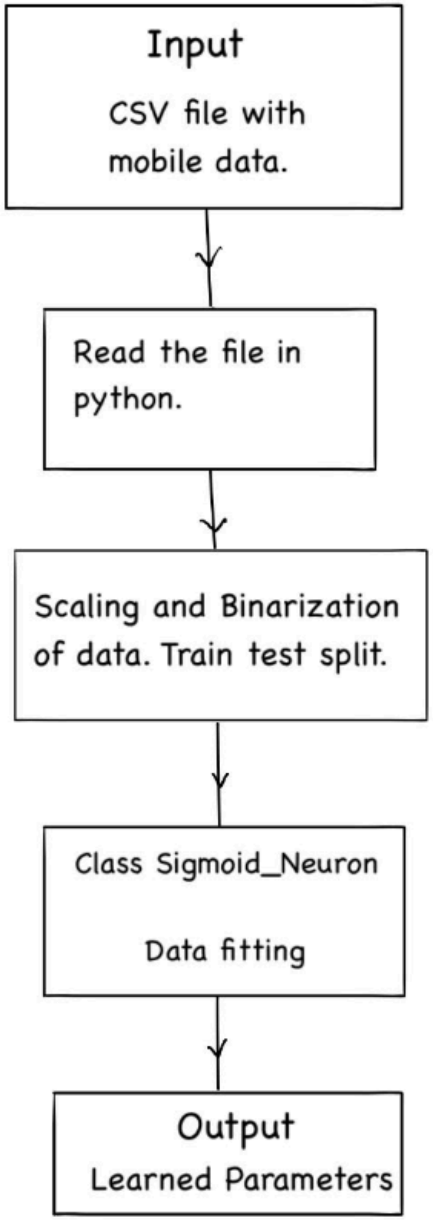
SIGMOID NEURON FOR MOBILE RATING CLASSIFICATION

Aniket Pal
244161002
Python Project

Motivation Behind the Problem

- Classification is a very basic and important problem in this world.
- We as humans classify objects in our day to day life(on limited data).
- There are many classification problems like image detection, movie recommendation etc.
- So it will be good to have model that will do this for us efficiently.
- I have taken the problem to classify mobile phones as 'likeable' or 'not likeable' based on various features. It is clearly a classification problem.
- Importance: Helps manufacturers and customers make informed decisions.
- To solve this problem I have used a very basic ML model called “Sigmoid Neuron”.

Block Diagram of the Solution



Input

CSV file with mobile data.

Read the file in python.

Scaling and Binarization of data. Train test split.

