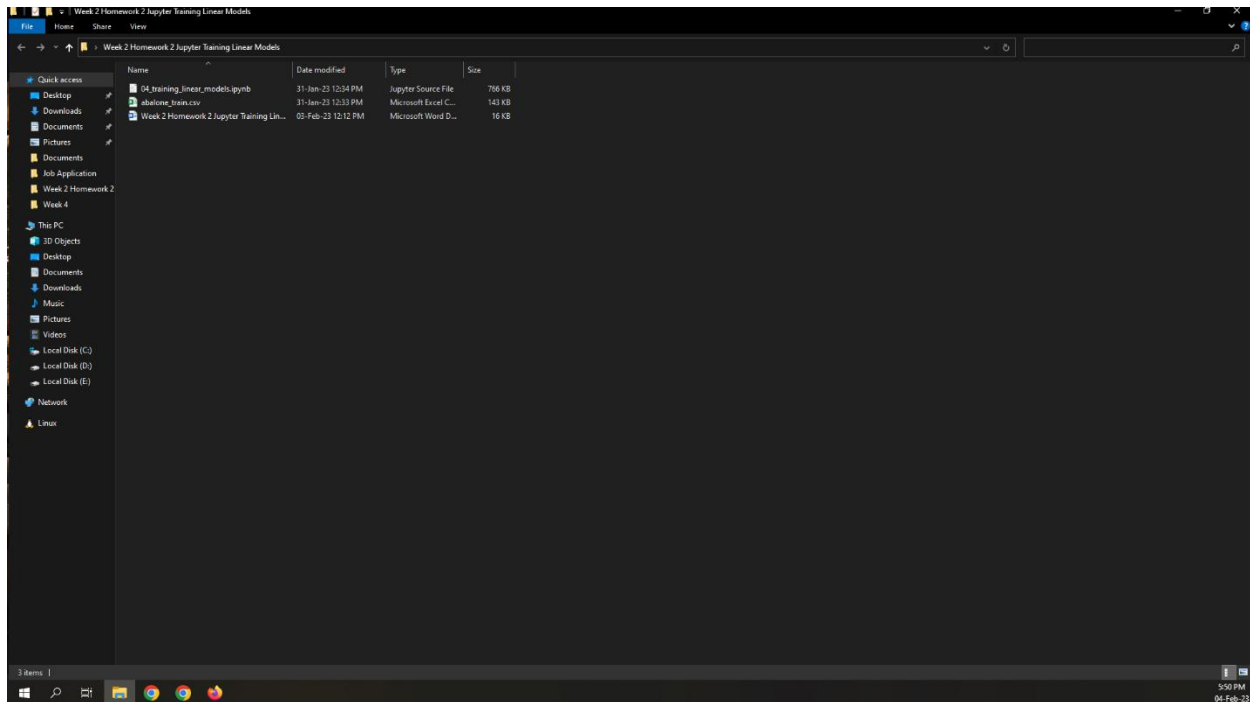
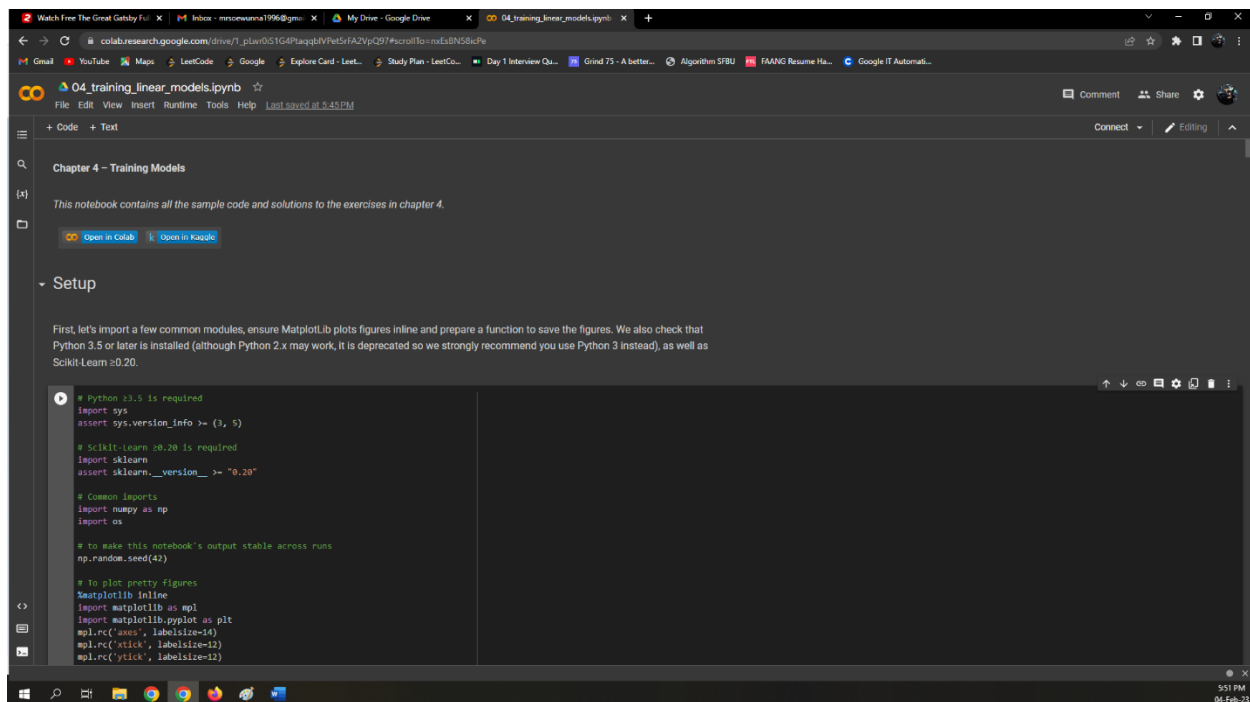


