Assignment 2

By: Abdulwahab Hawsawi & Abdulwahed Dahan

The input data for the GAI in details (Attach the data file).

The input data for the GAI is a the same data that would be used for the machine learning algorithm. It is historical data of some of the trips that were taken using the Wusool service from 2018 to 2023. The total count of the records is around 600,000.

The data was acquired from a website called OpenData: a Saudi website where different government agencies and organizations can publicly release some the data they collect. The dataset that will be used in the project comes The Human Resources Development Fund. The data is from 2018 to 2023. Each of the six years is associated with a CSV file that contains around 100,000 rows. While some of the rows gave odd numbers, e.g. the price of the trip is 0, these were very few.

The record files have a number of columns, 12 to be exact, but the program will use four of them: pickup longitude and latitude, and drop-off longitude and latitude.

Explain in details your GAI website with screenshots.

Our app is a python script that opens the files for all the records, take the relevant columns from every file, combines the data from all the records, normalize them, splits the data into training and testing dataset, then feeds it to the Generative Artificial Intelligence and machine learning algorithm. Then, the testing data will be used on both to see how accurate both models are within a specified range.

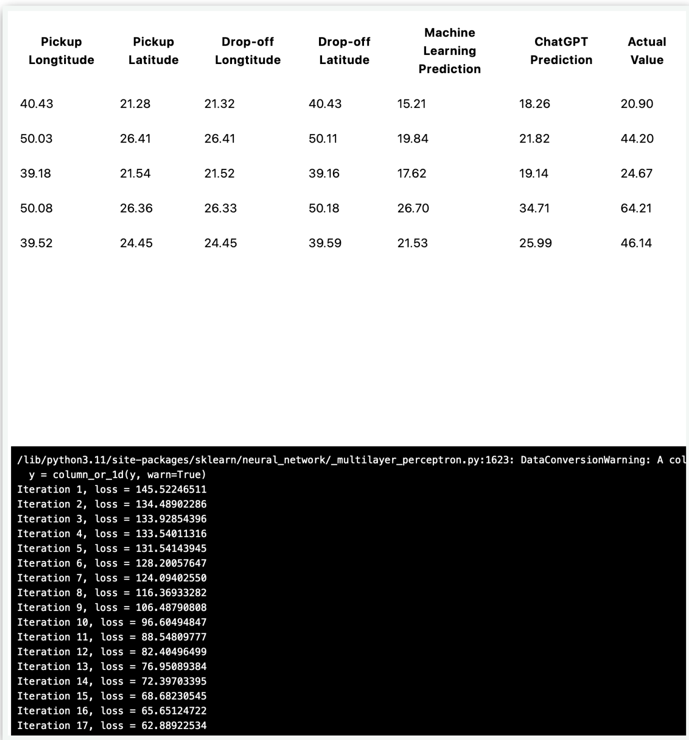
The model used for the machine learning algorithm is Multi-Layer Perceptron Regressor, which is a collection of perceptrons organized in layers between the input and output. The default settings will be used. The ones that are of interest are:

hidden_layer_size = 100

max_iter = 200

learning_rate_init = 0.001

hidden_layer_sizes=100



Pickup Longtitude	Pickup Latitude	Drop-off Longtitude	Drop-off Latitude	Machine Learning Prediction	ChatGPT Prediction	Actual Value
40.43	21.28	21.32	40.43	15.21	18.26	20.90
50.03	26.41	26.41	50.11	19.84	21.82	44.20
39.18	21.54	21.52	39.16	17.62	19.14	24.67
50.08	26.36	26.33	50.18	26.70	34.71	64.21
39.52	24.45	24.45	39.59	21.53	25.99	46.14

```
/lib/python3.11/site-packages/sklearn/neural_network/_multilayer_perceptron.py:1623: DataConversionWarning: A col
  y = column_or_1d(y, warn=True)
Iteration 1, loss = 145.52246511
Iteration 2, loss = 134.48902286
Iteration 3, loss = 133.92854396
Iteration 4, loss = 133.54011316
Iteration 5, loss = 131.54143945
Iteration 6, loss = 128.20057647
Iteration 7, loss = 124.09402550
Iteration 8, loss = 116.36933282
Iteration 9, loss = 106.48790808
Iteration 10, loss = 96.60494847
Iteration 11, loss = 88.54809777
Iteration 12, loss = 82.40496499
Iteration 13, loss = 76.95089384
Iteration 14, loss = 72.39703395
Iteration 15, loss = 68.68230545
Iteration 16, loss = 65.65124722
Iteration 17, loss = 62.88922534
```

Explain in details the output of your GAI website with screenshot of your results.

The GAI was supposed to take the data, pretend it is a machine learning algorithm, train on it, then take the testing data and test itself and give an accuracy score.

We managed to upload part of the data to LangChain in the form of a CSV file.

While there were ways to upload the CSV files, the model would not know how to train and test on it.

Explain how Machine Learning being used in your project with input data.

This step is done by first passing the training features, xtrain, and the training target, ytrain. Then, the test features, xtest_std, are passed to … test if the model has been trained correctly.

```
np_ytest = ytest.to_numpy()
```

✓ 0.0s

```
threshold = 0.8
accuracy = 0
for i in range(len(ytest)):
    if np_ytest[i] == 0:
        continue
    percent = prediction[i] / np_ytest[i]
    if percent >= threshold and percent <= 2 - threshold:
        accuracy += 1
        print("predicted: {}     Actual: {}".format(prediction[i], np_ytest[i]))
score = (accuracy / len(ytest))
print("accuracy within {:.0%} of the true value is {:.2%}".format(0.8,score))
```

To test the accuracy, the predicted value is divided by the actual value. This will give us, percentage wise, how close the values are. It will then be decided whether or not the prediction is accurate based on the tolerance level. For example, if the tolerance level is set to 0.9, then an "accurate" result will be within 10% of the true value. 0.8 means within 20% or the true value.

