Artificial Intelligence REPORT

Alejandro Garcia Pacheco

Github for all three tasks: https://github.com/AlejandroGarciaPacheco/AICW.git

Q1)

This report contains two algorithms. Heuristic and Dijakstra’s Algorithm. The user is allowed to select how they want to enter values in cells, either randomly or they can choose the probability of a number appearing in the grid. Each algorithm will display the amount of time taken, and the quickest path with a producing graph with axis of distance by size of the array. The results were taken as an average of 10 trials.

Heuristic Results:

* In a grid size 5 by 5 we get with random producing values the following time: 1.29676ms
* In a grid 100 by 100 we get with random producing values the following time: 23.93484ms
* In a grid size 5 by 5, the chosen probability of a number appearing in a cell is 0.1 and we get the following time: 0.64993ms
* In a grid size 100 by 100, the chosen probability of a number appearing is 0.1 and we get the following time: 22.81332ms

In Dijkstra Algorithm you can select the starting and finishing position. To allow an appropriate comparison, we will be choosing the same starting and finishing position. (0,0) and (width-1)(height-1)

Dijkstra Results:

* In a grid size 5 by 5 we get with random producing values the following time: 2818.90821ms
* In a grid 100 by 100 we get with random producing values the following time: 4393.36991ms
* In a grid size 5 by 5, the chosen probability of a number appearing in a cell is 0.1 and we get the following time: 4289.43467ms
* In a grid size 100 by 100, the chosen probability of a number appearing in a cell is 0.1 and we get the following time: 4013.78632ms

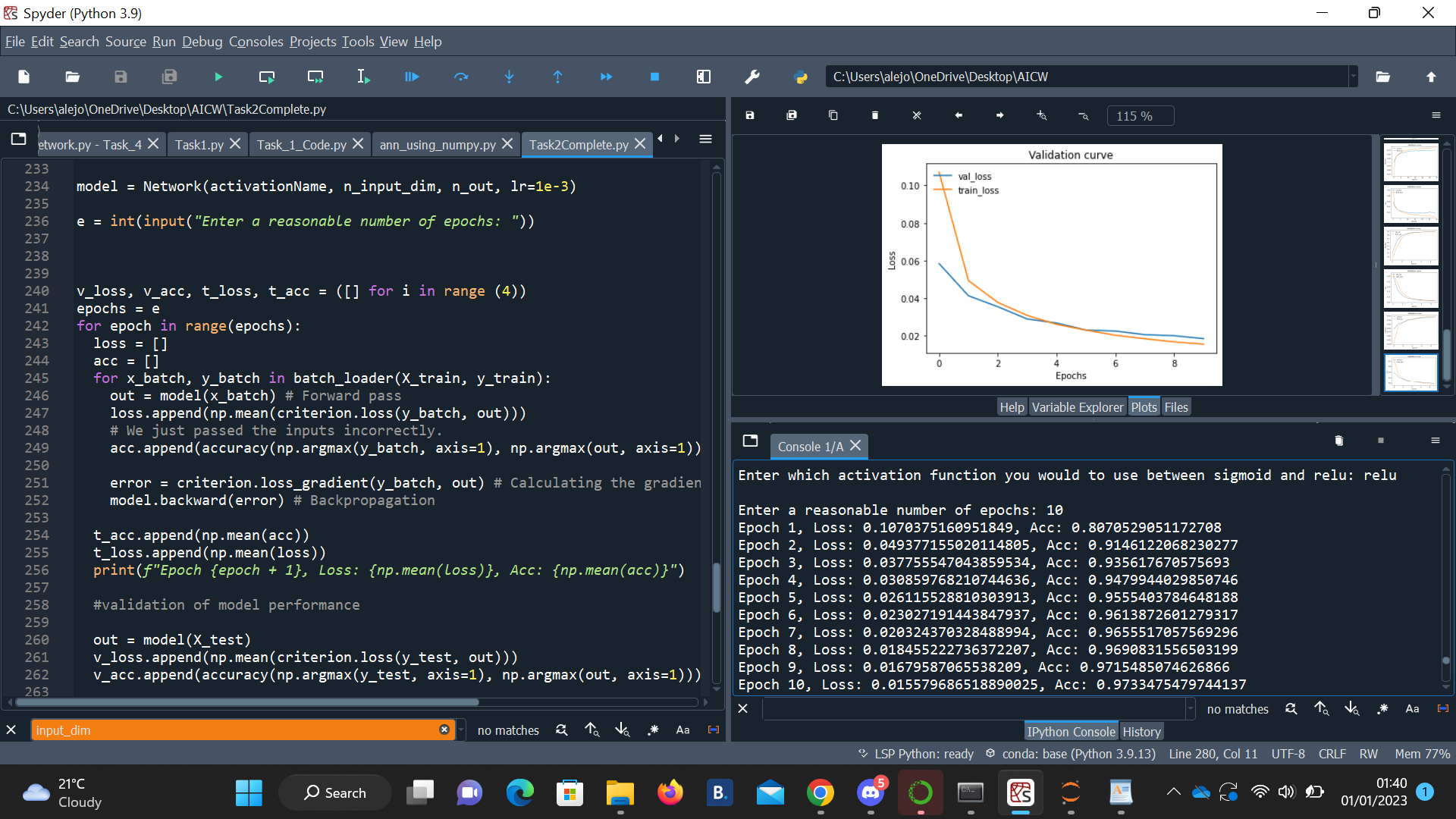
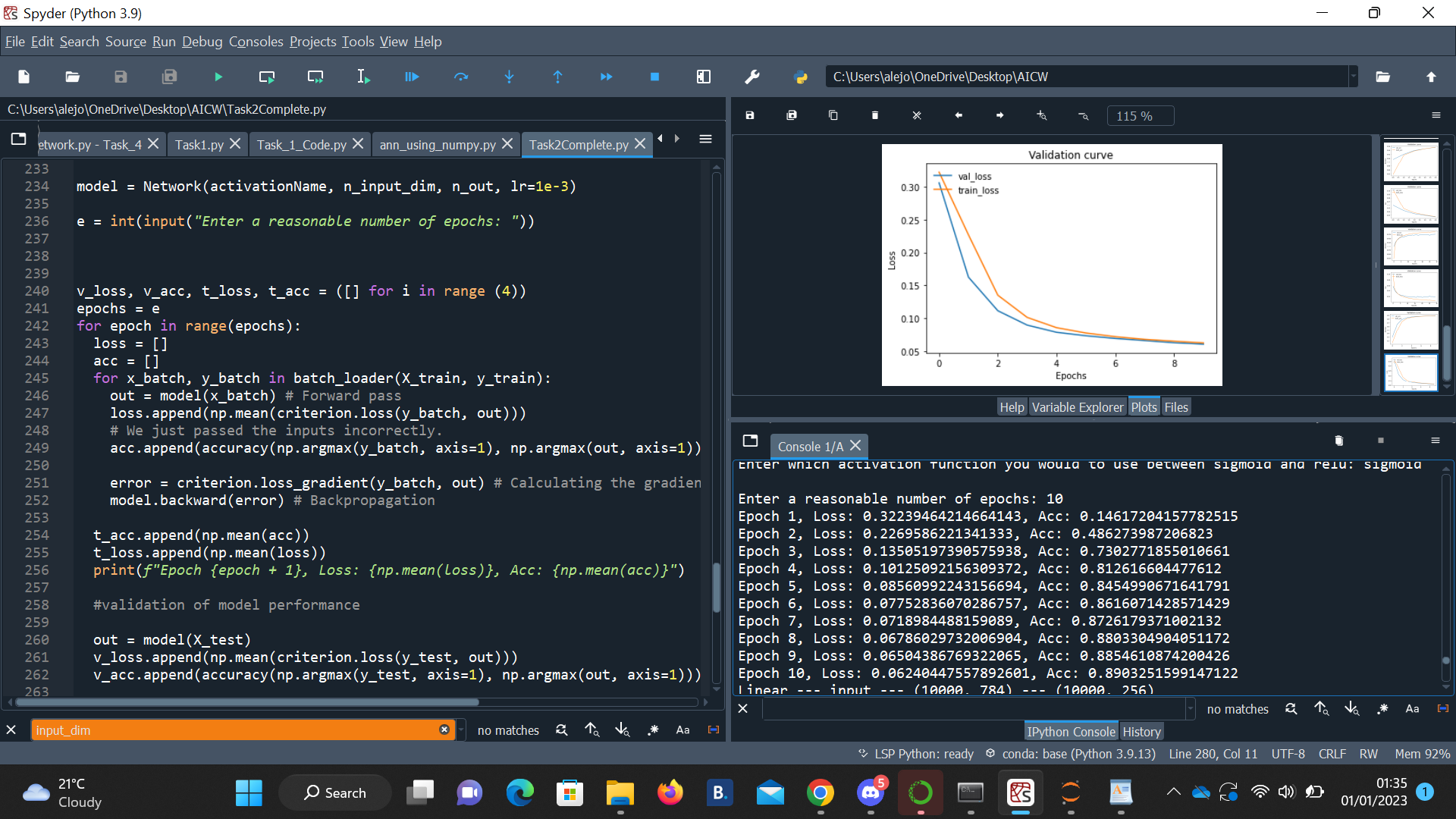
In a random producing value, we find a longer time taken to find the shortest path compared to the assigned probability to a number appearing on a cell for both algorithms. As expected, depending on the size of the array we get different results. Assigning for both algorithms the probability of a given number, it produces faster results than random distributing values in cells. The faster algorithm in a smaller size array, the heuristic algorithm produces faster results. However, in a larger size array, like 100 by 100, by far Dijkstra algorithm produces faster results.

Q2)