ipynb file and csv file in local drive.

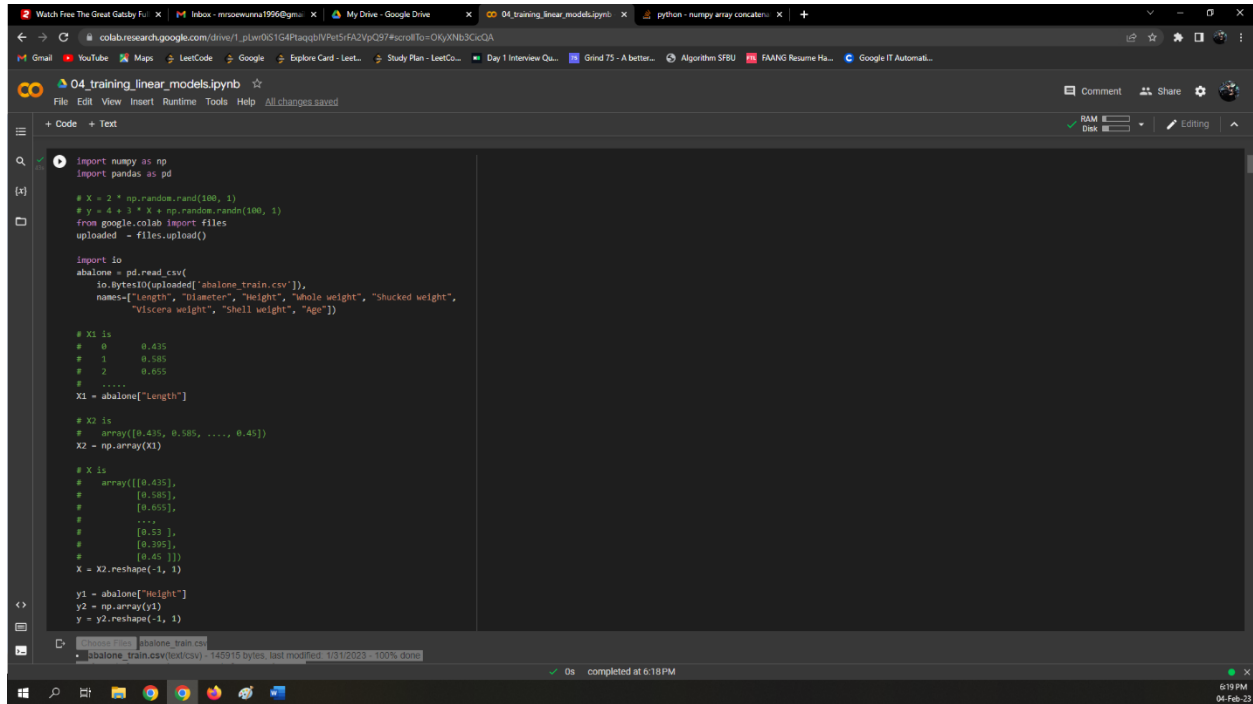


Importing ipynb file to Google Collab.



Modifying the 2 lines of code.

Reading data from a local drive and transform the data to fit the Python code.



