Foundations of Data Analysis - WS24
Lab assignment
Supervised learning

Due date: 09:45am on 11.12.2024

Description and Instructions

The maximum number of points achievable in this assignment is 100. Kindly follow the submission instructions carefully as failing to do so can **result in a penalty**.

- You should work on this assignment individually, however you are allowed and encouraged to discuss your approaches to the problems and questions with your colleagues.
- You are, however, not allowed to share your code! (Except for very small parts, in order to discuss problems with your peers.)
- Any act of plagiarism will be taken very seriously and handled according to university guidelines. Remember to cite every external source that you use!
- Upload your submission to Moodle as a jupyter notebook including both code and other answers. Your file should be named `<last_name>_<letter_first_name>.ipynb`, replacing `<last_name>` with your last name(s) and `<letter_first_name>` with the first letter of your first name, e.g., `luther_c.ipynb`.
- We provide a template file named `template.ipynb`. You are only allowed to use the libraries/functions that are imported in this file. If you wish to use different libraries, please ask for permission first.

  Hint: The imports aleady give away some functions to use for certain tasks.
- If you load the data it should be in the same directory as the notebook itself. This is done in the templates for you.
- For every time, a cell will not execute, we deduct points, so make sure your variable names are correct across cells and your code in general is running in the provided environment.
- We will execute your code in an environment created from the file `requirements.txt` [1]. We therefore strongly recommend that you recreate this environment locally as we will only count code answers that we can execute in the provided environment. To do so, you can create a virtual environment according to venv. Once you activated the environment, install all necessary packages from `requirements.txt`, e.g., by executing **pip install -r requirements.txt**. However, you can use conda environments (conda), too[2]. For more information and instructions consult the file `virtual_envs.pdf`.

Do not hesitate to email me (Christoph Luther) at `christoph.luther@univie.ac.at` or post on the discussion forum on Moodle with any questions you may have.

[1]In Python Version 3.10.14; If for any reason the specific version does not work for you, you can use any 3.10.XX but please let us know so in your assignment.

[2]e.g., `conda create --name fda python=3.10.14` + installing packages

Introduction

The purpose of this assignment is for you to put the theory about supervised learning algorithms we have discussed in the lectures into practice. You will implement some important concepts from scratch but also get to know a popular machine learning library for python, `scikit-learn`, among some other useful libraries, e.g., for visualisation. Please consult the respective documentations (found online) for questions on how to use the objects mentioned in this assignment.
You will first implement cross validation from scratch for a random forest classifier and in task 2 deal with non-linearly separable data.

You can download the data sets as well as the notebook template from ucloud:

- Link: https://ucloud.univie.ac.at/index.php/s/YRRLbcnsaxeYCxy
- Password: jjkNJBRDPG

Please make sure that you have access to the following seven files:

- `train.csv` and `test.csv` for task 1.
- `data_xor.csv` and `data_sphere.csv` for task 2.
- `template.ipynb`
- `requirements.txt`
- `virtual_envs.pdf`

1 Cross Validation for Random Forests [62 points]

In order to find the best hypothesis class among several candidates, you can use K-fold cross validation on the training data. Cross validation helps to evaluate risk-minimising hypotheses from a large number of different hypothesis classes that, for example, can vary by a hyper-parameter λ (hence, we will denote a single hypothesis class $\mathcal{H}_\lambda$). It works as outlined in Algorithm 1:

Algorithm 1: K-fold cross validation of hypothesis classes $\mathcal{H}_{\lambda_j}$, $j = 1, 2, ..., n$

```
Data: Data set $\mathcal{S} = \{(x_1^i, x_2^i, ..., x_d^i), y^i\}_{i=1}^m$;
Parameters: Number of folds K and index set $\mathcal{I} = \{1, ..., K\}$;
            hyperparameter set $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_n\}$;
1 Split $\mathcal{S}$ into $K$ (equally large) subsets $\mathcal{S}_1, ..., \mathcal{S}_K$;
2 for $\lambda_j$ in $\Lambda$ do
3     for $i$ in $(1, 2, ..., K)$ do
4         1. Find risk minimising hypothesis from hypothesis class $\mathcal{H}_{\lambda_j}$ using $S \setminus S_i$ as
           train data and denote it $h_{\lambda_j}^i$;
5         2. Calculate model risk $L_{\lambda_j}^i$ of $h_{\lambda_j}^i$ using $S_i$ as validation data;
6     end
7     Calculate average risk $L_{\lambda_j} = \frac{1}{K}\sum_{i=1}^K L_{\lambda_j}^i$;
8 end
9 return Parameter $\lambda_j$ that defines hypothesis class with lowest average risk.
```

For this task, you are required to use the data sets `train.csv` and `test.csv` that contain values for two features $X1$ and $X2$ and a target Y.