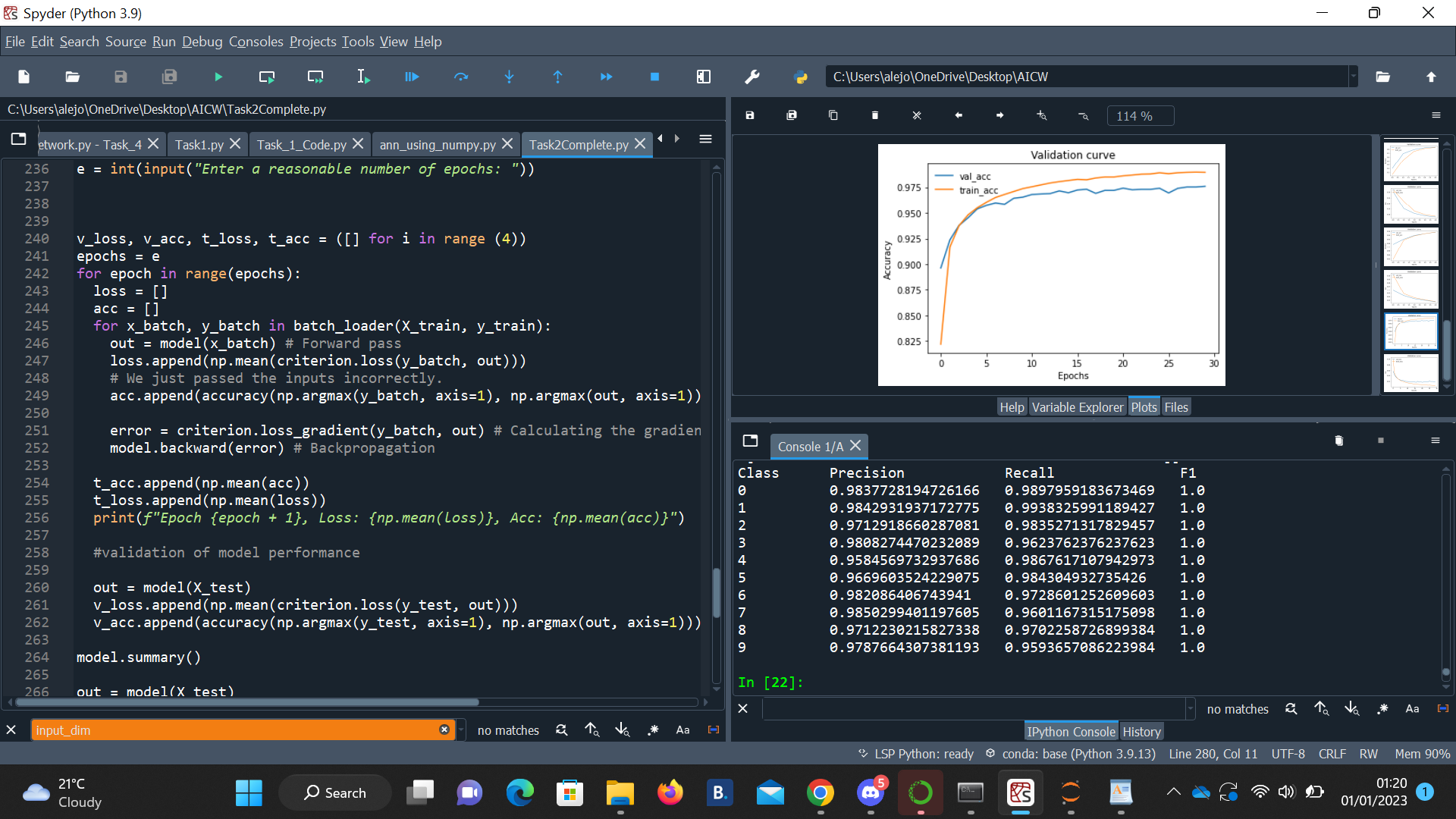
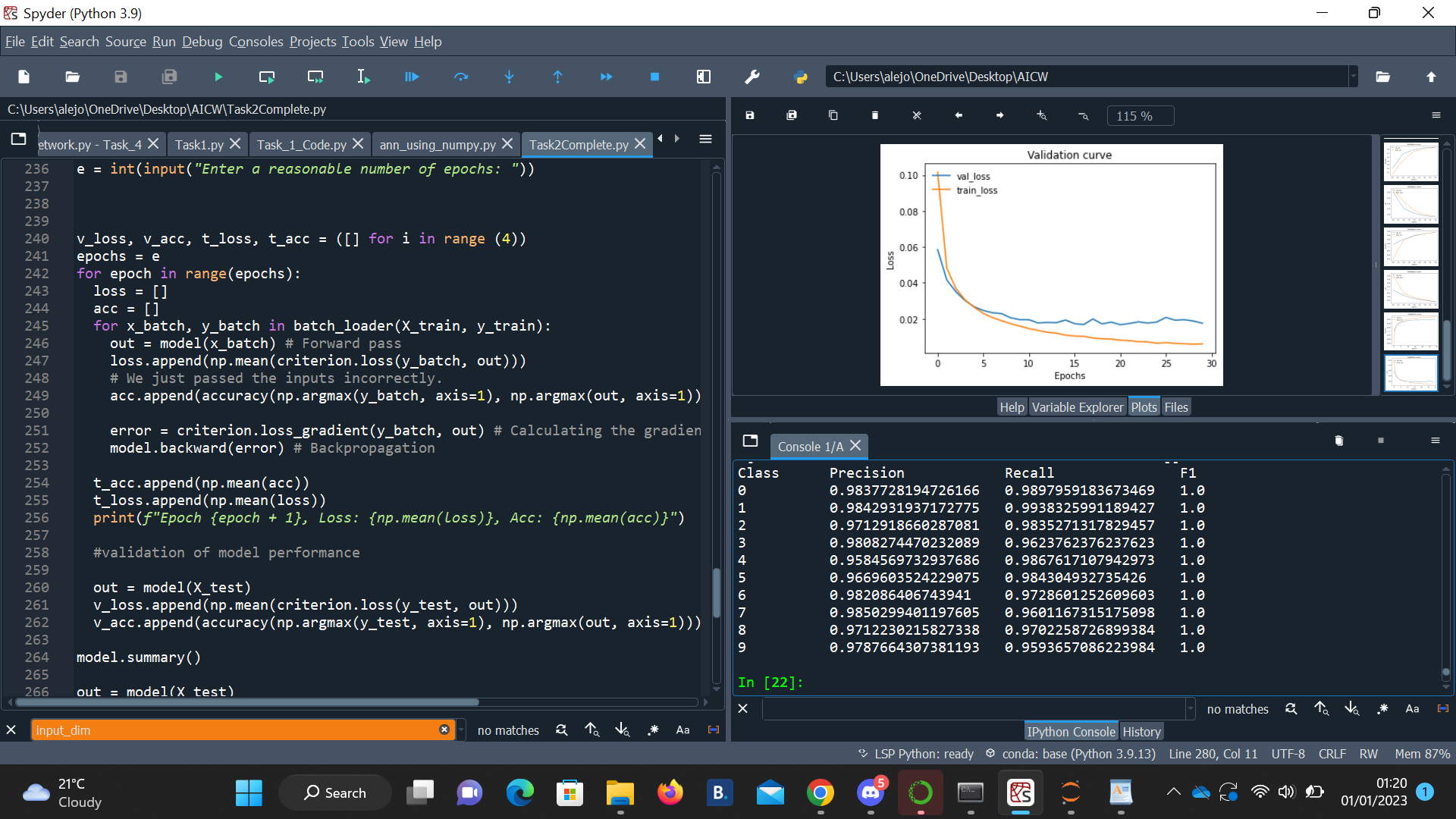
The model is built using a multi-layer neural network where the user can input which activation function as well as edit the number of nodes in hidden layers and inputs the number of epochs they want to execute.

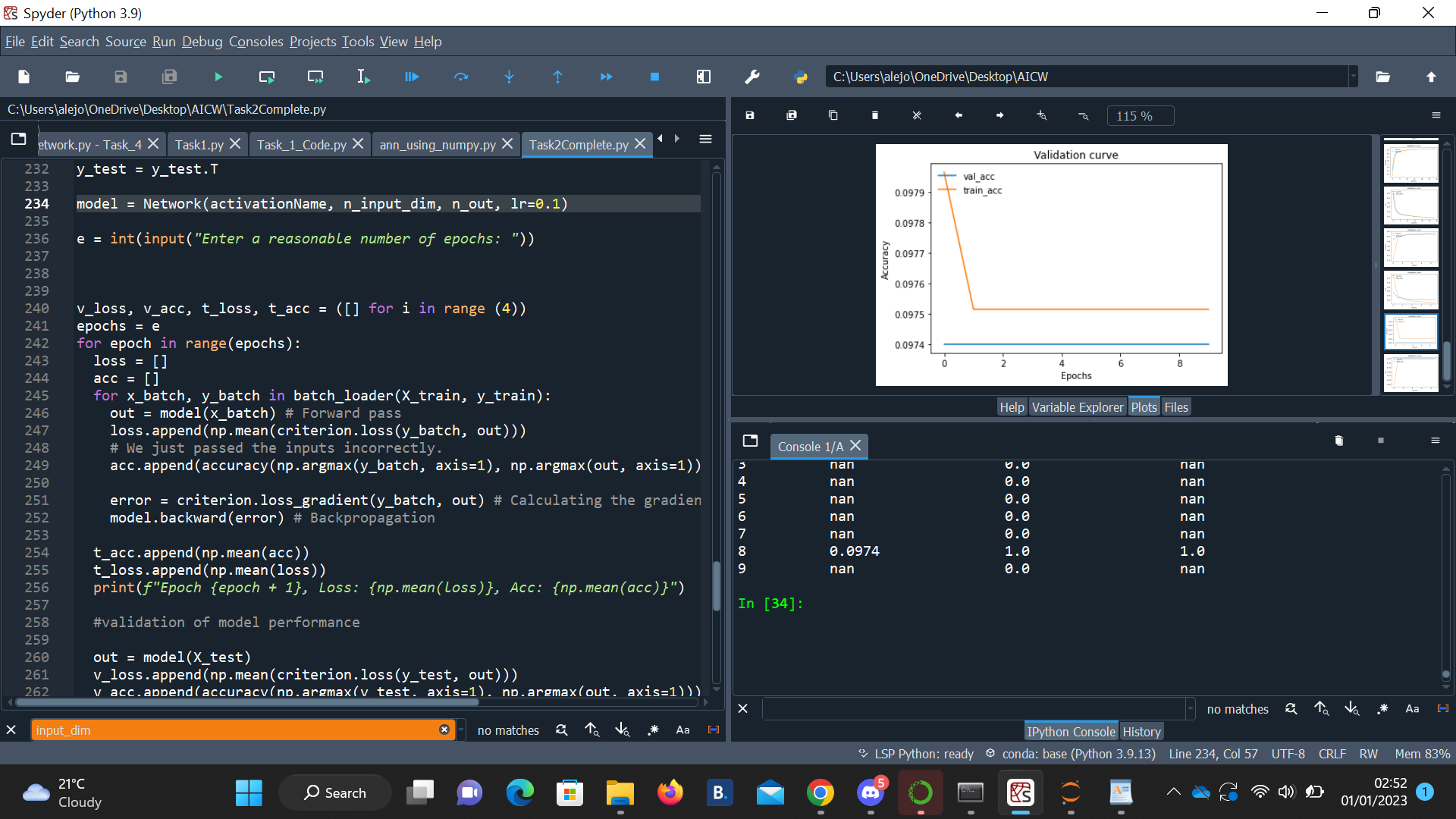
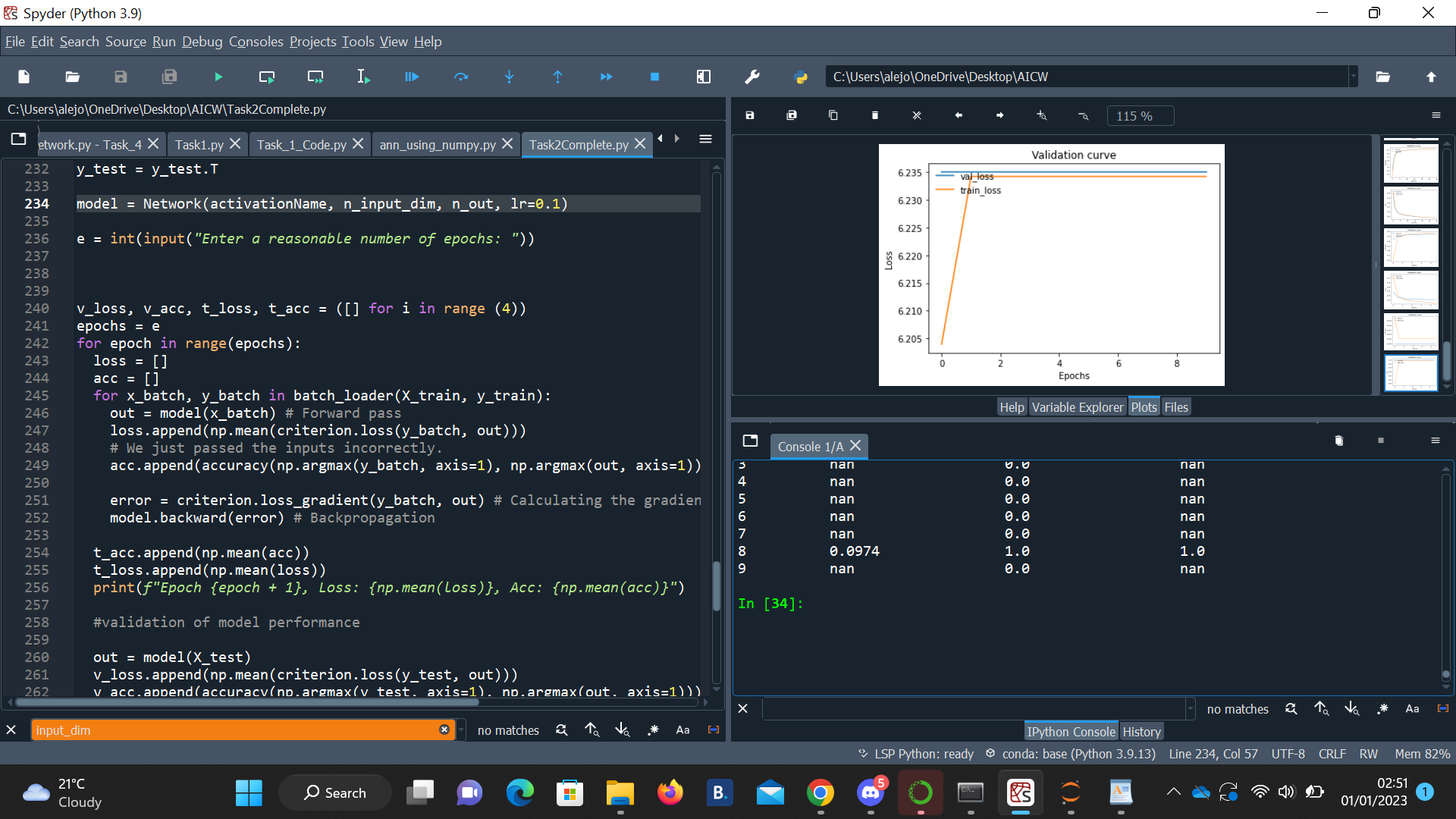
To undertake the same conditions for two comparisons, one being taken for a single layer and the other is for multiple hidden layers, they were both executed with the same hyperparameters. The trial was executed for 10 epochs, learning rate = 0.1 and using sigmoid as our activation function. We notice a big difference in the results. We see that the single hidden layer produces a much more accurate with minimal loss than the multi-hidden layer model. Similar results were shown running relu as the activation function.

