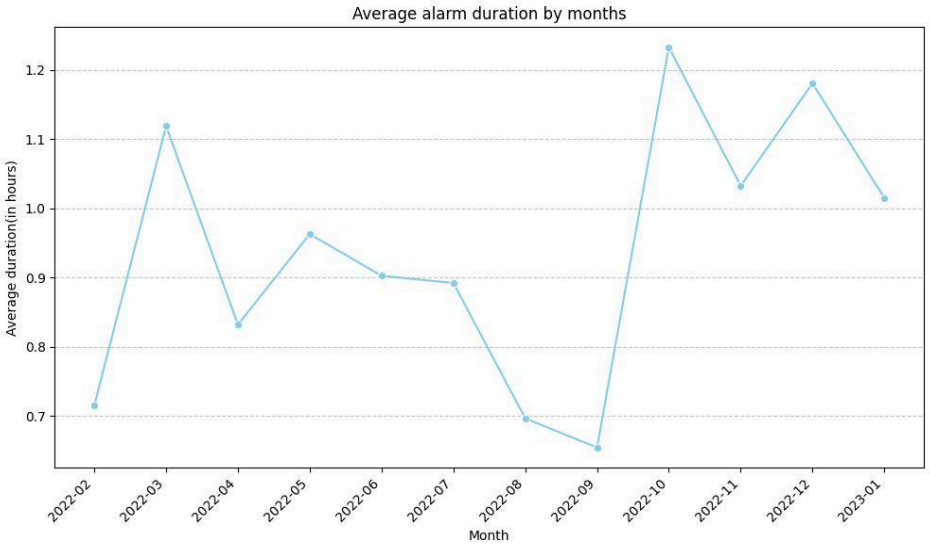
Exploratory Data Analysis

Weather+Alarms EDA:

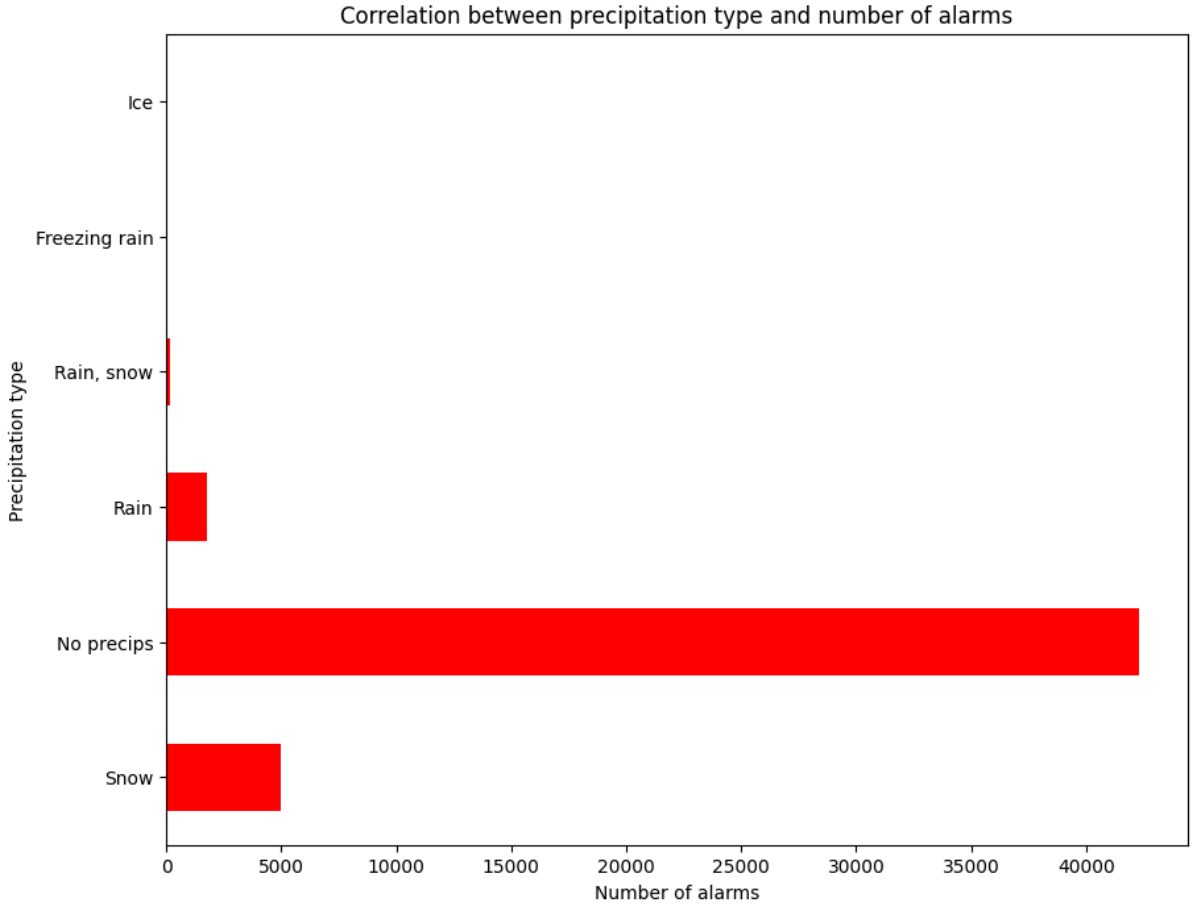
We started from quick analysis of parameters which we assumed could have the biggest correlations.
From here we can tell that the biggest amounts of precipitation appear in the South of Ukraine or sometimes in the North region (Chernihiv).



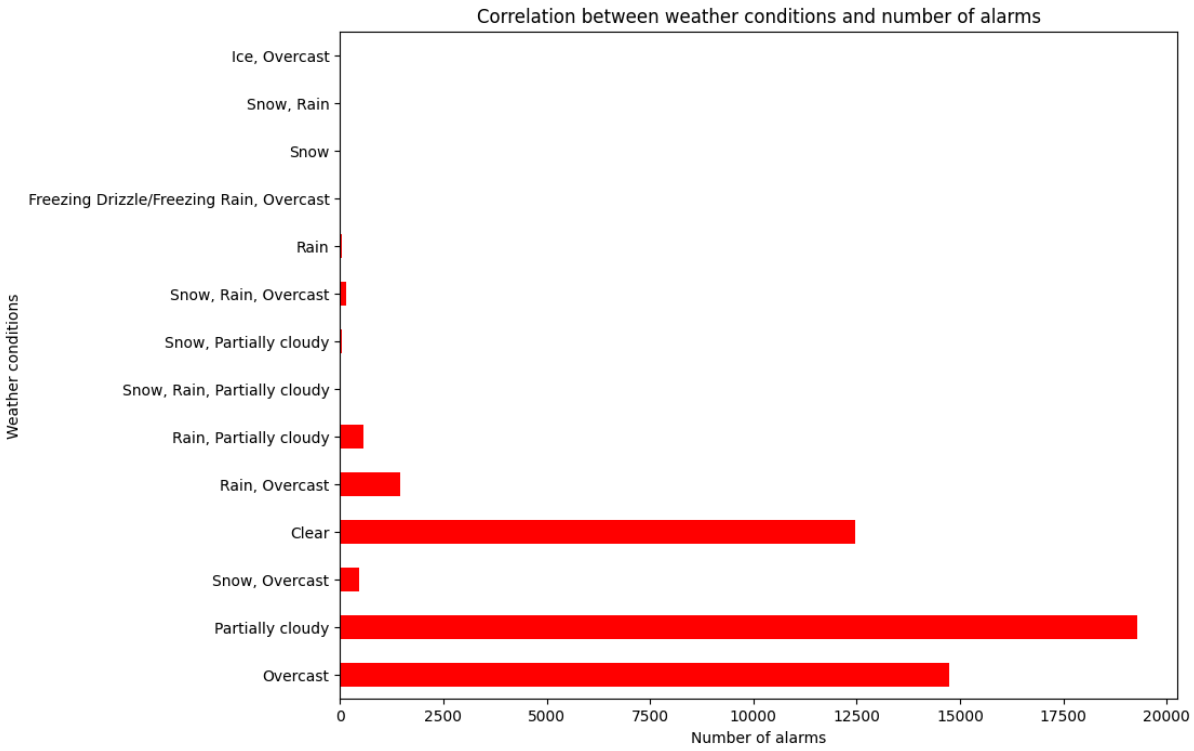
Throughout 2022 and 2023, there were sudden spikes in alarm durations recorded at the onset of the war and in October-November, coinciding with periods of blackouts. A sharp decline occurred during the summer, and there were no alarms recorded at the beginning of September.



