KU LEUVEN

Advanced Analytics in Business: D0S07a

# Advanced Analytics in Business: Lab Report

**Group 31:**
Aksoy, Barış, r0869901
Hegde, Medha, r0872802
Heller, Jack, r0862809
Zdravković, Aleksandra, r0869484

Leuven. May, 2022

# Assignment 1: Predictive Modeling on Tabular Data

## Problem Definition

Our goal in assignment 1 is to identify the most likely customers who will leave the bank. This is done using a variety of features that include information about the accounts of customers and their balances, as well as the activity of their accounts and any loans they may have with the bank and background information on the customer (education, family, etc.). This data is also provided for the trailing three months so that changes in patterns can be observed.

Due to the limited resources of the bank, the 250 clients that are predicted as most likely to churn will be contacted by the bank in an attempt to retain them. This means that our goal is not to effectively predict which customers will not churn as it is to find which customers will churn. Being at a large bank, customers will leave every month. However, we need to make sure the staff's time is used most effectively which means not wasting their time with customers who will not leave. This means that a measure like accuracy or AUC would not be appropriate. Instead, we should count the number of true positives in the top 250. This can be done using precision.

## Exploratory Data Analysis

We start by investigating what data we have and how it can be transformed into more valuable features. For a full description of the variables please visit:
http://seppe.net/aa/assignment1/. Immediately we can see many binary variables describing the customer's relationship with the bank. These will be very useful not on their own but when combined with trailing data. For example, if a customer was very active and then stopped using the account, this is a bad sign. Also if a customer has recently paid off a loan or cancelled insurance, they may be preparing to leave.

Other variables have a different story to tell. For example, the variables, 'customer_since_bank' is given as a year and month. This information is very useful as a customer who has been with the bank for significant time is less likely to leave. However, this needs to be converted to time as customer in either months or years to allow an easier interpretation of this decaying likelihood.

We have another set of variables that need to be featurized in another way. These are categorical variables where some are ordinal and some are not. For example, occupation codes are not ordinal. However, some are certainly more likely to churn than others. This is true of categories such as customer children as well. We also have some information on the customer's education level.

We also have information on the customer's postal codes. This poses an interesting challenge as some postal codes appear to have a much higher likelihood of churning than others. However, some post codes have so few customers that it is not completely reliable and creates a very high dimensional problem as there are many post codes in Belgium. Fortunately, these 4 digit numbers make up larger areas so we are able to easily aggregate these codes into larger areas by only using the first two numbers. For a visualization of this, see the map of 2 digit postal codes below.

When investigating the target data, is it important to note that this data is extremely unbalanced. With only about 3% of the whole dataset being churning

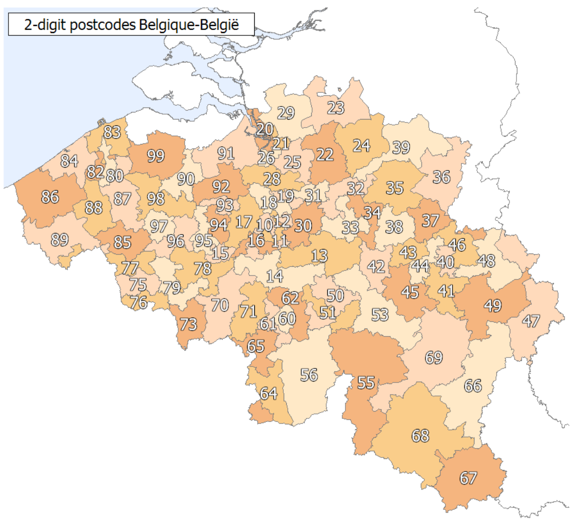


Figure 1: Belgian Two Digit Post Code Map

clients.

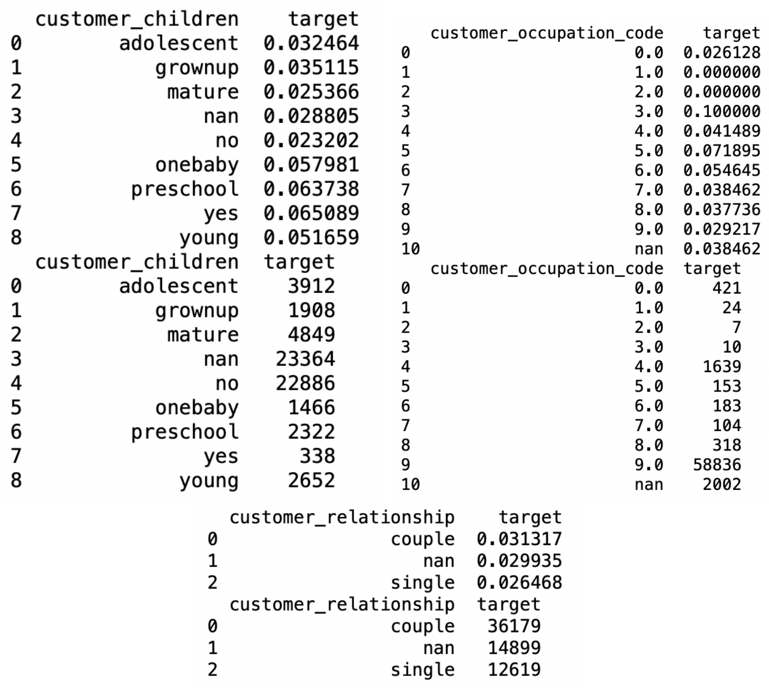
One challenge we did have is that not all customers fill in every field and we have incomplete information about every customer. Removing these customers from the dataset is not a good option as this will continue to be a problem in the test set. However, we can investigate how these missing data points impact their likelihood to churn.

We start by evaluating how a customers lack of information affects their likelihood of churning (Figure 2). We can see that while churning is pretty similar among all education levels with the exception of 6, customers who did not fill in their information are much less likely to churn. We can also see that the vast majority of customers did not include their education level information.

We can see that when we look at occupation codes, almost all of our customers have occupation 9. While evaluating solely on percentage churning by occupation it shows that some don't churn. However, with so few observations, it is difficult to say definitively that they really are different.

When doing the same evaluation for children, we see that customers with one baby, preschool children, young children, and those who simply answered yes are much more likely to churn. They are also a small population compared to the overall population. On the other side, those who did not answer or answered no make up almost all of the customers and have very similar likelihoods of churning.

