“Advanced Data Analytics”
SS 2021

Take-home exam 2

Simon Scheidegger

You have 24 hours to solve this take-home exam. The solutions to this exam sheet are due **Thursday, May 27th, 2021, 8.59am.**

You should answer the questions inside the *.py templates that are provided for every question (in the folder **Exam**), that is, exam_1.py, exam_2.py, exam_3.py. Any code in another file will NOT be corrected, i.e., Jupyter Notebooks will not be accepted for handing in solutions. All exam questions need to be solved by using the Python (3.x) programming language.

- Throughout the exam, we will ask you to save the results into a folder **res**.
- Please respect the folder name exactly. Note that if we say that you need to save the result into a folder: 'res/ex_1_a.csv', this means that you should create a CSV file called ex_1_a.csv, and this file should be stored into the folder 'res'.
- The code should run on Nuvolos.
- The script run_exam.sh will run all three answer sheets: exam_1.py, exam_2.py, exam_3.py.
- When we run this script on your Nuvolos account, the code does not run, you will get penalized.
- Please, once you have finished the exam, check that your code is running correctly by:
  a) opening a terminal in Nuvolos,
  b) going to the root of the exam folder and,
  c) typing the command: ./run_exam.sh.
  → This command will run all three codes and save the console output (what you printed with the function print()) and the error message into files: "out_exam_1.txt", "out_exam_2.txt", "out_exam_3.txt". Again, if when we launch './run_exam.sh' your code does not work, you will get penalized, so please check before submitting.

- To **submit your exam**, please proceed as follows:
  a) keep all your code in the exam folder in Nuvolos and check that the code runs correctly as described above.
  b) take a Nuvolos snapshot and name it "exam_2_2021".
  c) Download on your 3 codes exam_1.py, exam_2.py, exam_3.py, and the bonus answer file, and put it in a folder '<YOUR_FIRST_NAME>-<YOUR_LAST_NAME>-<YOUR_STUDENT_NUMBER>'. Submit this folder to **https://www.dropbox.com/request/cqtnf8y6ae2qAnnjOxjC**

- Note that the dropbox submission is a backup in case a problem occurs with Nuvolos. Unless there is a problem, we will ONLY correct your code on Nuvolos.

No late hand-in is accepted. A late hand-in will result in a grade of 1 (zero points) for the take-home exam.

You have to solve this take-home exam alone and therefore you have to submit your individual solutions. **No work in teams is permitted. Plagiarism will result in a grade of 1** (zero points) for this take-home exam.
Don’t forget to include comments into your source code. This will help us understand your thoughts when we correct it.

Moodle forum thread for questions during the exam:
https://moodle.unil.ch/mod/forum/view.php?id=1061852

We will answer your questions (first come first served) at 10:00, 13:00, and 18:00.

DO NOT answer each other's questions, it will be considered cheating

Good luck! Happy coding!

Exercise 1 (10 Points)

a) Load the dataframe 'dat_ex/pokemon.csv'

Produce a barplot which show on the x-axis the *types* of pokemons and on the y-axis the number of pokemon per *types*. Save the figure in: 'res/ex1_a.png'

!! The x-ticks should be readable. **!!**

b) Reduce your sample to the 3 *types* with the highest number of pokemons per types *types*. Create a single figure with a matrix of scatter plots. In each scatter plots you should show with different colors the 3 *types* on the x- and y-axis two of the characteristics of a pokemon: ['hp', 'attack', 'defense', 'special_attack', 'special_defense', 'speed']. Your figure should look somewhat like the example 'res/ex_1_b.png'. Note that your figure doens't have to have exactly the same color, markers etc. as te example. Your figure should nonetheless contains the same information and be at least as readable as the example.

Save this figure in 'res/ex1_b.png'

c) Reload the 'dat_ex/pokemon.csv' dataframe. Select only rows with a *types* in ['WATER','FIRE','GRASS','NORMAL'].

Define the variable **X** as the datagrame with columns ['hp', 'attack', 'defense', 'special_attack', 'special_defense', 'speed'].

As **y** you should transform columns 'type' into a matrix where each columns represent a specific type (['WATER','FIRE','GRASS','NORMAL']). In other words, in the matrix **y** the cell in the first row and first columns should be equal to 1 if the first pokemon of your data is of type 'WATER'.

Define the first 120 rows of this sample as a train sample, and define the rest as a test sample.

d) Train a SVM classifier with a linear kernel to predict the class **y** with the input **X**.
Select the hyperparameters 'C' (in sklearn) by finding the value C in [0.05, 0.1 , 0.15, 0.2 , 0.25, 0.3 , 0.35, 0.4 , 0.45, 0.5 , 0.55, 0.6 , 0.65, 0.7 , 0.75, 0.8 , 0.85, 0.9 , 0.95] which get the highest accuracy on the test sample.

e) Retrain a SVM on the full sample with the best parameter 'C'.

Print the confusion matrix with the line print("EX1e) confusion: '\n'", confusion_matrix)

Find the name of the pokemons which are support vectors.

Save the names of these support vector pokemons into a csv 'res/ex1_e.csv'

Exercise 2 (5 Points)

a) Load the dataframe 'dat_ex/pokemon.csv'

Define the matrix **X** as the dataframe with columns: ['hp', 'attack', 'defense', 'special_attack', 'special_defense', 'speed']

Train a Gaussian Mixture Model with 2 gaussians on the matrix **X**.