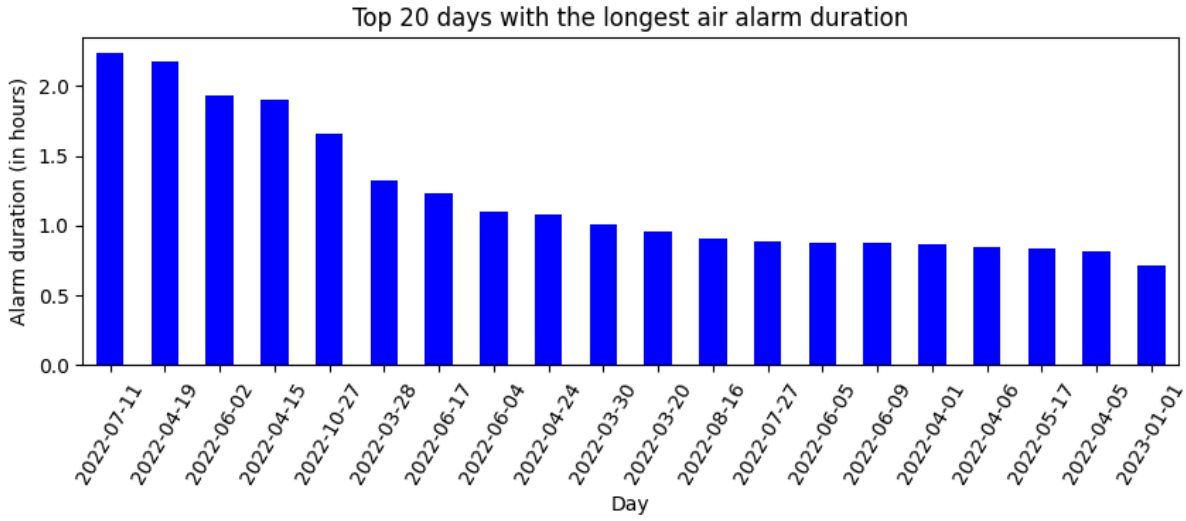
From the graph above we can see that the longest air alarms were in October 2022.



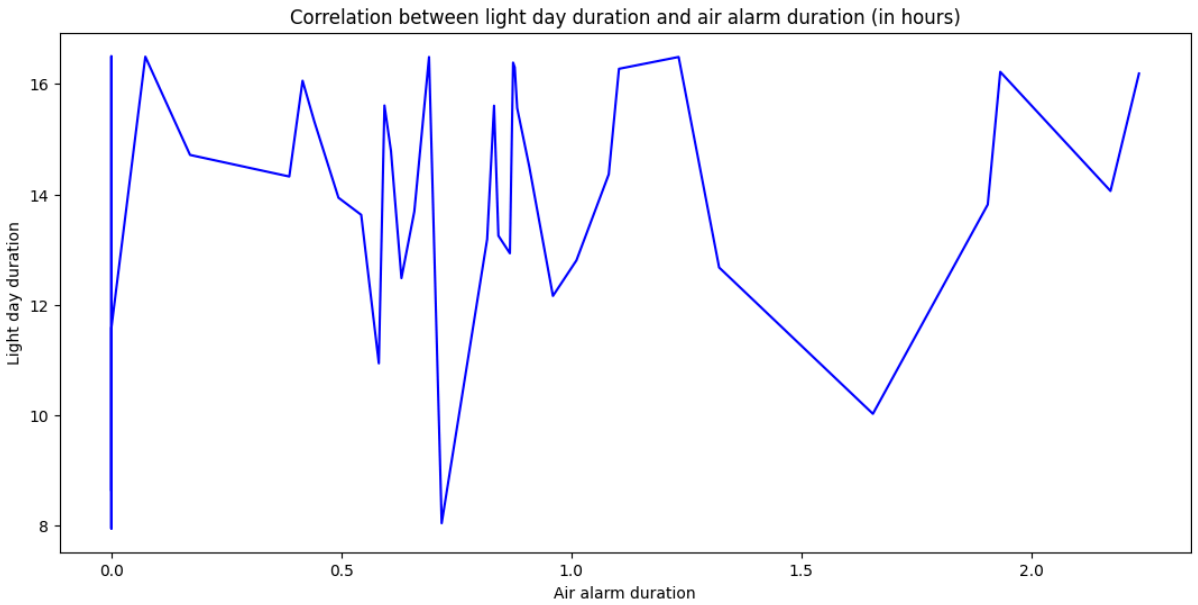
Alarms sounded most often when there was no precipitation. In 2nd place are moments with snowfall.



Contrary to expectations, the highest number of alarms does not fall on moments of clear sky, but on moments of partially cloudiness. Even the number of alarms during overcast is ahead of the number of alarms during clear skies.

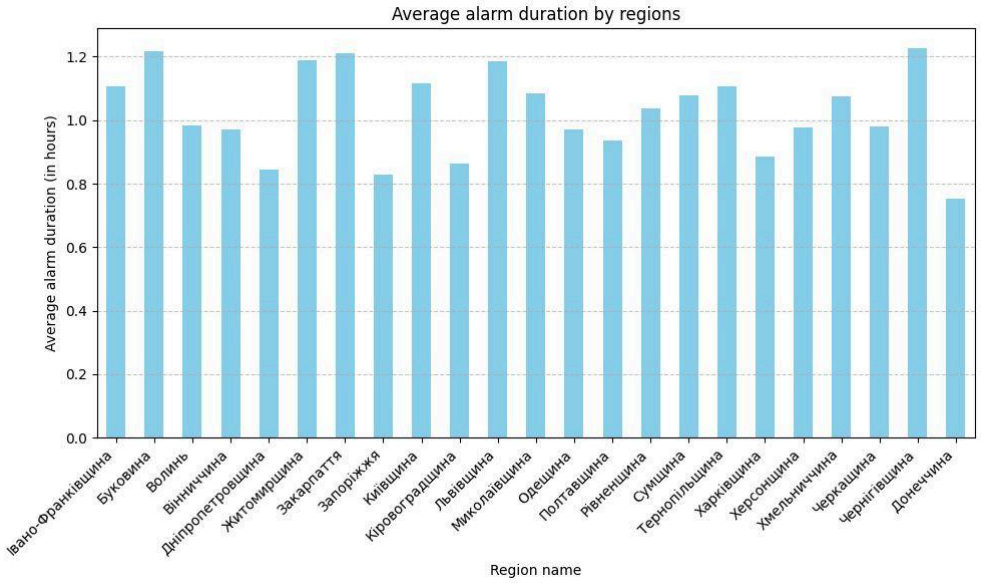


From the graph above we can see that July 11, 2022 it's the day with largest number of air alarms that sounded. Although the reliability of this schedule due to the lack of alarm reports for some periods, for example, for February 24, is somewhat doubtful.

