University of Amsterdam

Bachelor Thesis

Conspiracy videos on YouTube

April 24, 2021

Student:
Roan Schellingerhout

Supervisor:
Dr. Maarten Marx

1 Introduction

YouTube attracts an average of 34.6 billion page views per month, making it the world's largest video-sharing website and the second largest website on the entire internet (Neufeld, 2021). The overwhelming majority of those page views come from users watching videos, 70% of which are recommended to users by YouTube's algorithm (Cooper, 2020). All types of content get produced and consumed on the website. However, conspiracy content has been booming on YouTube (Donzelli et al., 2018). Alt-right (or far-right) and conspiracy channels are starting to grow their audiences, which could have many negative consequences for society at large. For example, the number of people who are distrustful of science is increasing, a development in which conspiracy content on YouTube plays a role. Whenever this increased distrust relates to important topics, such as believing in the efficacy of vaccines, it can create genuine dangers to the public. As it turns out, more than half of the American population has doubts about - or is definitely against - taking the COVID-19 vaccine (Rosenbaum, 2021). To better understand how YouTube's algorithm (and recommender algorithms in general) allow(s) conspiracy content to thrive, this research will investigate how quickly the algorithm develops a preference for conspiracy video; in other words: how many videos a user needs to watch before they get sent *down the rabbit hole*.

This research takes inspiration from the Dutch television program *Zondag met Lubach*, where a similar idea was executed on a smaller scale (Lubach, 2020). Lubach's experiment consisted of creating a new YouTube account on a never-used laptop, after which the account was used to watch a few (recommended) conspiracy videos, to see how this would affect the YouTube homepage of the account. After a mere three videos, the homepage of the account was filled to the brim with conspiracy content, mostly having to do with the coronavirus. The extremely interesting results that came from this micro-experiment formed the motivation to research this phenomenon more extensively.

1.1 Research question

To research this subject, the following research question has been formulated: *What is the impact of different watch strategies on the number of conspiracy videos that have to be watched until a user's YouTube-recommendations start preferring conspiracy content?* In this scenario, 'preferring' will be defined as the situation in which the amount of conspiracy videos present in the recommendations is significantly higher than that of the baseline.

In order to answer the research question, three sub-questions will have to be answered. These questions are the following:

- How do different watch strategies on YouTube influence the type of content that is recommended to a user?
- How long does it take for a YouTube recommendations to stop preferring conspiracy videos, once they have started doing so?
- What type of classifier performs the best when it comes to labeling conspiracy videos on YouTube?

2 Theoretical Framework

2.1 Filter bubbles on social media

Whenever the user of a website finds themselves in their own information universe, in which the content and recommendations play into the user's preexisting opinions and believes, they are in a filter bubble (Pariser, 2011). Users are by themselves in such bubbles and each bubble is unique. Different bubbles can have overlap, but each bubble is precisely tuned to an individual. In traditional media, a user makes a conscious *choice* what types of opinions they want to hear, for example by choosing to watch a broadcaster with a specific political opinion. Online this decision is implicit: based on the user's behavior, their content is filtered automatically by an algorithm, without explicit consent.

2.2 Filter bubbles on YouTube

Previous research has found that YouTube's recommendation algorithm runs the risk of creating filter bubbles. Roth et al. (2020) came to this conclusion after they analysed YouTube recommendations based on content. YouTube has two distinct types of recommendations: recommendations based on the user's viewing behavior and recommendations based on the content of the current video a user is watching. In their research, Roth et al. focused predominantly on recommendations based on content. They found that such recommendations could quickly lead to a decrease in information diversity (read: filter bubbles) and that this decrease happened sooner for videos with a lot of views; the more views a video had, the less diverse its related recommendations. They speculate that this can be explained by the fact that YouTube tends to store more information about videos with a high view count, allowing the algorithm to give better recommendations for such videos. They also predict that, whenever the algorithm has more information about a user to its disposal, it can combine said information with the information it has about a certain video, which could lead to an even stronger limitation of recommendations. According to Ledwich and Zaitsev (2019), a user's viewing behavior is responsible for approximately 70% of their recommendations; this behavior could therefore play a big role in the creation of filter bubbles on YouTube.