Class Sigmoid_Neuron

Data fitting

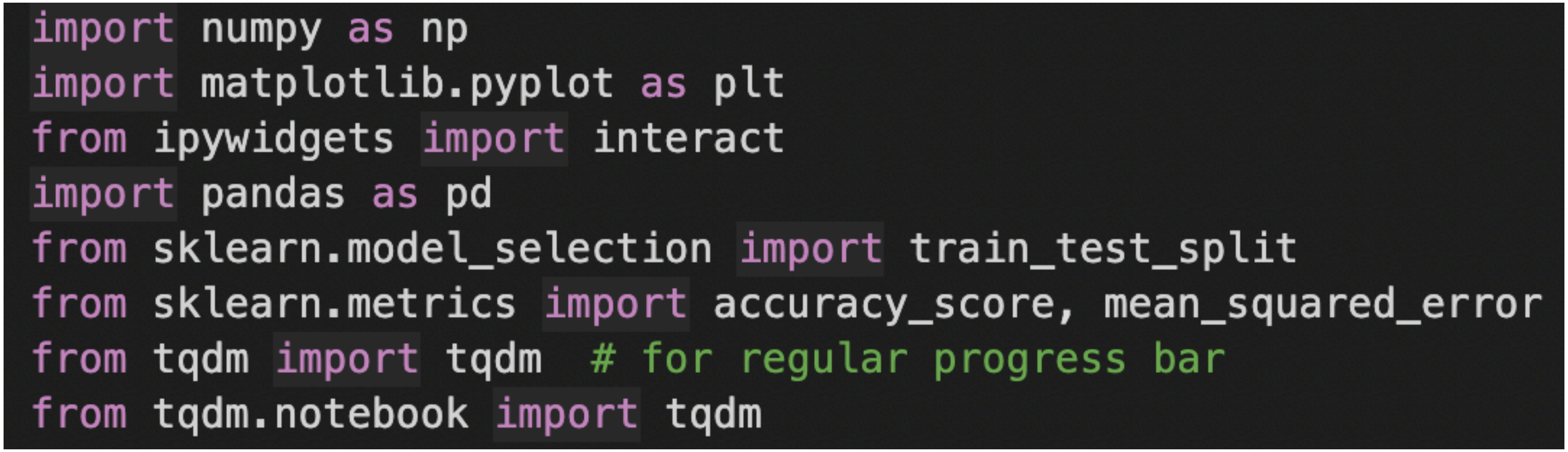
Output

Learned Parameters

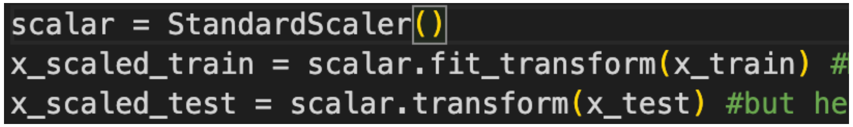
Pseudo-Code View of the Solution

- 1. Load data
- 2. Preprocess data (scaling, binarization)
- 4. Train-Test split
- 3. Initialise model(Randomly)
- 4. For each epoch:
-     - Compute predictions(To compute the loss)
-     - Calculate loss
-     - Update weights
- 5. Evaluate model accuracy

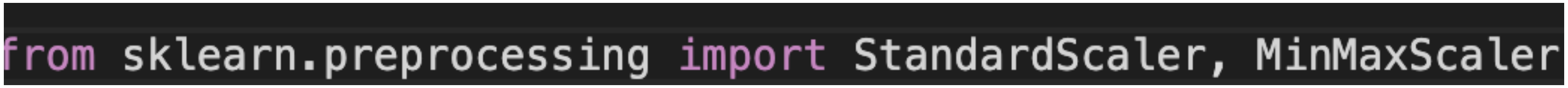
Some Important Python Features Used.



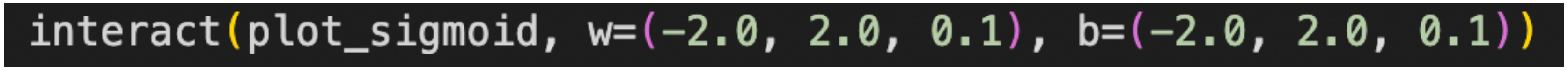
```python
import numpy as np
import matplotlib.pyplot as plt
from ipywidgets import interact
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, mean_squared_error
from tqdm import tqdm  # for regular progress bar
from tqdm.notebook import tqdm
```



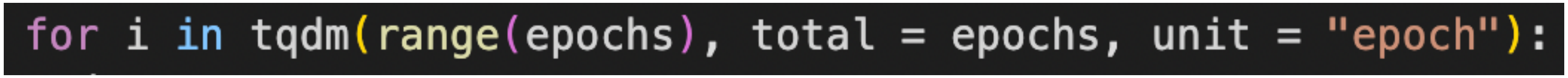
```python
scalar = StandardScaler()
x_scaled_train = scalar.fit_transform(x_train) #
x_scaled_test = scalar.transform(x_test) #but he
```