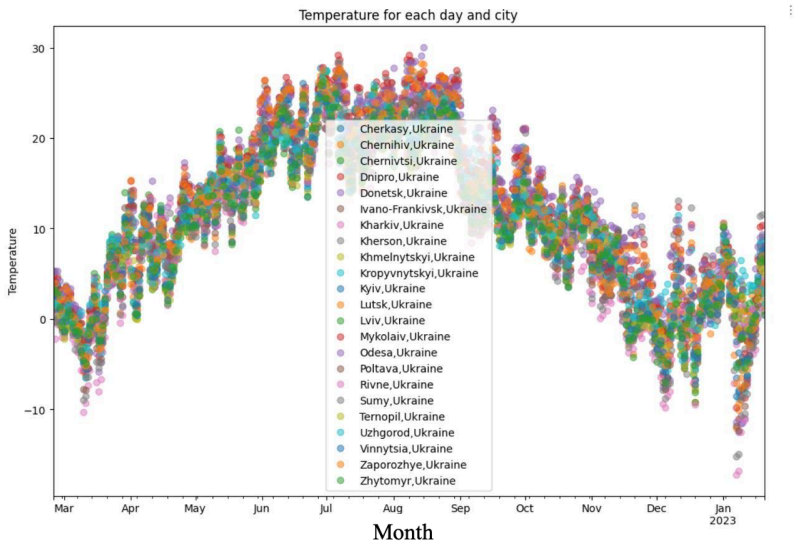


The graph above shows that there is no correlation between the duration of the air alarm and the length of daylight hours.

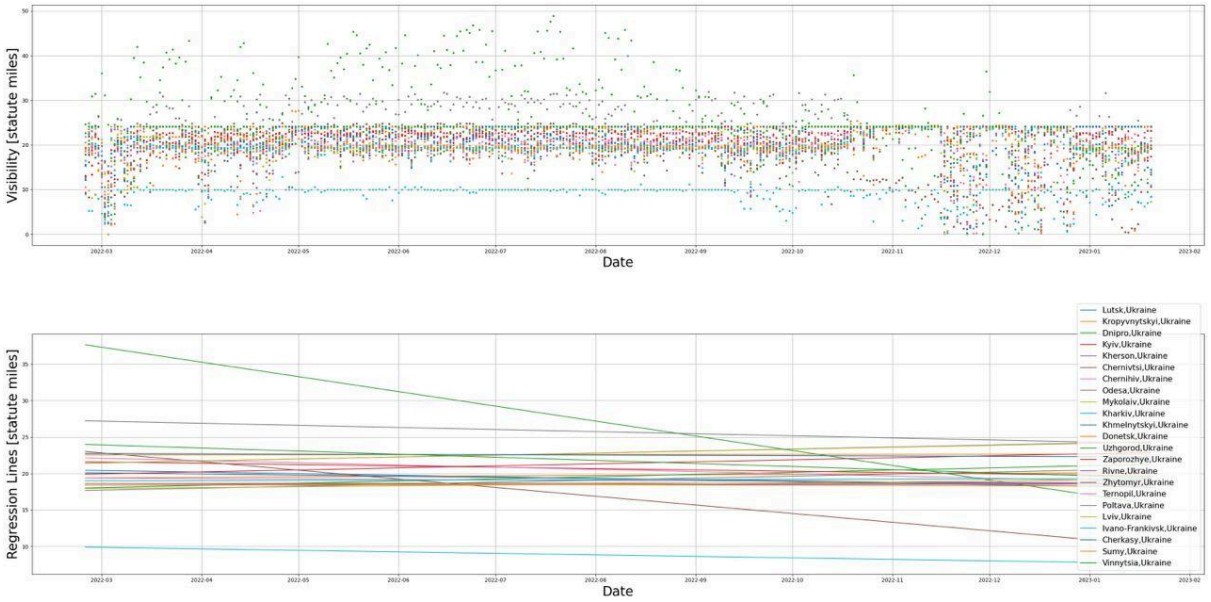
Another graph for average alarm duration by region throughout all time period:



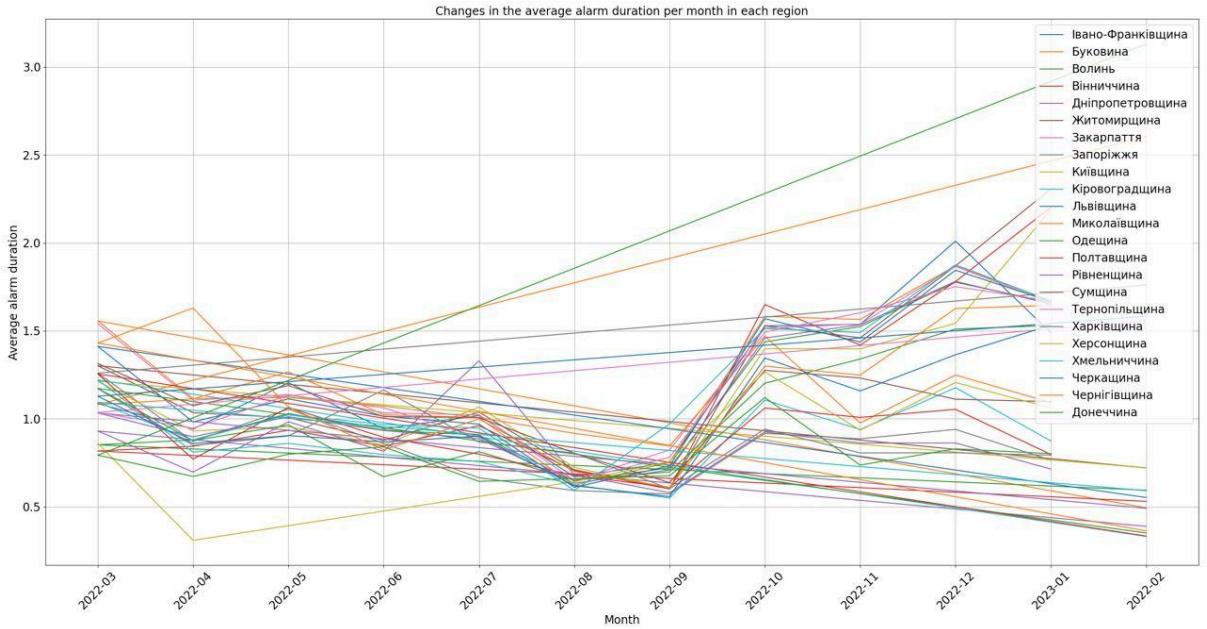
From the graph lower it's visible that the highest temperature throughout the year is in the South as well. The coldest regions though seem to be central and West parts of Ukraine.