2.3 Conspiracy content on YouTube

YouTube has limited rules with regards to the spread of conspiracy videos (YouTube, 2021). As long as the content does not directly incite violence or endangers the public health (e.g. misinformation about the COVID-19 virus), objectively incorrect ideas are allowed to be shared on YouTube. As a result, YouTube is a home to multiple conspiracy communities. Conspiracy theories such as 'the earth is flat and the government is hiding it from us', 'the world will end soon and only followers of this specific religion will be spared', and 'the world is ruled by cannibalistic, satanic pedophiles' (better known as QAnon) gather millions of views on the platform (Paolillo, 2018; Miller, 2021). Though such videos could be considered harmful to society, they are not suppressed by YouTube. Whenever a user shows interest in this type of content, they will be recommended similar videos, even when YouTube is aware of their harmful nature (Ledwich and Zaitsev, 2019; Maack, 2019).

2.4 The YouTube algorithm

The YouTube algorithm tries to recommend videos based on the expected watch time they will generate, rather than the probability of a user clicking on them (Covington et al., 2016). This decision was made in order to decrease the likelihood of misleading videos (also known as *clickbait*) being recommended. However, gathering feedback about videos through their watch time can cause a lot of noise, making it difficult to measure user satisfaction. As it turns out, even when a users enjoy a certain video, they are unlikely to watch it completely. On average, users watch around 50-60% of a video before they switch it off (Park et al., 2016). Though, videos that are well-structured, or especially interesting, can improve this percentage up till 70-80%, where nearly half of the viewers actually finish the video in its entirety (Lang, 2018). After a video has been watched, there is a 41.6% chance that the user decides to watch a recommended video. Which recommendation the user will choose, follows a Zipf-distribution ($\alpha = 0.78$) with regards to the position of the video in the list of recommendations (Zhou et al., 2010).

All in all, previous research has found that YouTube's algorithm is sensitive to filter bubbles and that it has a tendency to recommend conspiracy content. It is also speculated that the algorithm makes decisions based on the user's viewing behavior, which it combines with the content of videos. In order to keep the user on the website as long as possible, which is profitable for YouTube, the algorithm prefers recommending videos that it suspects the user will watch for a longer period of time, even when they may contain harmful content. Based on this information, further research can be done on the origination of filter bubbles and the spread of conspiracy content on YouTube. For example, little is known about how quickly a user's recommendations adapt to a user's behavior, even though this is a critical aspect when it comes to the creation of so-called *rabbit holes*. Furthermore, no research has been done into the way different types of videos (recommendations in different locations, random videos, etc.) influence YouTube's algorithm. For example, whenever a user primarily watches recommended videos, the algorithm could see this as implicit positive feedback, which could cause a snowball-effect.

3 Methodology

3.1 Watching conspiracy videos

In order to determine how different watch strategies affect the YouTube algorithm, a python script was created to automatically log into a Google account and proceed to watch YouTube videos. The script was made using Selenium WebDriver: a suite of tools used for browser automation.

3.1.1 Google login

Due to Google's strict policy regarding automation within their ecosystem, many obstacles are put into place to prevent users from logging into a Google account using automated software such as a selenium script. To circumvent this restriction, two steps had to be taken. Firstly, the selenium WebDriver had to be accompanied by the selenium-stealth package, which removes metadata about the current browser, so that it is less obvious that a WebDriver is being used. Additionally, because this metadata was removed, the Google login service was unable to check what browser the client was using. This results in a warning to the user that their current browser may be insecure, which prohibits them from logging in. To avoid this warning, the Google account needs to have been created within a WebDriver, such as Google's ChromeDriver or Mozilla's GeckoDriver. Therefore, all twenty accounts were manually created in ChromeDriver. Since Google accounts require a phone number verification upon creation, six free (prepaid) SIM cards were ordered from various providers in order to create the accounts. Each SIM card could create two to three accounts before it was blocked due to being used too many times.

