HS19

California Forest Fire Prediction

DSF PROJECT

YEHUDAH GOL
LUCA INAUEN
ROBIN SCHERRER
JEROM KÄMPFER

Table of contents

Explanation of Covariates

Fieldname	**Explanation**
tavg	daily temperature average
tmax	daily temperature maximum
tmin	daily temperature minimum
rh	relative humidity
pressure	pressure
wspd	windspeed
precip	daily precipitation
DSLR	days since last rain
HPW1	average temperature last week
HPW2	average temperature two weeks ago
HPW3	average temperature three weeks ago
weather_11	light rain
weather_12	rain
weather_19	blowing dust / windy
weather_20	patches of fog
weather_21	haze
weather_26	cloudy
weather_27	mostly cloudy
weather_28	mostly cloudy
weather_29	partly cloudy
weather_30	partly cloudy
weather_33	fair
weather_34	fair
weather_40	heavy rain

1 Introduction

With the expected rise in average temperature due to climate change and the resulting potential for drought, forest fires might become more common and spread more easily through drier forests, further increasing CO_2-emissions and posing a threat to humans and wildlife. For this reason, recognizing the risk of wildfires will become more important as well. This gives rise to the following question: Can machine learning be used to find a pattern behind forest fires and predict when a fire will break out using weather data?

In an attempt to predict forest fires, we are using the US forest fire dataset and weather data from Ontario, California to predict fires in the area around the weather station that provided us with the data. The problem will be structured as a classification problem, with the resulting prediction being a true/false value about whether a fire will break out for each day.

2 Data Collecting

First, we had to get the necessary data. We found a dataset on Kaggle that contained the information of all recorded wildfires in America between 1992 and 2015. The single observations contained information about the specific time, the size of the fire and latitude/longitude. The database had the form of a ".sqlite" file. After some research we could easily import it in RStudio. The first difficulty we faced while collecting the data was that the dataset didn't contain informations about the climate. So, we had to get the climate data from another source.

We found several websites that contained climate data. The challenge was to find a website that provided daily observations over several years without having too many voids. The only page we found that fulfilled our requirements was "Wunderground.com". So, we found a suitable data source, but it looked like the page owners didn't want to offer their data. When we tried to get the html source code, we realized that the page was generating the table with the needed data dynamically with JavaScript. Instead of the data we only got JavaScript calls as response. In order to get the data, we used a scriptable headless browser called PhantomJS. PhantomJS allows to simulate a real web browser for every single call. Our hope was that we could trick the webpage and receive the plain data instead of useless JavaScript calls. Unfortunately, we were disappointed. The page was still answering with plain JavaScript code. After some google research, we saw that Wunderground stopped the service for their API at the end of 2018. After this time no further API key could be generated without registering an own weather station to contribute climate data to the webpage. Luckily, after inspecting the network traffic with chromes developer tools, we noticed that the webpage itself was using an

API key to get the data from the webserver. We shorthand used the same key for our own API calls. We also found the API documentation of the time it was public. Fortunately, the API hadn't been changed since then. The webserver answered each API call with a beautifully formatted json object. The only downside was the number of days that could be received together in one API call. It was limited to a maximum of 31 days. As a result, we had to make an API call for each month between 1992 and 2015.

3 Data Cleaning

The first problem we faced while cleaning the data came with the weather data CSV files from noaa.gov. We tried to use this data source before we found the Wunderground webpage. This dataset was very large and took several separate, manual downloads because of the export size limitation imposed by noaa.gov. It was also very messy, and while we had attempted to clean it properly, we soon found out that it was almost impossible to bring it into a usable form. The dataset was inconsistent, with e.g. the temperature value often being NA and weather stations being added and removed over the course of a few years. After some attempts at cleaning, we had to switch to the dataset we got with web sracping from Wunderground and changed our observation area to a circle with a radius of 25 km around Ontario, California.