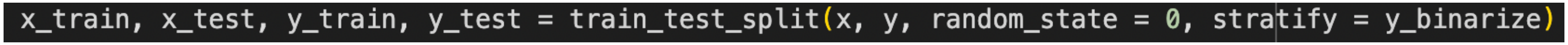
```python
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```



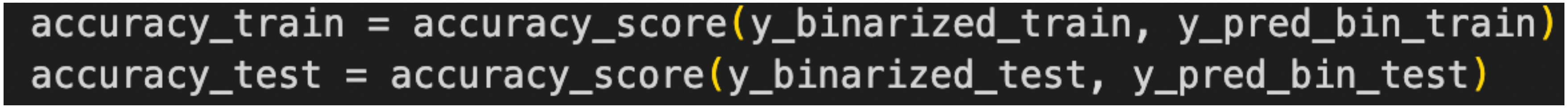
```python
interact(plot_sigmoid, w=(-2.0, 2.0, 0.1), b=(-2.0, 2.0, 0.1))
```



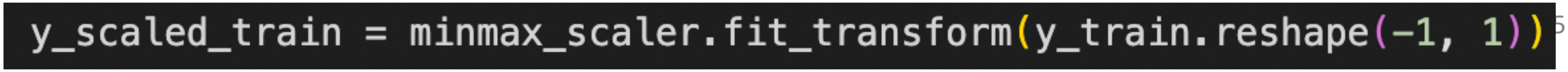
```python
for i in tqdm(range(epochs), total = epochs, unit = "epoch"):
```



```python
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state = 0, stratify = y_binarize)
```

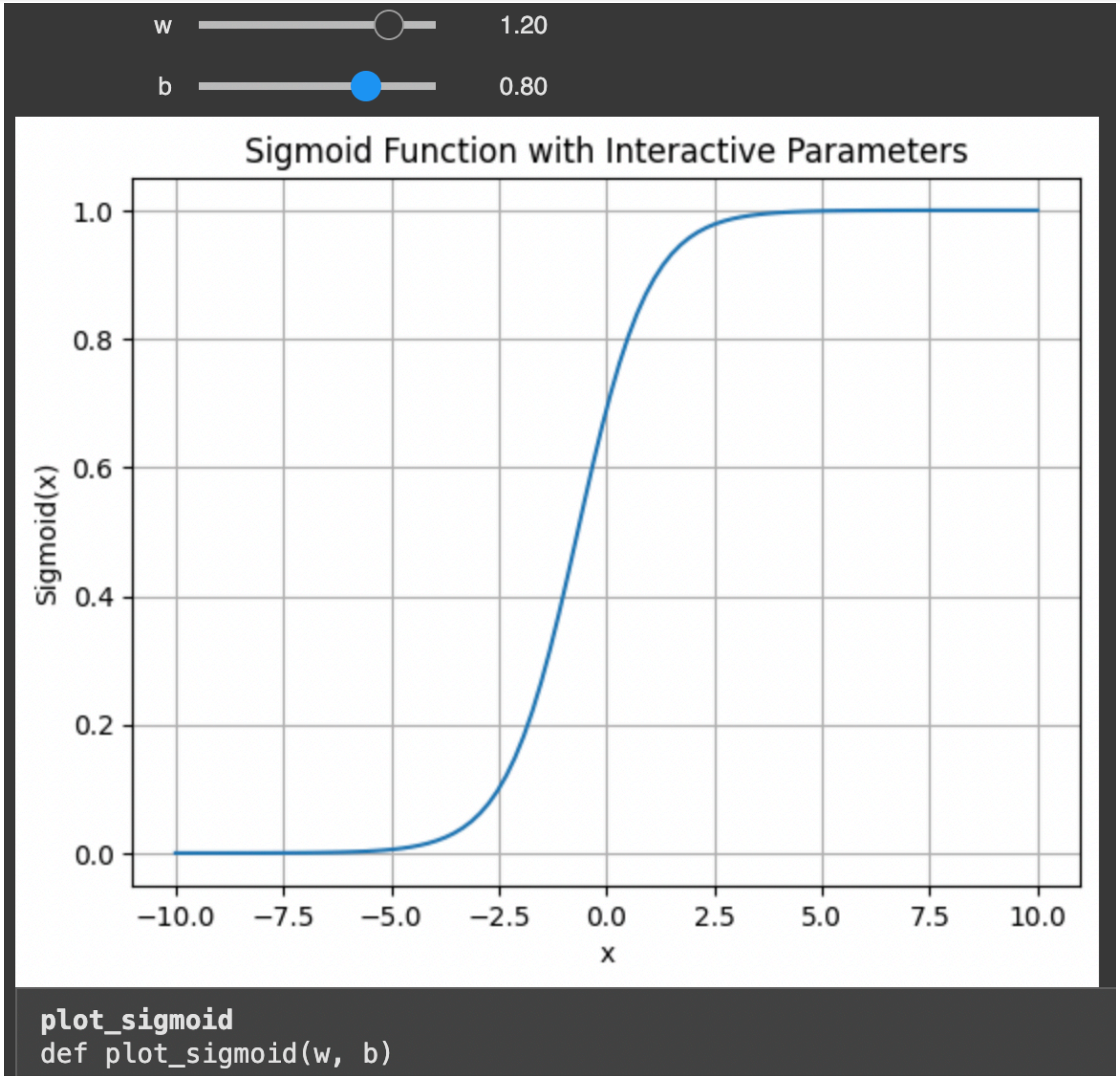