We do one more analysis in this fashion on customer relationships. Here we see that being in a relationship appears to have very little effect on if they will churn and not answering is also not a strong indicator.



```
  customer_children    target
0        adolescent  0.032464
1           grownup  0.035115
2            mature  0.025366
3               nan  0.028805
4                no  0.023202
5           onebaby  0.057981
6         preschool  0.063738
7               yes  0.065089
8             young  0.051659
  customer_children  target
0        adolescent    3912
1           grownup    1908
2            mature    4849
3               nan   23364
4                no   22886
5           onebaby    1466
6         preschool    2322
7               yes     338
8             young    2652
```

```
   customer_occupation_code    target
0                       0.0  0.026128
1                       1.0  0.000000
2                       2.0  0.000000
3                       3.0  0.100000
4                       4.0  0.041489
5                       5.0  0.071895
6                       6.0  0.054645
7                       7.0  0.038462
8                       8.0  0.037736
9                       9.0  0.029217
10                      nan  0.038462
   customer_occupation_code  target
0                       0.0     421
1                       1.0      24
2                       2.0       7
3                       3.0      10
4                       4.0    1639
5                       5.0     153
6                       6.0     183
7                       7.0     104
8                       8.0     318
9                       9.0   58836
10                      nan    2002
```

```
  customer_relationship    target
0                couple  0.031317
1                   nan  0.029935
2                single  0.026468
  customer_relationship  target
0                couple   36179
1                   nan   14899
2                single   12619
```

Figure 2: Probabilities of Churning by Category

## Featurization

Here we featurize the data as described. We:

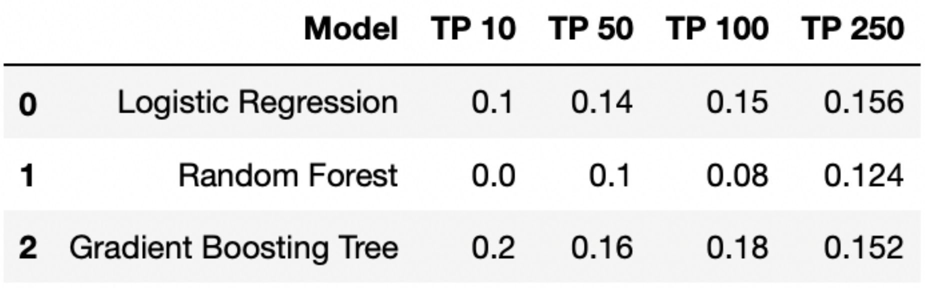
- Remove Client ID
- Create a feature to identify if they were homebanking 3 months ago and stopped
- If their savings account balance changed
- If the customer has a current checking account
- If there was a change in the customer's loan or mortgage status
- Postal code is dropped as it will be added back in later
- We use a median imputation for how long the customer has been with the bank as this data is rarely NA.

We next work to deal with the class imbalance problem. We will do this by using random oversampling. We do this at this point as to make sure we have a validation set to evaluate on for tuning and we only create observation training reliant features on the training set. We found that oversampling was a key step in this process.

We now scale all the training data and perform the same transform on our validation set.

## Modeling

Here we can see that at every level, gradient boosting trees performed the best on the validation set. For this reason, it appears to be the best model to proceed with. We can also see the precision as various amounts of observations taken. For gradient boosting, it may be better to take less than the top 250, the precision is much better for the top 100. Of course, more churners were found in the top 250 but the benefit of pursuing fewer clients is that less resources and investment is required and there is a greater ROI.



|  | Model | TP 10 | TP 50 | TP 100 | TP 250 |
|---|---|---|---|---|---|
| 0 | Logistic Regression | 0.1 | 0.14 | 0.15 | 0.156 |
| 1 | Random Forest | 0.0 | 0.1 | 0.08 | 0.124 |
| 2 | Gradient Boosting Tree | 0.2 | 0.16 | 0.18 | 0.152 |

Figure 3: Model Performances at Different Number of Candidates Included

## Reflection

When we see the second half of the test set, we had 35 true positives out of the top 250. This is encouraging performance after seeing 36 true positives on the first test set. We had 39 true positives on the validation set so while we did have some degradation in performance, it was acceptable. We were very careful to not continuously submit models to the test set but we instead performed tuning and different approaches on the validation set. This is why we saw a similar performance.

While the definition of the target variable is not bad the way it is, it may be helpful to we see from our validation set that the true positive rate is better when only taking the top 100 candidates. Our recommendation would be that the bank perform a cost benefit analysis to evaluate if the additional resources required to contact 150 more people is worth the payoff of finding the extra people. This could be done through a breakeven analysis of what is an acceptable true positive rate and then finding a good threshold.

# Assignment 2: Predicting Location of Swimming Pools in Satellite Images

## Problem Definition

Our data set consists of 14,912 satellite images of swimming pools in the south of France. All images are the same size (512x512) and the pools are located in the centre of every image. All images are assumed to contain only one swimming pool. The metadata.json file contains the bounding polygon's coordinates. We aim to use this data to predict the location of the pools using convolutional neural networks (CNNs).

The swimming pools were of varying shapes and so we explored predicting the following:

1. the smallest upright bounding box containing the pool (Object Localisation)
2. the polygon shape of the pool (Semantic Segmentation)

## Preprocessing

### Metadata file

The metadata file containing the bounding polygon coordinates was used to extract:

1. the smallest upright bounding rectangle's coordinates: The openCV function *boundingRect* was used for this purpose. We divided each coordinate by 4 since the input images will be later resized to 128x128 from their original size of 512x512.
2. the grayscale masks of each image containing the polygon area: The openCV function *fillConvexPoly* was used to obtain an image containing only the polygon of the pool (filled in black) and the remaining space filled in white. These images were saved locally to be used as masks for the second model.

   Both our models can handle images of different sizes since we are resizing them during preprocessing.

## I. Object Localisation

The purpose of object localisation is to locate the tightest rectangular bounding box centred on the swimming pool in the images. Using the images and the bounding box coordinates, we will train a ResNet50 CNN model to be able to predict these coordinates.