As soon as we got the data in RData-format, it was possible to load it into the workspace immediately and begin with the data cleaning. This data was fairly clean, but a lot of data transformation was still necessary. Since the forest fires dataset and the weather dataset were gathered from separate sources, merging the two sets would be necessary later on.

First, we used the anytime-library to transform the timestamps from Unix-time to Epoch format. Then we created a list of variables to be dropped. This list was expanded over the course of the preparation process, as some variables turned out to be useless later in the project.

The weather observations were done hourly, which was not a very practical frequency for the classification. Whenever there was a fire at a specific date, it was necessary to sum up the observations into daily values. For this, we first added an index indicating the number of the day of each observation. With this, it was possible to loop through the data and calculate the daily values. A decision on how the summarising should happen had to be made for each variable. Some variables such as the temperature at the time of the observation or the relative humidity had to be converted into an average. Others, i.e. discrete variables like weather type or wind direction, could not reasonably be converted that way. With those we counted the occurrence of each value and picked the most common value for each day. The calculated daily values were then collected into a new data frame.

Unfortunately, it turned out that the total precipitation and the sum of the hourly precipitation over a day did often not match. We decided that the hourly precipitation was most likely more precise since the total precipitation seemed to ignore small values and thus calculated the total from the hourly values. The total precipitation from the original data was removed.

We added some more variables by calculating the number of days since it last rained and the temperature of the previous weeks using custom functions in the 000_Functions.R file. The idea behind this was that drought and heat periods can lead to significantly increased risk of a fire outbreak. An overview of all the covariates and their description is listed at the beginning of the paper.

The weather type variable had to be made usable as well. Two variables with the same meaning were present in the dataset: weather type as number and the corresponding phrase of the weather type. This was not a format we could have used because if the numbers had been weighted by an algorithm, it would have distorted the result since the correlation would most likely not have been linear in the variable. We used the phrases instead and one-hot encoded them. For a similar reason, we removed wind direction as well.

With this, the weather data was ready for merging. To prepare the fire data, only the fires above size class A were selected as those below often just had the size of a campfire and would spark fairly randomly. We also removed any following fires after the first of a day, since one per day was all we needed for classification. This was then converted into the y-variable. After that, all other variables in the set except the date and the y-variable were dropped. The two datasets could now be merged with the join-function from the dplyr library.

4 Model Building

Throughout our project, we intended to work with algorithms from the parametric as well as from the non-parametric family in order to tackle the problem from different angles. We decided to cover these families with logistic regression and KNN because we already had some practice with these two during the bootcamp. Moreover, we chose neural network as a third method in order to complement our model selection with a more complex approach.

4.1 Preparation

The first trial of all three models brought similar results. All models performed quite satisfactory in terms of the empirical testing error as you can see with the example of the logistic regression approach on the bar chart on the right. Nevertheless, all models were pointless since they did just predict a no-fire for almost every observation. The good performance based only on the circumstance that the fire observations accounted only for approximately 10% of the whole dataset. Therefore, we decided to adjust our training dataset by increasing the percentage of fire observations. In order to do so we dropped a certain percentage of random no-fire observations and got a new dataset with approximately 50% fire and no-fire observations. We called this dataset the “adjusted training sample”.