3.1.2 The watch strategies

After all accounts had been created, they were subdivided into four distinct watch strategies, making for a total of five accounts per strategy. The first watch strategy is the simplest one. The bots following it will watch random, non-conspiracy videos from a dataset. This watch strategy is used as the baseline to compare the other three strategies to. The second strategy is similar to the first: the adhering bots watch random conspiracy videos from a dataset. This strategy was not added to study the origination of filter bubbles, since it is unlikely that a filter bubble will come from this strategy. Considering the video choices will be all over the place, it will be difficult for the algorithm to determine the specific interest of the user. However, comparing how the algorithm responds to conspiracy content in general as opposed to regular content might still yield interesting results. The second-to-last strategy starts off in the same way: it chooses a random conspiracy video from a dataset to watch. However, after having watched the initial video, it will start watching the recommended videos displayed next to the current video. These recommendations consist of a combination of recommendations based on the content of the current video and the personalized recommendations of the user. By using this strategy, bots are likely to go *down the rabbit hole* and eventually end up in a filter bubble. Finally, the last strategy will be similar to the previous one, though with one alteration: rather than choosing a recommended video from the list of recommendations next to the current video, it will choose a recommended video from the YouTube homepage of the account. Compared to the third strategy, this will lead to the user watching more personalized recommendations rather than content-based recommendations, possibly speeding up the creation and/or increasing the strength of the filter bubble. Each strategy will be executed by five different accounts in order to decrease the probability of a random streak of videos altering the result. The individual accounts will watch a total of fifteen videos as described by their watch strategy, for a total of three hundred videos watched by the script.

3.1.3 Running the bots

After the accounts were logged in, they started watching YouTube videos according to their watch strategy. However, some restrictions were put into place to make sure the bots did not take too long (considering three hundred videos had to be watched in total, some limitations had to apply). For example, the bots were not allowed to watch videos over an hour long, nor were they allowed to watch live streams, as those could theoretically go on infinitely. Additionally, the random videos at the start of the third and fourth strategy were first manually inspected to make sure the bots would not start the experiment by watching a falsely flagged conspiracy video. Considering the way in which the dataset was created, it is possible that some videos that are flagged as conspiracy videos are, in reality, normal videos. This could happen whenever a conspiracy channel uploads a regular video for once (e.g. a holiday video, promotion of some product, etc.). These *false positives* are far and few between, however, having one of them be selected as the first video for strategy three or four had to be prevented, as that could greatly alter the final results. With these restrictions in mind, the following script was created and run for all twenty bots, keeping track of the videos they watched and the homepage recommendations they had after each video:

Algorithm 1: Watch YouTube videos according to a watch strategy

Data: User information and a video dataset
Result: The watched videos and homepage recommendations of the user

```
initialize WebDriver;
log into Google account;
for twenty videos do
    if there is a recommendation to be watched then
        go to the link;
    else
        pick a random video to watch based on usertype;
        determine how long it will get watched;
        go to the link;
    get video metadata and store for overview of watched videos;
    watch video for given amount of time;
    if usertype == 3 then
        pick recommendation next to current video to watch next;
        determine watch time for found recommendation;
    go to YouTube homepage;
    store current recommendations for overview;
    if usertype == 4 then
        pick homepage recommendation to watch next;
        determine watch time for found recommendation;
return watched videos and homepage recommendations;
```

Running the script for all twenty bots resulted in two different datasets: the first containing the videos watched by the bots, including their view count, likes/dislikes, and upload date; and one containing the homepage recommendations for all bots, after each number of videos watched. To determine the influence of the watch strategies on the algorithm, the recommendations were labeled as being either conspiracy or non-conspiracy videos by the classifier. By then grouping all recommendations by their watch strategy and the number of videos watched before them (i.e. the recommendations after the third video watched by all bots with strategy one), it was possible to calculate aggregates about general statistics of the recommendations, such as view count and video duration, and the percentage of conspiracy videos present amongst them. This lead to four groups with fifteen entries of different statistics (one for each video watched). In order to find out whether any of the differences between the four groups were significant at any point, a number of ANOVAs were performed.