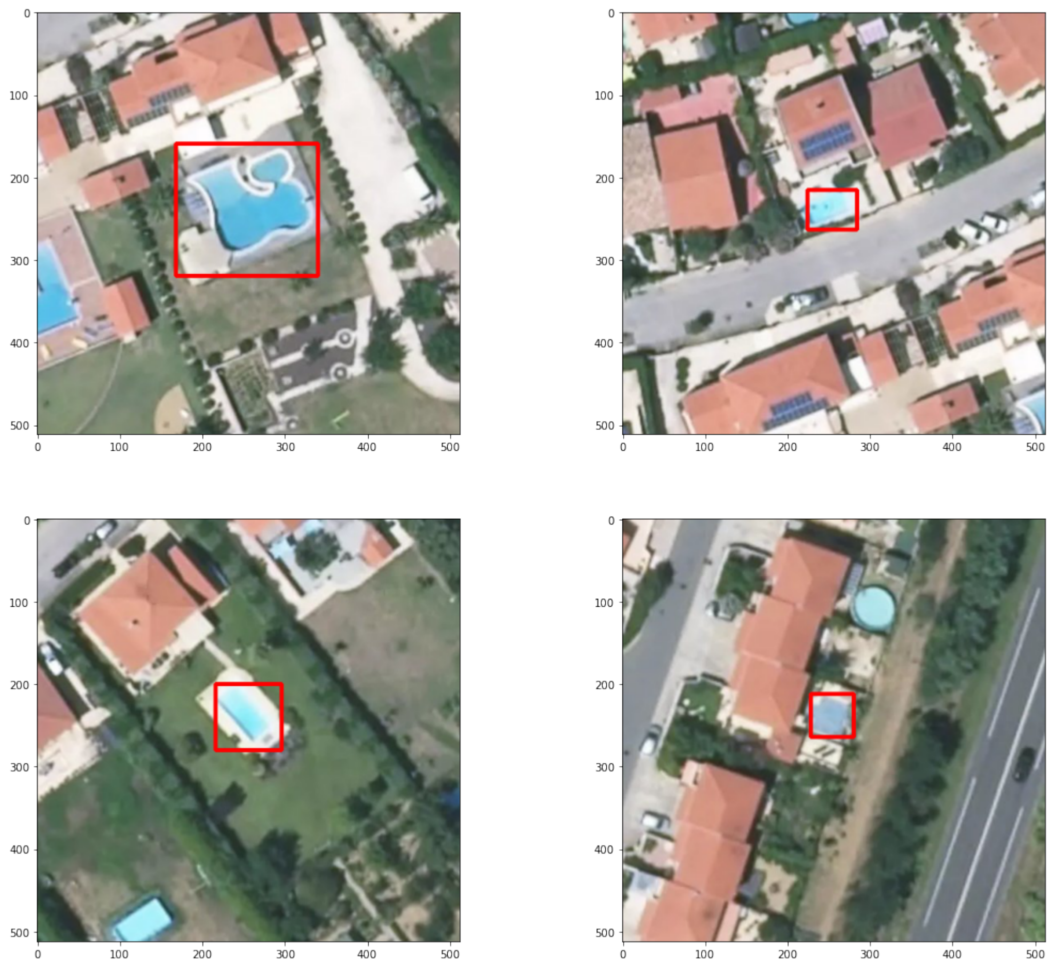


Figure 4: Visualising the Bounding Box

### Feature Extraction

The keras image preprocessing utilites were used to load images into PIL format and then convert this into Numpy array. This was then passed through the ResNet50 *preprocess_input* function which converts the input images from RGB to BGR, and then zero-centers each color channel with respect to the ImageNet dataset, without scaling. The resulting output for each image is a 128x128 matrix with the zero-centered BGR coordinates for each pixel. This is the input format required for our model.

We then split our dataset (image data and their corresponding bounding box outputs) in 80-10-10 split for training, validation and testing, respectively.

### Model Definition: ResNet50 Architecture

We created a custom model using the ResNet50 model which is a 50-layer convolutional neural network pretrained on more than a million images from ImageNet database. The base layers were frozen so that the ImageNet weights were not retrained. The final pooling and the fully connected top layers were not included so that we could add our own final layers - these are the only layers which will be trained on our dataset.

The pre-trained model also contains Batch Normalization layers which help prevent overfitting.

We added a Flatten layer which converts the inputted layer into a one dimensional layer. This is then inputted into a final Dense Layer with 4 nodes corresponding to the 4 coordinates of the bounding box. A ReLu (Rectified Linear Unit) activation is applied since our required output conists of non-negative continuous numbers and the ReLu activation function does this by deactivating neurons whose output is less than zero. The final model has 2,712,964 parameters, of which only 262,148 will be trained.

## Model Configuration

**Loss function** A loss function is used to compare the model's predicted values to the ground truth values of the bounding box coordinates. We will use the Mean Squared Error for this purpose.

**Optimizer** The *Adam* optimization algorithm is used to update the network's weights at every epoch in order to "guide" the model towards minimizing the loss, here the MSE.

## Model Training

The model is then trained on our training data set and is also validated at every step using our validation data set. We have trained the model for 20 epochs and used a batch size of 500 since we found that this combination works most efficiently with our current capacities of time and computing power.

From the output of the training, we see that both the training and validation losses reduce after each epoch. The final loss (MSE) for the training set computed at the end of 20 epochs is 2.62 from an initial loss of 476.33 at the first epoch. The final validation loss is 3.87.

## Visualizing Training

Both the training and validation losses drop fairly quickly by the 2nd epoch with very small decreases towards the minimum in the subsequent epochs, indicating that the model learns quickly and is able to predict the bounding box for each image in the training and validation data sets well.

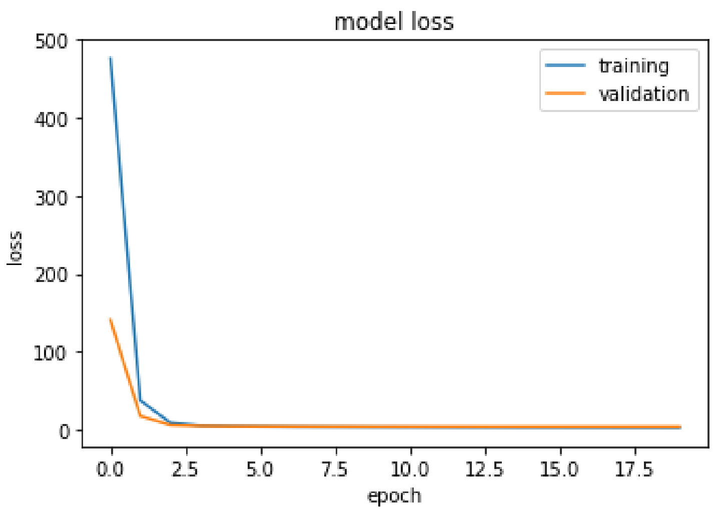


Figure 5: ResNet50 Training Process

## Model Evaluation

The model was evaluated on the test data set using 2 metrics:

1. Mean Squared Error: The MSE was found to be **4.02**, which is not far from the model's 2.62 on the training data. This means that the average squared difference between the observed and predicted values for all 4 coordinates of the bounding box is 4.02 pixels.