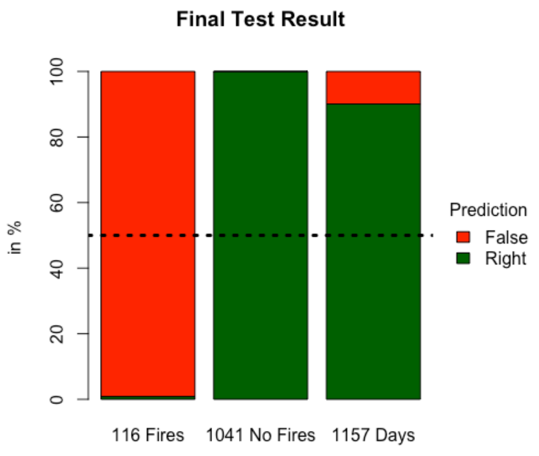
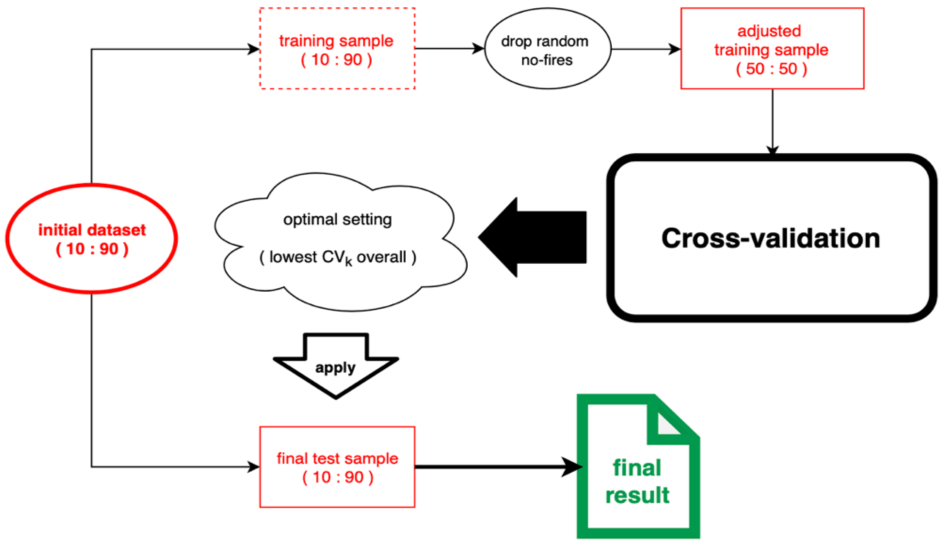


Figure 1, logistic regression: final test sample based on the training with the initial dataset

We knew that this was a quite unconventional measure and test results based on this unrealistic dataset would not be persuasive. We circumvented this problem by taking out 20% of the initial dataset as a final test set which we called “final test sample”. This set was built up with an actual distribution of fires and no-fires and was therefore appropriate for a final, realistic and credible test of our models.

The whole procedure from the initial dataset up to the final result is illustrated in the flow chart below. The next subchapter explains the step cross-validation and how we determined for every model an optimal setting which we finally applied on our final test sample.



training sample (10 : 90)
drop random no-fires
adjusted training sample (50 : 50)
initial dataset (10 : 90)
optimal setting (lowest CV_k overall)
Cross-validation
apply
final test sample (10 : 90)
final result

Figure 2, The whole procedure from the initial dataset up to the final result.

4.2 Cross-Validation

4.2.1 KNN and Logistic Regression

For KNN and logistic regression the procedure of cross-validation was very similar. Using cross-validation, we took the adjusted training sample and trained and tested it with the KNN and the logistic regression algorithm. This step was not only laborious in terms of programming, but it included also several difficult conceptual decisions. How should the training and test setting look like, or more precisely, how many and which covariates and different numbers of nearest neighbours (only for KNN) should our models take into account? We wanted to compare the performance of our model for different numbers of covariates and nearest neighbours, so we agreed to train and test it with different settings and to calculate the cross-validated empirical error (CV_k) for every setting in order to compare them.

The decision concerning the number of covariates was quite complex for KNN as well as for logistic regression. Since we worked with more than twenty different covariates, we faced a numerical problem: There are billions of possibilities how one can choose X covariates out of 25. Hence, we decided to just drop the weather-type covariates in order to reduce the number of different covariates. We dropped these covariates because a lot of the information they included was already provided by other covariates. The weather type “rain” for example is already indicated by the covariate “precip” which shows how much it rained during a specific day.

