ExamUppgift#1 – Effective python (git, Venv , testing)

Individual exam.

Publish: 2024-09-1

Due date: 2024-09-14 kl 23:59 – (after special extension)

Points and betyg : Ttl point 200 , Point >140 => G , Points > 180 => VG

Project 1:	We need to create a python desktop app that makes our work to find the best **regressor** or best **classifier** for the data we get from our customers simple and informative as possible. Important: Kindly note • This project will be built into **new Python 3 virtual environment**, and in git repo (optional: GitHub **private** repo) with **declaration** of: Requirements.txt, .gitignore **(15 points)** • **Using Python OOP** only. • **Testing** for every class and method below is a **must**. *70% of task points are for testing and 30% for coding.* The project consists of these parts: 1- Read in from the user via terminal , configuration file or Graphic user interface what type of supervised ML they need to find is it **regressor** or **classifier**. **(3 points).** 2- Read in and validate the path or/and .csv file name. **(3 points).** 3- Read in and validate dependent target (y) by reading column name after printing all columns name to the user. **(3 points).** a. Will be great to validate at their choice is right by checking if dependent value is **continuing** or **Categorical**. **(5 points).**

4- then the app needs to do:

a. Check if the data is ready for the machine learning process or not.
b. In case it is not we need to print report to terminal at data is not ready and exit. **(3 points).**
    i. Or : In case it is not: Will be great if we **report why it is not ready** and give hints about why it is int ready like:
    we have missing data in these columns …. Fill in them and rerun the app.

    data has Categorical / String data and need to create dummies, do u like to create / convert them? (yes or no) …. Etc… **(5 points).**
c. In case data is ready: and it is **regressor** we need to find: **(20 points).**
    i. Program in python the needed to create ML model of each **regressor** we learned:
    ii. LiR,
    iii. Lasso,
    iv. Ridge,
    v. Elastic Net,
    vi. SVR,
    vii. ANN Deep learning model (you can use ANN Class we created before)
    viii. And print a report per each **regressor**.
        1. best parameter after applying GridSearchCV, CV =10. **(Optional FOR ANN MODEL)**
        2. MAE,
        3. RMSE,

4. R2 Score.
5. Your feedback/conclusion: which is the best for this data!

d. In case data is ready: and it is **classifier** we need to find: **(35 points).**
   i. Program in python the needed to create ML model of each **classifier** we learned.
   ii. LoR,
   iii. KNN,
   iv. SVC,
   v. ANN learning model (you can use ANN Class we created before)
   vi. And print a report for each **Classifier**.
      1. best parameter after applying GridSearchCV, CV =10. **(Optional FOR ANN MODEL)**
      2. Plot the confusion matrix per each.
      3. Print classification report per each.
      4. Your feedback/conclusion: which is the best for this data!

e. At the end we need to ask the user if they agree with your feedback about the best ML model. And if yes what the name of the model to dump it out. **(8 points).**

Project 2:	**Computer game: What is your lucky number?** Let's program a game between CPU and human player (min 18 years old) with python. Important: Kindly note • This project will be built into **new Python 3 virtual environment**, and in git repo (optional: GitHub **private** repo) with **declaration** of: Requirements.txt, .gitignore **(15 points)** • **Using Python OOP** only. • **Testing** for every class and method below is a **must**. *70% of tasks' points is for testing and 30% for coding.* **Game steps and rules:** 1. Once we start the game, CPU asks to input the player's full name and CPU saves it into variable **player_name**. player_name should be a string of characters only (no numbers) and one whitespace only between first name and last name. ***(5 points)*** 2. Then CPU asks to input player birthdate as yyyymmdd, CPU validate this input: it is (a year in yyyy, a month in mm and a day in dd.) If all ok, then we save it into variable **player_birthdate.** ***(5 points)*** 3. CPU calculates the player age from the player_birthdate, and CPU saves it into variable called **player_age.**

(5 points)

4. CPU compares if player_age more or equal 18 (as current years is 2022) if True so we move to step #5, if not CPU asks for both steps #2 and #3 again.

(5 points)

Once we have the right player_name and age:

5. CPU generates a lucky list of 9 integers between 0-100 and saves it in variable called **lucky_list**.

(5 points)

6. CPU generates a lucky number between 0-100, and saves it into variable called **lucky_number**, and adds it to the lucky_list so lucky list will have 10 integers now.

(5 points)

7. CPU prints the lucky_list and asks player to pick the lucky number from the lucky list, CPU saves the user input into variable called **player_input.**

(5 points)

8. If the player chooses the luck number, then CPU print: **"Gongrats, game is over! And you got lucky number from try#{tries_count}** 😊 **"** ,where tries_count is variable to count how many times the player tried to get the lucky number.
   then CPU prints to player:
   "Do you like to play again? (Input y: Yes, and n:NO)
   if *yes* so we go back to step #5, if *no* so we just

exit!

(10 points)

9. Otherwise wrong guess, CPU generates new list that contains numbers from the lucky_list and has only numbers that *differ 10* from the lucky number and save it into **shorter_lucky_list**. Check this example:

   lucky_list = [5,1,20,99,70,12,22,2,89,15]
   Luck_number = 12
   player_input = 70
   Shorter list will be between **min 12-10 =2** and **max 12+10 = 22**
   So
   shorter_lucky_list = [5,20,12,22,2,15]

   (20 points)

10. So, CPU print: **"this is try#{tries_count} and new list is: {shorter_lucky_list}, choose the lucky number?"**

    (10 points)

11. If the player chooses the lucky number so we are in step #8 again.

    (5 points)

12. Otherwise, CPU deletes the wrong number, that player entered in step#10, from the shorter lucky list and back again to step #10.

    (5 points)

13. So, game ends upon choosing the lucky number or shorter_lucky_list is just 2 integers left!