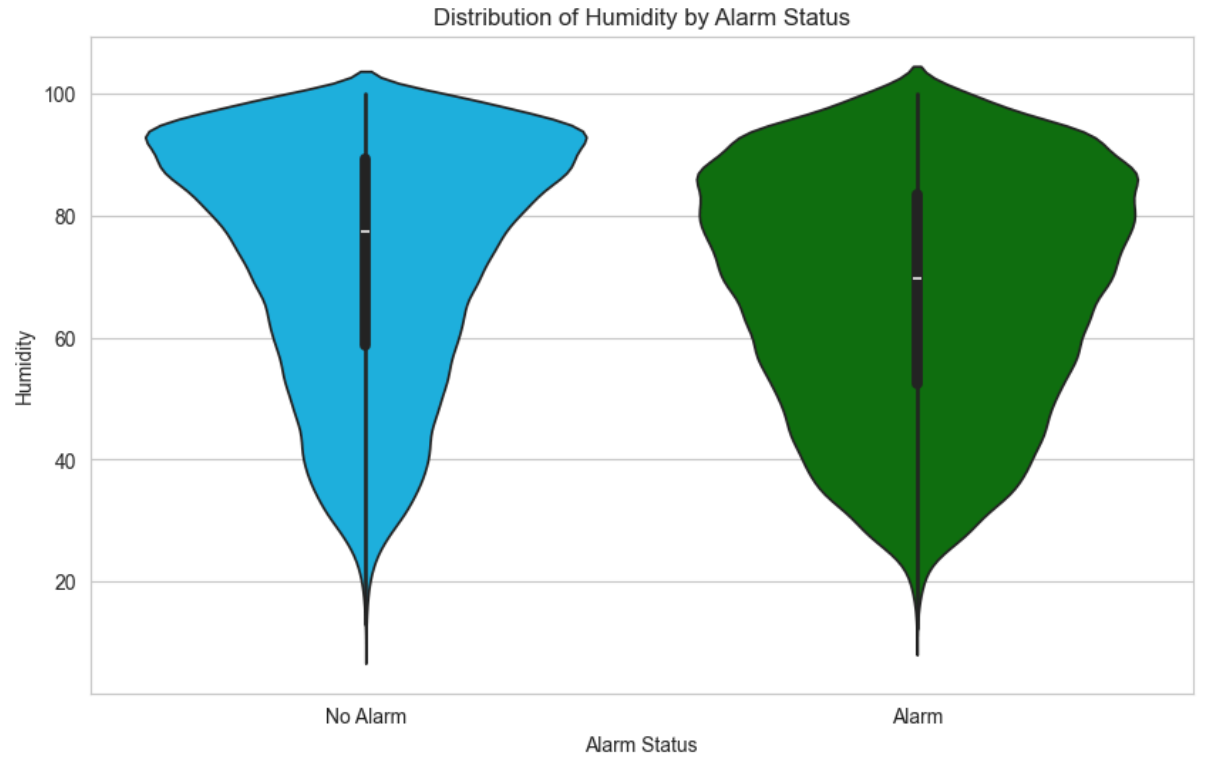
Visibility among prevailing parts of regions is pretty much stable. There're several exceptions though. For example, during 2022-23 years time period visibility level was decreased in particular central and West parts of Ukraine.



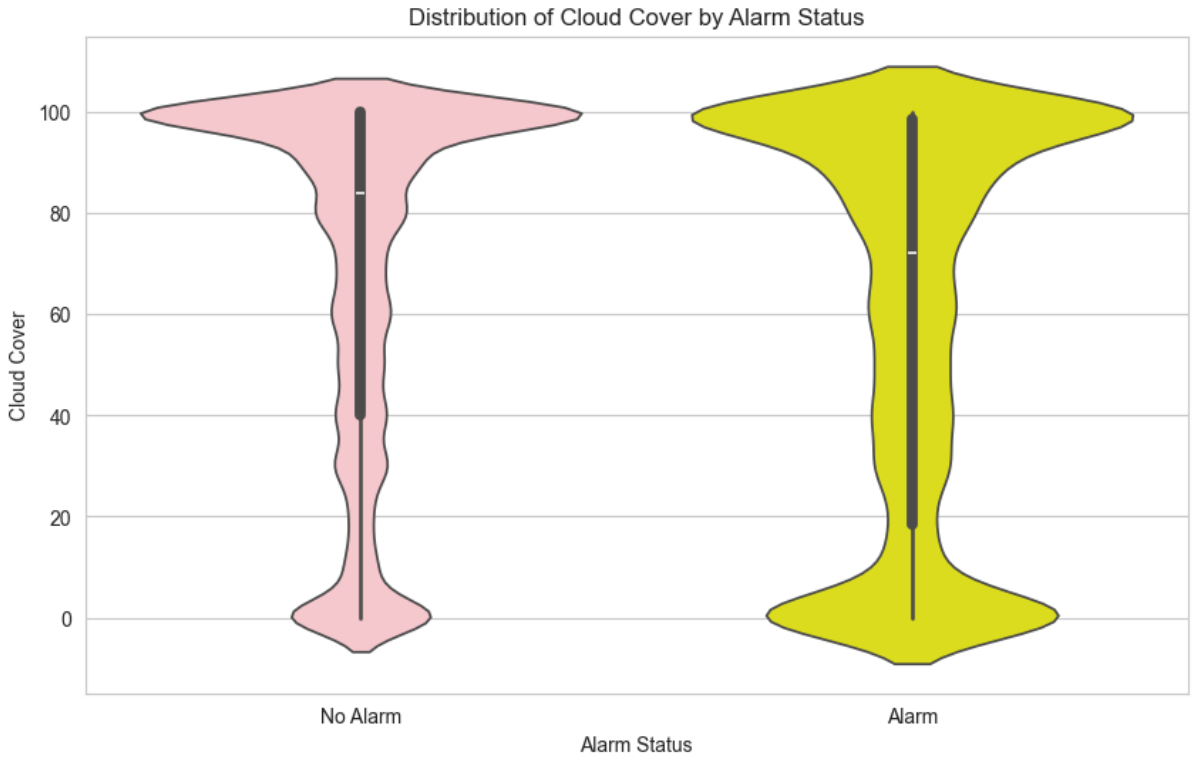
Lower we can observe a general trend of decreasing air alarm durations during the summer on a regional basis. Similarly, we notice that in certain southern and eastern regions of Ukraine, it steadily increased. Such regions are in the minority. There is almost complete similarity with the previous graph on air alarm data. The main changes remain unchanged here as well.



On this graph we see that the median humidity level appears to be higher for the “Alarm” than for the “No Alarm”. This suggests that there is a tendency for humidity levels to be higher when the alarm is on.



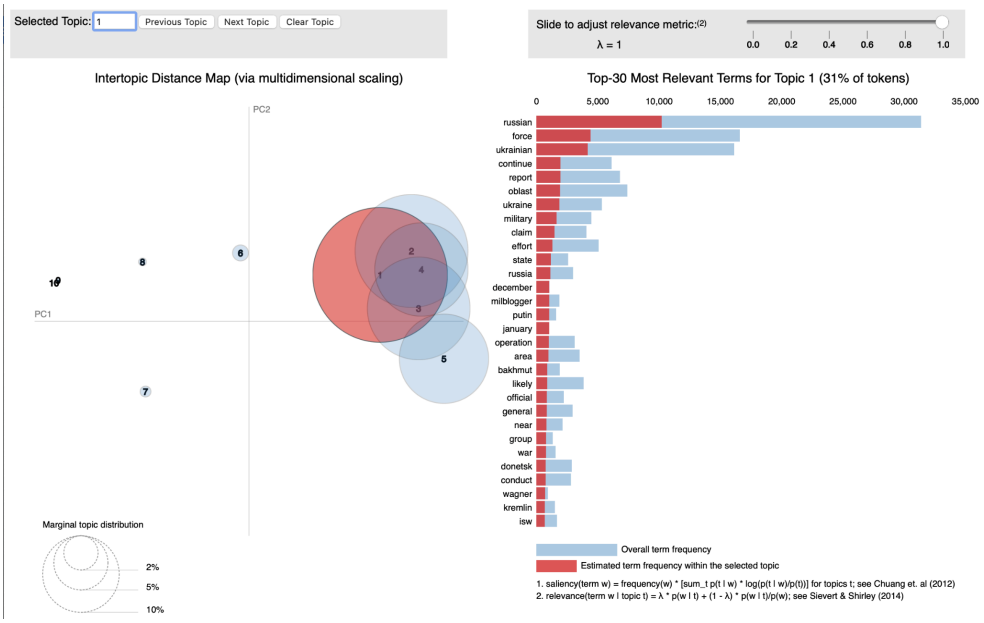
In this graphic we can see that the distribution of cloud cover seems to be similar for both "No Alarm" and "Alarm", so it doesn’t really vary for both states. That means that, surprisingly, this parameter suggests a weak correlation.