The screenshot shows a Google Colab notebook titled "O4\_training\_linear\_models.ipynb". The code in the notebook is as follows:

```
import numpy as np
import pandas as pd

# X = 2 * np.random.rand(100, 1)
# y = 4 + 3 * x + np.random.randn(100, 1)
from google.colab import files
uploaded = files.upload()

import io
abalone = pd.read_csv(
    io.BytesIO(uploaded['abalone_train.csv']),
    names=["Length", "Diameter", "Height", "Whole weight", "Shucked weight",
           "Viscera weight", "Shell weight", "Age"])

# X1 is
# 0      0.435
# 1      0.585
# 2      0.655
# .....
# X1 = abalone["Length"]

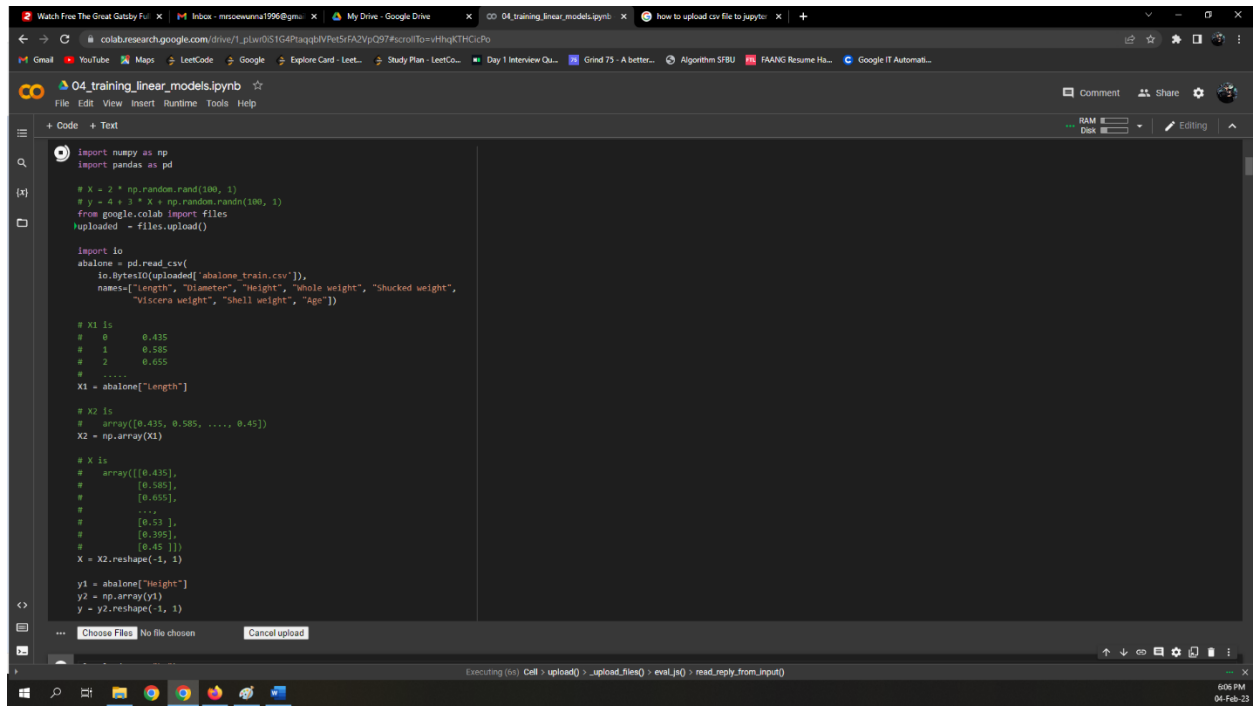
# X2 is
# array([0.435, 0.585, ..., 0.45])
X2 = np.array(X1)

# X is
# array([[0.435],
#        [0.585],
#        [0.655],
#        ...,
#        [0.43 ],
#        [0.395],
#        [0.45 ]])
X = X2.reshape(-1, 1)

y1 = abalone["Height"]
y2 = np.array(y1)
y = y2.reshape(-1, 1)
```

The bottom status bar indicates that the file "abalone\_train.csv" was uploaded successfully (145916 bytes, last modified: 1/31/2023, 100% done) and the code was executed at 6:18 PM on 04-Feb-23.

Choose the file to upload.