relu is the more efficient as it starts with a higher accuracy than sigmoid with a small learning training loss. As we perform the training, we notice that relu reaches 0.97 accuracy result in comparison to sigmoid where at the tenth echo, reaches 0.89. We can see that sigmoid learns at a much higher rate than relu so even after lowering the run rate from an accuracy of initiating at 0.81 whereas sigmoid starts at 0.14. Sigmoid has a higher learning rate as the results present the accuracy result has a significant increase per iteration through training and due to its larger drop in train losses, almost halving per each echo as shown in screenshot 1, where the first three echoes dropped almost half the values per iteration.

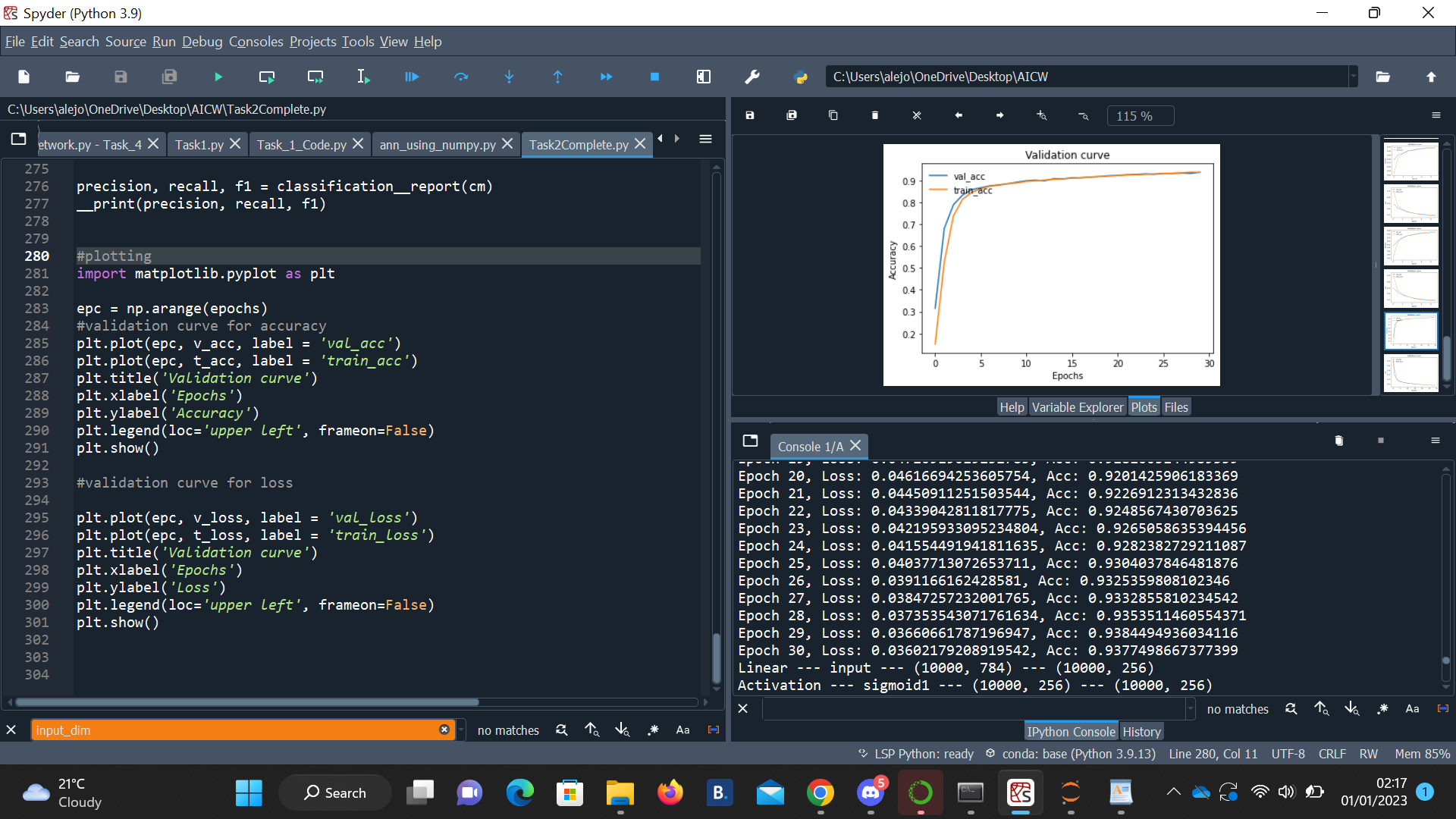
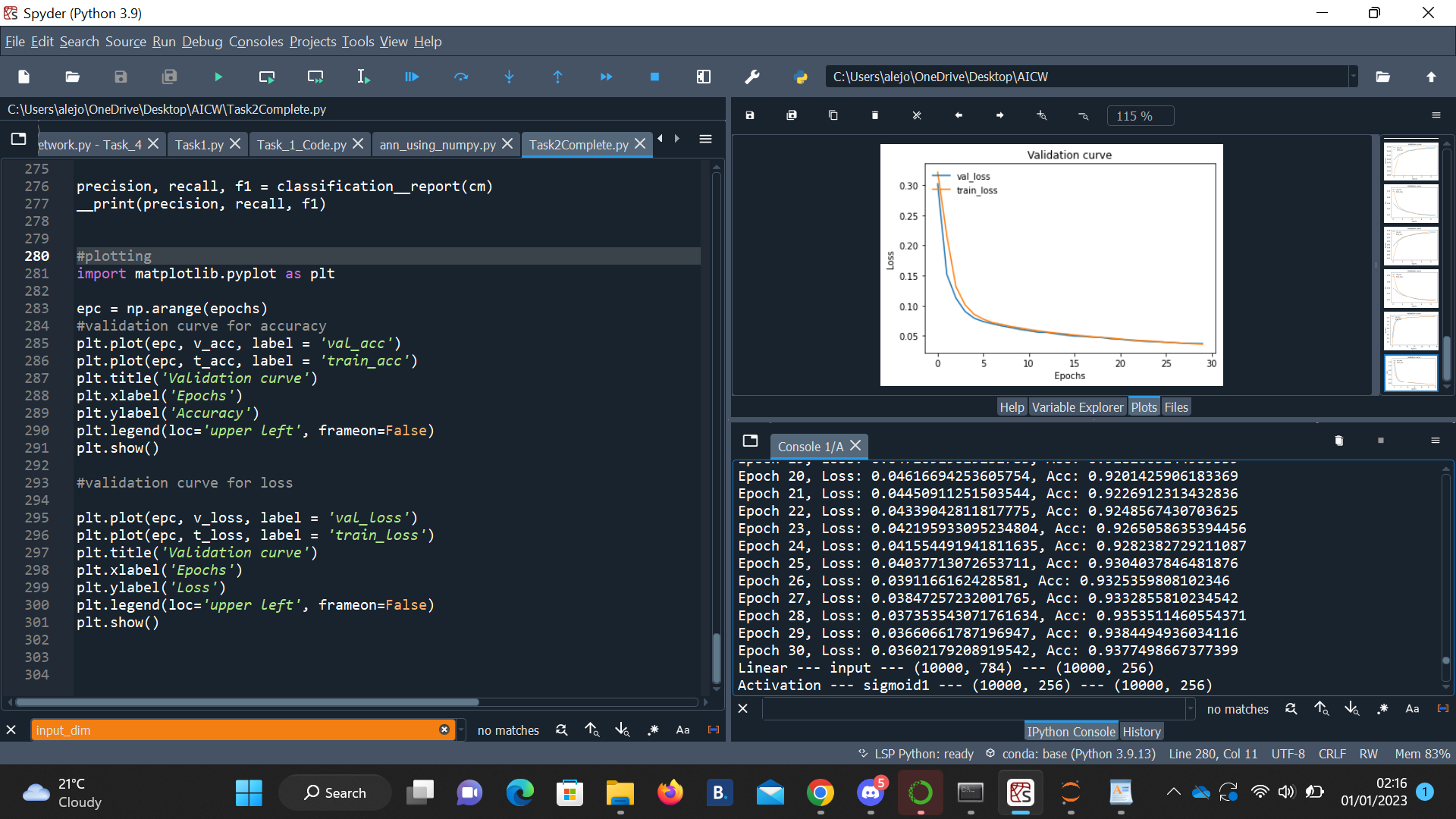


This graph shows the validation curve for validation accuracy and training loss specifically for relu. This Is another trial executed under 50 echos with the same lr value. We can see the results have improved. In the first graph, we can see that the training loss starts high, roughly at 0.1012... and as we perform the function train, we can see that the training and validation loss slowly decreases. A significant drop in train loss is between is between epoch 1 and 2 and a significant increase in training accuracy as show in graph 2, between echo 0 and 5 where the accuracy increases from 0.81 to near 0.94. As epoch gets closer to 50, the accuracy increases closer to 1 and train loss slowly decreases to 0.