Graphic shows that it's not strong correlation between alarm status and cloudcover

ISW_EDA:

We decided to include the use of a model for Latent Dirichlet Allocation. For this part of analysis it was necessary to tokenize data and create a bag of words from our ISW csv. While using LDA, we create n different abstract topics to create relations between attributes in the dataset.
Here you can see an example of visualizing ISW data for topic 1. Lower there will be a full file with visualized LDA for the dataset which anyone is free to download and use.



File is here.

ISW reports, tf-idf vectors;

In order for the model to be able to work with text data, it is necessary to vectorise them. In this project, we experimented with 2 different methods of text vectorisation. LDA and TF-IDF.

TF—IDF, short for Term Frequency-Inverse Document Frequency, is chosen for its ability to weigh the importance of words in a document relative to a corpus. This method calculates a numerical value for each word indicating its significance in a document compared to the entire corpus. In our vectorized dataset, we selected the top 700 words based on their TF-IDF scores, effectively capturing the most relevant terms for analysis.

We can see work of Tf-IDF for all dataset in this wordcloud

LDA, or Latent Dirichlet Allocation, is another method we experimented with for text vectorization. Unlike TF-IDF, LDA is a generative probabilistic model that assigns topics to each document and words to each topic based on their probability distributions. While TF-IDF focuses on the importance of individual words, LDA aims to uncover underlying topics within a corpus, offering a different perspective for text analysis.

As a result after training accuracies of our models we have this results

LDA (100 topics):

Linear — 86.05%

Logistic — 92.95%

TF-IDF(700 words):

Linear — 85.32

Logistic —99.98

<u>Merged dataset</u>

Final dataset have this dimensions: (191120, 735)

700 columns are for TF-IDF

7 Columns for war events

1 Column – region ID

27 – all weather features

```
corr_humidity, _ = stats.pointbiserialr(final_dataset['airstrike'], final_dataset['day_humidity'])
print("Correlation between airstrike and day_humidity:", corr_humidity)
```
Executed at 2024.03.28 20:03:30 in 43ms

```
Correlation between airstrike and day_humidity: -0.23437882985043998
```

```
corr_humidity, _ = stats.pointbiserialr(final_weather_alarms['is_alarm'], final_weather_alarms['day_humidity'])
print("Correlation between is_alarm and day_humidity:", corr_humidity)
```
Executed at 2024.03.27 21:33:38 in 21ms

```
Correlation between is_alarm and day_humidity: -0.13956313160662662
```

Correlation between word airstrike and humidity is similar to correlation between humidity and column is_alarm

Prepare Data

Weather+Alarms

1. Feature Engineering (generate new features):
   We added some new features to dataset with weather and alarm status:
   num_of_reg_alarm - this column shows number of regions where are air alarm in the one time row,
   num_of_alarms24 - shows number of regions with alarms for 24 hours,
   light_day_duration - shows how long light day was,
   event_duration - shows how long alarm was,
   seq_num_of_alarm - shows sequence number of alarms for each day.

ISW

1. NLP

Talking about data preparation, firstly we used a column with extracted text from our original ISW dataset from Hometask #2. The data from here went through all cleaning steps: lowering, cleaning from punctuation signs, any symbols, dates, numbers, etc. Then we deleted all stopwords, after which data went through 2 types of lemmatizing models.
We tried to test the data on Stemming and Lemmatizing models, and then decided to stop our choice on the second one.
Pre-final part was filtering by frequency and importance. We left approximately 700 words in the dataset out of 5000 original columns.

Final part of this step was a vectorization of the model and creating a new dataset with vectorized data only.

2. PCA

Will be added later.

Merged dataset

On March 28th we added the encoding of region_id for the final dataset. As it was a last-minute update, we decided to keep in the report both results for final dataset and for final dataset with region encoding.

Separate data: train & test

Which of the 2 suggested approaches did we use?

In our analysis, we decided to use the approach of splitting the data into time series. This decision was made after considering two proposed approaches, namely time series splitting and an 80-20 split. Considering the nature of our data, which exhibits temporal dependencies and sequential patterns, the time series splitting method actually yielded better results in terms of accuracy. By employing time series splitting, we aim to avoid potential issues and effectively evaluate the accuracy of our models' predictions. Additionally, for training one of the models (Logistic Regression), we set the value of n_splits to 5, so that during each training iteration, the data used for training the model does not overlap.

Train models

Linear Regression -
Which of the 2 suggested approaches did we use?
To evaluate this model we used hyperparameters
`positive=True,` (force model to produce only positive predictions)
`fit_intercept=False` (determines whether to calculate the intercept)

After evaluating we got such results for metrics:

With region_id encoding

```
accuracy: 85.32%
MSE: 0.03
```

without:

```
accuracy: 87.48%
MSE: 0.02
```

Model coefficients:

```
import joblib

model_linear = joblib.load("linear_regression_model_test.pkl")
```

```
print(model_linear.coef_)
```

```
[0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 1.48703717e-04 0.00000000e+00 1.24551323e-03
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 7.57009605e-01 2.36354012e-05 4.40226691e-03
 6.92253475e-03 0.00000000e+00 5.14437970e-02 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
```

Logistic Regression - 98-99% of accuracy

Which of the 2 suggested approaches did we use?

Firstly we tried to train several models using 80 to 20 split. With this approach we have received next accuracy results:

```
/Users/tsaebst/pythonProject3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:469:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

Accuracy: 0.8950136040184178
Confusion Matrix:
[[27457  1188]
 [ 2825  6754]]
```

It was the average accuracy.

Then we changed the split to the Time Series Split with value 5. We have divided our main dataset into 2 parts - one for model training and one for accuracy assessment.

