“Advanced Data Analytics”
SS 2021

Take-home exam 1

Simon Scheidegger

You have 24 hours to solve this take-home exam. The solutions to this exam sheet are due **Thursday, April 1st, 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 sheet will NOT be corrected. 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_1_2021".
  c) Download on your 3 codes exam_1.py, exam_2.py, exam_3.py and put it in a folder '<YOUR_FIRST_NAME>-<YOUR_LAST_NAME>-<YOUR_STUDENT_NUMBER>'. Submit this folder to **https://www.dropbox.com/request/MSgFkkdcJGurPkVLvfrD**

- 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=1035123

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!

<u>Exercise 1 (10 Points)</u>

a) Load the data in **'data/StudentsPerformance.csv'** into a pandas dataframe and:

- Transform the columns so that they can be used by a quantitative model,
- if the columns are binary (two possible values), replace those values with 0, and 1 if the columns are categorical (more than two possible values) create one column per possible value with 1/0 indicating if the row belongs to that particular category.
- Create a column y with the average of math, reading, and writing score.

IMPORTANT: save the first 5 rows of your dataframe into the folder res: 'res/ex_1_a.csv'

b) Create the following exploratory graphs and save them as *.png in the folder res.

- Create a single figure 'res/ex_1_correlation_x.png' in which we can see the correlation between the three reading graphs (you can use subplot to put multiple graphs into one figure)
- Create a figure 'res/ex_1_correlation_xy.png' with 5 subfigure, one for gender, race/ethnicity, parental level of education, lunch test, and preparation course. Each subfigure should illustrate the correlation between the variable of interest and the variable 'y' created in a). The type of plot for each subfigure can be different.
- Create a figure 'res/ex_1.dist.png' with four subfigure, one for, math score, reading score, writing score, and one for 'y'. Each sub-figure should illustrate the shape of the distribution of the variable of interest.

c) Define the first 700 rows of the dataset as your train sample and the last 300 as your test sample.

- Take as input all non-score variables and try to predict the variable 'y' with an OLS model and a random forest. Train your models on the training sample and measure the performance on the test sample.
- Compute the mean absolute error, mean square error, and R^2 on the test sample for both models. Organize those three result per model into a single pandas dataframe and save it into a CSV ('res/ex_1_c_oos.csv')
- Code a random cross-validation procedure where you: a) select randomly 30% of the dataset to be a test-sample, b) train the random forest on the train-sample, and measure performance on the test sample.
- Run your random cross-validation procedure 100 times and show in a single performance the mean, std, min, max, 5% quantile, 99%quantile, and median of the mean square error, mean absolute error, and R^2. Organize those results into a single pandas dataframe and save it into 'res/ex_1_c_oos_CV.csv'

Exercise 2 (10 Points)

In this exercise, we will use the dataset **'data/bankrupt.csv'**. The variable of interest 'y' will be the binary "Bankrupt?" which is equal to 1 if the company went bankrupt, 0 otherwise.

a) Compute the correlation matrix between all the variable and save it into 'res/ex_2_corr.csv'

b) Create a single dataframe containing the mean, std, min, max, 5% quantile, 99%quantile, and median of each feature and save it into res/ex_2_descr_1.csv

c) For each feature in the dataframe (all columns except 'Bankrupt?). Create a graph with two histograms on top of each other. Each figure should show two **density** in 100 bins. By density, we mean that the histogram should be standardized such that the heights on the figure represent a probability. The first density should correspond to the distribution columns for bankrupt firms, the second for non-bankrupt firms.
Make sure that we can see both distributions and compare them (i.e., if the two histogram overlap, we should be able to see the value of the histogram below the other)
Save each of these graphs into a folder 'res/fig_c/'.
The name of each graph should be x+'.png' where x is the name of the column where:
i) you replaced all space with underscore
ii) you remove all % sign
iii) you remove all parenthesis '(' or ')'
iv) you remove all '/'
v) you remove all currency symbols.

d) Split the sample as follow: row 0-6000 -> train sample, rows 6001-6820 --> test sample

Train on the train three different models on the training sample: a k-nearest neighbour (kNN), a Naive Bayes, and a decision tree. You can select the models' hyper-parameter as you wish.

In a single dataframe, report each model's in- and out-of-sample:

- recall
- precision
- F1 score
- True Positive Rate
- False positive Rate
- True Negative Rate
- False negative rate

Save the results in a single dataframe called 'res/ex_2_perf.csv'

Note: on point d), you can use a library for the model, but you should estimate the performance measure using only NumPy and pandas.

e) An ROC curve is a graphical representations of a binary classifier’s performance. We have given you all the *ingredient* to construct such curve in the class and you can find a good definition on it here: https://en.wikipedia.org/wiki/Receiver_operating_characteristic.

Use the out-of-sample forecast of your logistic regression model to draw an ROC curve in a plot and save it in 'res/ex_2_roc.png'. For this question, you can use the packages matplotlib, pandas, and NumPy but not other packages.

HINTS:

- An ROC curve is a graph that shows on the X-axis the **False positive rate**, and on the Y-axis the **True positive rate** for different decision thresholds.
- The logistic output is a probability of being in one class instead of another. The decision threshold is the probability above which you decide to classify the points into one class instead of another.
- To plot a ROC curve, you will need to compute **True positive rate** and **False positive rate** for a grid of decision threshold, then organize your result to plot the curve.

Exercise 3 (10 Points)

Use the same data as in exercise 2.

Using only NumPy and pandas as external libraries, you will code a k-Nearest-Neighbours algorithm. You can use any algorithm you wish, but the simplest to code is the "brute force."

We provide you with a template of a class. All you have to do is complete the functions (remove the "pass" and write some code!)

To get full marks in this exercise, you can never use any LOOPS in the KNN class, in other words your code should not use the key word ***for***.

a)

__init__(self,n_neighbors, dist_type = 1): should simple store the number of neighbors 'n_neighbors' in the object, and the dist_type.
If dist_type == 2, your algorithm should use the *euclidean* distance to determine nearest neighbors
If dist_type ==1, your algorithm should use the *manathan* distance to determine nearest neighbors (you may have to use google to find the definition of the *manathan* distance)

b)

def fit(self, X, y): should take an X and y and store it in the object. If you chose to use any other algorithm than "brute force," you can also do some preliminary computation in this function. To simplify, assume y is always a vector containing 1 or 0 values only.

c)

def predict_single_prb(self,x): takes as an input a vector of predictors (same dimension as the number of columns in X) and yields out a probability of this new point being in category 1.

d)

def predict_multiple_prb(self, X): takes as input a matrix of new points and yields a vector of prediction using **self.predict_single_prb**. fs

e) Use your KNN class, the training sample defined in exercise 2, and train 3 models. One with n_neighbors=5, one with n_neighbors=10, one with n_neighbors=15. All three model should be using the **manathan** distance (dist_type=1)
For each model, compute the out-of-sample probability of being bankrupt for row 27,28 and 29 of the test sample (that is, row 6027,6028,6029 of the full dataframe)
Save the predicted probabilities in a single dataframe and save it in 'res/ex_3_manathan.csv'

f) Compute the exact same prediction as in point e) but with euclidean distance (dist_type=2). Save the results in 'res/ex_3_euclid.csv'.

g) Does the predicted probabilities change between e) and f)? Explain why. Put your answer in a *.txt file here: 'res/ex_3_g.txt'.