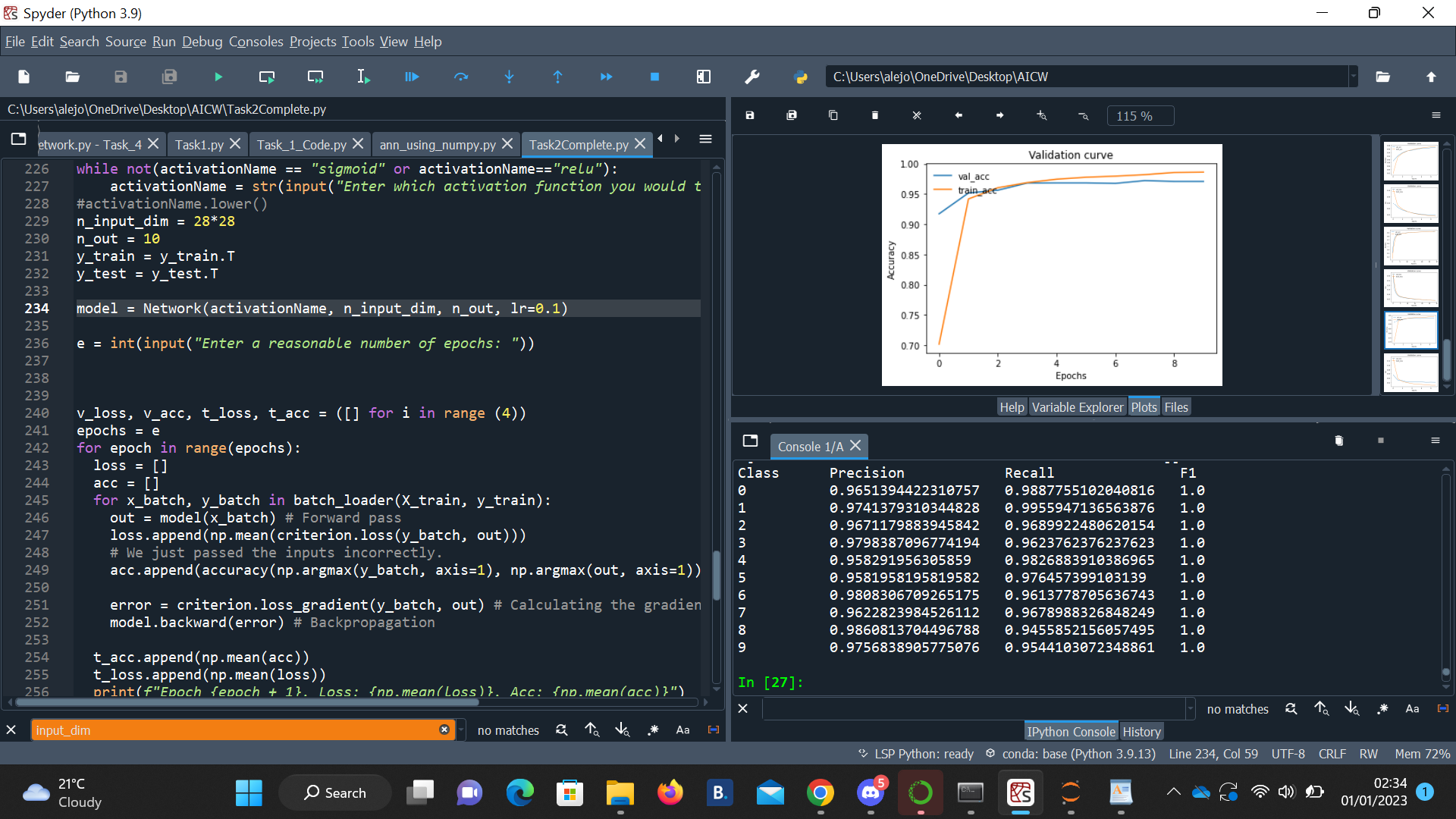




This graph shows the validation curve for validation accuracy and training loss specifically for Sigmoid. This graph was executed under the following hyperparameters: lr=0.001, echos = 50. sigmoid is more likely to generate non-zero value in dense representations. The graph shows a almost a linear scope in both graph, this may perhaps mean that the learning rate for this activation function is not the most adequate learning rate linearly from a large initial value to a small value. It deals with the overfitting issue by using Dropout and regularization, ensuring the accuracy is higher. It also deals with the issues by easing on the regularization that was causing the Dropout to yield not optimal results as non-sparse coefficients are reduced.



I then decided to run another trial, but this time increasing the learning rate slightly to 0.1 with only 10 epochs. These were the results:



This new trial shows better results than the previous graph. Although I run less epochs, the initial starting loss is much lower than the previous and the starting accuracy is much higher. This new parameter determines optimal weights to produce better results. However, increasing the learning rate is very large and we will skip the optimal solution. If it is too small, we will need too many iterations to converge to the best values as the clear example from the previous graph.

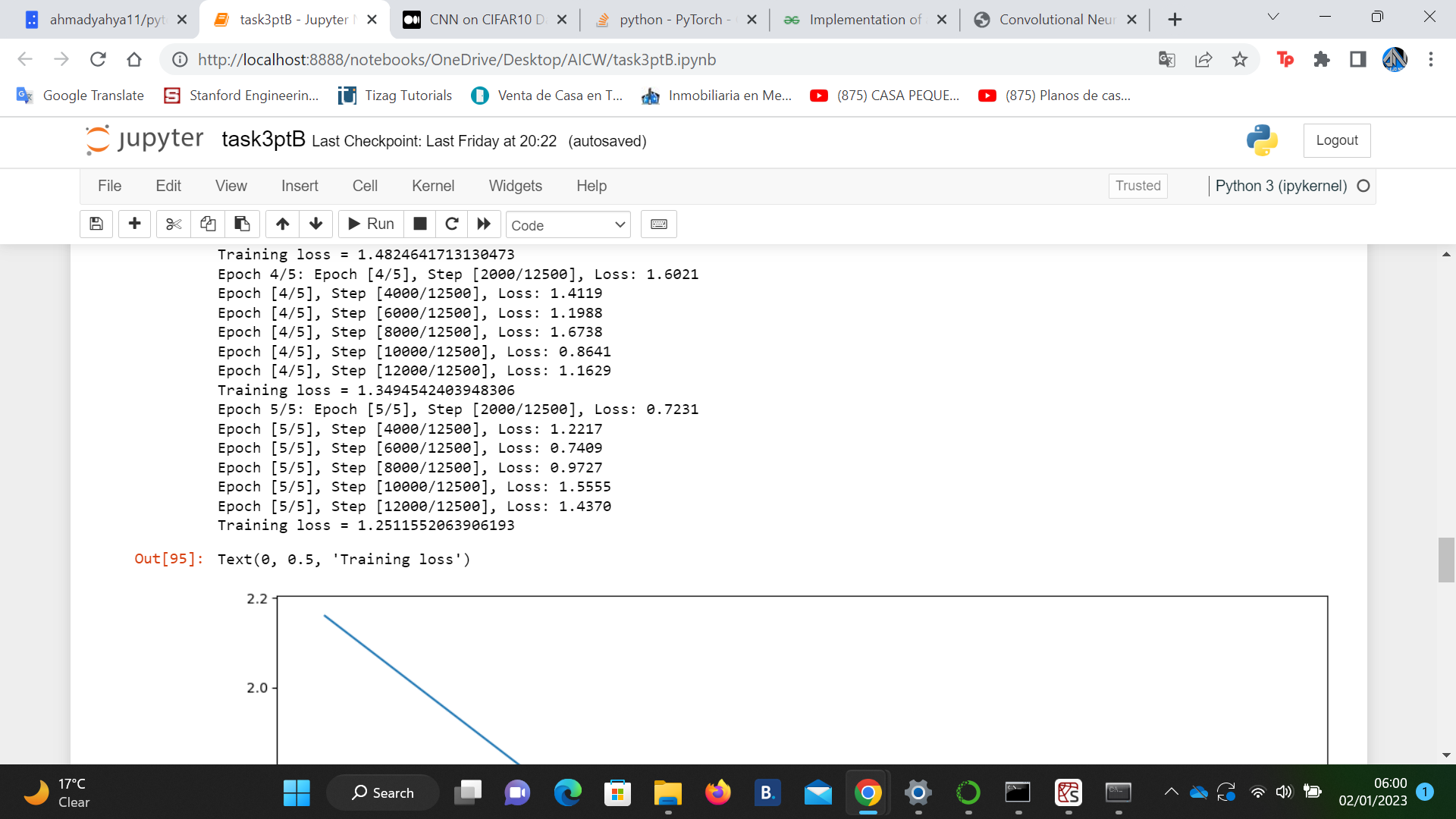
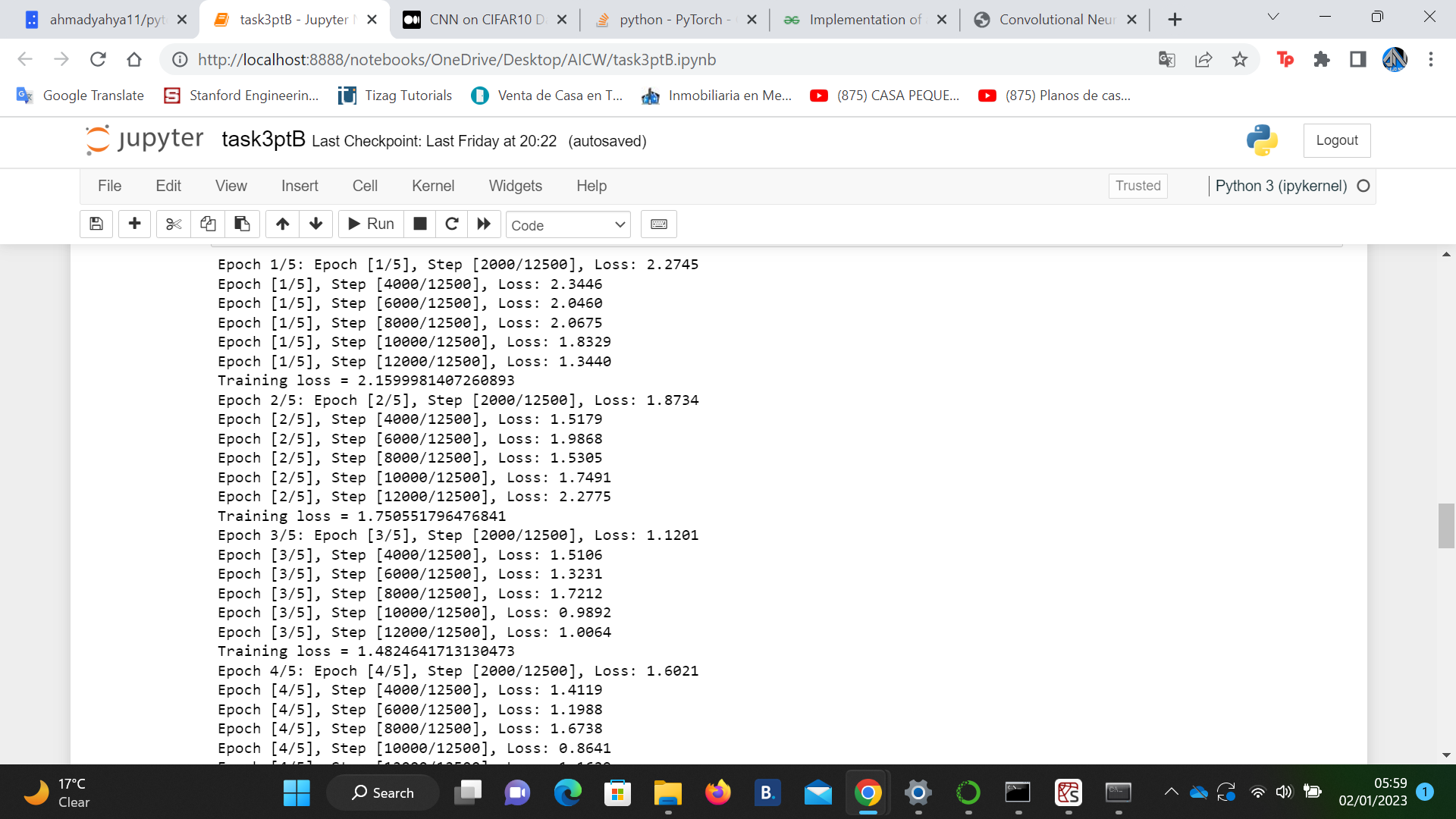
Conclusion:

This function does not have the vanishing gradient problem. This function is computationally inexpensive because it has a fixed derivative slope as it has no exponential terms, and so the learning process is faster. Analyzing both graphs, relu looks faster than sigmoid as the convergence is greater than sigmoid. However, choosing a high learning rate may produce undesired results. For example, I chose lr to be 0.1 and I got NA for my classification report, Low accuracy and high training loss, none of which were decreasing throughout the training process