Use your model to assign a class to each pokemon in the sample.

For each class, find the pokemon *TYPE* most present and store it into a datarame: **class_type**

Print the name of those type with the line print("EX2a) confusion: '\n'", **class_types**)

b)

In this exercise you will try to find how many gaussians we need to have more than one class with the same **class_type**. In other words, how many gaussians in the mixture of gausians results in at least two classes having the same most present *TYPE* in it.

To do so: define N = 2
1) Retrain the Gaussian Mixture with N gaussians.
2) Compute the most present type in each class.
3) If no two class have the same most present type, increase N (N=N+1) and repeat.

Exercise 3 (15 Points)

dat_ex/otp.csv contains data on trains being on time or not. You are going to try to predict if a train arrives on time or not.

a) Load the data on trains and drop all rows with at least one missing variable.

Create a binary "y" variable equal to 1 if the trains arrived on time, and equal to 0 if the train did not arrive on time.

In a single figure show on the x-axis the hour of the day at which the train arrive, and on the y axis the percentage of train which arrive on time per hours. In the same graph, show with a horizontal line the average number of trains which arrive on times. Save this figure in 'res/ex3_a.png'

Does the hours of the day contains information about the probability that a train is on time? Your answer should contains two parts:
i) give an answer which only analyze the figure ex3_a.png
ii) give an answer which only think rationally about why a train is or isn't more likely to be on time at a particular time of the day.

Print your answer with:
print('Answer 3_a_i)',**your_answer_part_i**)
print('Answer 3_a_ii)',**your_answer_part_ii**)

!! In the second part of this exercise, you will train three neural networks, and test their performance. For each exercise you will have to select the first 80% of the sample as a training sample. The next 10% as a validation sample. The last 10% will serve as a test sample. **!!**

b) Use a LSTM neural network with a single 10 neurones layer to predict if the next train is on time. Train the model with only 1 epoch of training. You can chose the other hyper-parameter of the network.

Your input should be a time-series of the y-variable with 10 lags.

Print the dimensions of the shape of your training X = X_training_shape with the line: print('EX3b) shape:',X_training_shape)

Compute the confusion matrix on the test sample, print it with print("EX3b) confusion: '\n'", confusion_matrix)

c) Train a second LSTM network with same hyper-parameters and number of epochs as in Ex3 b).

As an input use a TWO-DIMENSIONAL set of time-series: 1) the y-variable with 10 lags, 2) a time-series of the hours of the timeStamp columns with 10 lags.

Print the dimensions of the shape of your training X = X_training_shape with the line: print('EX3c) shape:',X_training_shape)

Compute the confusion matrix on the test sample, print it with print("EX3c) confusion: '\n'", confusion_matrix)

d) Train Deep-Forward Neural Network to predict if a train is on time or not.

As an input variable you should use the *origin station*, the *next_station*, and the time from the *timeStamp* column.
You need to pre-process the *origin station* such that you have one column per possible *origin station* with a value equal to 1 if the train came from this station.
Similarly, you need to pre-process the *next_station* such that you have one column per possible *next_station* with a value equal to 1 if the train came from this station.

If you pre-process the data correctly, you should have 344 columns as input to your neural network.

Train y neural network with ReLu activation function and three layers of respectively 64,32, and 16 neurones (not counting the output layer).

print the confusion matrix with the line: print("EX3d) confusion: '\n'", confusion_matrix)

e) If you can chose a threshold to maximize the recall of the three neural network trained previously, what would be the highest recall you could obtain for each model (hint, you don't need to code anything to answer this question)

Bonus 3 (2 Points)

!! We know you have not seen this in class (that's why it's a bonus), you may need to google some definitions to be able to answer it correctly **!!**

!!You can chose to answer this question in a .txt file *bonus.txt*, or if you work on paper, with a picture of your work *bonus.png* **!!**

Every month, I want to predict stock returns (R_t) at time t with prices (P), earning-ratio (E), and inflation rate (I) at time t-1, and t-2.

I organize my input for each month as a matrix 2 by 3. For example:

	price	ER	I
t-1	P_1	E_1	I_1
t-2	P_2	E_2	I_2

That means that if my training sample has N points, I my training set will be a tensor of dimension: (N,2,3)

I wish to use a neural network with the following architecture:

- A single *1D Convolutional Layer* with
  - one output (filters = 1 in tensorflow 2)
  - a tanh activation function.

Write down the equation F of the neural network as a prediction function: R_t = F(P_1,P_2,E_1,E_2,I_1,I_2)

a) Define explicitly and clearly every single parameters in your equation (hint, if you done your job correctly, you should have 7 trainable parameters, including the constant).

b) If I change the architecture of my neural network to the following:

- First, a *1D Convolutional Layer* with
  - two neurones (filters = 2 in tensorflow 2)
  - a *swish* activation function.
- A second *Dense Layer* with
  - An input dimension of 2 (the output of the convolutional layer)
  - A single neurones (the model's output)
  - A *tanh* activation function

How many parameters, does the model have? If you don't know, take a guess and tell us you guessed randomly. You won't be penalized, we just want to test the wisdom of the crowed.