Here's the results of training:

Model coefficients

```
import joblib

model_log = joblib.load("logistic_regression_model.pkl")
```

```
print(model_log.coef_)
```

```
[[-2.90810464e-01  2.61573577e-01  1.05150495e+00 -1.34213141e+00
   3.29305807e-01 -2.49575694e-03  4.09811859e-02  4.18183979e-02
  -2.33970913e-01 -2.82044493e-02  3.92269678e-01  1.90706764e-02
   5.08296007e-02 -2.39504705e-02 -5.34533641e-02 -1.79270650e+00
  -7.60198214e-01  9.11925229e-02  7.91136600e-02 -1.54641640e-02
  -1.04674730e-04 -1.41029102e-02  2.99683757e-03  4.08113001e-03
   4.52877102e-03 -3.34768007e+00  3.16721334e-01 -6.19411564e-02
  -5.91282110e-02  2.00069814e+01  2.77768606e-02  1.24994308e-01
   1.00125102e-01 -1.38617197e+00  3.85673365e+02 -1.45371603e+00
  -5.28393155e-01  4.67972043e-01 -1.25870360e+00  3.71018998e+00
  -7.12241813e-01  2.63397013e-01  2.43020058e+00  3.07866398e+00
  -2.69625216e+00  5.48794956e-01  2.65565777e+00 -2.25181833e+00
```

Evaluation:

Metrics of Logistic Regression Model without region_id Encoding:

```
Accuracy: 99.98%
Precision: 1.00
Recall: 1.00
MSE: 0.00

Process finished with exit code 0
```

Evaluate models

1) **LogisticRegression**

Before using hyperparameters, model gave us such accuracy:

```
Середня точність моделі: 99.86%

Process finished with exit code 0
```

then we tried to evaluate it using GridSearchCV, but it took too much time and our hardware can't handle this, so best hyperparameters were identified by numerous testing and analysis. To accelerate search process, we tested only with 30000 of data raws and got such results:

```
# clf = LogisticRegression(max_iter=30000) 98.19
# clf = LogisticRegression(max_iter=30000, solver='lbfgs') 98.19
# clf = LogisticRegression(max_iter=30000, solver='lbfgs', penalty='l2') #98.19
# clf = LogisticRegression(max_iter=50000, tol=0.001) 98.01
# clf = LogisticRegression(max_iter=50000, solver='newton-cg') 98.16
# clf = LogisticRegression(max_iter=50000, class_weight='balanced') 98.47%
# clf = LogisticRegression(max_iter=50000, class_weight='balanced', dual=False) 98.47
# clf = LogisticRegression(max_iter=50000, class_weight='balanced', random_state=42) 98.47
# clf = LogisticRegression(max_iter=50000, class_weight='balanced', fit_intercept=True) 98.47
# clf = LogisticRegression(max_iter=50000, class_weight='balanced', fit_intercept=False) 98.45
# clf = LogisticRegression(max_iter=50000, class_weight='balanced', solver='liblinear') 98.34
# clf = LogisticRegression(max_iter=50000, class_weight='balanced', solver='sag') 87.01
# clf = LogisticRegression(max_iter=50000, class_weight='balanced', C=np.logspace(-4, 4, 2)[1]) 98.95
# clf = LogisticRegression(max_iter=50000, class_weight='balanced', solver='newton-cg', C=np.logspace(-4, 4, 2)[1],
                                                                                    # penalty='l2') 99.36
# random_state не впливає dual не впливає
# clf = LogisticRegression(max_iter=50000, class_weight='balanced', solver='newton-cg', C=np.logspace(-4, 4, 2)[1],
                                                                          # penalty='l2', tol=0.000001) 99.45
```

So, with such hyperparameters, model accuracy was improved from 98.19% to 99.45.

The best parameters set for our model is:

`max_iter=50000,`

`class_weight='balanced',` (is used to handle imbalanced class distribution in dataset)

`solver='newton-cg',` (optimization algorithm to find the parameters of logistic regression)

`C=np.logspace(-4, 4, 2)[1],` (used to prevent overfitting by adding a penalty term to the loss function)

`penalty='l2',` (encourages smaller weights)

`tol=0.000001` (stopping criterion for the optimization algorithm)

And as a result accuracy of a model, trained on full dataset was improved from 99.86% to 99.98

With encoding:

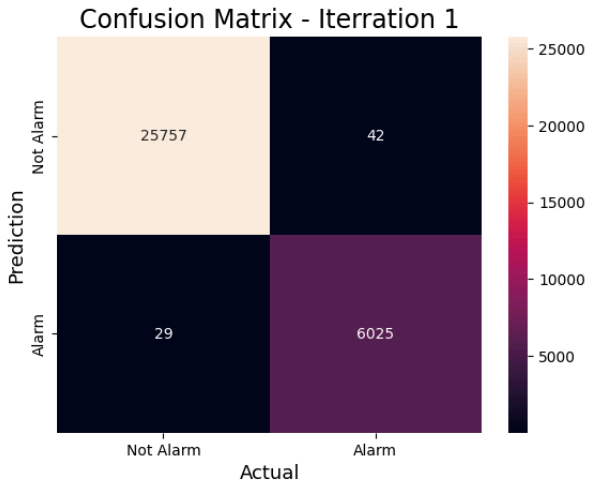
```
Accuracy: 99.98%
Precision: 1.00
Recall: 1.00
MSE: 0.00
R-squared: 1.00
```

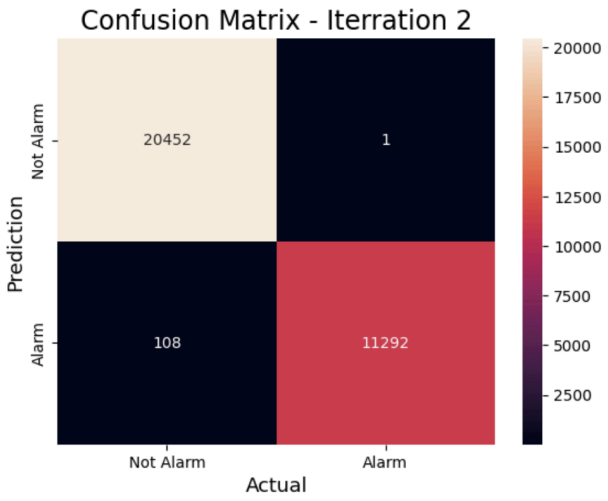
without:

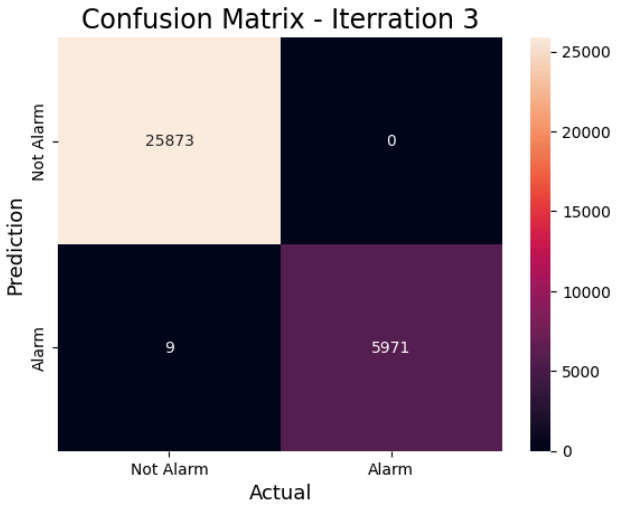
```
Accuracy: 99.98%
Precision: 1.00
Recall: 1.00
MSE: 0.00

Process finished with exit code 0
```

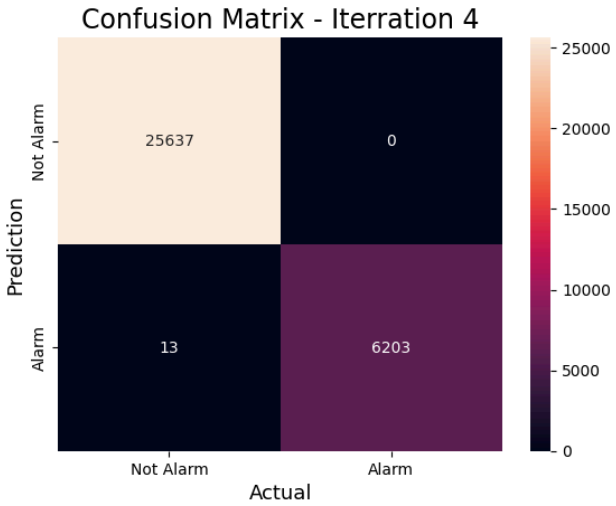
a. Calculate confusion matrices for the tuned models. Explain what do metrics mean?