2. Intersection Over Union: The IOU is used to measure the overlap between the ground truth and the predicted bounding boxes. It is calculated as the region of overlap divided by the combined region covered by the bounding boxes. The closer this number is to 1, the more accurate our model is at predicting the correct bounding box coordinates. For our model's prediction on the test data set, the average IOU is **0.86**.

## Visualizing Predictions

Below, we visualize the model's prediction on a random image from the test data set. The green bounding box on the right is the predicted output of our model.

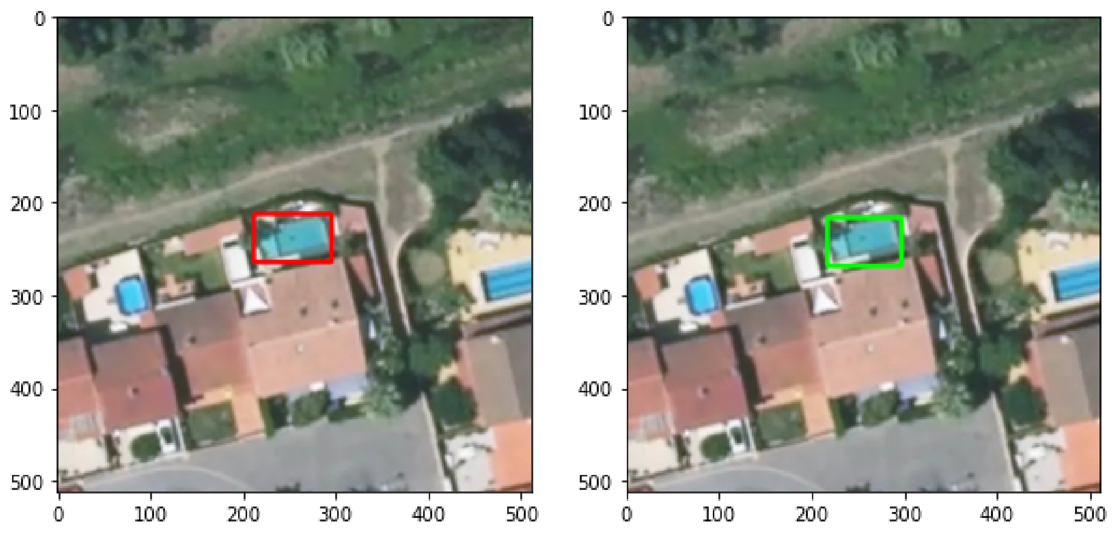


Figure 6: ResNet50 Prediction (Red: Ground Truth, Green: Prediction)

## II. Semantic Segmentation

Semantic segmentation is the process of pixel-wise prediction where each pixel is classified as either belonging to the object (pool) or not. The inputs to the model will be similar to those of the previous model. The output, however is no longer the 4 bounding box coordinates. Instead, the output is a Numpy array representation of an image where each cell in the matrix represents a pixel with 1 if it belongs to the bounding polygon (pool) and 0 if it does not.

### Preprocessing

The input images are preprocessed in a slightly different manner than for the first model. The image is first resized to 128x128 and then the RGB coordinates are normalized by dividing by 255.

The output (masks) are the same dimensions as the input image (128x128), but each pixel only contains one value instead of 3 values for the RGB coordinates, since the masks are in gray scale.

**What is a *mask*?** A mask is an image that is modified such that pixels belonging to the same object have the same color. They are the target output of the segmentation model since they represent the area of the image that we are trying to identify. For our problem, since we have only one image, we only have one colour (white) demarcating the area of the pool in the image.

In the images below, we see the original images and their corresponding masks:

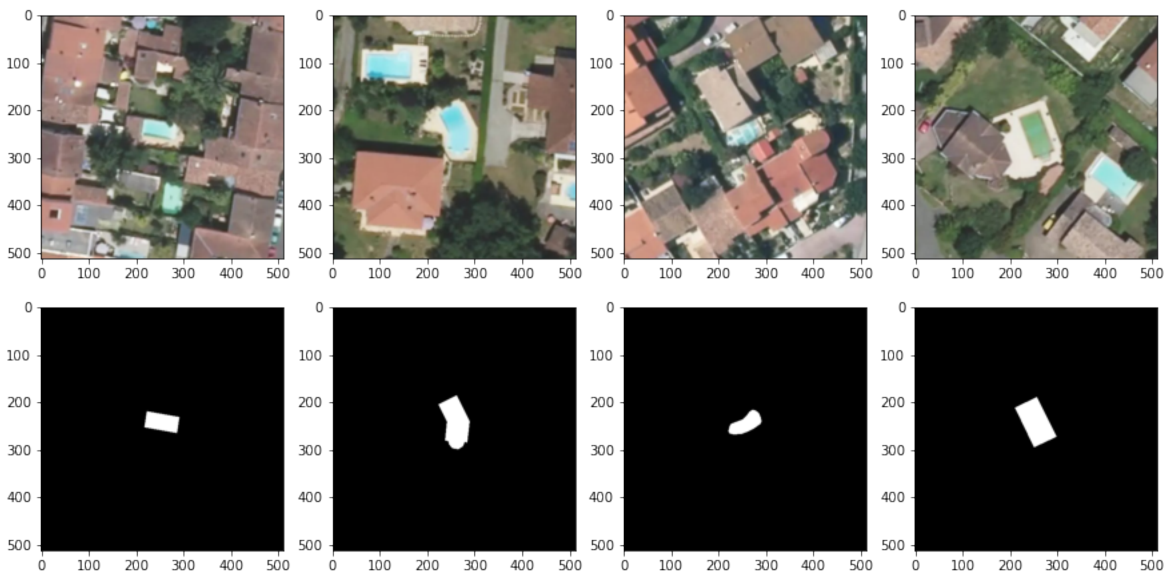


Figure 7: Visualising Masks

## Model Definition: U-Net Architecture with MobileNetV2

The U-Net model is a CNN architecture that is designed for fast and precise segmentation of images. It consists of a contraction path (encoder) and expansive path (decoder). The former uses traditional convolutional layers while the latter uses symmetric transposed convolutions which performs upsampling, a method to increase the "resolution" of an inputted image. The U-net is a Fully Convolutional Network (FCN).

We are using the MobileNetV2 architecture for the contraction path of the U-Net model since it is pre-trained and will help the final model converge much faster. We then replace certain layers with Upsampling and Convolutional layers to form the second half of the U-Net.

