ELEN-0062: Introduction to Machine Learning

Project 3 - Human Activity Prediction

Maxime Goffart
180521

Olivier Joris
182113

Academic year 2021 - 2022

Contents

1 Introduction

In this report, we will present and justify the different techniques we used for the third project of the course. The goal is to provide the reasoning behind our choices and to present the results obtained on Kaggle.

2 K-nearest neighbors

First, we decided to use the K-nearest neighbors method because it was the one provided by the pedagogical team with the assignment. Yet, the reasons that made us trying multiple versions of this technique are the facts that it is easy to interpret and to use.

2.1 1-NN

Our first submission was the one provided with assignment[1] that was generated with the 1-nearest neighbor model[2].
This submission yields a score of 0.52 on Kaggle.

2.2 25-NN

Based on 2.1, we decided to increase the value of K of the K-nearest neighbors method to 25. The motivation behind the choice of this value for K was theoretical. Since the dataset is huge and can contain errors in the measured values, we though that increasing the value of K would increase the precision of the model because each prediction will not depend on a single neighbor that could have been misclassified.
This technique is implemented in the file `knn_basic_25.py`. This submission yields a score of 0.54 on Kaggle.

2.3 55-NN and 49-NN

We decided to keep using the K-NN method to study how well it could perform on the assignment. We decided to study the accuracy of the KNN technique by using a 10-fold cross validation technique with the negative log loss function as a scoring measure. We obtained the following graph of accuracies depending on the value of K:

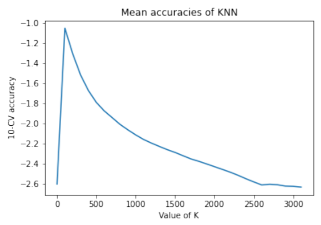

Figure 1: 10-CV accuracy of KNN depending on value of K

We could see that there is a peak. By using a divide and conquer technique, we found that the peak was obtained for values of K around 50. Thus, we decided to study the accuracy in a small range of K values around $K = 50$. We obtained the following graph:

[1]File `example_submission.csv`.
[2]Files `toy_script.py` or `knn_basic_1.py`.

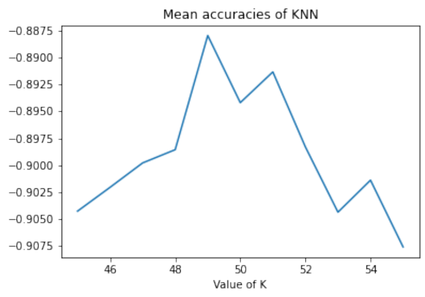

Figure 2: 10-CV accuracy of KNN depending on value of K

Based on the divide and conquer approach performed, we decided to use 55 neighbors. The choice of this value was motivated by the fact that we observed a peak around this value.
This submission yields a score of 0.53142 on Kaggle and the implementation of it is inside the file `knn_basic_55.py`.

Then, based on the graph of figure 2, we decided to use 49 neighbors. The choice of this value was motivated by the fact that it yields the highest score on the learning set by using a 10-fold cross validation strategy with the negative log loss function as a scoring measure.
This submission yields a score of 0.52571 on Kaggle which is less than the one for 55-NN. Yet, we expected a higher score based on the graph of figure 2. The implementation is available inside the file `knn_basic_49.py`.

Based on the two previous results, we came to the conclusion that using "vanilla" K-nearest neighbors was not sufficient. Thus, we decided to find new techniques explained in the following sections.

2.4 Multiple K-nearest neighbors models

We took a look back at the slides provided with the assignment and we focused on the features (sensors) of the datasets. We noticed that the features are not in the same units. Thus, we thought about using multiple 1-nearest neighbor models with one 1-NN model per feature. This idea was motivated by the fact that we though we would obtained a better accuracy by using one 1-NN per feature because measurements in different units would not be mixed.
We implemented this approach in the file `knn_splitted_1.py`. This yields a score of 0.52 on Kaggle which is the same as the toy script provided with the assignment.

2.5 Mutiple K-nearest neighbors models and samples modification

After the disappointing result obtained in 2.4, we decided to take a better a look at the datasets. We noticed that some samples were vectors full of -999999.99. We though that these measurements would badly influenced the models. Thus, we decided to replace each sample that was a vector full of -999999.99 by a vector full of 0.
We implemented this approach in the file `knn_splitted_1_filtered.py` and the score obtained on Kaggle was 0.53142.

2.6 Conclusion on K-nearest neighbors

After all the different tries perform using K-nearest neighbors models, we came to the conclusion that "vanilla" K-NN were not sufficient for this project.

The codes for the different plots are available inside the Jupyter notebooks `knn_basic.ipynb` and `knn_splitted.ipynb`.