3.2 Leaving the filter bubble

3.3 Machine learning

3.3.1 Data gathering

To answer the research question, it is necessary to determine which YouTube videos can be considered conspiracy videos. Considering the large amount of videos getting recommended, determining each video manually is simply not possible. There are two possible ways to solve this problem. Firstly, there is a dataset which contains nearly 7000 YouTube channels that have been manually labeled based on their political view - almost 3000 of which were labeled as conspiracy channels (Ledwich and Zaitsev, 2019); whenever a video is made by one such channel, it can be considered a conspiracy video. However, due to the enormous amount of existing YouTube channels, the odds of a video being uploaded by a channel that is not present in this dataset are very large. For those videos, a supervised machine learning classifier was used. To optimize performance, five different classifiers have been trained and compared: k-nearest neighbors, support-vector machine, neural network, logistic regression, and ridge regression.

In order to train these machine learning algorithms, a training dataset was created. To get a labeled dataset of conspiracy and non-conspiracy videos, use was made of the aforementioned channel dataset made by Ledwich and Zaitsev (2019). For each channel in that dataset, the title, description, and transcript of the ten most recently uploaded videos were downloaded using YouTube's API. Videos uploaded by a conspiracy channel were then labeled as conspiracy videos, and videos uploaded by a channel from a different category were labeled as normal videos. Additionally, the channel description and channel keywords (which are used for targeted advertising on YouTube) were added to each video. The final dataset contains 65.683 unique YouTube videos, 22.156 of which are considered as conspiracy videos.

3.3.2 Data cleaning

However, this dataset was not yet suitable for machine learning, as the data was still messy. Therefore, multiple steps were taken in order to clean the data. Firstly, the two classes (conspiracy and non-conspiracy) were balanced, so that the classifier would not develop a bias for non-conspiracy videos. Rather than opting for balancing the two classes through the use of class-weights (a technique where weights are attributed to classes, thereby telling the classifier that getting a prediction correct for a certain, underrepresented class is more important), the choice was made to under-sample the data in order to equalize both classes (both containing 22.156 videos, for a total of 44.312 videos) (Lemaître et al., 2017; Sun et al., 2006). As there was plenty of data in the dataset, under-sampling was more convenient than implementing class-weights. After both classes had been balanced, the text for each video had to be translated into English. Since the original dataset by Ledwich and Zaitsev (2019) also contained channels by non-English speakers, these videos had to be automatically translated. Then, a few common cleaning methods were applied: all text was converted to lowercase, after which special characters, such as emojis were removed, whereafter stop words were removed and all words were stemmed using the porter stemmer (Karaa, 2013). Finally, each video was TF-IDF vectorized to allow the classifiers to function.

3.3.3 Performance optimization

After splitting the dataset into a training, test, and validation set, the hyperparameters of each algorithm were tuned to get the optimal performance (Feurer and Hutter, 2019). Performance was measured using four distinct metrics: the accuracy, which shows the share of correct predictions; the recall, which shows what fraction of truly positive samples were correctly labeled as such; the precision, which shows what part of the positive predictions were correct; and the F1-score, which is the harmonic mean of the recall and precision (Sokolova and Lapalme, 2009). For each classifier, different configurations of hyperparameters (such as the kernel and the penalty-parameter) were systemically tested - each possible combination was tried. The classifiers were trained on the training set and the optimal hyperparameters were determined based on the performance of the classifiers on the validation set. By saving these performance measures for every configuration, for every classifier, the optimal configuration for every classifier could be determined. Lastly, the classifiers were equipped with their optimal hyperparameters and then tested for the final time on the test set. By comparing the performance of every optimally configured classifier on the test set, the best-performing classifier could be chosen (Reitermanova, 2010).

Additionally, the added value of using a machine learning ensemble was measured. By having each classifier make a prediction for all videos in the dataset, a new dataset was created, wherein the features were the predictions made by the different classifiers. By using all possible combinations of classifiers, and then having different neural networks use those features as input, a machine learning ensemble was created. This ensemble was then optimized in a similar way to the classifiers individually.