It should be noted that simialar to the ResNet50 model, the MobileNetV2 model has Batch Normalization layers after every Convolutional layer. This reduces overfitting due to its slight regularization effect.

## Model Configuration

**Loss and Metrics Functions** We use the complement of the Sørensen–Dice coefficient as the loss function to be minimized during training.

We are also using the Sørensen–Dice coefficient as a metric to evaluate the model. It works in a similar way to the Intersection over Union metric but is more often used for image segmentation models. It is a measure of spatial overlap that ranges form 0 to 1, where 1 indicates complete overlap between ground truth and predicted polygons. It is calculated as 2 times the area of intersection divided by the sum of their areas.

We are also looking at Recall and Precision as metrics. In image segmentation, a pixel is classified as "positive" if it belongs to the object and as "negative" if it does not. Recall measures the proportion of actual positives that were identified correctly. Precision, on the other hand, measures the proportion of predicted positives that were actually correct.

**Optimizer** We are using the NAdam algorithm as the optimizer for the model. It is essentially the Adam algorithm with Nestrov momentum.

**Callbacks** We are using the Keras callbacks API to:

1. Reduce learning rate by a factor of 0.1 when the Dice Coefficient loss has stopped improving for 4 epochs
2. Stop training when the Dice Coefficient loss has stopped improving for 4 epochs, and retain the weights of the last step of training

This helps reduce the chances of overfitting by preventing the model from overtraining.

### Model Training

The model is then trained on our training data and is also validated at every step using our validation data. We have trained the model for 10 epochs only.

From the output of the training, we see that both the training and validation losses reduce after each epoch. The final Dice Coefficient for the training set computed is 0.89 while for the validation set is 0.87. The final Precision and Recall for the validation set is 0.91 and 0.83, respectively.

### Model Evaluation

The model was evaluated on the test data set using the following metrics :

- Dice Coefficient: **0.87**
- Recall: **0.83**
- Precision: **0.91**

Based on these metrics, we evaluate that the model generalizes well to new data since they did not reduce when compared to the same metrics measured on the validation data set. It proves to be a good predictor of the area in the satellite image that belongs to a swimming pool.

### Visualising Predictions

Below, we visualize 3 predicted mask images that is outputted by the model on the test data set. The middle column is the ground truth while the last column on the right is the prediction. We can see that the model does a good job of classifying pixels, especially on the last image where it appears to be predicting better than the ground truth mask itself.

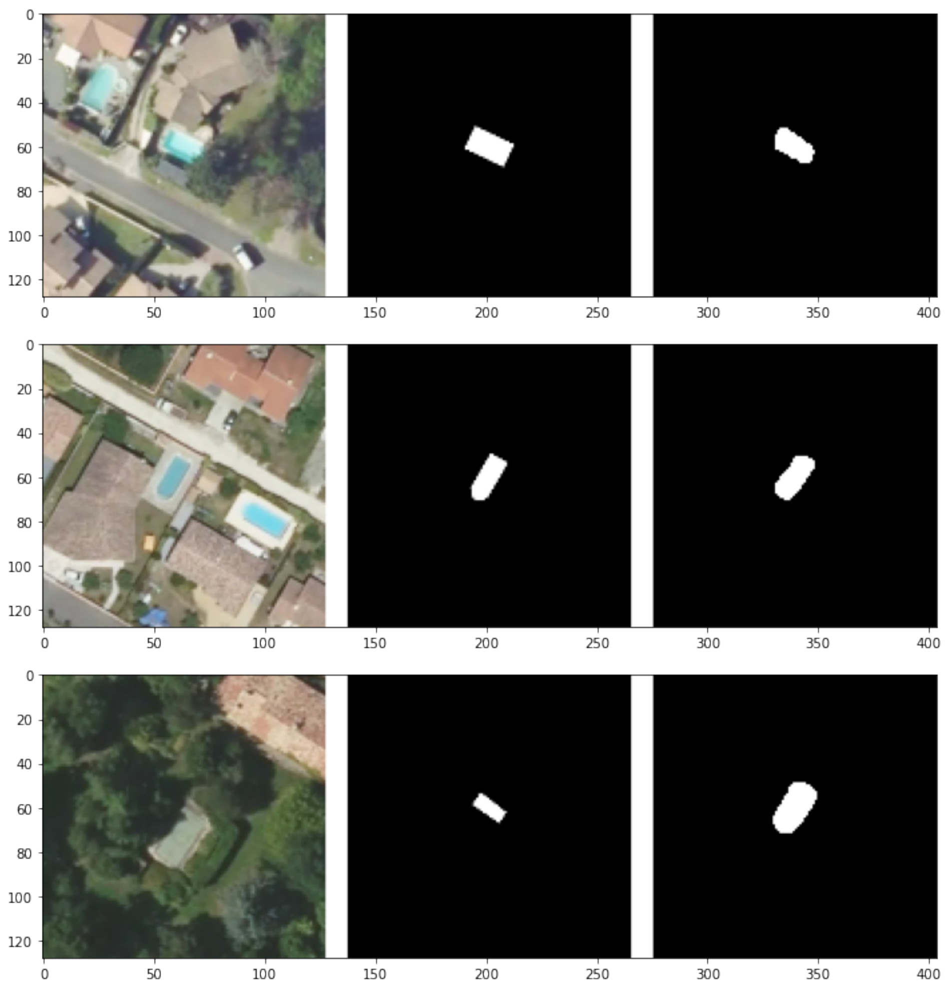


Figure 8: Visualising U-Net Predictions (Left: Image, Middle: Ground Truth Mask, Right: Predicted Mask)

## Conclusion

Overall, we find both models to perform well at their respective tasks of Object Localisation and Semantic Segmentation. We used MSE and IOU as evaluation metrics for the former and the Dice Coefficient, Recall and Precision for the latter model. Both models scored fairly high on all these metrics.

It should be noted that we did not encounter overfitting in our models as indicated by the fact that the evaluated metrics did not fall by a large amount when tested on new data. Both models contained Batch Normalization layers which helped prevent overfitting. We also used Early Stopping in the training of the Segmentation model to assist this.

However, if it had been required, we would have used techniques such as image augmentation (random crop, random translation, random zoom, etc), Dropout and/or Regularization to prevent overfitting.

# Assignment 3: Predicting on Streamed Textual Data From Twitch

## Background

The aim in assignment 3 is to predict the streaming channels in Twitch using live chat messages. The study consists of three steps: First we construct streaming textual data frame for chat messages coming from two different channels. Then a predictive model for channels is trained based on historical live chat messages. Finally, the model is deployed to the data source in order to make predictions as more streams come in. The Spark framework is used for all steps of the assignment.

