Assignment 1

Machine Learning FCT NOVA
Claudia Soares, Ludwig Krippahl, Ricardo Ferreira

October 11, 2022

1 Dates and Rules

This assignment is for groups of 2 students.

The assignment can be submitted between October 10 and October 23. The deadline for submitting the assignment is October 23, 17:00 Lisbon time. There is an additional tolerance up to October 23 at 23:59 for solving any problems with your submission. This period should be used exclusively for this purpose. No submissions will be accepted after 23:59 of October 23.

You must submit by email to your Tutorial class instructor a zip file containing, at least, these five files, named exactly as specified (names are case-sensitive):

- TP1.txt This is the questions and answers file;
- TP1.py This is a Python 3.x script that can be used to run your code for this assignment;
- NB.png The plot of training and cross-validation errors for the KDE kernel width of your implementation of the Naïve Bayes classifier;
- REGRESS-TR-VAL.png The plot of the mean squared error versus the degree of the regression model;
- REGRESS-PRED-VS-TRUE.png The plot of the predicted miss distance versus the true miss distance.

Optionally, you can include other python modules if you wish to separate your code into several files.

2 Regression: Collision avoidance in Space

2.1 Our problem

Our Lower Earth Orbit (LEO) is occupied by a growing number of active satellites and space debris. It is essential for our life on Earth to monitor and prevent future collisions that can damage useful satellites, and, most importantly, create an even larger quantity of debris surrounding us (for more information on the space debris problem visit the European Space Agency's website).

Many observatories around the globe continuously monitor man-made objects in LEO and collect large datasets of their observations. These observations are analysed in the context of pairs of objects, that are believed to be engaged in a close encounter. Your task will be to use real data collected by the European Space Agency (ESA) to try to predict the distance of each two objects in the time of closest approach (TCA) of an encounter — an important measure of how dangerous the encounter really is.

2.2 The data

Load the data from this link and explore it. Notice you have seven columns in your dataset. Three of them refer to the objects relative position in 3D, other three contain the relative velocity in 3D and the last column contains the *miss distance*, the smallest distance that will separate the two objects in the near future. The miss distance is what you will predict.

Start by splitting the data in 80% for training and 20% for testing, e.g., by using `train_test_split` from the library `sklearn.model_selection`. Remember you will try different models and thus you should have a validation procedure in place. Next, you will standardize the data (check Scikit-Learn webpage documenting the class `sklearn.preprocessing.StandardScaler`).

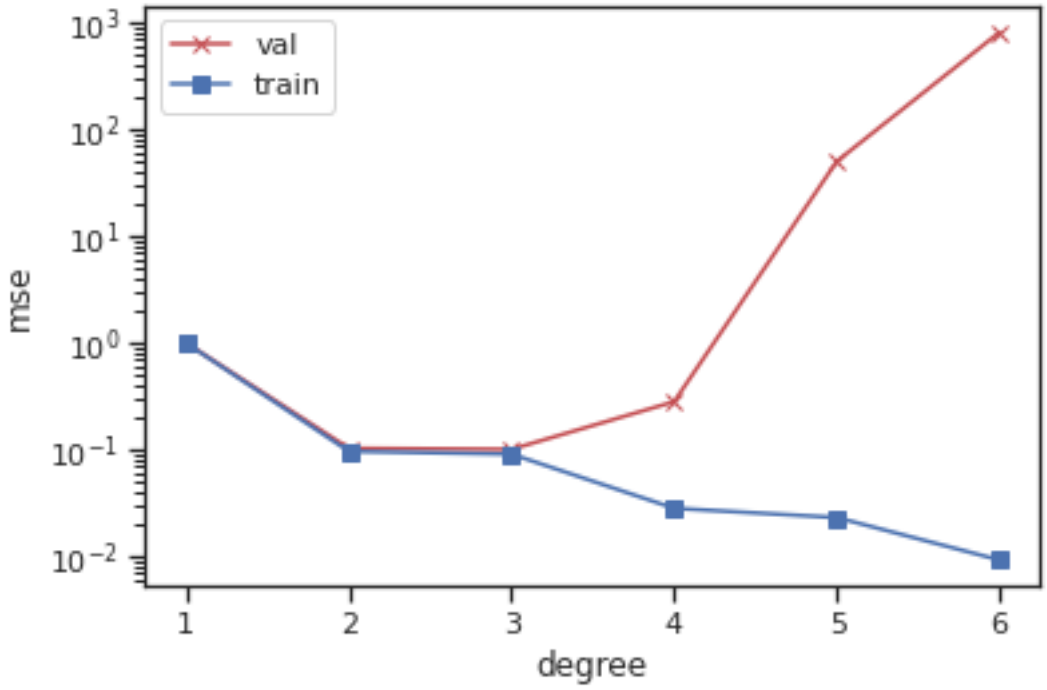

Figure 1: Example of a training and validation mean squared error plot across model degrees.