Using the default values, and a tolerance value of 0.8, the training finished in 2 minutes and 48 seconds. The accuracy was 60.04%

Experimentation

Changing the default configurations yielded the following results:

- Increasing learning_rate_init from 0.001 to 0.01 increased the accuracy to 41.92%
- Reducing hidden_layer_sizes from 100 to 9, while keeping learning_rate_init at 0.01, increased the accuracy to 51.99%. In addition to the accuracy increase, the speed of the

training has been reduced to 32 seconds. However, this only happened twice. Afterwards, every subsequent try stuck around the high-30s in terms of accuracy.

- Returning to the default made the accuracy around 48%.
- Removing the 2023 dataset increased the accuracy to around 62%! It also improved the training time by reducing it to 1 minute and 11 seconds.
- Removing the 2023, 2022, 2021, and 2020 dataset reduced the accuracy to around 52%.

It seems the 2023 dataset a lot of datapoints that cause the model to learn bad information. The best accuracy is ensured when it is removed.

Here is an example of the correct predictions the model made, within 20% of the true value:

```
predicted: 16.588202975272583     Actual: [19.58]
predicted: 19.283030191163917     Actual: [21.03]
predicted: 12.477689107341178     Actual: [13.86]
predicted: 33.008337765388745     Actual: [36.85]
predicted: 13.713172506810213     Actual: [14.36]
predicted: 17.816510138561256     Actual: [17.99]
predicted: 41.1749594935717     Actual: [47.5]
predicted: 12.569987010028548     Actual: [12.83]
predicted: 18.803941741631622     Actual: [21.53]
predicted: 27.954936273942362     Actual: [31.47]
predicted: 19.39791361048876     Actual: [23.87]
predicted: 56.79312560781528     Actual: [57.28]
predicted: 12.043761513819394     Actual: [14.84]
predicted: 25.318241351192427     Actual: [30.97]
predicted: 36.06199556468952     Actual: [36.2]
predicted: 44.66638947478601     Actual: [53.71]
predicted: 18.3545418828127     Actual: [19.98]
predicted: 44.03601936627344     Actual: [53.51]
predicted: 17.928681381734116     Actual: [18.5]
predicted: 11.09421141803097     Actual: [12.5]
predicted: 31.864427695440742     Actual: [34.86]
predicted: 45.23347002466219     Actual: [53.5]
predicted: 18.48219847887576     Actual: [22.08]
predicted: 55.34303045032199     Actual: [61.77]
predicted: 35.538979929053234     Actual: [41.91]
```

Here is an example of wrong predations the model made, outside the 20% threshold:

```
predicted: 14.891993882367721     Actual: [10.29]
predicted: 16.543319829980707     Actual: [11.74]
predicted: 25.66390355747419     Actual: [32.42]
predicted: 12.021776819200028     Actual: [29.02]
predicted: 16.080404686506274     Actual: [25.69]
predicted: 16.845504158069552     Actual: [10.4]
predicted: 46.814253557478985     Actual: [65.35]
predicted: 45.65399081722004     Actual: [28.72]
predicted: 21.94609959890758     Actual: [28.04]
predicted: 19.18207195507338     Actual: [12.62]
predicted: 19.89415920682546     Actual: [15.98]
predicted: 29.794461985282158     Actual: [20.83]
predicted: 27.63512063252011     Actual: [20.35]
predicted: 31.10180645881539     Actual: [23.55]
predicted: 21.30391623409994     Actual: [14.92]
predicted: 17.18945103520575     Actual: [10.06]
predicted: 22.42646289916289     Actual: [17.58]
predicted: 52.46125661285843     Actual: [42.37]
```

When reducing the threshold to within 10%, the accuracy got reduced to 37.41%. Here is some correct predictions:

```
predicted: 31.395891433741085     Actual: [28.87]
predicted: 43.78802753134829     Actual: [40.43]
predicted: 19.283030191163917     Actual: [21.03]
predicted: 14.700580474193039     Actual: [13.41]
predicted: 13.198462354363143     Actual: [12.02]
predicted: 12.477689107341178     Actual: [13.86]
predicted: 32.990348785457805     Actual: [32.43]
predicted: 21.27987671952219     Actual: [20.26]
predicted: 28.759570480971977     Actual: [27.52]
predicted: 13.713172506810213     Actual: [14.36]
predicted: 17.816510138561256     Actual: [17.99]
predicted: 12.569987010028548     Actual: [12.83]
predicted: 31.928337888766627     Actual: [30.9]
predicted: 30.419756331913785     Actual: [29.52]
predicted: 18.369481394272047     Actual: [16.99]
predicted: 56.79312560781528     Actual: [57.28]
predicted: 26.74646165833557     Actual: [26.71]
predicted: 43.209470174047986     Actual: [41.48]
predicted: 23.304870047361096     Actual: [23.22]
predicted: 27.357784584000992     Actual: [25.9]
```

The Machine Learning tools used in the project.

- Sckkit-learn: this library provided to model to be used in the app.
- Pandas: to read the CSV files
- OpenAI and OS: to use chatgpt