We have chosen *loltyler1* and *asmongold* Twitch channels as data source for the assignment. *loltyler1* is one of the most popular online streaming channels on Twitch for video games (mainly League of Legends) with 5 million followers. On the other hand, *asmongold* is a popular channel streaming mostly about another video game World of Warcraft with 3,2 million followers. However, live streams about some hot topic contents like Heard-Deep trial also take place in this channel. Note that both channels are targeting English speaker audiences.

## Exploratory Data Analysis

We start by pulling streaming textual data from two Twitch channels using pyspark library in python. The data frame constructed consists of 41739 observations. 30928 of them are coming from the channel *asmongold* and 10811 of them from *loltyler1*. We can deduce that the dataset can be considered as imbalanced.

In the data frame constructed, there are 4 variables in total whose all have string datatype:

- **channel:** describes to which channel the chat message was sent. The level is either #asmongold or #loltyler1 . It will be used as target label in the classifier model.
- **datetime:** shows the date and time when the message was sent
- **message:** textual feature showing the message sent to the live chat
- **username:** stores the information which twitch user sent the message

We also want to examine the volume of messages sent by each twitch user. There are 11671 different users in the dataset and they sent 3,57 messages on avarage to either of channels' chat. As seen in the Figure 9, one of the user named *fossabot* has sent 536 messages while the second highest number is 118. We can consider this user as extremely active since the number of messages it sent is significanly higher than the mean(3,57) and the upper quartile(3.00). Also note that it is active in both channels with 400 messages in *loltyler1* stream and 136 messages in *asmongold* stream.

If we go deeper with the user *fossabot* concerning its remarkable domination in both streams' chat, we realize that most of the messages it sent are repetitions of each other. They are mostly personalized by tagging other users and consist of some hyperlinks, which is similar to the working pattern of a spam bot. Considering its possibility of being bot and its level of activity in both channels, especially in *loltyler1*, we might consider removing it from the dataset later in the modelling part.

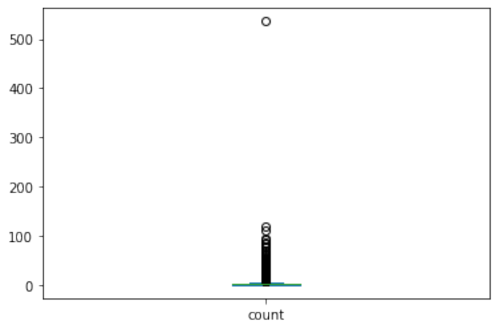


Figure 9: The Boxplot of the Number of Messages Sent per User

Before proceeding to the modelling part, we split the dataset randomly into training, validation and test parts with the proportion of 0.60, 0.20 and 0.20 respectively.

## The Predictive Model

Initially, a pre-trained model was used to encode and predict the model. These have been trained by John Snow labs and only a small amount of training is required to then implement this model. This is done using only the messages and classifies channels based on this. This process uses a document assembler, universal sentence encoder, and a deep learning classifier in a pipeline. The document assembler assembles all messages. The next step is to use a universal sentence encoder. This is a pretrained model that turns a sentence into encodings that are a 512 length vector of numbers. This vector is also created using a deep averaging network. This vector is then put into a deep learning classifier. This has also been pretrained for running on sentence embeddings. This model alone performs very well but others were evaluated to investigate if different featurizations would improve performance.

Next, a model was created using just the usernames and classifying using that. These usernames are tokenized, hashed, converted into inverse document frequency and then a classifier is applied. This model only predicted one class indicating that this information was not enough to effectively classify. A logistic regression and decision tree was implemented and neither worked well. It is possible that other types of encodings may perform better as well as other models.

Lastly, the pretained model is fitted removing the messages of the username *fossabot* as discussed in the EDA part of the analysis. This model works as same as the first model described above, expect being fitted on a subset of the dataset. It will be considered as the message sub-classifier only to check whether it improves the model performance.

Generally, it makes sense that a high performance pretrained model that is tuned to our data would perform better than a classifier that is constructed locally.

## Performance

Below is the table of performances of the three models that were primarily evaluated. One based on the pretrained model, one classifying on the username, and a combined model. The pretrained model is by far the best model and really the only one that had good discriminatory power.

This makes sense as the two streams were talking about very different topics (The Depp/Heard trial, and League of Legends). As they may have similar users, the messages will most likely be quite different as the words used will likely have limited overlap. However, by using this data it may make it less generalizable as these channels talk about more than just these topics so if we use this model at a different date, they may be discussing different topics and the model will not be well trained for this.

| Model | Accuracy | Recall |
|---|---|---|
| Message Classifier | 86% | 60% |
| Message Sub-Classifier | 85% | 59% |
| Username Classifier | 75% | 0% |
| Combined Classifier | 75% | 0% |

Lastly, We compare the pretrained model classifying only based on messages with its sub-model as described in the previous section. We decide to drop the idea of removing *fossabot* user from the dataset since it didnt improve model and its not an automated process which can deal with upcoming streams as they come in online.

## Challenges

There were challenges with running this model for online predictions. These came from Java version problems with the local machine not able to load the pipeline used by the classifier. This problem persisted when trying to load only the model and not the entire pipeline. This issue was avoided for training and evaluation of the model as it was done on Google Colab.

## Conclusion

We successfully pulled data from two twitch streams and created a dataset that allowed for an offline training of a classifier. A pretrained model using just messages has performed well. Although this performance may not be reproducible going forward. Online prediction of this model has also been unsuccessful.

# Assignment 4: Ukrainian TikTok Data Analysis

## Introduction

In this assignment, data was collected from TikTok during April 2022. This was done using the tags: ”ukraine”, ”russia”, ”ukrainewar”, ”ukrainerussiaconflict”, ”ukrainewarrussia”, ”supportukraine”, ”ukrainevsrussia”, ”prayforukraine”, ”russianarmy”, and ”Z”. With this set of videos collected, some are randomly chosen and then those users are navigated to and from there, suggested users are navigated to and selected. It is important that this was done in Belgium and some of the data is based on geolocation data.

The data attributes collected about every video are: the user, song, and associated tags. Comments were not included as this needed to be done without signing in. In the process of analyzing this graph, the entire dataset will not be analyzed at the same time as that results in analyses that are extremely computationally expensive to run on a local machine and are not very insightful as more specific, insightful questions should be asked to glean better information from the graph.