1. (1 points) An important tenet in data analysis is to 'look at your data'. A very first step is to just browse the data set yourself. Print the first 10 rows of the data set.

2. (2 points) Create an appropriate plot to see how (un-)evenly the values of the target are distributed. Use the train data.

3. (2 points) With `seaborn.pairplot()` you can visualise pairwise distributions of all variables in a data. Apply the function to the train data and display the plot.

4. *(6 points)* Train a random forest classifier using `sklearn.ensemble.RandomForestClassifier()` using an appropriate data set and make a model prediction on the test data. Additionally, print the train and test accuracy.

   Note: For `sklearn.ensemble.RandomForestClassifier()` *set* `random_state=0` *but do not set any other kwargs.*

5. *(3 points)* Compute the confusion matrix using `sklearn.metrics.confusion_matrix()` and visualise it appropriately. Use the test data.

6. *(5 points)* Write a function to calculate the model risk w.r.t. the 0-1 loss function directly from the confusion matrix. This means the input to the function shall be the matrix returned by `sklearn.metrics.confusion_matrix()` and the output the respective estimated model risk. Execute the function and print the risk from subtask 4.

7. *(4 points)* Create a plot with two subplots: On the left, a scatter plot of the features colour coded by the target and on the right, a plot showing the decision regions of the random forest from subtask 4.

8. *(4 points)* Create a learning curve for the random forest classifier and plot it. You are free to choose the steps for sample size on the x-axis.

   Note: Again, use `random_state=0` whenever you define a `sklearn.ensemble.RandomForestClassifier()` *but do not set any other kwargs.*

9. *(6 points)* Taking into account the results from above, what problem may the classifier suffer from? Explain in at most three sentences while explaining which results point to the problem.

10. *(16 points)* Perform 10-fold cross validation (using the training data) for a random forest to find the best value for the argument `max_depth` of the `sklearn.ensemble.RandomForestClassifier()`. Search over the values $\{1, 2, 3, 5, 10, 20\}$ and evaluate the model w.r.t. 0-1 loss. You have to write the code for cross validation from scratch and are only allowed to use `sklearn.ensemble.RandomForestClassifier()`, its methods `fit()` and `predict()`, `sklearn.metrics.zero_one_loss()`, the `numpy` library as well as every default python function.

    Note: Again, use `random_state=0` whenever you define a `sklearn.ensemble.RandomForestClassifier()` *but this time you also have to define the argument* `max_depth`.

11. *(3 points)* Train the best performing model from task 10 on the complete train data set and evaluate it with respect to 0-1 loss on appropriate data.

12. *(4 points)* Compare the average number of nodes per tree as well as the average maximum depth of each tree of the classifiers from subtasks 4 and 11.

13. *(2 points)* Compare the model from subtask 11 to the one from subtask 4. Which performs better? Why do you think this is the case?

14. *(4 points)* Limiting the depth of a decision tree can have advantages, which is used in a regularisation technique called pruning. Look up pruning and in your own words explain, what it is and what it is helpful for. Explain how decision trees are used in a random forest. Use at most five sentences.

2 On Linear Separability [38 Points]

In this task you are asked to analyse two data sets, `data_xor.csv` with features $X1$ and $X2$ and a target Y and `data_sphere.csv` with features $X1$, $X2$ and $X3$ and a target Y.

1. (6 points) Make a 2D scatter plot of the features of `data_xor` and a 3D scatter plot of the features of `data_sphere` colour coded by their labels. Plot them side by side as subfigures of a plot and correctly label all axes.
2. (5 points) Split both data sets into training and validation set with an appropriate ratio, train a logistic regression model for either data set and print their accuracy.
3. (3 points) Interpret the results from subtasks 1 and 2 together.
4. (4 points) Find a model that can classify `data_xor` perfectly, train the model and evaluate its accuracy.
5. (6 points) Write a function to project the features of `data_sphere` into another space. Create the scatter plot in this space and apply a linear classification model to the new data. You have to achieve 100% accuracy on the training data.
6. (6 points) Explain why and how your models from subtasks 4 and 5 were able to classify the data perfectly. Also explain what your function to project the features does exactly (if applicable, explain what it does geometrically).
7. (4 points) Explain Kernel Support Vector Machines and the Kernel trick in your own words. Use at most five sentences.
8. (4 points) Use a kernel SVM from `sklearn.svm` to classify the sphere data.

Good luck!