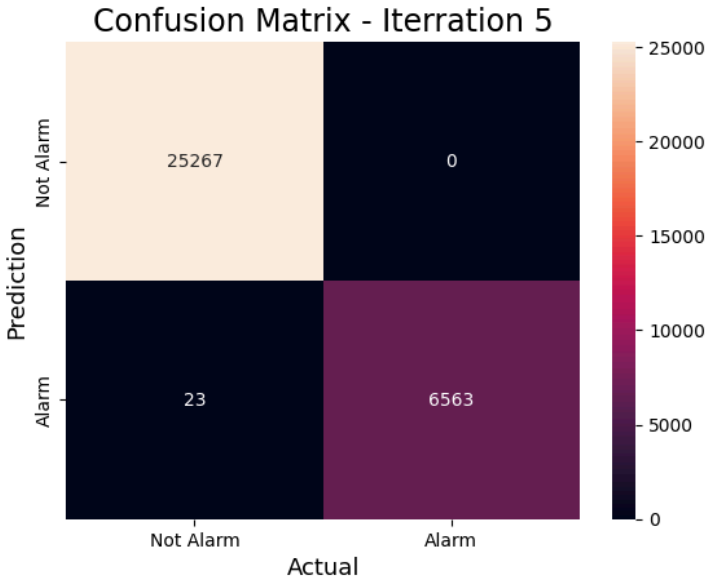
We built Confusion Matrix to have a visual representation of how our model predicts alarms. Here you can see 5 different results for each iteration.



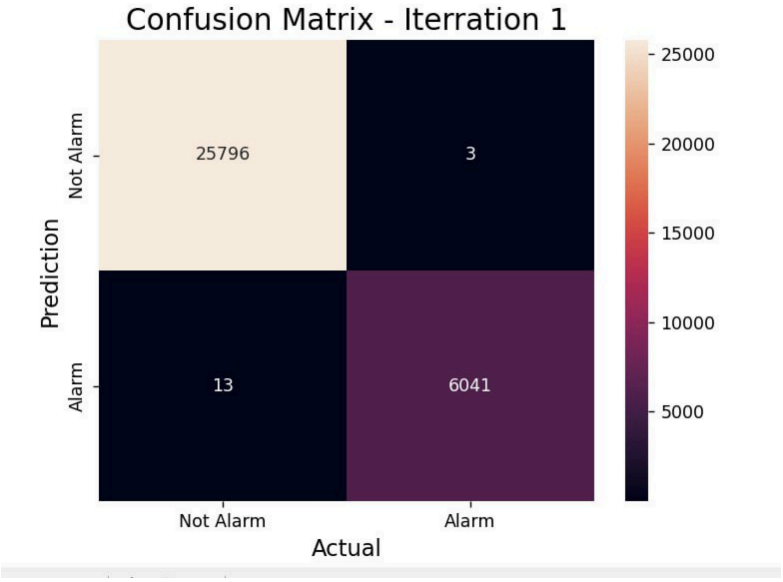








They’re made via a dataset without regions_id encoding.
After the encoding we got other CMs. We’ll attach a screenshot of Confusion Matrix for Logistic Regression model.



You are able to check out all MS collection here: [Confusion Matrix for dataset with region_id encoding.]

Talking about metrics - that is another tool that could be handy in analyzing the accuracy and principles of work of models from different angles.
In our work we use such metrics as Precision, Recalls, Accuracy, MSE, and R-squared.

Accuracy is a ratio of correct predictions to total predictions. For us it’s the most straight-forward and intuitively understandable explanation of model success.

Whereas precision is a ratio of true positives to all positives, that is sum of true positives and false positives. When saying true positive, we mean items which are correctly labeled as positive.

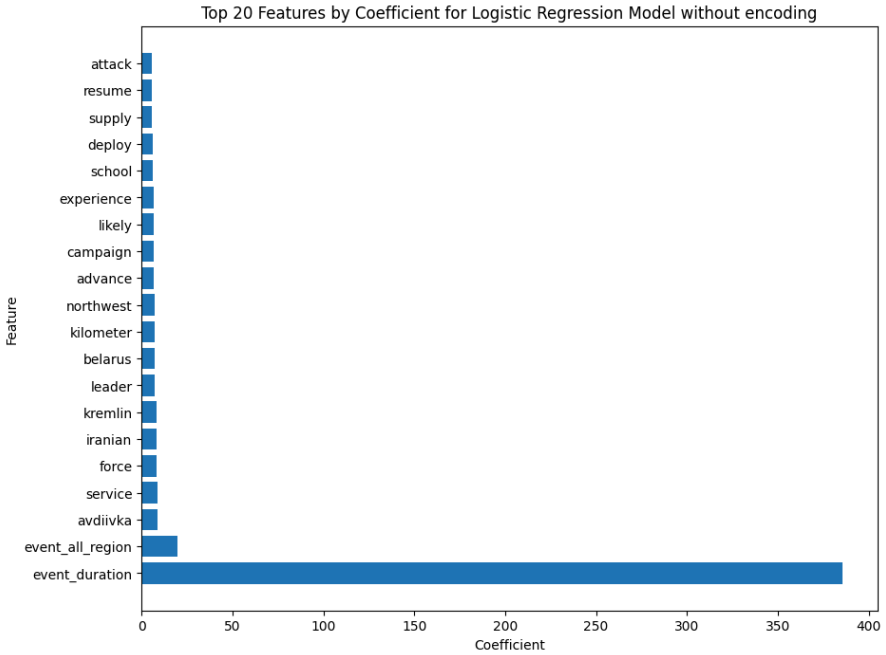
Talking about recalls, it measures the fraction of positives that were correctly marked as so in a classification part. That is the ratio of true positives to the sum of true and false negatives. So here we have the same numerator, while the denominator is diametrically opposite.
These are the most basic metrics, which we obviously decided to include.

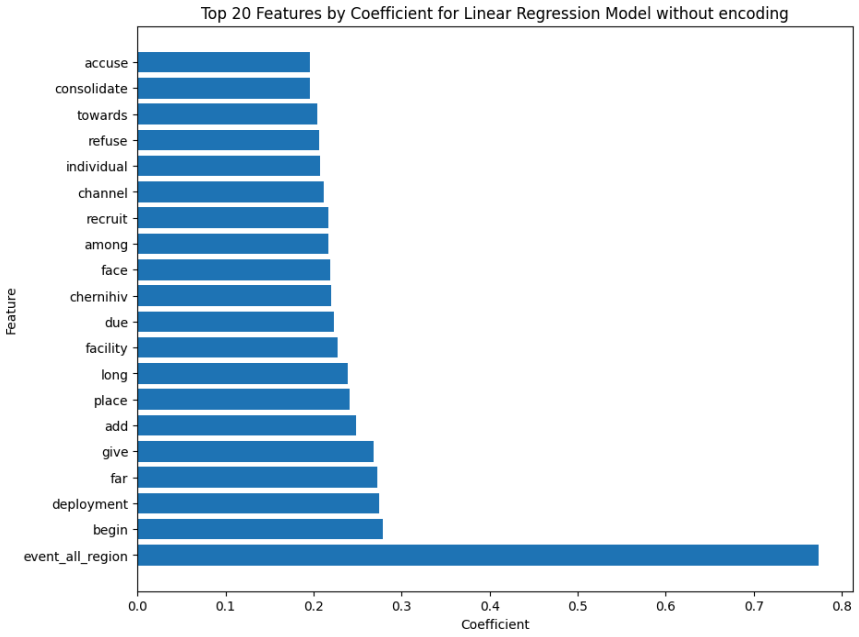
Other interesting metrics are MSE and RS.
The last one is a proportion of variance of y that has been explained by the independent variables in the model. As it was explained in the article we used to get more general information about the metrics, this metric provides an indicator of quality (how good is it) of fit and a measure of how well samples that are not seen could be predicted.