## Approach

We will investigate different aspects of this dataset. These include which accounts and users are most influential in certain areas, which users are working to create a certain narrative around the war, and finally how quickly the topic of videos can change. Influential accounts can identify who has the greatest reach in propagating information through the network and who is most listened to.

By finding users, specifically news outlets and what tags they use, we can identify any potential bias in their reporting. While certain tags are relatively objective (for example, ”ukraine”, ”russia”, ”ukrainerussiaconflict”), others are not very objective (”supportukraine”, ”prayforukraine”). This can help identify how these news outlets are going to portray the events there.

Finally, due to the dynamicism of TikTok, the information a user is consuming can change very quickly. We will investigate how quickly the topic can change in this network from the war to a completely unrelated topic. It is important for TikTok to be able to do this as it keeps users on the site by keeping them engaged with new, exciting content so it must be able to predict and deliver content the user wants at that time very quickly.

## Users Talking About Ukraine and Russia

Next, we are looking into a subgraph containing users who are talking about Russia or Ukraine (see Figure 10). The whole graph was filtered such that we are left of only with users who are creating videos that are mentioning presidents Zelensky and Putin in their description, or have one of the tags ”ukraine”, ”russia”, ”ukrainewar”, ”ukrainerussiaconflict”, ”ukrainewarrussia”, ”supportukraine”, ”ukrainevsrussia”, ”prayforukraine”, ”russianarmy”. Edges between the nodes represent the *recommends* relationship.

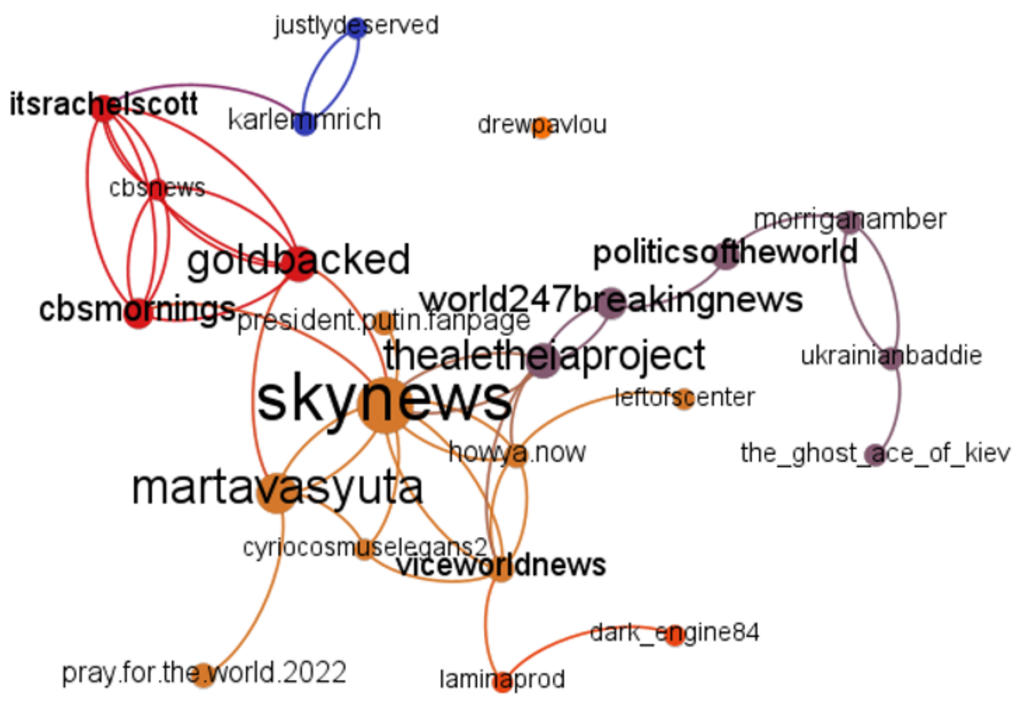


Figure 10: Users Talking About Ukraine and Russia

The colors in the graph represent the communities distinguished after performing community mining, while the node size depicts the betweenness centrality measure.

A community is a subset of the network that forms a coherent subnetwork. In our graph we see 4 communities. The orange cluster is positioned around users *skynews* and *martavasyuta*. Sky News is a British television news channel, and Marta Vasyuta is a Ukrainian TikTok influencer. She publishes content regarding the war in Ukraine. The red cluster is centered around *cbsnews*. CBS is an American television channel. In the same cluster we also see Rachel Scott, a journalist of the American ABC News.

Betweenness centrality identifies nodes that are strategically positioned in the network, meaning that information will often travel through that person. Such an intermediary position gives that person power and influence. In our graph, very influential users that are talking about the Ukraine and Russia are *skynews, martavasyuta, cbs, itsrachelscott*. We can see how communities and the most influential nodes coincide to an extent.

## Tags Analysis

Narratives are grouped not only around users, but also around the tags used in videos. In Figure 11 we see the most popular tags in our entire TikTok network. Next to the tag title, we also see the exact number of times said tag was used throughout the graph. Unsurprisingly, the most popular tags include #fyp, #foryou, and other variations of it. Authors use these tags with the hope of gaining a bigger outreach, and ending up in the TikTok's For You page.

Tags talking about Ukraine make up only a smaller portion of our data, with #ukraine being used 4113 times, and #ukrainewar 1275 times.

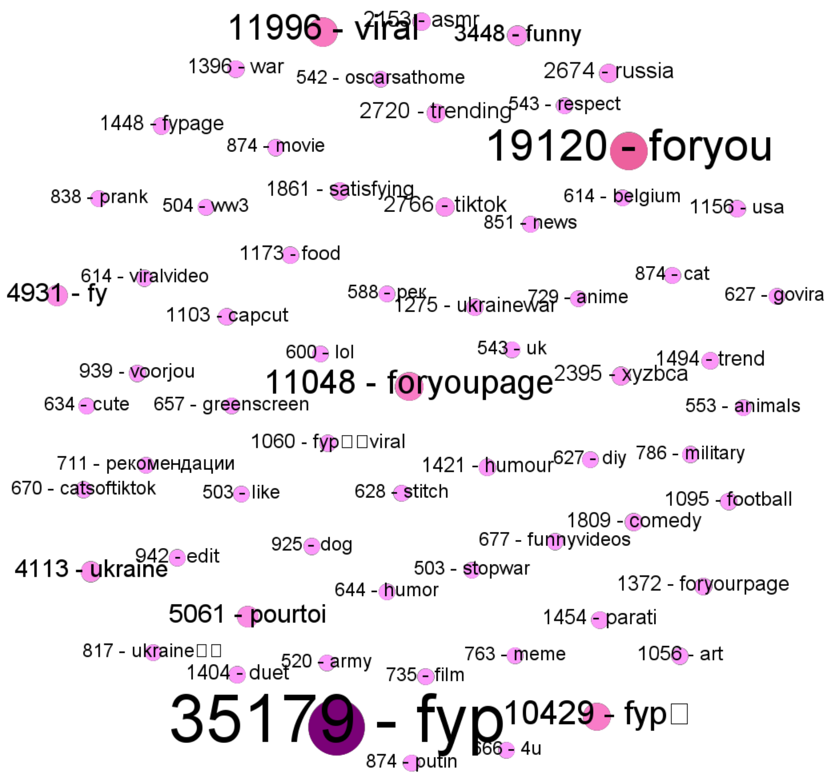


Figure 11: The Most Popular Tags

Next, we explored all of the tags that are either tagged on the same video that contains the tag #ukraine, or are tagged on the video recommended by the video with the tag #ukraine.