2.3 Polynomial Regression

Now you will use the `LinearRegression` class in `sklearn.linear_model`, jointly with the `PolynomialFeatures` class in `sklearn.preprocessing` to explore different models. The models to be tested are polynomial regression models with degrees from 1 to 6.

For each polynomial model, compute train and validation mean squared error. Plot the validation and training mean squared error across the different model degrees, obtaining a plot similar to Figure 1.

For all the degrees from 1 to 6 plot the true (x-axis) versus the predicted (y-axis) values of the miss distance. You should have 6 plots, one for each degree value, with blue points for the training set and red points for the validation set. In each plot you should also draw the line of slope 1 passing through zero.

Finally, after choosing your best model, estimate your true error when applying it to future unseen data.

3 Classification: Detecting Bank note fraud

The goal of this assignment is to parametrize, fit and compare Naive Bayes classifiers. The data set is inspired on the banknote authentication problem in the UCI machine learning repository, but the data was adapted for this assignment.

3.1 The data

The data are available on .tsv files where each line corresponds to a bank note and the five values, separated by tabs, are, in order, the four features (variance, skewness and kurtosis of Wavelet Transformed image and the entropy of the bank note image) and the class label, an integer with values 0 for real bank notes and 1 for fake bank notes. The training set can be found here, while the test set is here.

3.2 Classification

You must implement your own Naïve Bayes classifier using Kernel Density Estimation for the probability distributions of the feature values. You can use any code from the lectures, lecture notes and tutorials that you find useful. Also, use the KernelDensity class from sklearn.neighbors.kde for the density estimation. You will need to find the optimum value for the bandwitdh parameter of

the kernel density estimators you will use. Use the training set provided in the TP1_train.tsv. The second classifier will be the Gaussian Naïve Bayes classifier in the sklearn.naive_bayes.GaussianNB class. You do not need to adjust additional parameters for this classifier.

After training your classifiers and fine tuning all parameters, compare the performance of the two classifiers, identify the best one and discuss if it is significantly better than the others.

In addition to the code, you must include a plot with the training and cross-validation errors for the optimization of the bandwidth parameter of your Naïve Bayes classifier (the plot should be named NB.png). This plot should have a legend identifying which line corresponds to which error.

After selecting your best model, assess the true error in future unseen data.

4 Questions

You must answer a set of questions about this assignment. The question files are available here. Right-click the links and save the chosen file locally. You must include the file with your answers in the zip file when submitting the assignment.

5 Implementation notes

Process the data correctly, including randomizing the order of the data points and standardizing the values.

Determine the parameters with cross validation on the training set, leaving the test set for the final comparisons.

Use the same bandwidth value for all the Kernel Density Estimators in each instance of your Naïve Bayes classifier, and try values from 0.02 to 0.6 with a step of 0.02 for the bandwidth.

Use the default kernel ('gaussian') for the Kernel Density Estimators.

When splitting your data for cross validation use stratified sampling.

Use 5 folds for cross validation

Use the fraction of incorrect classifications as the measure of the error. This is equal to 1-accuracy, and the accuracy can be obtained with the score method of the classifiers provided by the Scikit-learn library.

For your Naïve Bayes classifier, you can implement your own measure of the accuracy or use the accuracy_score function in the metrics module if you find it useful.

For comparing the classifiers, use the approximate normal test and McNemar's test, both with a 95% confidence interval

Plagiarism

Plagiarism is any attempt to make you seem the author of text, code or any work which is not really yours. Plagiarism breaks the necessary trust for fair evaluation and any student submitting plagiarized work will fail the course immediately. If you rely on material other than what was provided in this course you must identify it and credit your sources. Also note that you will be graded on the work you actually did, so even if you avoid plagiarism by crediting your sources you still need to do your own implementation.