Question 1:

NAND Gate

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

In the NAND gate , the result is 0 only if all the inputs are 0, otherwise it is 1.

{
If $x_1w_1 + x_2w_2 + b > 0$, then Y' = 1
Else Y' = 0

For Row 1

where we initialise x1 = 0, x2 = 0, w1 = 1, w2 = 1, b = -1
then,

$$x_1w_1 + x_2w_2 + b$$
$$= x_1(1) + x_2(1) + (-1)$$
$$= 0.(1) + 0.(1) + (-1) = -1$$ (Below threshold value. So, y' = 0)

So, the output in this case is 0, But the required output is 1.

By changing the value of b to 1.
then,

$$x_1w_1 + x_2w_2 + b$$
$$= x_1(1) + x_2(1) + (1)$$
$$= 0.(1) + 0.(1) + (1) = 1$$ (Above threshold value. So, y' = 1)

From perceptron rule, this is correct.

For Row 2

where we initialise x1 = 0, x2 = 1, w1 = 1, w2 = 1, b = 1
then,

$$x_1w_1 + x_2w_2 + b$$
$$= x_1(1) + x_2(1) + (1)$$
$$= 0.(1) + 1.(1) + (1) = 2$$ (Above threshold value. So, y' = 1)

So, the output in this case is 1 and the required output is also 1. From perceptron rule, this is correct.

For Row 3

where we initialise x1 = 1, x2 = 0, w1 = 1, w2 = 1, b = 1 then,

$$x_1w_1 + x_2w_2 + b$$
$$= x_1(1) + x_2(1) + (1)$$
$$= 1.(1) + 0.(1) + (1) = 1 \text{ (Above threshold value. So, y' = 1)}$$

So, the output in this case is 1 and the required output is also 1. From perceptron rule, this is correct.

For Row 4

where we initialise x1 = 1, x2 = 1, w1 = 1, w2 = 1, b = 1

then,

$$x_1w_1 + x_2w_2 + b$$
$$= x_1(1) + x_2(1) + (1)$$
$$= 1.(1) + 1.(1) + (1) = 1 \text{ (Above threshold value. So, y' = 1)}$$

So, the output in this case is 1 but the required output is 0.

a. By changing the value of w1 = -2.

then,

$$x_1w_1 + x_2w_2 + b$$
$$= x_1(-2) + x_2(1) + (1)$$
$$= 1.(-2) + 1.(1) + (1) = 0 \text{ (Below threshold value. So, y' = 0)}$$

But this will give incorrect output for Row 3.

b. By changing the value of w2 = -2.

then,

$$x_1w_1 + x_2w_2 + b$$
$$= x_1(1) + x_2(-2) + (1)$$
$$= 1.(1) + 1.(-2) + (1) = 0 \text{ (Below threshold value. So, y' = 0)}$$

But this will give incorrect output for Row 2.

c. By changing the value of w1= -1, w2 = -1 and b = 2.

then,

$$x_1w_1 + x_2w_2 + b$$
$$= x_1(-1) + x_2(-1) + (2)$$
$$= 1.(-1) + 1.(-1) + (2) = 0 \text{ (Below threshold value. So, y' = 0)}$$

Gives correct output for Row 4.

1) Let’s check for Row 1 for the same inputs.

x1 = 0, x2 = 0, w1 = -1, w2 = -1, b = 2

then,

$$x_1w_1 + x_2w_2 + b$$
$$= x_1(-1) + x_2(-1) + (2)$$
$$= 0.(-1) + 0.(-1) + (2) = 2 \text{ (Above threshold value. So, y' = 1)}$$

2) Let’s check for Row 2 for the same inputs.

x1 = 0, x2 = 1, w1 = -1, w2 = -1, b = 2

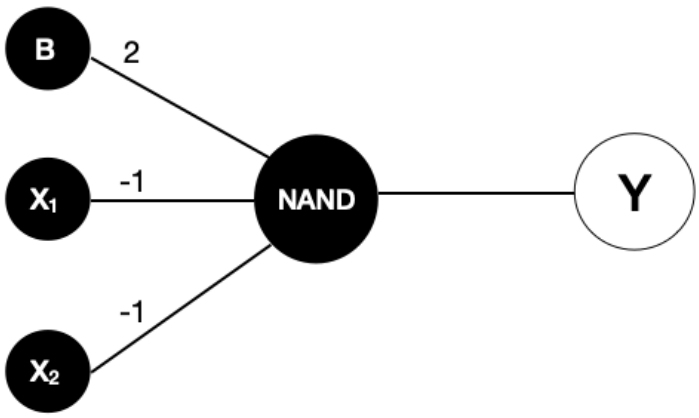
then,

$$x_1w_1 + x_2w_2 + b$$
$$= x_1(-1) + x_2(-1) + (2)$$
$$= 0.(-1) + 1.(-1) + (2) = 1 \text{ (Above threshold value. So, } y' = 1)$$