Classifier performance

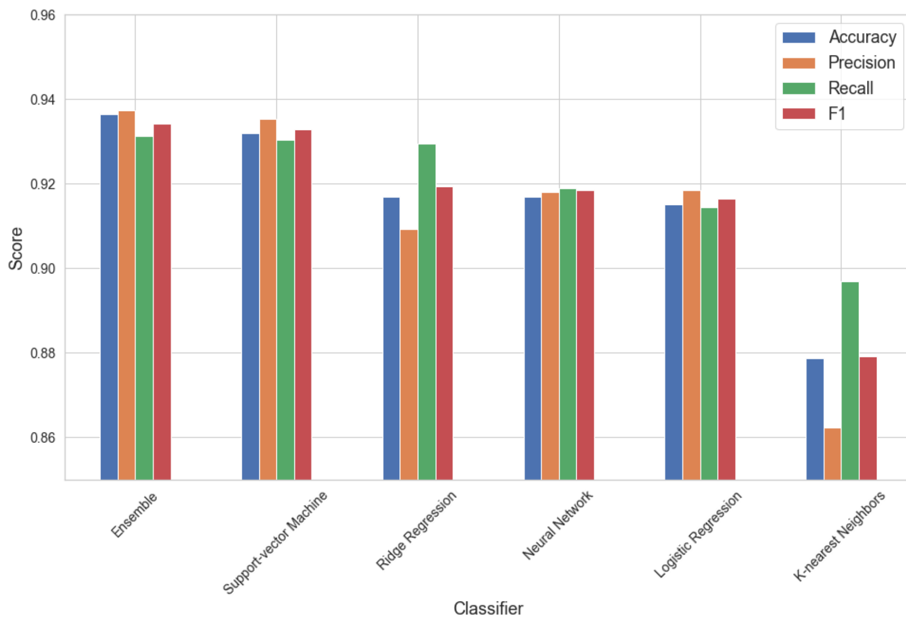

Figure 1: Metrics for each classifier with optimized hyperparameters

4 Results

4.1 The recommended content

4.2 Leaving the filter bubble

4.3 Machine learning

The hyperparameter tuning lead to impressive scores for all classifiers. When making prediction for the test set, the best-performing classifier is the support-vector machine making use of the Radial Basis Function (RBF) kernel and a penalty parameter (C-value) of 10. In second place, there is a two-way tie for accuracy between ridge regression with a sparse-cg solver and penalty (alpha) value of 0.1, and the neural network using the identity activation function with 10 hidden layers of 10 neurons. However, ridge regression has a slightly better F1-score, though this difference is neglectable (0.9193 as opposed to 0.9184). The neural network and ridge regression are closely followed by logistic regression with an L2 penalty, a penalty (C) value of 20 and a newton-cg solver. The worst-performing classifier is also the simplest of the bunch: the k-nearest neighbors classifier (K=1). Although its performance is still formidable, it does substantially worse than the others. An overview of all metrics for each classifier can be seen in figure 1. The ten best-performing configurations for each classifier can be found in appendix A.

Noteworthy is the fact that the optimal ensemble actually outperforms the support-vector machine by a slight margin. This ensemble, consisting of the SVM, the neural network, and surprisingly, the k-nearest neighbor classifiers, gets slightly higher scores than the runner-up across the board. The ensemble had a 16-way tie for best-performing parameters, all of which contained at least the SVM, neural network, and k-NN classifiers.

Though the ensemble outperforms the other classifiers, it has a significant drawback: its training time is significantly larger than that of the individual classifiers. Support-vector machines are infamous for their slowness when there is a lot of training data, and neural networks can require a lot of training time whenever the number of neurons gets large (Burges and Schölkopf, 1997; Kamarthi and Pittner, 1999). Requiring both algorithms to run will therefore require a lot of additional training time. Considering the marginal performance increase, the cost outweighs the benefit. As a result, when taking everything into account, the support-vector machine is the best classifier for labeling conspiracy videos on YouTube.

5 Discussion

To do.

