Bicycle traffic approximation with Decision trees

Miska Tammenpää
Computer Science student at Aalto University

Introduction

I am an avid cyclist. I travel around the Helsinki area by bike quite a lot. There may be a lot of other cyclists on the move when I wish to go cycling. Thus it would be nice to know what level of cycle traffic to expect. I can approximate this with a machine learning model based on decision trees.

Problem Formulation

The data points we will use in our machine learning model are hours of the day. Their features are the measured (mean) temperature and precipitation during the hour and a set of different attributes related to the time of observation such as the hour, the weekday, the month and the day of the month.

The label for each datapoint is a classification of the amount of cyclists. The labels are gained from a mapping (as shown in table 1) of the measured absolute amounts of cyclists to a discrete range of 1-3, where 1 = low traffic, 2 = medium and 3 = high. We use classes instead of absolute values for two reasons. 1. Absolute values don't tell us much in practice. 2. It's easier in our case to learn a useful model with classes.

Class	Cyclists
1	0-70
2	70-170
3	170→

Table 1: Amounts of cyclists and their labels

Method

We combine datasets of three different hourly observations; the precipitation, the weather and the amount of cyclists. All observations are made in Kaisaniemi, Helsinki from Jan 1st 2014 to Dec 31st 2020. Of course, this pivots the solution to be "prediction of cyclist traffic in Kaisaniemi" but the impact of this is trivial as the method can be easily expanded to contain data from any desired place. I used only the data from Kaisaniemi to simplify the dataset. The cyclist data is from Helsinki Region Infoshare (HRI, 2020) and the weather data is from the Finnish Meteorological Institute (FMI, 2021).

The method implementation is done in Python. We use the *pandas* (pandas, 2021) library for storing and handling the data, and *scikit-learn* (scikit-learn, 2021) for the machine learning models and model evaluation. First we create the labels from the data according to Table 1. The mapping is chosen simply as a personal evaluation of how much traffic is *low*, *medium* and *high*. Then we parse the hour, the day (of the month), the weekday, the month and the year from the datetime data. After this we remove all "null-rows" or rows, which contain at least one empty field. In the end we're left with 61 333 data points.

We are dealing with time series data that is highly non-linear and is to be separated into classes. Thus it makes sense to use a powerful non-linear classifier like decision trees. Decision trees are also very useful when working with feature-rich data as we don't have to know beforehand which features are the most meaningful to the data. However, simply using a single decision tree often leads to high overfitting. Due to this we will also use a random forest and compare the results.

The implementations used for the decision trees and the random forest are *scikit-learn*'s *DecisionTreeClassifier* and *RandomForestClassifier* structures. The supported loss functions for these are information gain (*entropy)* and gini impurity. While gini impurity is computationally more efficient (divineml, 2020), we will try using both and compare the outcomes. For training and validation, we use 55 000 of the first datapoints. The rest is used for testing. The data is split into training and validation sets with *scikit-learn*'s *train_test_split* -method with a 80/20 ratio respectively. Cross validation is unnecessary as the dataset is very large. For the random forest we try different amounts of decision trees ranging from 100 to 2 000 with a step size of 100.