After this reduction 11 covariates remained, so the numerical problem was still there. Online, we tried to find variants of both algorithms which could solve this problem in an elegant manner but without any success. No other solution available, we finally decided to generate 1000 random combinations of different covariates for every specific number of covariates between 2 and 11 (note that for 11 covariates, we could obviously only test one combination since there is only one way of selecting 11 covariates out of 11). In the case of logistic regression, a setting finally consisted of the number of different covariates and their names. We then applied the logistic regression model on all these generated settings using cross-validation.

In the case of KNN, the final settings included one additional determinant: the number of nearest neighbours. Because of the relatively small dataset we decided to train and test the KNN model only with 1 to 10 nearest neighbours. For every given number of nearest neighbours between 1 and 10 we then tried all different numbers of covariates between 2 and 11, whereby we generated again 1000 random combinations of covariates for every given number of covariates. After the cross-validation we got a specific final list for both models (we called it "Endlist"), which included, among other performance measurements, the CV_k for all different settings of the specific model.

As a final step we had to identify which of the large number of different settings should be the optimal one obligated to use for the final test of both algorithms. We decided to focus in both cases on the empirical error and just choose the particular setting which provided the lowest CV_k overall. We did this despite the fact that this measurement might again have covered a high rate of false negatives just as during our first trials with the initial dataset. Nevertheless, we considered this as the optimal strategy because we feared that our model would predict a fire for almost every observation in case we had focused on the false-negative rate. However, concerning further improvements of our model, this might be a spot to take a deeper look at.

4.2.2 Neural Network

Using cross-validation we determined the covariates that performed the best in terms of the empirical error. We had to choose a different approach for the neural network than we used for logistic regression and KNN because building the model for the neural network took about 15 minutes. First of all, we split the data in a training and testing set. In terms of trial and error we found out that three covariates correlated very well among each other and used those as the “base covariates”. These were wspd, tmax and rh. In a next step we looped over every further variable in our dataset and built ten times a new model for the combination of these four covariates based on the training data. For each model we made a prediction for our testing data and calculated the empirical error. At the end of the loop we checked which combination of covariates led to a smaller empirical error than the combination of just wspd, tmax and rh. Based on this cross validation we identified wspd, tmax, rh, tavg, weather_11, weather_21, weather_26, weather_28, weather_29, weather_30, weather_33 as our covariates for the neural network.

We also had to choose the number of hidden layers of the neural network. It was very easy to determine this number since if we did choose less or more than two layers the neural network performed very badly. Only with two layers we received usable results.

5 Model Estimation

5.1 KNN

5.1.1 Results

The optimal setting for the KNN approach was built up as follows:

Number of covariates:	5
Covariates:	rh, HPW1, tmin, wspd, pressure
Number of nearest neighbours:	10

So, the setting with these five covariates and ten nearest neighbours provided the lowest CV_k overall. Once we got the optimal setting of our model, we could apply it on the final test sample. The results are presented in the table below. The bar chart indicates how KNN performed in predicting the two separate situations and how it performed overall (the bar on the right). The pie chart shows the share of all different types of errors and true predictions.