6 Planning

Table 1: Planning

Week	Handelingen	Afgehandeld
28/03-03/04	Hyperparameters optimaliseren	Ja
03/04-10/04	Classifier-ensemble optimaliseren	Ja
11/04-17/04	Google accounts maken, deelvraag 3 maken	Ja
18/04-24/04	Inleiding uitbreiden	Ja
25/04-01/05	Uitvoering experiment, labelen met classifier	
02/05-08/05	Deelvraag 1 schrijven, beginnen deelvraag 2	
09/05-15/05	Deelvraag 2 afschrijven	
16/05-22/05	Beginnen met discussie schrijven	
23/05-29/05	Discussie afschrijven	
30/05-05/06	Abstract schrijven, tekst proof-readen	
06/06-12/06	Laatste aanpassingen en verbeteringen	
13/06-19/06	Inleveren scriptie en verdediging	

References

Burges, C. J. and Schölkopf, B. (1997). Improving the accuracy and speed of support vector machines. *Advances in neural information processing systems*, pages 375–381.

Cooper, P. (2020). How does the youtube algorithm work? a guide to getting more views.

Covington, P., Adams, J., and Sargin, E. (2016). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198.

Donzelli, G., Palomba, G., Federigi, I., Aquino, F., Cioni, L., Verani, M., Carducci, A., and Lopalco, P. (2018). Misinformation on vaccination: A quantitative analysis of youtube videos. *Human vaccines & immunotherapeutics*, 14(7):1654–1659.

Feurer, M. and Hutter, F. (2019). Hyperparameter optimization. In *Automated Machine Learning*, pages 3–33. Springer, Cham.

Kamarthi, S. V. and Pittner, S. (1999). Accelerating neural network training using weight extrapolations. *Neural networks*, 12(9):1285–1299.

Karaa, W. B. A. (2013). A new stemmer to improve information retrieval. *International Journal of Network Security & Its Applications*, 5(4):143.

Lang, P. (2018). Youtube average view duration - the 50% rule.

Ledwich, M. and Zaitsev, A. (2019). Algorithmic extremism: Examining youtube's rabbit hole of radicalization. *arXiv preprint arXiv:1912.11211*.

Lemaître, G., Nogueira, F., and Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research*, 18(1):559–563.

Lubach, A. (2020). De online fabeltjesfuik — zondag met lubach (s12).

Maack, M. M. (2019). 'youtube recommendations are toxic,' says dev who worked on the algorithm.

Miller, D. T. (2021). Characterizing qanon: Analysis of youtube comments presents new conclusions about a popular conservative conspiracy. *First Monday*.

Neufeld, D. (2021). The 50 most visited websites in the world.

Paolillo, J. C. (2018). The flat earth phenomenon on youtube. *First Monday*.

Pariser, E. (2011). *The filter bubble: What the Internet is hiding from you.* Penguin UK.

Park, M., Naaman, M., and Berger, J. (2016). A data-driven study of view duration on youtube. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 10.

Reitermanova, Z. (2010). Data splitting. In *WDS*, volume 10, pages 31–36.

Rosenbaum, L. (2021). Escaping catch-22—overcoming covid vaccine hesitancy.

Roth, C., Mazières, A., and Menezes, T. (2020). Tubes and bubbles topological confinement of youtube recommendations. *PloS one*, 15(4):e0231703.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437.

Sun, Y., Kamel, M. S., and Wang, Y. (2006). Boosting for learning multiple classes with imbalanced class distribution. In *Sixth international conference on data mining (ICDM'06)*, pages 592–602. IEEE.

YouTube (2021). Youtube community guidelines & policies - how youtube works.

Zhou, R., Khemmarat, S., and Gao, L. (2010). The impact of youtube recommendation system on video views. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 404–410.

Appendices

A Hyperparameter tuning

Ensemble	Activation	Layers	Neurons	Accuracy	Precision	Recall	F1
svm, nn, knn	logistic	1	20	0.939318	0.948923	0.931204	0.93998
svm, nn, knn	relu	1	20	0.939318	0.948923	0.931204	0.93998
svm, nn, knn	identity	1	1	0.939318	0.948923	0.931204	0.93998
svm, nn, knn	identity	10	1	0.939318	0.948923	0.931204	0.93998
svm, nn, knn	logistic	1	10	0.939318	0.948923	0.931204	0.93998
ridge, svm, nn, knn	tanh	10	1	0.939318	0.948923	0.931204	0.93998
svm, nn, knn	tanh	1	10	0.939318	0.948923	0.931204	0.93998
svm, nn, knn	tanh	1	20	0.939318	0.948923	0.931204	0.93998
svm, nn, logr, knn	identity	1	1	0.939318	0.948923	0.931204	0.93998
svm, nn, logr, knn	identity	1	10	0.939318	0.948923	0.931204	0.93998

Table 2: Ensemble.