```python
accuracy_train = accuracy_score(y_binarized_train, y_pred_bin_train)
accuracy_test = accuracy_score(y_binarized_test, y_pred_bin_test)
```



```python
y_scaled_train = minmax_scaler.fit_transform(y_train.reshape(-1, 1))
```

Interactive Plot



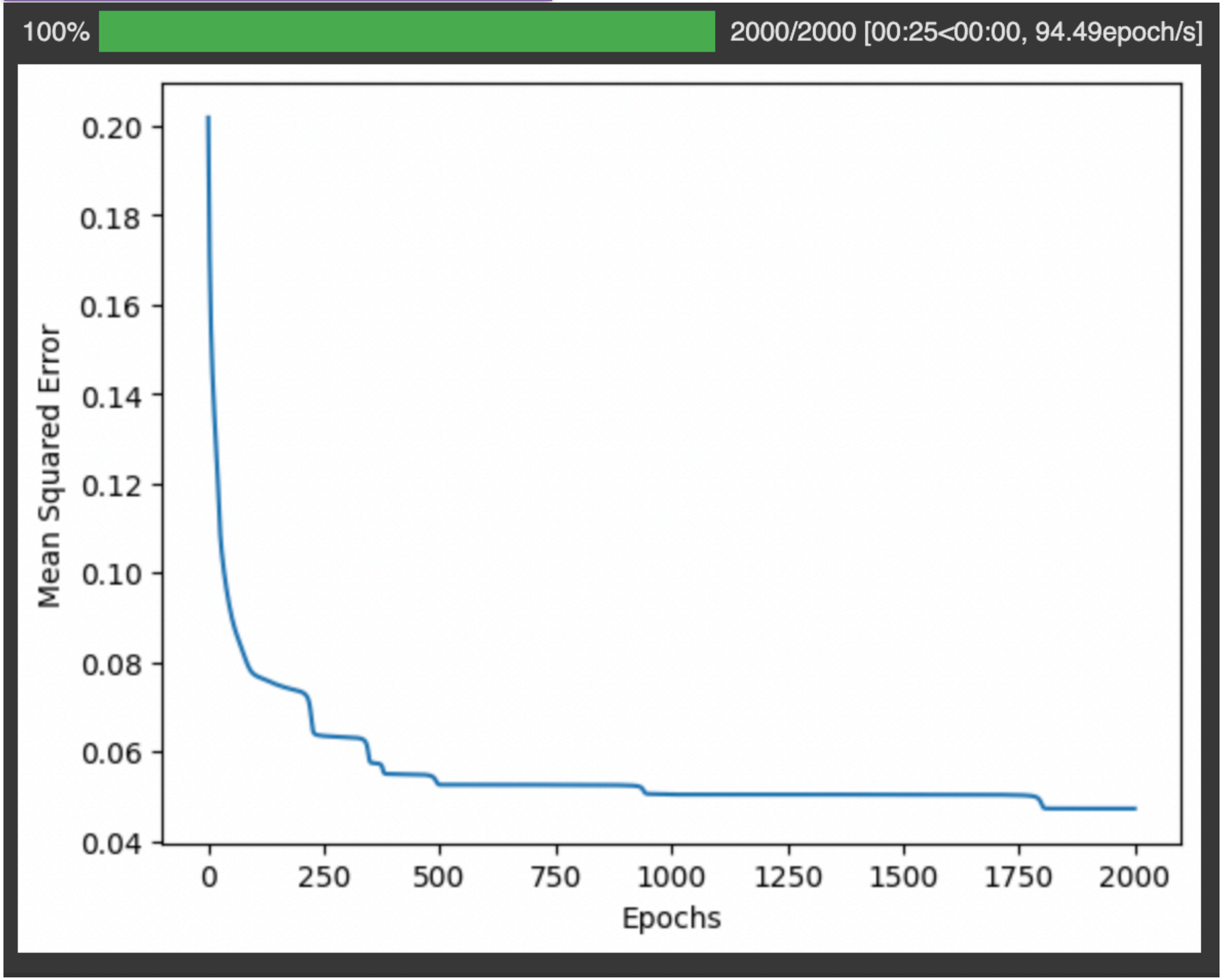
w 1.20

b 0.80