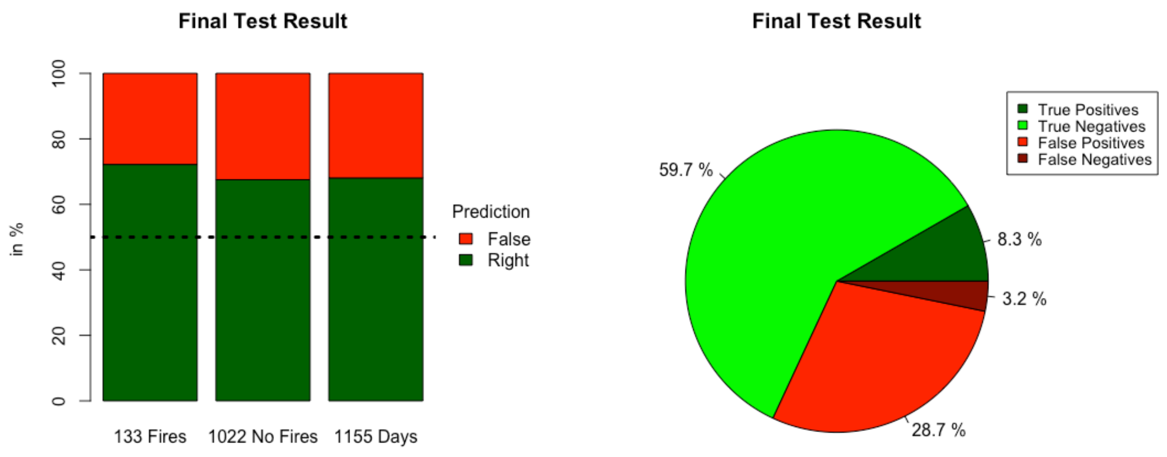


Figure 3, KNN: final test result based on the training with the adjusted training sample

5.1.2 Interpretation

The empirical error in the amount of 31.9% is not really satisfying. Nevertheless, the result in the final test describes an improvement in comparison to the one we got using the setting based on the initial dataset with only 10% fire outbreak observations. The reason for this is the much lower false-negative rate. It seems like the higher percentage of fire observations within the cross-validation dataset helped the KNN-algorithm to substantially improve its fire-forecasting performance at the expense of a significant higher empirical error due to an increasing rate of false-positives. However, from a safety-related perspective predicting a majority of all fire outbreaks and accepting a certain percentage of false alarms is better than foresee hardly any fire.

The simultaneous increase of the empirical error and the right-positive-rate gets more comprehensible when we look at the operating principle of the KNN-algorithm. It just predicts according to the K most similar observations. If we have 90% of no-fire observations within our dataset the chance that under the K-nearest neighbours of a new observation are several no-fires is higher than if only 50% of all observation within the dataset are no-fire observations. But this conclusion only holds if the observations of the fires and the no-fires are to some extend similar. This leads us to a possible explanation why the KNN-algorithm was not able to perform better. There might be days with optimal conditions for a forest fire outbreak, but the outbreak itself needs a trigger. In many cases this trigger represents not natural factors like our covariates, but coincidentally factors like human incautious for example. Such factors were not taken into account during our analysis. Consequentially, days with and without a fire outbreak may look exactly the same in terms of the covariates used. So, an observation of a no-fire might possibly have more fire than no-fire neighbours and therefore KNN fails to predict correctly. The same logic may apply for the opposite.

5.2 Logistic Regression

5.2.1 Results

The optimal setting for the logistic regression approach was built up as follows:

Number of covariates:	8
Covariates:	HPW2, tmax, pressure, wspd, HPW3, precip, DSLR, tavg

So, the setting with the listed eight covariates provided the lowest CV_k overall. Once we got the optimal setting of our model, we could apply it on the final test sample. The results are presented in the table below. The bar chart indicates how KNN performed in predicting the two separate situations and how it performed overall (the bar on the right). The pie chart shows the share of all different types of errors and true predictions.

