1. **PROJECT TITLE:**

**Data complexity aware RideNN ensemble classifier for dynamic classification**

**2. HARDWARE REQUIREMENTS**

OS-Windows 10

RAM-8GB

ROM-More than 100 GB

GPU-Yes

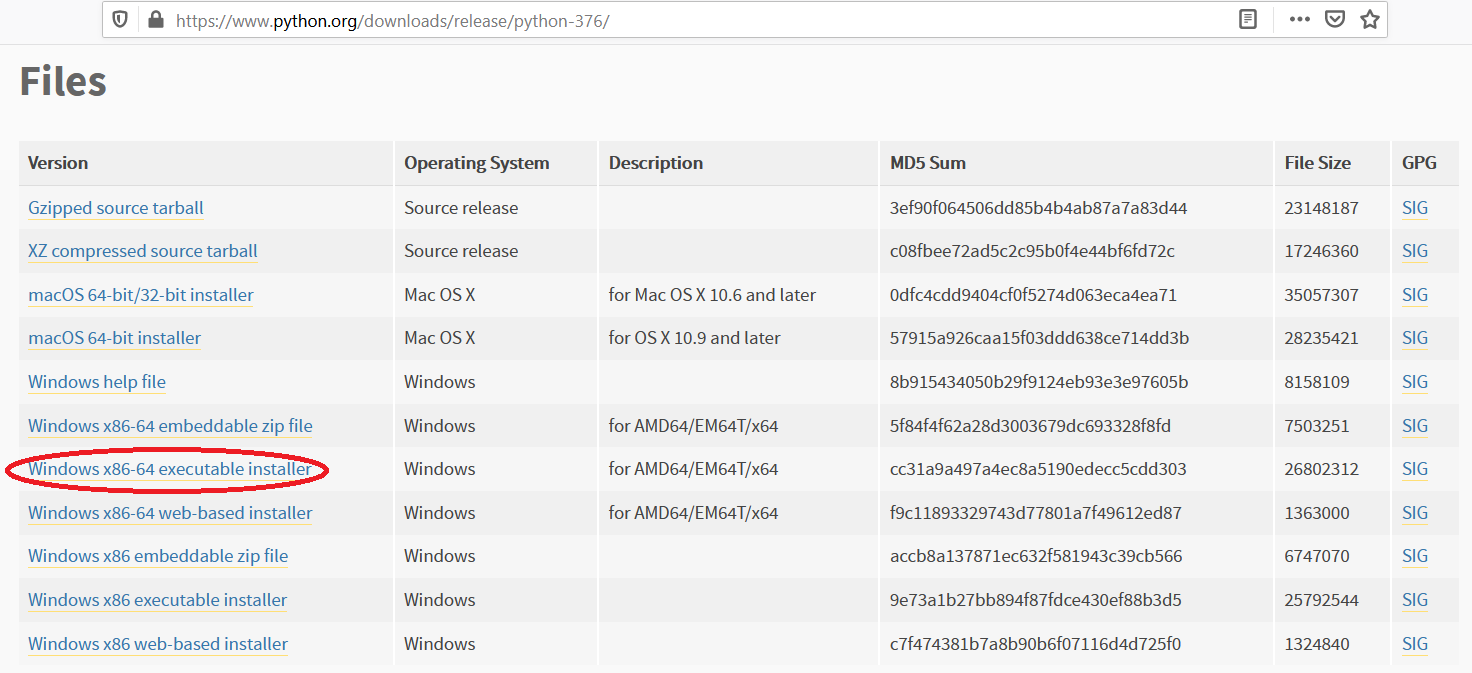
CPU-1.7 GHz

**3. SOFTWARE REQUIREMENTS**

Software name(**Python**): Version: 3.9.11

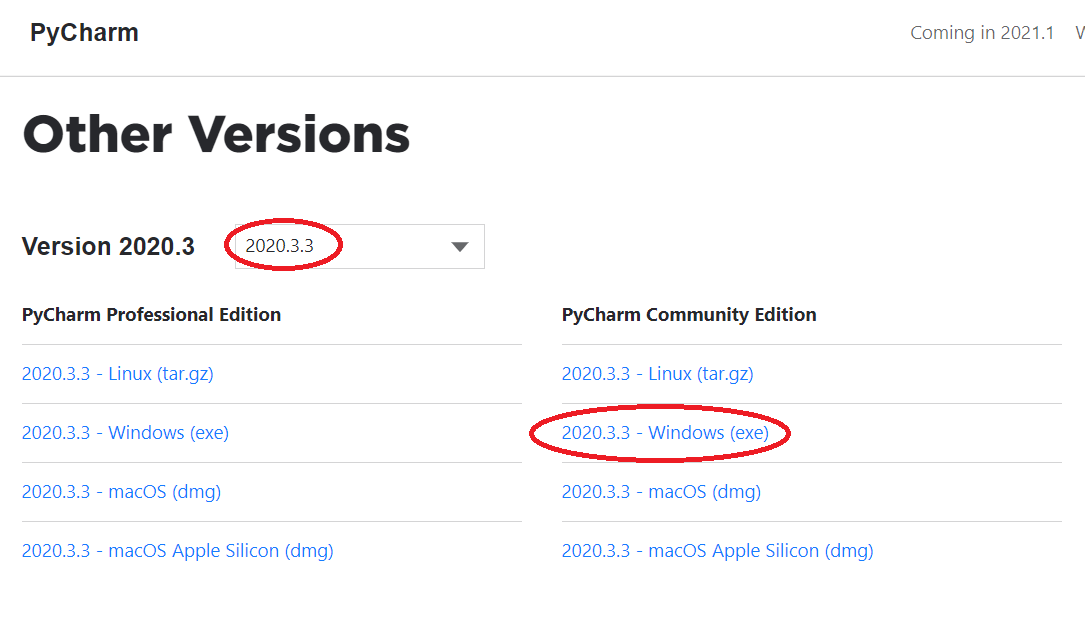
(Download link: <https://www.python.org/downloads/release/python-376/> )

Click -> Windows x86-64 executable installer.



Software name: **PyCharm**: Version: 2020.3.3

(Download link: <https://www.jetbrains.com/pycharm/download/other.html>)



(For installation procedure, please refer the doc “steps to install python.doc”)

**4. HOW TO RUN**

**Step 1**: Loading the project in PYCHARM

* Open pycharm
* Go to File, select Open browse the project from your drive and select it. So that the project will get loaded into the Pycharm.
* For the first time, Pycharm will take some time to load the settings.
* Please wait if any process is loading on the bottom of the screen.
* Check the Project Interpreter (File -> Settings -> Project: 285520-> Project Interpreter).

If this location “(C:\Users\---\AppData\Local\Programs\Python\Python39-64\python.exe) is not presented, then add this ‘python.exe’ from the installed location.

* In Pycharm Terminal(bottom left), type the comment “pip install -r requirements.txt”

**Step 2**: Run the program and getting the results

* From 'current project folder' window in pycharm, Open 285520**-> Main->GUI.py**’ and click run button
* In GUI window,
* 1) Enter Training data(%)(eg:60,70,80,90)
* 2) Click START, after some time the result will be displayed

3) Click Run Graph to view the current result graph.

[Expected Execution time expected: **15 – 20 minutes**]

* **Step 3**: Generate the graphs plotted in the paper
* From 'current project folder' window in pycharm, open ‘285520-> Main->Result\_graphs.py’, and click run button.

**5. IMPORTANT PYTHON FILE AND DESCRIPTION:**

**Main-> GUI.py**: User Interface, code starts here

**Main-> Run.py**: Main code

**Proposed\_ADA\_ViT ->Ridder\_NN.py :** Dynamic ensemble selection of component classifiers using RideNN – test data ‘t’ – weighted ensemble  predict the class label by N classifier – calculate the distance of  ‘t’ each rideNN – assign weight to each classifier based on distance of the test data ‘t’  -- Final prediction is the maximum weighted class label

**Main-> Result\_graphs.py**: displays graphs in paper.