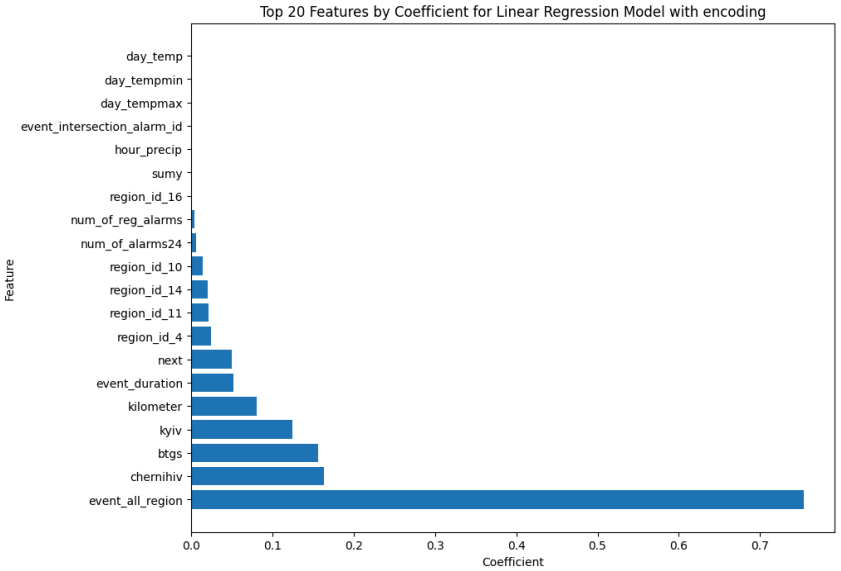
Lastly, MSE describes mean squared differences between actual and predicted values. Thanks to squaring the difference, the impact and "visibility" of big errors on the overall error metric is much more distinguishable. So that model's mistake will be more lethal if it differs significantly from the original data.

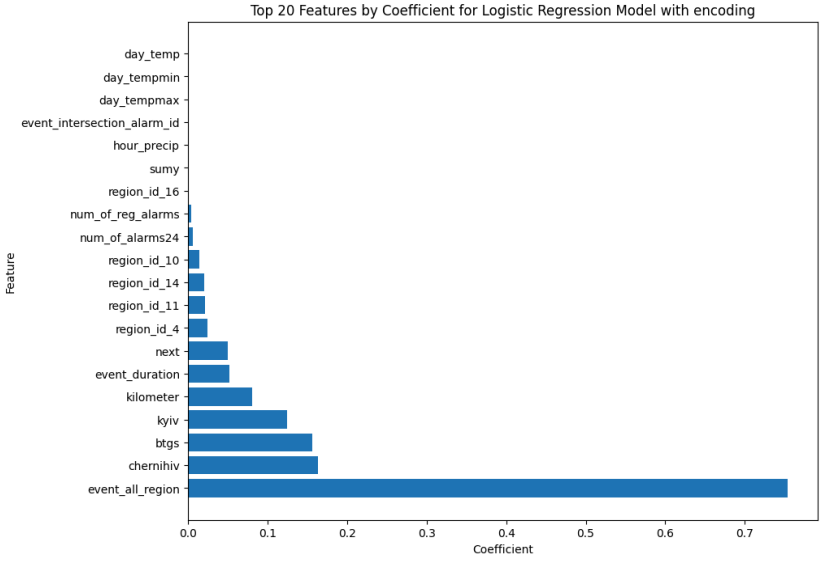
b. What are the top 20 features for each of your models? Draw them and their weights.

Here are some graphs visualizing top-20 features for several tasks:




Top 20 Features by Coefficient for Linear Regression Model without encoding
accuse
consolidate
towards
refuse
individual
channel
recruit
among
face
chernihiv
due
facility
long
place
add
give
far
deployment
begin
event_all_region
Feature
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Coefficient


Top 20 Features by Coefficient for Linear Regression Model with encoding
day_temp
day_tempmin
day_tempmax
event_intersection_alarm_id
hour_precip
sumy
region_id_16
num_of_reg_alarms
num_of_alarms24
region_id_10
region_id_14
region_id_11
region_id_4
next
event_duration
kilometer
kyiv
btgs
chernihiv
event_all_region
Feature
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Coefficient


Top 20 Features by Coefficient for Logistic Regression Model with encoding
day_temp
day_tempmin
day_tempmax
event_intersection_alarm_id
hour_precip
sumy
region_id_16
num_of_reg_alarms
num_of_alarms24
region_id_10
region_id_14
region_id_11
region_id_4
next
event_duration
kilometer
kyiv
btgs
chernihiv
event_all_region
Feature
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Coefficient

- Write what you find the hardest part of the work in a few sentences.

According to the poll in our group, the hardest parts probably were during NLP understanding and implementation, choosing the vectorisation approach, merging data from different datasets and model improvement through features addition Of course, model training was the most time consuming part. However, it wasn't the hardest part.

A few words about work distribution:

EDA for Weather was done by Katerina, Vladyslav and Vladyslava. EDA for Alarms was Matviy's part of work. ISW EDA was Ivan's part of work, so was ISW reports, tf-idf vectors as Ivan was the person who did the prevailing part of TF-IDF. Vladyslava added description to the graphs and all EDA parts.

Data preparation was divided between Matviy, Vladyslava and Ivan. Vladyslava and Ivan did the NLP part, Matviy and Ivan vectorized it and tried LDA. We decided to follow up our dataset with PCA in the nearest future, to decrease the dimensionality with saved relations. It will make the time complexity of model training even less than now.

Vladyslava did the encoding for region_id, after which we received 2 final datasets (with and without One Hot Encoding).

Features were added to the weather dataset by Katerina, Vladyslav and Ivan. Vladyslava made One Hot Encoding for region_id column in the final dataset.

Matviy, Vladyslava and Ivan trained Linear and Logistic Regression models. During this stage we were experimenting with the approaches to training in our group, so we discussed the most fitting (hyper)parameters and updated our testing with various metrics.

Model evaluation was a part of Ivan and Matviy's work. Confusion matrix was done by Vladyslava, so were descriptions to these parts of the report.

Link to our GitHub repo:

https://github.com/synthco/4-pythonic-squad

Additionally, some useful links:

https://www.kaggle.com/code/funxexcel/p2-logistic-regression-hyperparameter-tuning

https://medium.com/@hckecommerce/comprehensive-text-preprocessing-nlp-natural-language-processing-fe295978523e

https://ayselaydin.medium.com/4-bag-of-words-model-in-nlp-434cb38cdd1b

https://medium.com/@yashj302/lemmatization-f134b3089429

https://ayselaydin.medium.com/2-stemming-lemmatization-in-nlp-text-preprocessing-techniques-adfe4d84ceee

https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

https://medium.com/@corymaklin/latent-dirichlet-allocation-dfcea0b1fddc

https://neptune.ai/blog/exploratory-data-analysis-natural-language-processing-tools

https://www.kaggle.com/code/funxexcel/p2-logistic-regression-hyperparameter-tuning