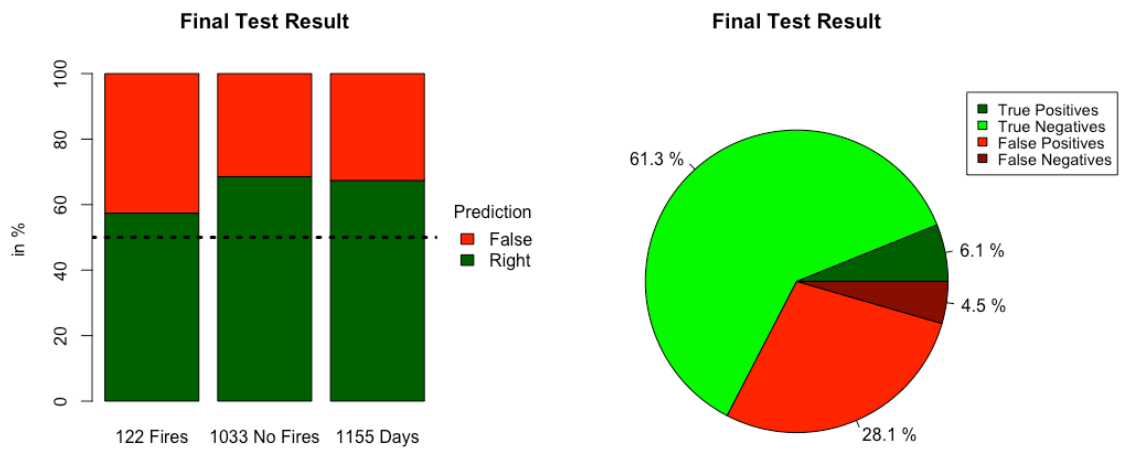


Figure 4, logistic regression: final test result, based on the training with the adjusted training sample

5.2.2 Interpretation

The result for the logistic regression shows a pattern similar to the one we observed within our KNN approach. We can observe an increase concerning the fire forecasting performance as well as an increase of the false-positive-rate and the empirical error. So, the adjustment of the initial data set towards the adjusted training sample we used for the cross-validation had again a noticeable impact. From a safety-related perspective we can state once more that the optimal setting provided by the adjusted training sample is preferable to the one of the initial dataset. It allowed us to foresee a slight majority of all fires. Nevertheless, we have also to state that logistic regression performed worse than KNN, not only referring to the empirical error, but especially also regarding to the rate of correctly predicted fires.

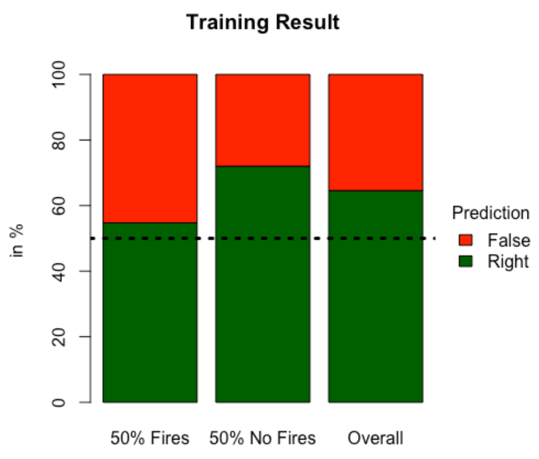


Figure 5, logistic regression: result of the training of optimal setting with the adjusted training sample

The comparison with the training results (see the bar chart on the right) gives us some indications for the reasons behind these rather disenchanting findings. At first glance, it seems strange that the empirical error in the training is higher than the one in the final test. But the reason for this becomes apparent when we also look at the percentage of right predicted fires and no-fires during the training. Both are quite similar to the ones reached during the final test. Furthermore, the algorithm performs better in predicting no-fires than forecasting fires in both cases. Consequentially, the decrease of the empirical error is not a surprise since the better performance in recognizing no-fires weighs more during the final test due to the higher share of no-fire observations within the final test sample. Having said this, we can state that the testing performance is similar to the one during training. Thus, it appears that overfitting is not a problem for our logistic regression approach. Otherwise we would observe a more noticeable difference between training and testing results. On these grounds, underfitting seems to be the major problem here. What attracts attention is the fact that the algorithm was noticeably better in predicting no-fires than fires. We can only make assumption about the reasons behind this phenomenon. One explanation could be that the used covariates are only good in detecting weather conditions which make forest fire outbreaks substantially unlikely, but they fail in covering the real mechanism behind a fire outbreak. The fact that the two precipitation measures "precip" and "DSLR", both factors which reduce the probability of an outbreak of a forest fire significantly, are part of the optimal setting is another indication which speaks for this conclusion.