The screenshot shows the same Google Colab notebook as before, but with a file upload dialog box open at the bottom. The dialog box has a "Choose File" button and a "Cancel upload" button. The status bar at the bottom indicates that the code is executing and the file upload process is ongoing.

csv file is uploaded.

```

import numpy as np
import pandas as pd

# X = 2 * np.random.rand(100, 1)
# y = 4 + 3 * X + np.random.randn(100, 1)
from google.colab import files
uploaded = files.upload()

import io
abalone = pd.read_csv(
    io.BytesIO(uploaded['abalone_train.csv']),
    names=["Length", "Diameter", "Height", "Whole weight", "Shucked weight",
           "Viscera weight", "Shell weight", "Age"])

# X1 is
# 0      0.435
# 1      0.585
# 2      0.655
# .....
X1 = abalone["Length"]

# X2 is
# array([0.435, 0.585, ..., 0.45])
X2 = np.array(X1)

# X is
# array([[0.435],
#        [0.585],
#        [0.655],
#        ...,
#        [0.53 ],
#        [0.395],
#        [0.45 ]])
X = X2.reshape(-1, 1)

y1 = abalone["Weight"]
y2 = np.array(y1)
y = y2.reshape(-1, 1)

```

abalone\_train.csv  
 abalone\_train.csv(text/csv) - 145915 bytes, last modified: 1/31/2023 - 100% done  
 Saving abalone\_train.csv to abalone\_train.csv

42s completed at 6:07 PM

Changing 100 to 3320 and the problem has been solved.

```

X_b = np.c_[np.ones((3320, 1)), X] # add x0 = 1 to each instance
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)

```

Or

We need to define nrows = 100.

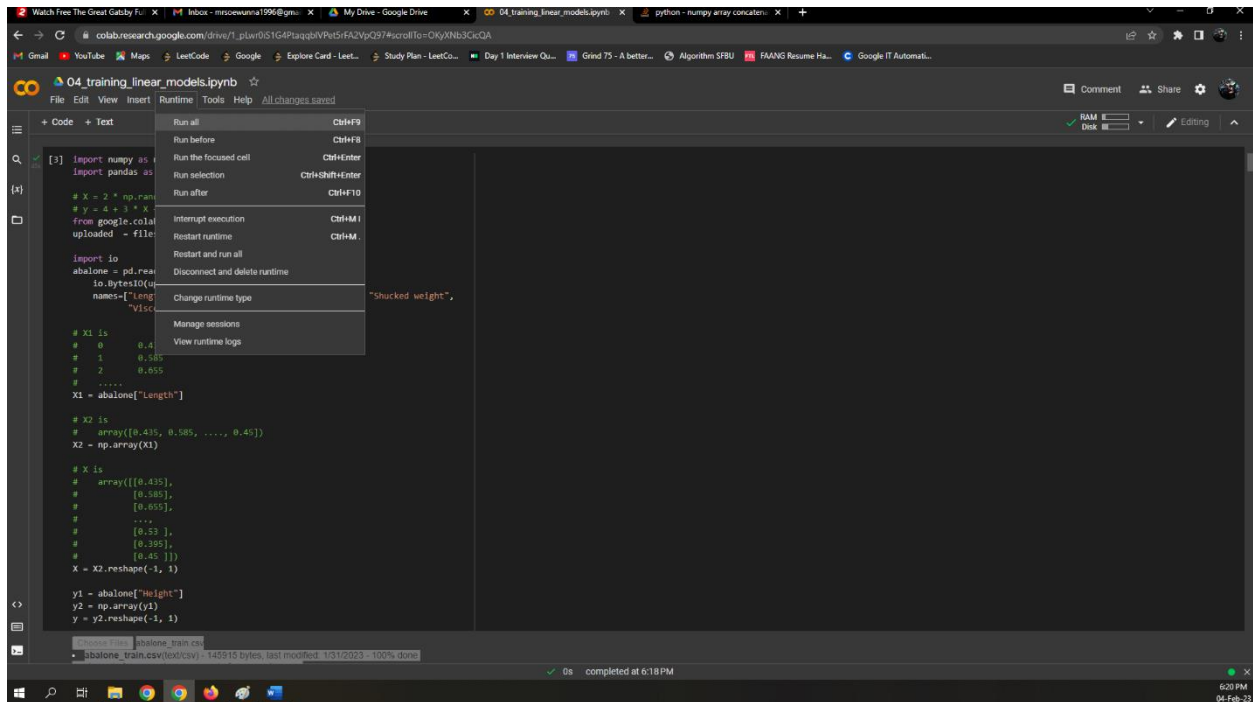
```

import io
abalone = pd.read_csv(
    io.BytesIO(uploaded['abalone_train.csv']),
    names=["Length", "Diameter", "Height", "Whole weight", "Shucked weight",
           "Viscera weight", "Shell weight", "Age"], nrows = 100)

# X1 is