Kernel	C	Accuracy	Precision	Recall	F1
rbf	10.0	0.936309	0.945473	0.928747	0.937035
rbf	100.0	0.935557	0.942289	0.930713	0.936465
rbf	1.0	0.925276	0.930163	0.922850	0.926492
poly	10.0	0.916499	0.946017	0.886978	0.915547
poly	100.0	0.915246	0.944940	0.885504	0.914257
linear	1.0	0.913741	0.917944	0.912531	0.915229
poly	1.0	0.909729	0.935065	0.884521	0.909091
linear	10.0	0.904965	0.907882	0.905651	0.906765
linear	100.0	0.898195	0.905830	0.893366	0.899555
rbf	0.1	0.878887	0.878906	0.884521	0.881705

Table 3: Support-vector machine.

Activation	Layers	Neurons	Accuracy	Precision	Recall	F1
identity	10	10	0.923019	0.935484	0.912039	0.923613
identity	25	10	0.921013	0.933031	0.910565	0.921661
relu	10	10	0.919007	0.917561	0.924324	0.920930
identity	10	20	0.916750	0.906056	0.933661	0.919652
relu	10	20	0.915998	0.912221	0.924324	0.918233
tanh	10	10	0.915747	0.914592	0.920885	0.917728
relu	1	1	0.915747	0.931876	0.900737	0.916042
tanh	25	20	0.915496	0.919052	0.914988	0.917016
tanh	10	20	0.914744	0.920178	0.912039	0.916091
logistic	1	1	0.913741	0.925516	0.903686	0.914470

Table 4: Neural network.

Solver	Alpha	Accuracy	Precision	Recall	F1
auto	0.1	0.918506	0.919118	0.921376	0.920245
sparse_cg	0.1	0.918506	0.919118	0.921376	0.920245
sag	0.1	0.918255	0.919902	0.919902	0.919902
auto	1.0	0.917252	0.923497	0.913514	0.918478
sparse_cg	1.0	0.917252	0.923497	0.913514	0.918478
sag	1.0	0.917252	0.923497	0.913514	0.918478
sag	10.0	0.878385	0.893002	0.865356	0.878962
auto	10.0	0.878134	0.892549	0.865356	0.878743
sparse_cg	10.0	0.878134	0.892549	0.865356	0.878743
auto	100.0	0.812437	0.854545	0.762162	0.805714

Table 5: Ridge regression.

Penalty	C	Solver	Accuracy	Precision	Recall	F1
l2	20	newton-cg	0.918506	0.924107	0.915479	0.919773
l2	20	saga	0.918506	0.924107	0.915479	0.919773
l2	20	sag	0.918506	0.924107	0.915479	0.919773
l2	10	sag	0.916249	0.920831	0.914496	0.917653
l2	10	newton-cg	0.916249	0.920831	0.914496	0.917653
l2	10	saga	0.916249	0.920831	0.914496	0.917653
l2	10	lbfgs	0.915747	0.919506	0.914988	0.917241
l2	20	lbfgs	0.914744	0.921432	0.910565	0.915966
none	1	sag	0.913992	0.923848	0.906143	0.914909
none	10	saga	0.913240	0.922461	0.906143	0.914229

Table 6: Logistic regression.

K	Accuracy	Precision	Recall	F1
1	0.889669	0.888456	0.896314	0.892368
3	0.888415	0.882212	0.901720	0.891859
4	0.879137	0.908899	0.848157	0.877478
5	0.873370	0.858482	0.900246	0.878868
6	0.873119	0.882441	0.866830	0.874566
2	0.872618	0.935043	0.806388	0.865963
7	0.868355	0.848891	0.902703	0.874970
8	0.867603	0.867382	0.874201	0.870778
9	0.861585	0.835672	0.907125	0.869934
10	0.859579	0.854397	0.873710	0.863946

Table 7: K-nearest neighbors.