5.3 Neural Network

5.3.1 Results

The optimal setting for the neural network approach was build up as followed:

Number of covariates:	11
Covariates:	wspd, tmax, rh, tavg, weather_11, weather_21, weather_26, weather_28, weather_29, weather_30, weather_33

So, this combination of covariates performed best. Once we got the optimal setting for our model, we could apply it on the final test sample. The result is presented in the charts below. The first bar of the bar chart shows how the algorithm performed in predicting the outbreak of a fire while the second bar reveals its performance for predicting days with no fire outbreaks. The third bar indicates the performance overall. This means that we were able to forecast 58.3% of all the forest fires.

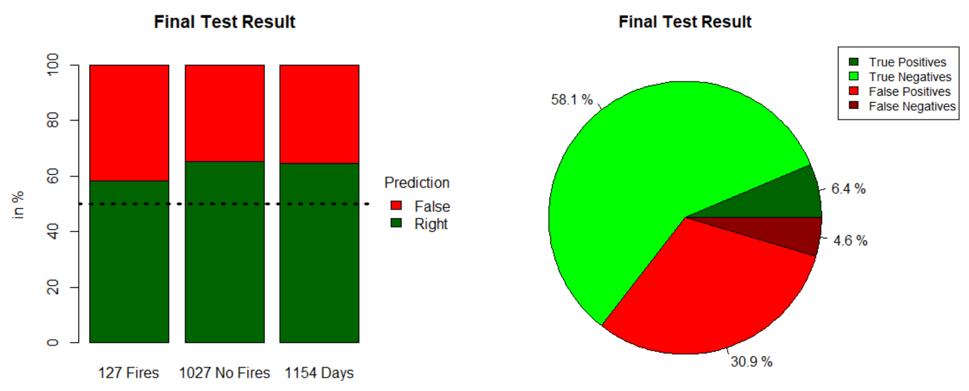


Figure 6, neural network: final test result, based on the training with the adjusted training sample

5.3.2 Interpretation

The results of the neural network based on the FinalTest-sample were similar to the results we observed with KNN and logistic regression. The neural network performed better than KNN but performed worse than logistic regression. In total we got 1'154 observations in the FinalTest-sample. The neural network predicted 357 false positives, 53 false negatives, 74 correct positives and 670 correct negatives. This leads to an empirical error of 35.5%. Although this number is relatively high, we would say it isn't too bad. Even if the climate fosters the breakout of wildfires, it doesn't mean that there must be a wildfire for sure. Also creating predictions with a neural network was very tricky. We found a convenient library (neuralnet) that did a lot of work for us. The difficulty was to understand what the function does exactly. The neuralnet package expected us to define a function for smoothing the result of the cross product of the covariates (or neurons) and the weights. The only two option for this parameter was "logistic" and "tanh". As we tried out, we constantly got better results with tanh function than with the logistic function.

The logistic activation function $f(x) = \frac{1}{1+e^{-x}}$ ranges from 0 to 1. It's also known as the sigmoid activation function. This function has some downsides when it is used for neural networks. Because the output of the function is always between 0 and 1 it could lead to the vanishing gradient problem where the gradient of weight vanishes or goes down to zero. So, the weight wouldn't be updated. Another problem is that the function isn't zero centred. This means the output can't be negative. This property makes it harder to optimize the weights because the weights are not allowed to move in all possible (negative) directions. In addition to this it is highly compute-intensive to calculate the derivative $f'(x) = \frac{1}{1+e^{-x}}(1 - \frac{1}{1+e^{-x}})$.[1]

The tanh (hyperbolic tangent) function $f(x) = 2 * \log(2x) - 1$ ranges from -1 to 1. Therefore, the function is zero centred in contrast to the logistic activation function and thus overcomes the non-zero centric issue of the logistic activation function. However, the vanishing gradient problem is still existing because the output of the tanh function can also be 0 and cause that the weights aren't updated. Also, the derivative of the tanh function is highly compute-intensive $f'(x) = 1 - (\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}})^2$. [1]

These findings also coincide with our observations we made during the project. If we used the logistic activation function for building our neural network model, we faced the problem that sometimes the model building method of the neuralnet package couldn't find suitable weights to represent the underlying relationship of the data. This problem didn't occur if we used the tanh function.