Table 2: Single decision tree

	Gini	Entropy
Training accuracy	1.0	1.0
Validation accuracy	0.838	0.837

Table 3: Random forest with 200 trees

	Gini	Entropy
Training accuracy	1.0	1.0
Validation accuracy	0.877	0.880

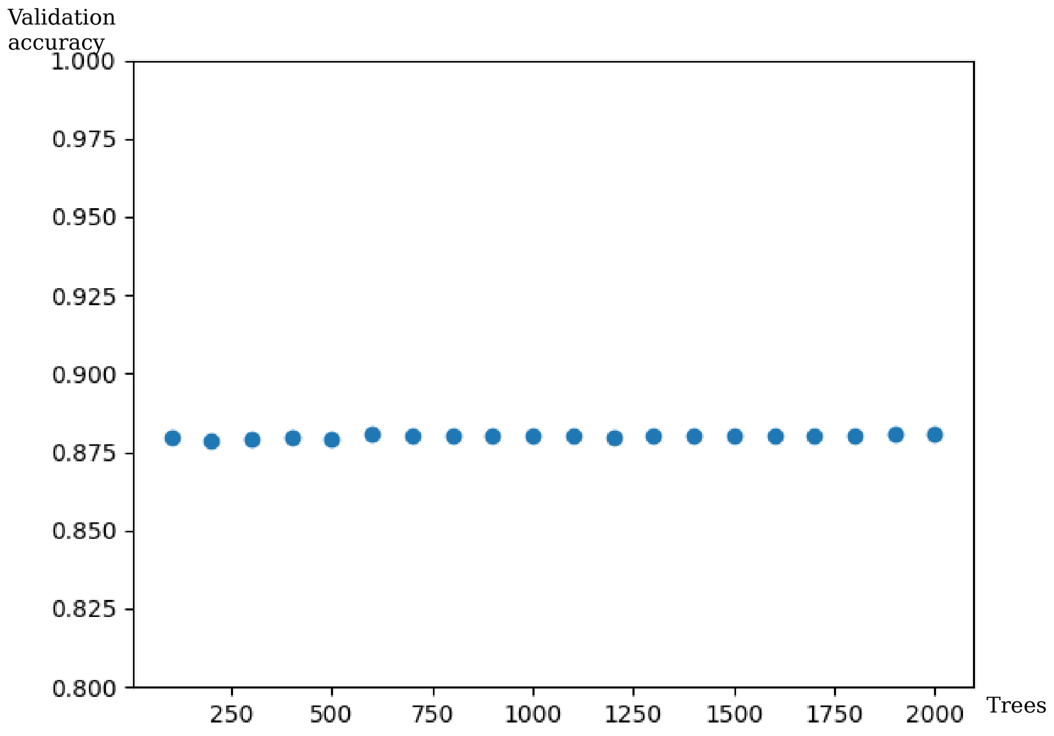

Figure 1: Random forest with increasing amount of trees

Results

From tables 2 and 3 we see that both methods are able to fit the training data perfectly, strongly indicating overfitting. However, they also both have a high validation accuracy so this notion is, in part, debunked. Nevertheless, there is a clear difference. A way to decrease overfitting here might be to use principal component analysis (PCA) before fitting the model. This way the less meaningful features in the data could be disregarded.

We also see from figure 1 that when increasing the amount of trees in the random forest, the validation accuracy remains effectively constant. As such, using larger amounts of trees in the forest is, in our case, unnecessary.

From the models in the tables, we choose the random forest with information gain as it's loss function, as it is the model with the highest validation accuracy. We use this model for predicting results on the testing dataset (6 333 datapoints). The resulting accuracy score is ~0.839 which is somewhat lower than the validation accuracy but still fairly high. As discussed in the first paragraph, this as well could possibly be improved in the future with PCA.

Conclusion

We have used decision trees and random forests to predict bicycle traffic levels from time series data. After training and validation, we chose our hypothesis to be a random forest with 200 trees and information gain as its loss function, since it had the highest validation accuracy of 0.880. The resulting test accuracy was 0.839.

Clearly, predicting bicycle traffic from time series data can be done to a high degree of precision. It is worthwhile, however, to explore some improvement opportunities. One of these is applying PCA on the dataset before training the model. The data could also be modified to not include data from the nights as that is largely irrelevant and only leads the ML model in the wrong direction. Another opportunity is exploring different classification methods such as deep learning models.

The application could also be expanded to include data from more areas and thus its predictions will be of more use to the user as they can then check traffic in specific areas and even decide where to go biking based on the traffic levels.

Bibliography/References

Cyclist data:
https://hri.fi/data/fi/dataset/helsingin-pyorailijamaarat

Weather data:
https://en.ilmatieteenlaitos.fi/download-observations

Inspiration for code:
https://www.pluralsight.com/guides/machine-learning-for-time-series-data-in-python

Gini impurity vs. Entropy
https://divineml.com/difference-between-entropy-and-gini-impurity-in-the-decision-tree/

pandas
https://pandas.pydata.org/

scikit-learn
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html