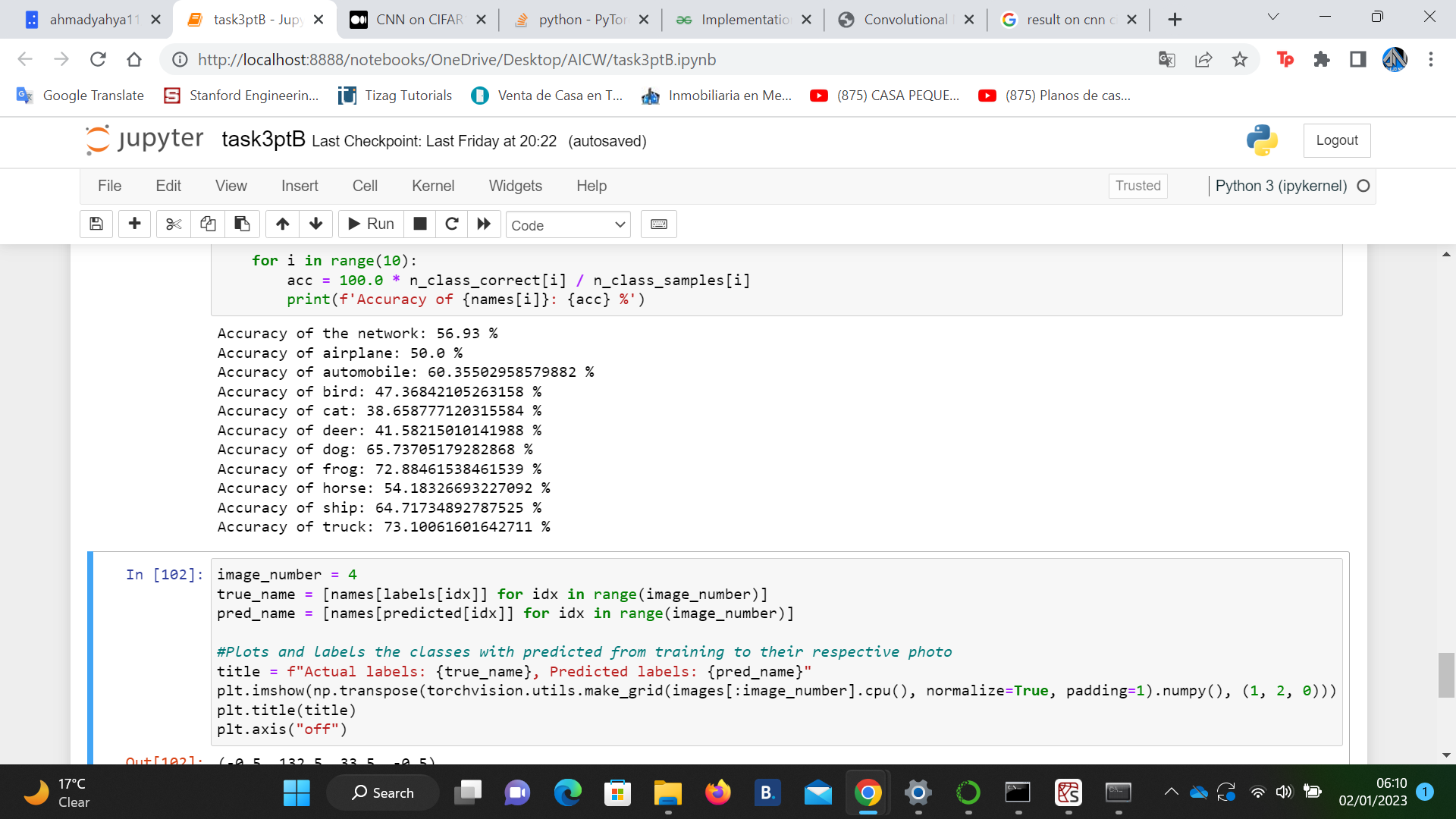
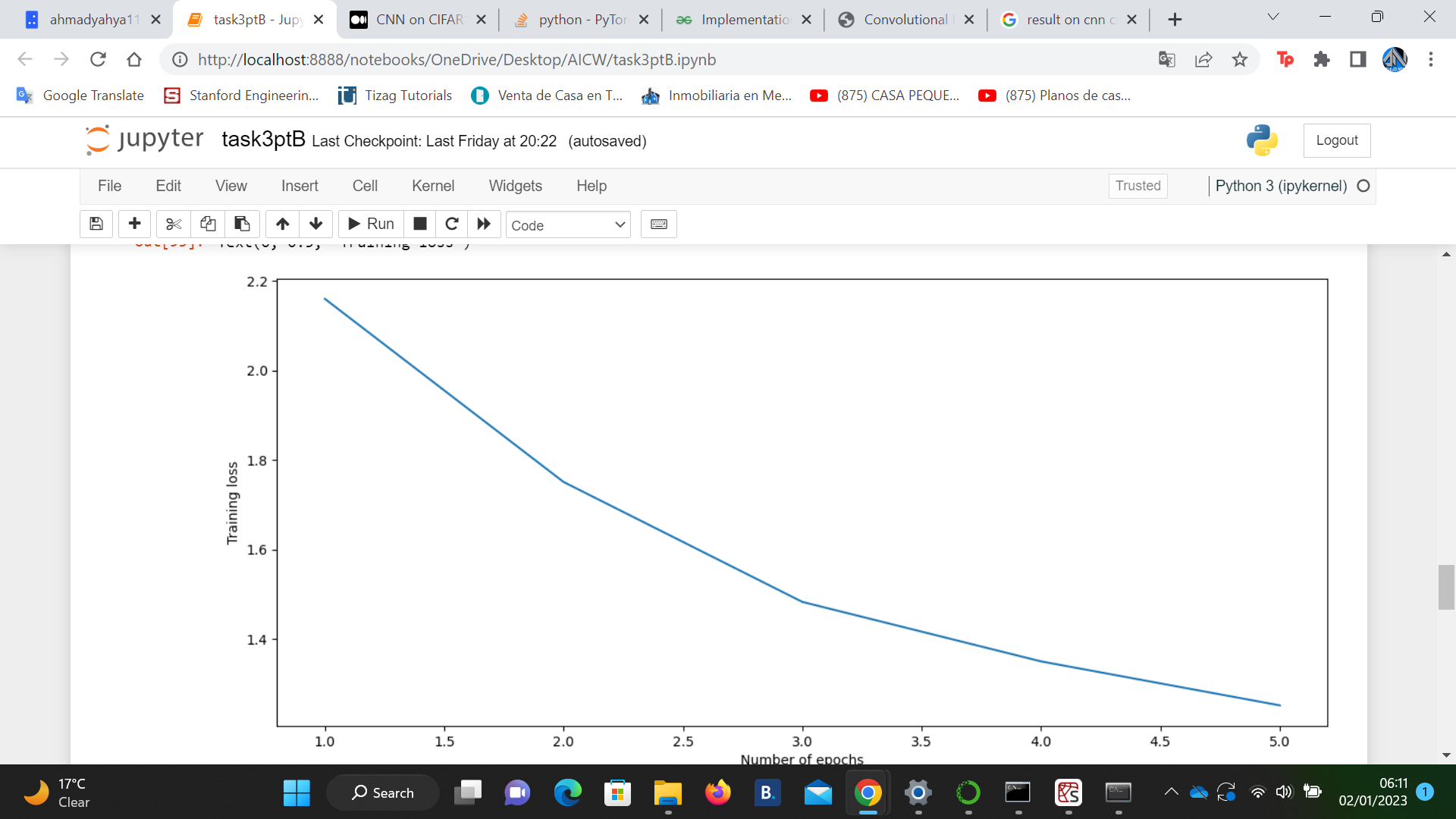
Q3)a

The model is built using a sequential because it is faster and easier to build neural network and skip over the part where we have to implement them forward() function as the sequential class forward() function is embedded in.

The results below are a single hidden layer with the following hyper parameters 5 epochs, lr=0.01, batch\_size=4.



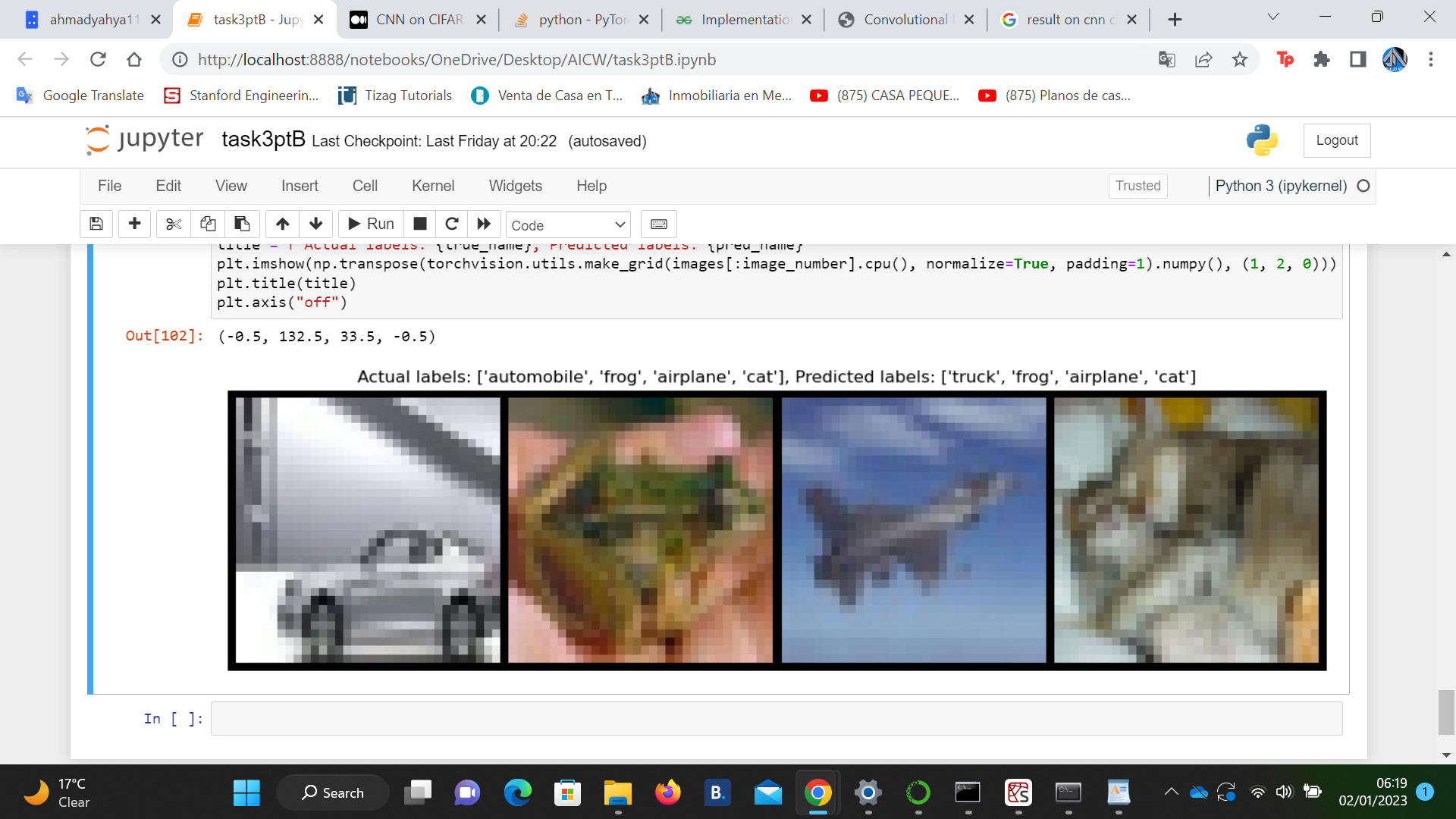
The results are not what is expected. The training loss starts very high and gradually decreases to 1.251... at the end of the 5th epoch. Not enough training iterations are provided to allow the machine to learn and provide better results.



Although, there were a few number of epochs, the machine was able to decrease the number of training loss through training. On the first set of hyperparameters the widths are getting smaller and there is no implementation of any regularisation. It can be observed that training with a learning rate of 1e-2 over 5 iterations we get a test accuracy of 56.93% but it's clear that the test loss is increasing as the training loss is decreasing, implying the network is overfitting. The CNN predicts at less than 50% for cat,bird and deer.

It's not easy to tell if it improves accuracy between single and multi-layer CNN. It seems that training loss and accuracy doesn't diverge from the training accuracy so if the model was left training for more epochs it could give out better results for both networks.