[1] Source: https://towardsdatascience.com/analyzing-different-types-of-activation-functions-in-neural-networks-which-one-to-prefer-e11649256209

6 Conclusion

In Conclusion, out of the three models tested, the KNN model performed the best, since it had the lowest empirical testing error and reached the highest percentage of correctly predicted fires at the same time. Although the KNN model performed the best, it would probably still be insufficient for a real-world application because the empirical testing error is still too high. There are a variety of possible reasons for this. Firstly, often no fires happen even though the weather conditions are ideal for a wildfire. The reason for that is that, for wildfires to happen, most of the time an external trigger is needed, for instance a campfire that got out of control. There is a lot of randomness involved in a wildfire happening, which means that there is a lot of noise, which makes it hard to find the real pattern. Additionally, it seems plausible that the covariates used in this project are rather good in detecting conditions which make fire outbreaks unlikely, but they insufficiently cover the real pattern of a such an outbreak. All of this complicates a successful training of a model that properly predicts wildfire outbreaks.

To enhance the models, there are some improvements that could be made. For example, for the logistic regression and KNN, not all possible combinations of the available 11 covariates were used. This means that there possibly are combinations of covariates which would lead to better results. Additionally, an enhancement of the models could be achieved by using regularization in our models. By introducing regularization, we could eliminate unrelated covariates and the search for the optimal setting would be more effective which would potentially improve the testing performance.

Another approach to get a better model would be to focus less on the empirical error and instead try to minimize the false-negative-rate. In the case of wildfires, one could argue that a higher empirical error due to more false alarms is to some extent preferable if it goes along with a higher rate of correctly predicted fire outbreaks. For this purpose, we could adjust our procedure and instead of choosing the setting with the lowest empirical error as the optimal we could for example focus on the setting with the highest percentage of correctly predicted fires. Another approach would be the alteration of the critical probability which determines the probability an observation has to reach in order to be classified as a fire. In our models, we constantly worked with a critical probability of 50%. By lowering the critical probability, we would be able to get a lower false negative rate. Although both modifications would undoubtedly lead to a higher empirical error and more false alarms they could still be reasonable for a real-world application on account of a better fire-forecasting performance.

Finally, we can state that our models might not yet be good enough for a real-world application. Nevertheless, they show that machine learning certainly has some potential to help foreseeing forest fires and reduce their humanitarian, environmental and economic damage.

Declaration of authorship

"We hereby declare

• that we have written this thesis without any help from others and without the use of documents and aids other than those stated above;

• that we have mentioned all the sources used and that I have cited them correctly according to established academic citation rules;

• that we have acquired any immaterial rights to materials I may have used such as images or graphs, or that I have produced such materials myself;

• that the topic or parts of it are not already the object of any work or examination of another course unless this has been explicitly agreed on with the faculty member in advance and is referred to in the thesis;

• that we will not pass on copies of this work to third parties or publish them without the University's written consent if a direct connection can be established with the University of St.Gallen or its faculty members;

• that we are aware that our work can be electronically checked for plagiarism and that we hereby grant the University of St.Gallen copyright in accordance with the Examination Regulations in so far as this is required for administrative action;

• that we are aware that the University will prosecute any infringement of this declaration of authorship and, in particular, the employment of a ghostwriter, and that any such infringement may result in disciplinary and criminal consequences which may result in our expulsion from the University or our being stripped of our degree."

By submitting this academic term paper, we confirm through our conclusive action that we are submitting the Declaration of Authorship, that we have read and understood it, and that it is true.