```

Running all blocks of code.



```
[3] import numpy as np
import pandas as pd

# X1 is
# 0 0.4
# 1 0.585
# 2 0.555
# ...
# X2 is
# array([0.435, 0.585, ..., 0.45])
X2 = np.array(X1)

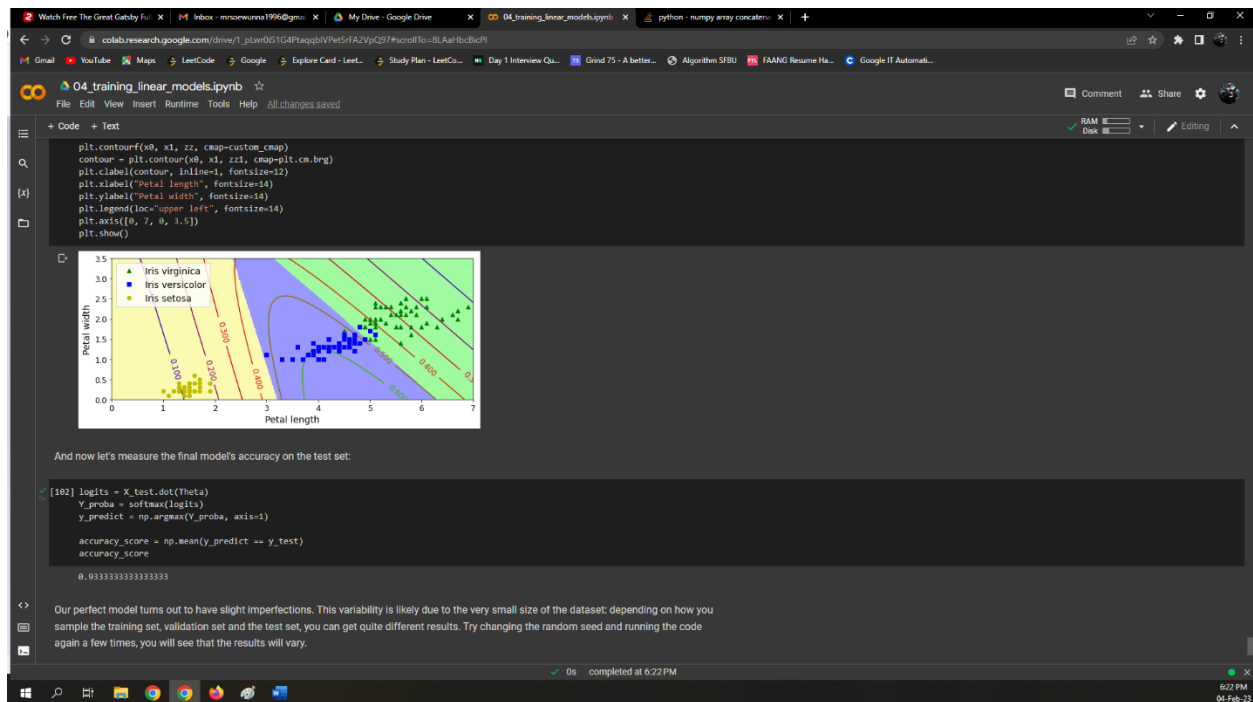
# X is
# array([[0.435,
#         0.585,
#         0.555,
#         ...,
#         0.53 ],
#        [0.395,
#        [0.45 ]])
X = X2.reshape(-1, 1)

y1 = abalone["height"]
y2 = np.array(y1)
y = y2.reshape(-1, 1)
```

abalone\_train.csv (text/csv) · 145915 bytes, last modified 1/31/2023 · 100% done

completed at 6:18 PM

The entire notebook has run successfully.



```
plt.contourf(x0, x1, z2, cmap=custom_cmap)
contour = plt.contour(x0, x1, z2, cmap=plt.cm.brg)
plt.xlabel("Petal length", fontsize=14)
plt.ylabel("Petal width", fontsize=14)
plt.legend(loc="upper left", fontsize=14)
plt.axis([0, 7, 0, 3.5])
plt.show()
```

And now let's measure the final model's accuracy on the test set:

```
[102] logits = X_test.dot(weights)
y_proba = softmax(logits)
y_predict = np.argmax(y_proba, axis=1)

accuracy_score = np.mean(y_predict == y_test)
accuracy_score
```

0.9333333333333333

Our perfect model turns out to have slight imperfections. This variability is likely due to the very small size of the dataset: depending on how you sample the training set, validation set and the test set, you can get quite different results. Try changing the random seed and running the code again a few times, you will see that the results will vary.

completed at 6:22 PM