```
plot_sigmoid
def plot_sigmoid(w, b)
```

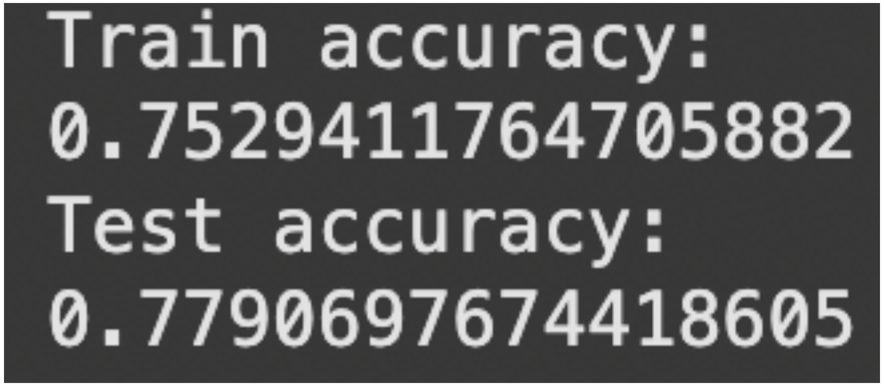
Loss vs no. of epochs

- We can see that total time taken is 25 sec.
- 49 epochs per second.
- Total no. of epochs - 2000.



```
100%  2000/2000 [00:25<00:00, 94.49epoch/s]
```

Output of the Model



```
Train accuracy:
0.7529411764705882
Test accuracy:
0.7790697674418605
```

Observations from the Results

- Learning rate tuning improved model performance.
- Standardisation and binarisation were crucial preprocessing steps.
- Interactive plots were useful for understanding parameter behaviour.
- Our model classifies correctly 75% of the times on training set and around 78% on the test set.

Learnings

- Basics of Python.
- Object oriented programming.
- Some standard libraries like, numpy, Pandas, etc.
- File Handling.
- Analysing textual and Image data.
- Plotting histogram, pie charts.
- Learned how to make 3d and interactive plots.
- Experience with data preprocessing, such as scaling and binarising.
- Implemented a simple machine learning model (Sigmoid Neuron).
- Understood how hyper-parameter tuning affects model performance.

Thank You