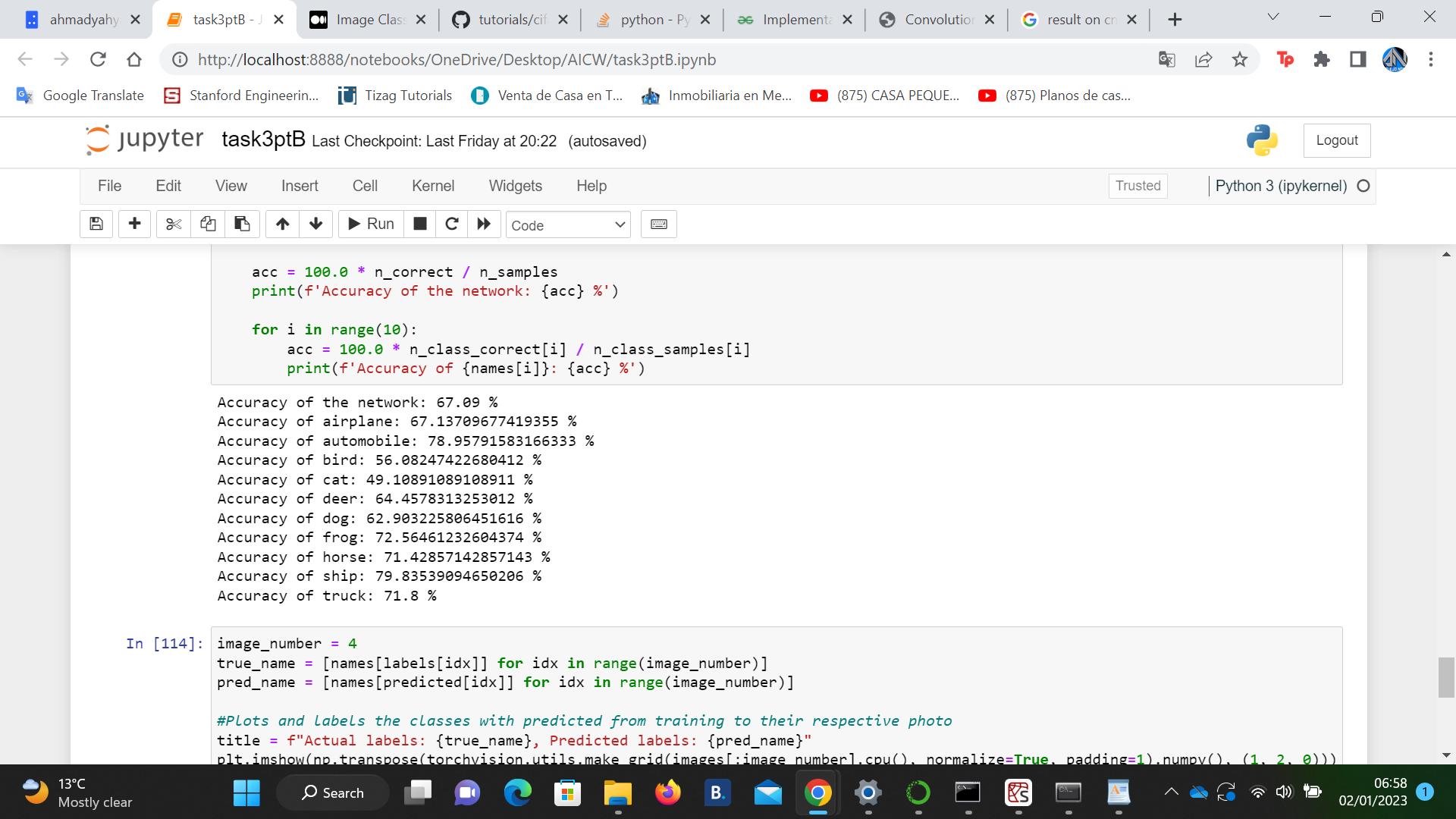
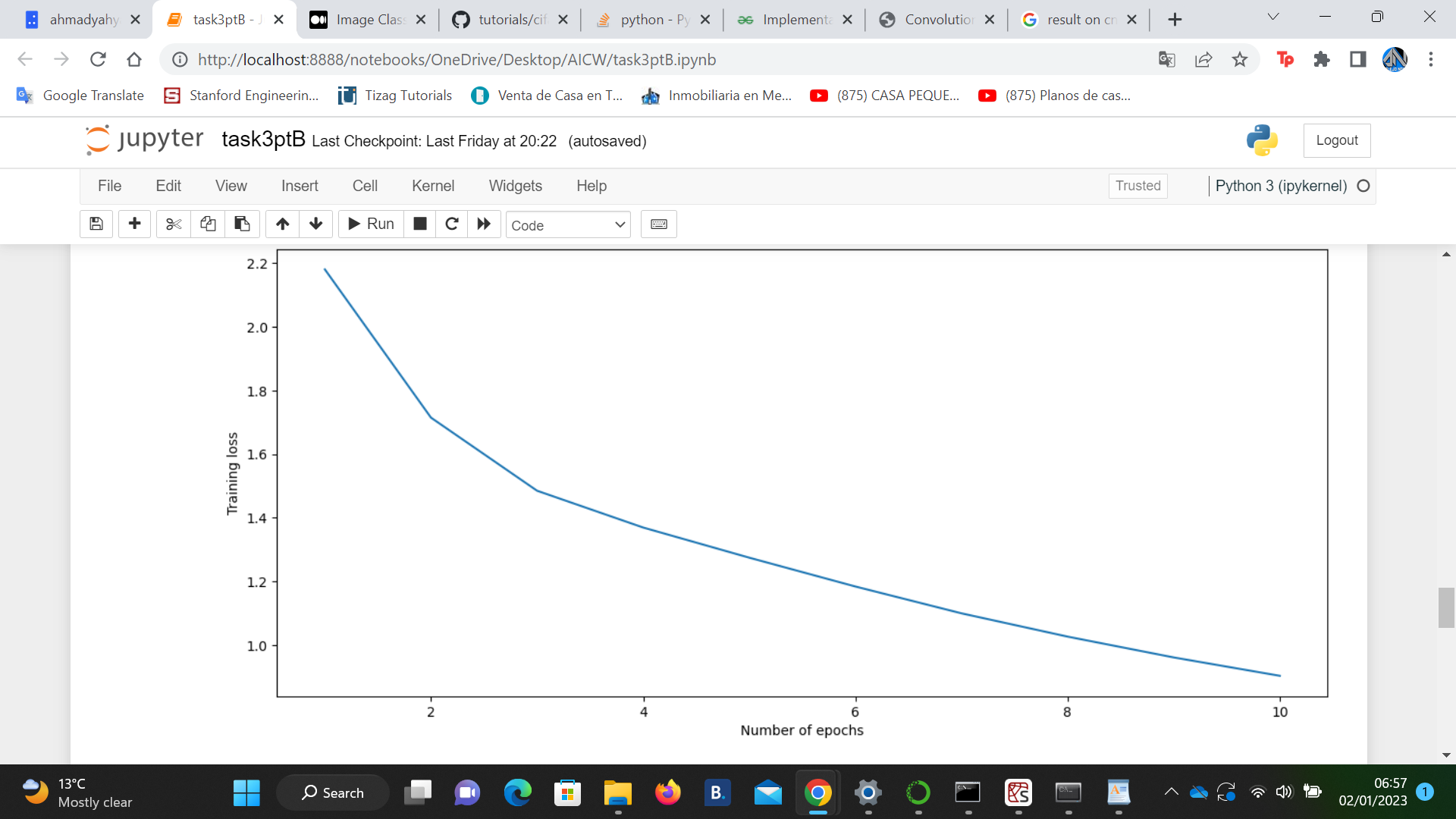
Now changing batch size. As expected, using smaller batch size (slightly) increases accuracy but hinders performance. More steps are needed to complete an epoch but the model updates more often as well. Large batch size executes code faster at a great cost.



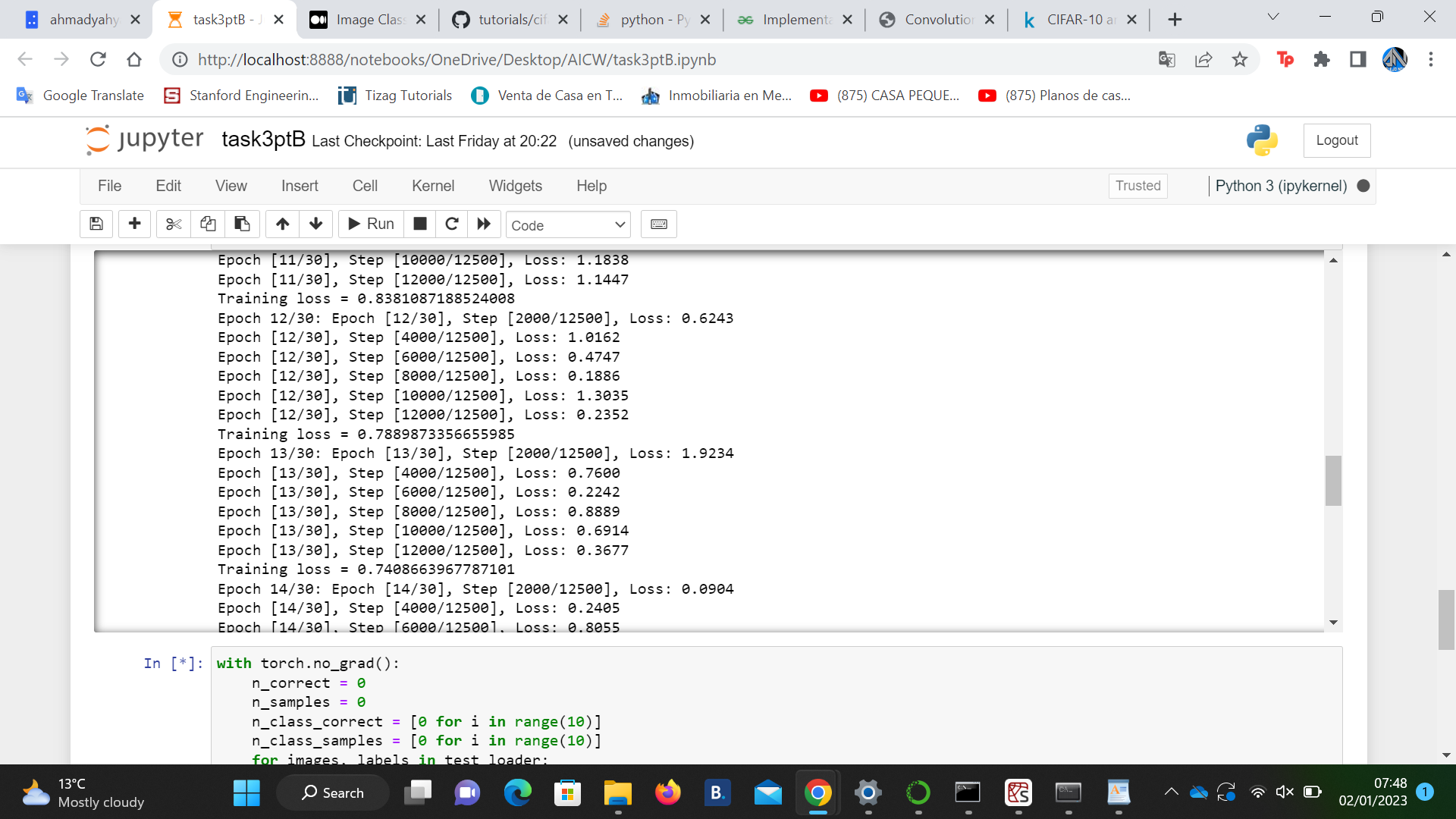
However, the machine was able to predict the following images with the respective classes. Everything is predicted except for automobile.