In the Figure 12 we see different tags connected to the #ukraine. The size and the color of the node are determined by the number of times each node was at the distance that is at most 2 (in terms of relationship *has*) from the #ukraine. It is no surprise that this tag is greatly connected to the *for you* tags, as they make the vast majority of our tags. Apart from the *for you* tags, we see that #ukraine does imply that the video is talking about the Ukraine-Russia war, as all other tags seem to be on the same topic (*russia, war, ukrainewar, putin, zelensky, ww3, standwithukraine, etc.*)

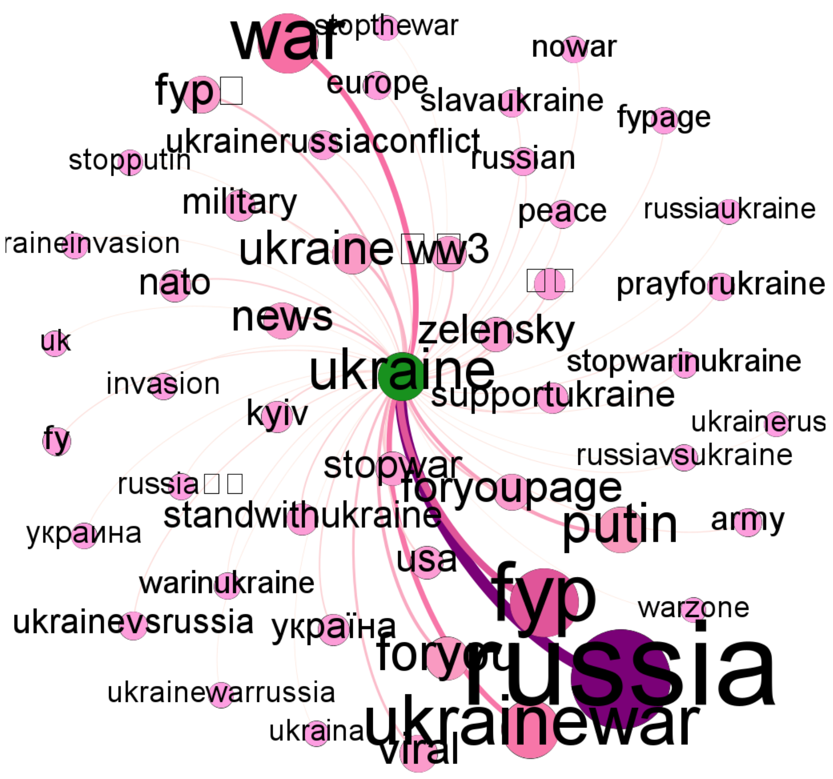


Figure 12: Tags Connected to #ukraine with Depth at Most 2

## From the War to the Kardashians

In Figure 12 we see how tags that are at the distance of 2 relationships *has* seem to stay on the same topic.

However, it is interesting how content presented to the user can change very quickly. Figure 13 shows the shortest paths between #ukrainewar and #kardashians, as well as between #ukrainewar and #haircut.

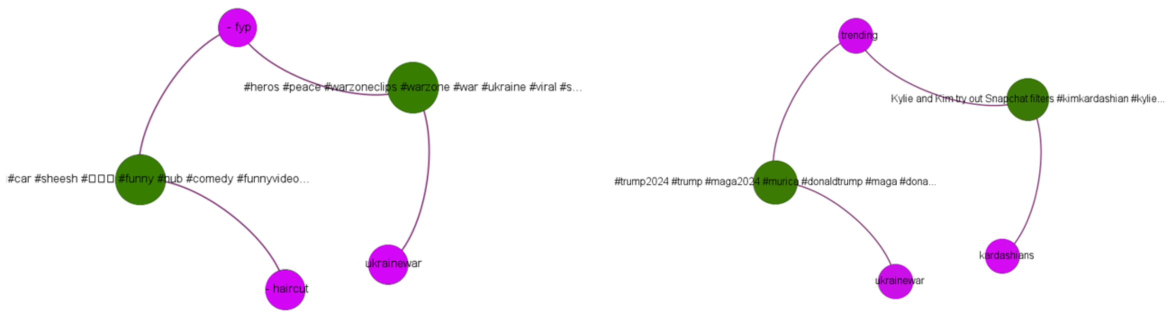

- fyp
#heros #peace #warzoneclips #warzone #war #ukraine #viral #s...
#car #sheesh #□□□ #funny #hub #comedy #funnyvideo...
ukrainewar
- haircut
trending
Kylie and Kim try out Snapchat filters #kimkardashian #kylie...
#trump2024 #trump #maga2024 #murica #donaldtrump #maga #dona...
kardashians
ukrainewar

Figure 13: Shortest Paths Between #ukranianwar and #kardashians or #haircut

Green nodes depict videos with parts of the video description shown, while purple nodes represent the tags. We can see how the Kardashians video contains tag #fyp, which in turn appears in the video talking about the USA, and potentially the war, as it also has the tag #ukrainewar. It is clear when evaluating this that while a person can consume quite a bit of information on the war over TikTok, it is also quite easy to get away from this content with relative ease.

## Conclusion

In this section we evaluated how different news outlets are connected and what that means for their influence. The biggest take away from this is that Sky News plays a very central role to the conversation surrounding the war in Ukraine. However, it should be noted that while this dataset was collected during the war, it is not solely about the war and also that the geolocation of the data being collected can influence the results. Also there are certain tags that go along very closely with the #ukraine tag. For example, #russia and #war. Finally, while there is an entire web of information connected to the war on TikTok, it is very easy to escape this with only a few clicks and be looking at the Kardashians or haircuts.