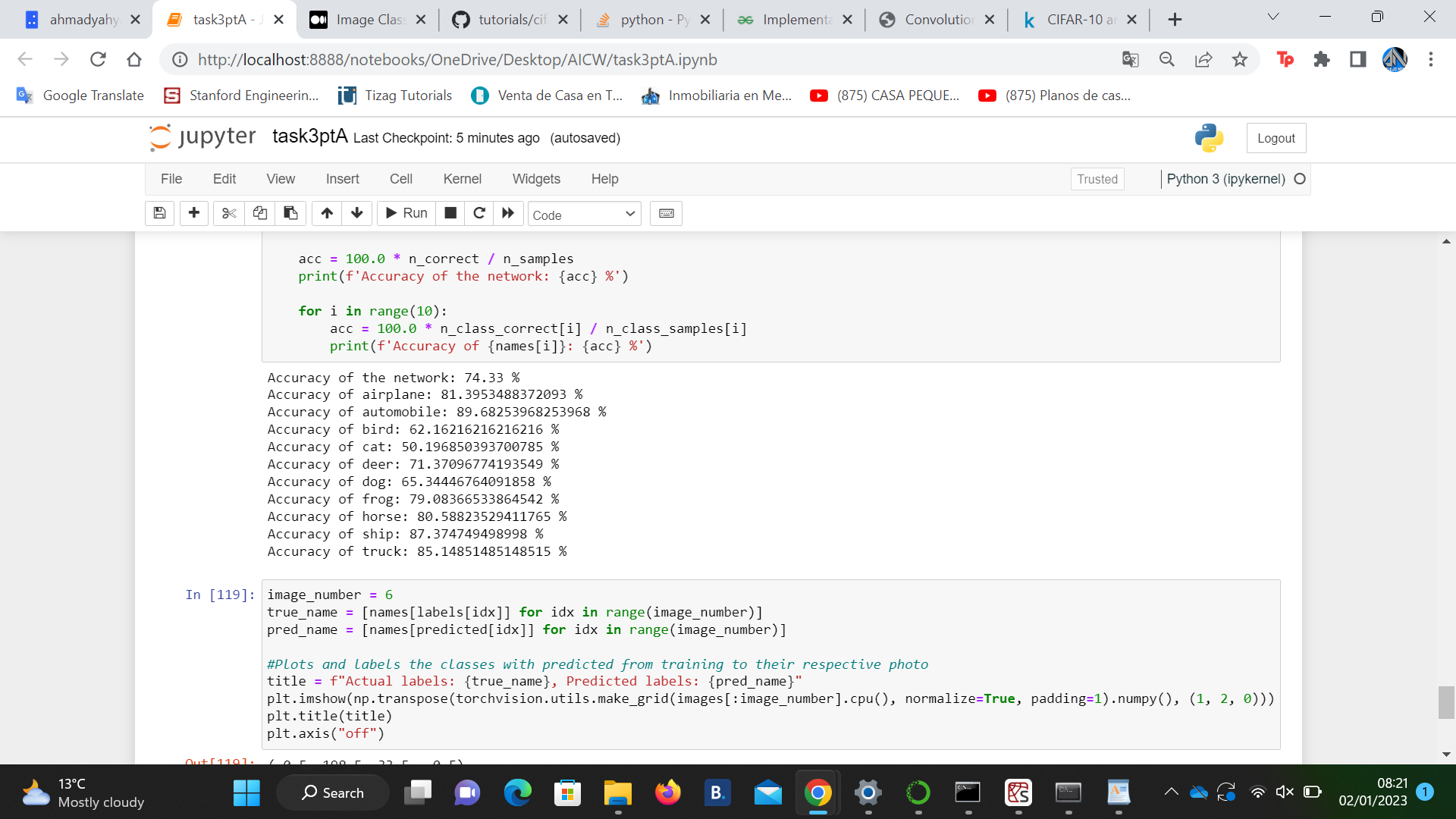
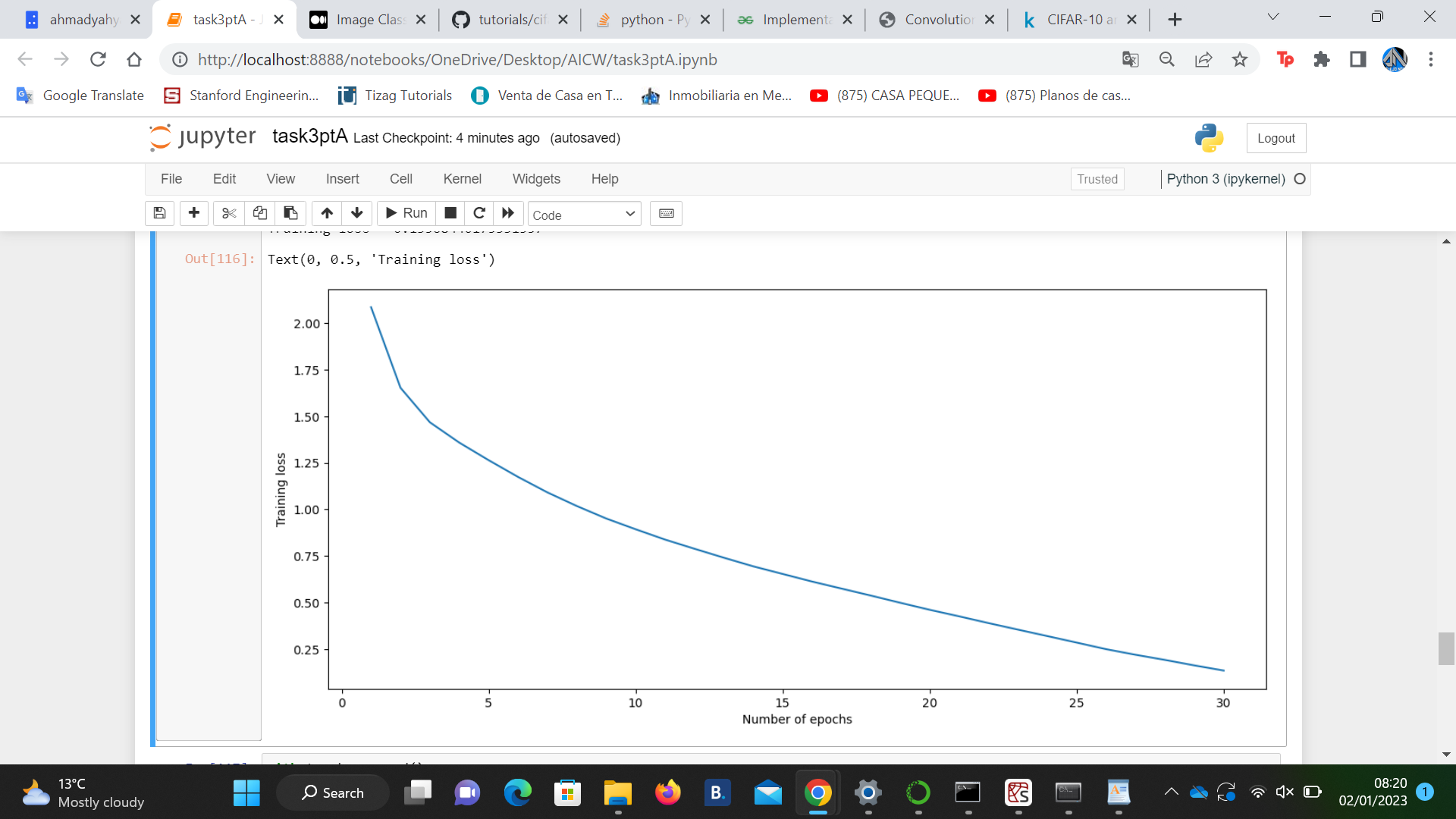
I changed the hyperparameters to be: learning rate 0.001 and epochs = 10 and batch size = 4.

These were the results:



Same parameters but epochs= 30

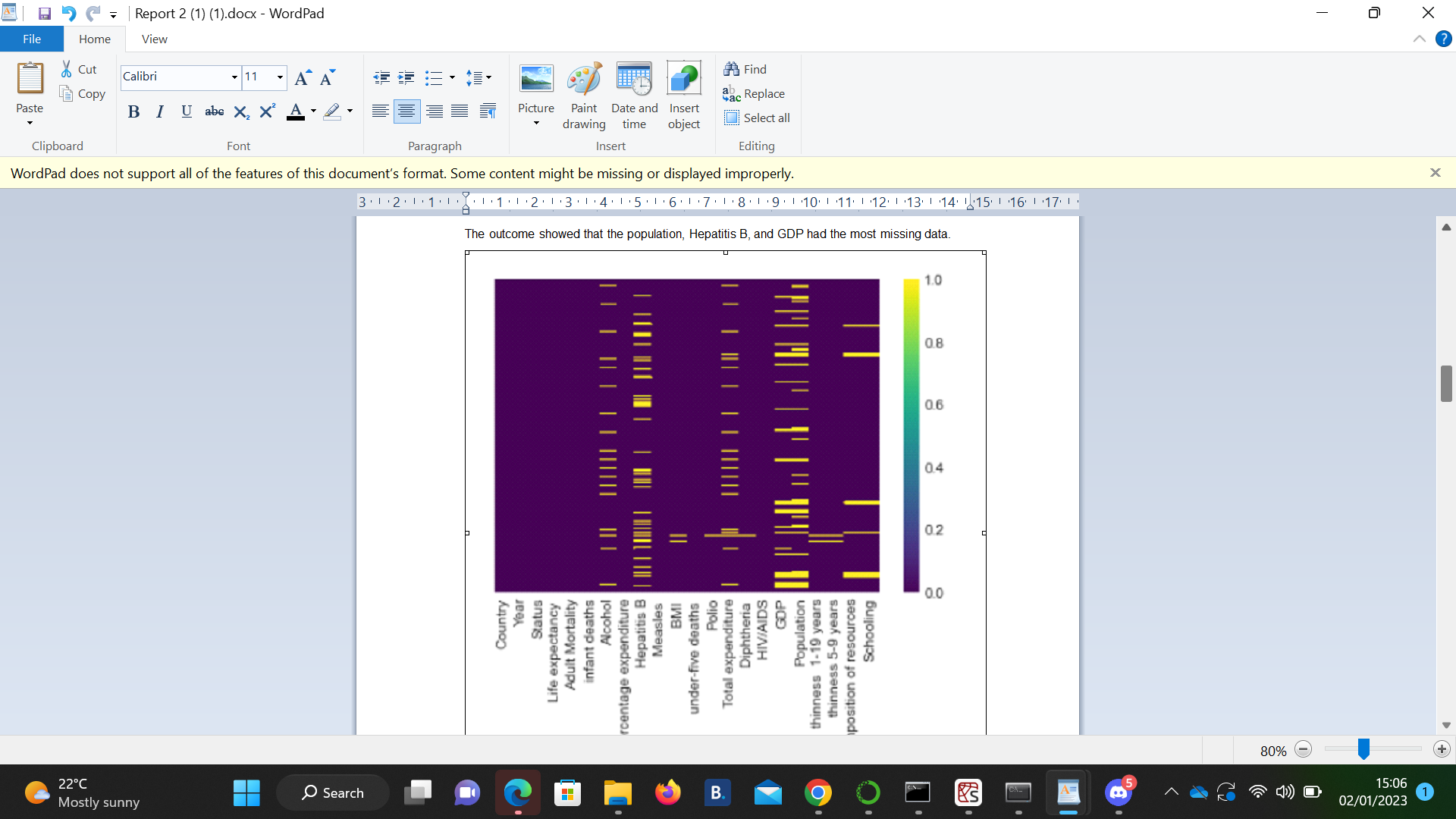
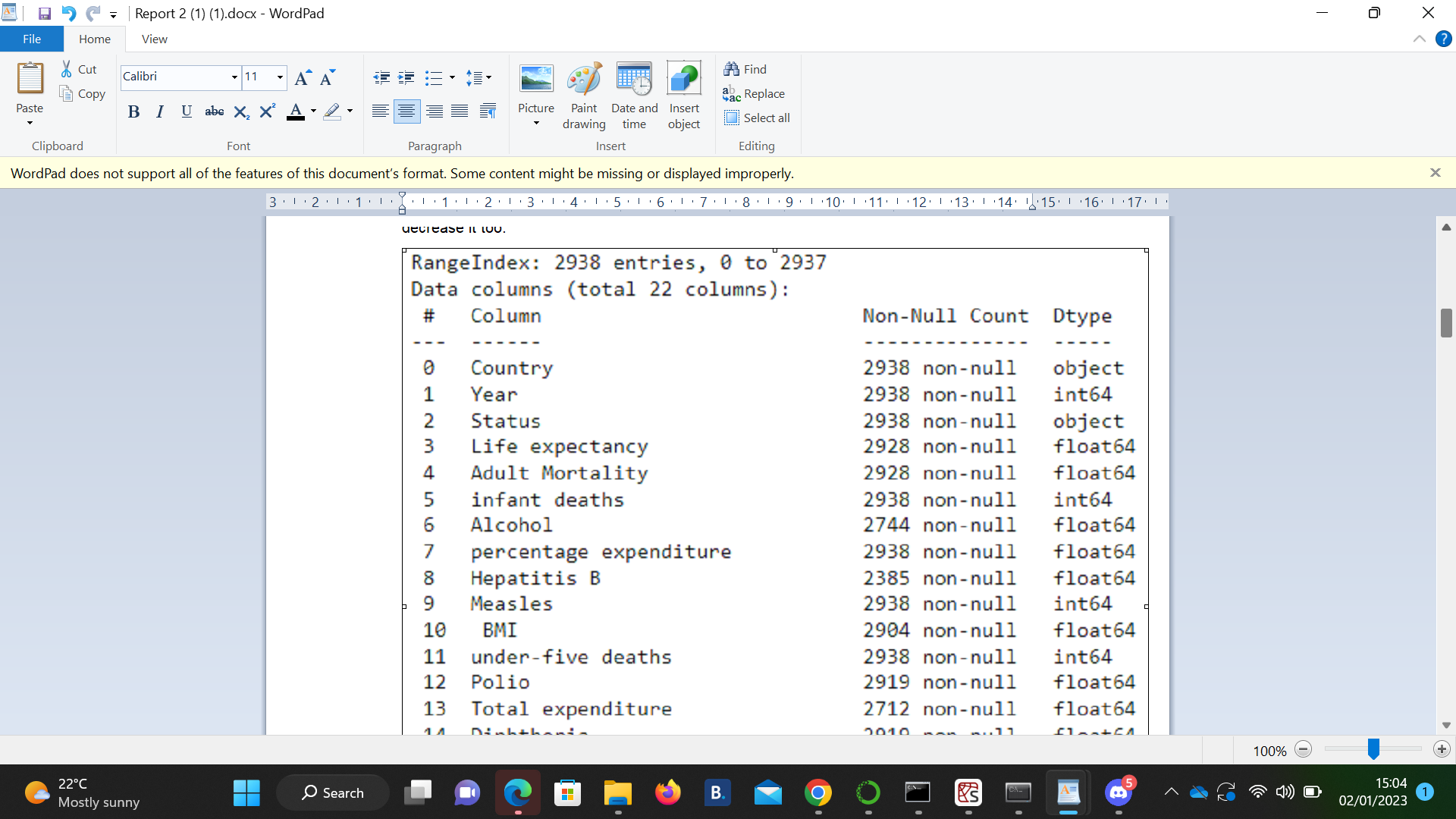
Plotting how well this configuration goes reveals something interesting: the model overfits after during the 12 and 13th epoch! We can see the training loss slowly increased while the training accuracy steadily increases. This is more visible in the plot of the model loss during the epochs. The accumulated Loss in the validation set clearly goes up after the 12th epoch and comes back down on the 14th epoch



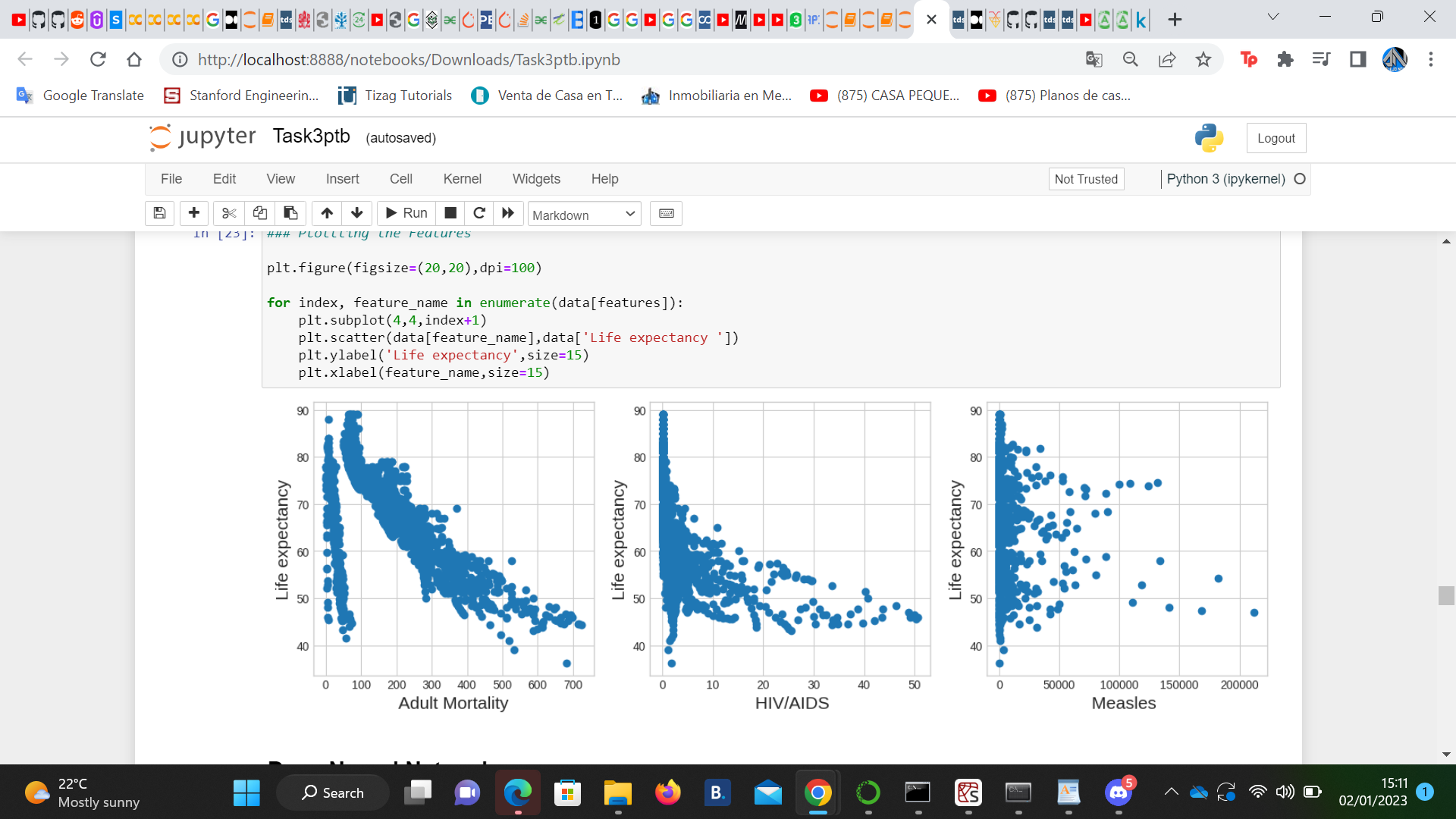
Except for cat, bird and dog, the rest have very high predicting accuracy.

Q3)b

This project is based on the dataset ‘Life Expectancy.csv’. The World Health Organization (WHO ) data repository monitors each nation's health status as well as a number of other relevant variables. For the goal of analyzing health data, the data sets are made available to the general public. The WHO data repository website affiliated with Kaggle was used to acquire the data set pertaining to life expectancy, health variables for 193 nations, and its equivalent economic data was obtained from the United Nations website. Only the most representative critical elements from each category of health-related factors were selected. It has been noted that, during the past 15 years, the health sector has seen significant growth, which has improved human mortality rates, particularly in emerging nations when compared to the past 30 years. Our main goal is to find the factors that affects the life expectancy, specifically the ones which increase the expected life in the countries and those factors that decrease it too.



When the data were first visually inspected, several values were missing. Since the data sets were from WHO, I didn't see any obvious mistakes. In Python, Interpolate imputation method have been used to handle missing data. The outcome showed that the population, Hepatitis B, and GDP had the most missing data. As shown in the second figure.



When I weigh against, say, population, HIV, we can see there is a correlation between HIV/AIDS and adult mortality as it shows to have a serious impact. This does not mean that the vast majority of adult mortality is directly dependent on HIV but a clear indication that adults that range between 40 and 60 tend to show high mortality rates in this age range. These could be for a number of reasons such as, the country a person lives in could also impact the life expectancy.

Q1) <https://bradfieldcs.com/algos/graphs/dijkstras-algorithm/>

By: Abul Salam february 13, 2021. https://levelup.gitconnected.com/dijkstra-algorithm-in-python-8f0e75e3f16e

Q2) By: [Aayush Agrawa,](https://medium.com/@aayushmnit?source=---two_column_layout_sidebar----------------------------------) June 13, 2018. <https://towardsdatascience.com/building-neural-network-from-scratch-9c88535bf8e9>

## By: [Maciej Balawejder](https://medium.com/@maciejbalawejder?source=---two_column_layout_sidebar----------------------------------) July 27, 2021. <https://medium.com/analytics-vidhya/neural-network-mnist-classifier-from-scratch-using-numpy-library-94bbcfed7eae>

By: Piotr Babel Jun 11, 2019. <https://towardsdatascience.com/deep-neural-networks-from-scratch-in-python-451f07999373>

*by* [*Deepak Battini*](https://www.tech-quantum.com/author/deepak/) [*November 3, 2018*](https://www.tech-quantum.com/implementing-drop-out-regularization-in-neural-networks/)<https://www.tech-quantum.com/implementing-drop-out-regularization-in-neural-networks/>

By: Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov. https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

Batches implementation- by sentdex 1st May 2020. <https://www.youtube.com/watch?v=TEWy9vZcxW4>

*By Savvy Sahai, 28 Nov 2022.* <https://www.projectpro.io/recipes/generate-classification-report-and-confusion-matrix-in-python>

# Calculating Loss with Cross-Entropy – by Sentdex 23rd Jan 2021. <https://www.youtube.com/watch?v=dEXPMQXoiLc>

By: [yawen-d](https://github.com/yawen-d) 3rd february 2019. https://github.com/yawen-d/Neural-Network-on-MNIST-with-NumPy-from-Scratch/blob/master/Neural\_Network.py

Q3)a By: unknown August 29, 2021. <https://androidkt.com/convolutional-neural-network-using-sequential-model-in-pytorch/>

By: [wesleyacheng](https://github.com/wesleyacheng) <https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html#sphx-glr-beginner-blitz-cifar10-tutorial-py>

By: [MK Gurucharan](https://www.upgrad.com/blog/author/gurucharan/) july 28, 2022. <https://www.upgrad.com/blog/basic-cnn-architecture/>

By dataJamesson, 2 years ago. <https://www.kaggle.com/code/datajameson/cifar-10-object-recognition-cnn-explained/notebook>

By: Steffan Fiott November 30, 2018. <https://www.stefanfiott.com/machine-learning/cifar-10-classifier-using-cnn-in-pytorch/>

Aby: run Purakkat 4th September 2020. <https://medium.com/swlh/image-classification-with-cnn-4f2a501faadb>

Q)3)b

Mastering Markdown. https://guides.github.com/features/mastering-markdown/

Matplotlib: Visualization with Python. https://matplotlib.org/

Seaborn User guide and tutorial. https://seaborn.pydata.org/tutorial.html

Geeks For Geeks 2018 Python. https://www.geeksforgeeks.org/python-pandas-dataframe-info/

The Pandas Development Team 2020 pandas.DataFrame.describe, Available at: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html

Will Koehrsen 2018 Visualizing Data with Pairs Plots in Python. https://towardsdatascience.com/visualizing-data-with-pair-plots-in-python-f228cf529166

Geeks For Geeks 2020 Python. https://www.geeksforgeeks.org/python-pandas-dataframe-corr/

Statistics How To, Regression Analysis: Step by Step Articles, Videos, Simple Definitions. https://www.statisticshowto.com/probability-and-statistics/regression-analysis/

Life Expectancy Dataset; https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who?datasetId=12603&sortBy=voteCount