3) Let's check for Row 3 for the same inputs.
x1 = 1, x2 = 0, w1 = -1, w2 = -1, b = 2
then,

$$x_1w_1 + x_2w_2 + b$$
$$= x_1(-1) + x_2(-1) + (2)$$
$$= 1.(-1) + 0.(-1) + (2) = 1 \text{ (Above threshold value. So, } y' = 1)$$

Therefor, our NAND gate model is achieved with w1 = -1, w2 = -1 and b = 2.



Code : -

```
import numpy as np
class Perceptron(object):
    def __init__(self, input_size, lr=1, epochs=100):
        self.W = np.zeros(input_size + 1)
        self.epochs = epochs
        self.lr = lr

    def activation_fn(self, x):
        return 1 if x > 0 else 0

    def predict(self, x):
        z = self.W.T.dot(x)
        a = self.activation_fn(z)
```

```
        return a
    def fit(self, X, d):
        for _ in range(self.epochs):
            for i in range(d.shape[0]):
                x = np.insert(X[i], 0, 1)
                y = self.predict(x)
                e = d[i] - y
                self.W = self.W + self.lr * e * x

if __name__ == '__main__':
    X = np.array([
        [0, 0],
        [0, 1],
        [1, 0],
        [1, 1]
    ])
    d = np.array([1, 1, 1, 0])
    perceptron = Perceptron(input_size=2)
    perceptron.fit(X, d)
    print(perceptron.W)
```

Question 2:

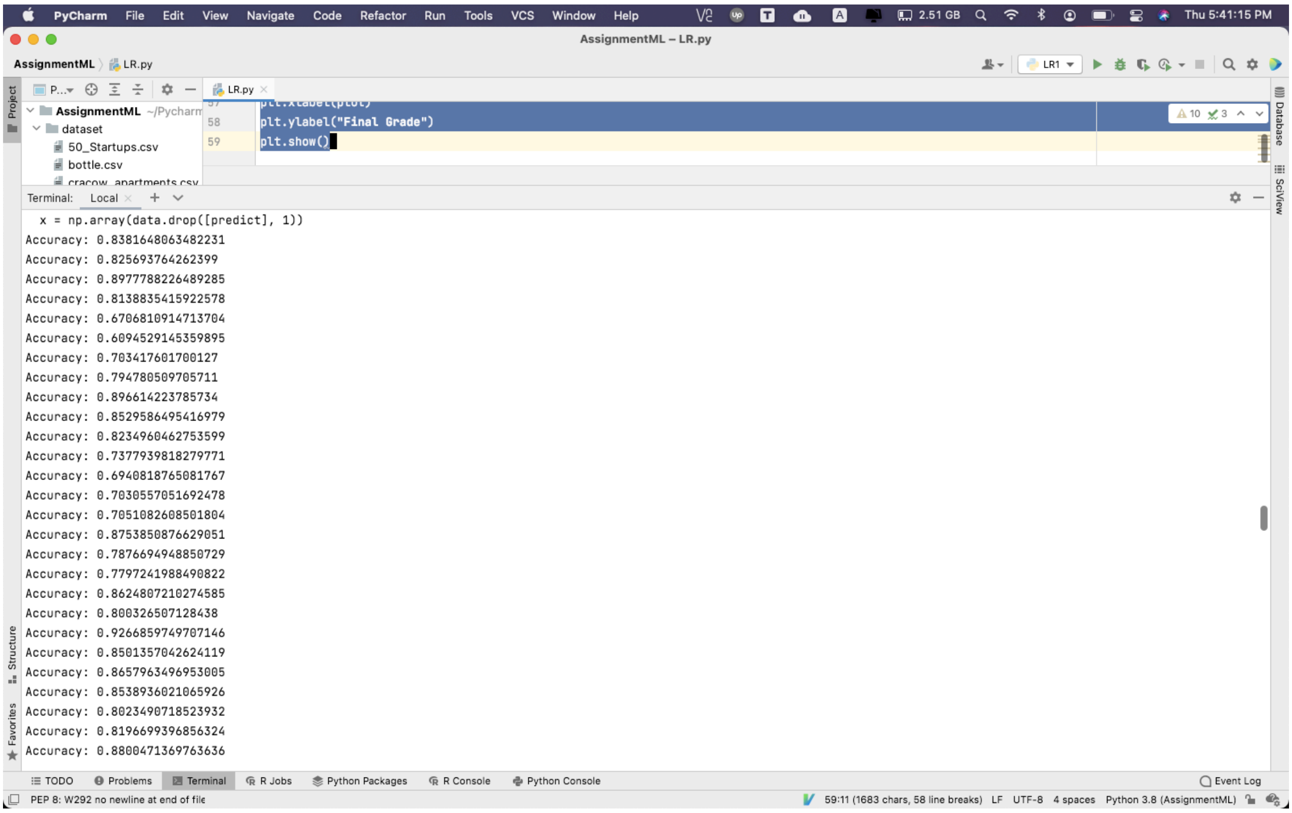
Part 1

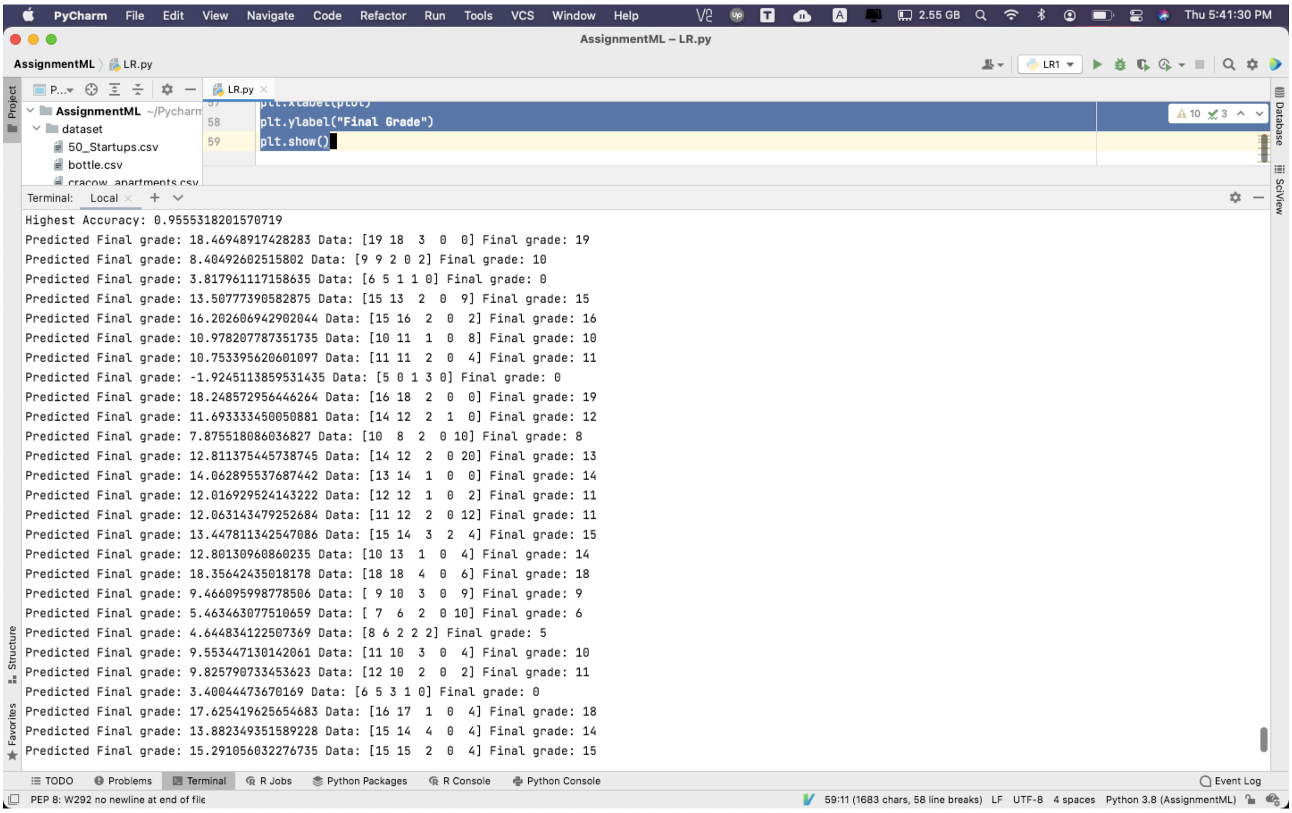
Yes, we can use the perceptron learning for Linear Regression. The perceptron learning algorithm is mainly used for (binary) classification problems because of its capability to handle linear separability. It uses a discrete function (step function) as the activation function. The basic step function returns two outputs - 0 or 1.

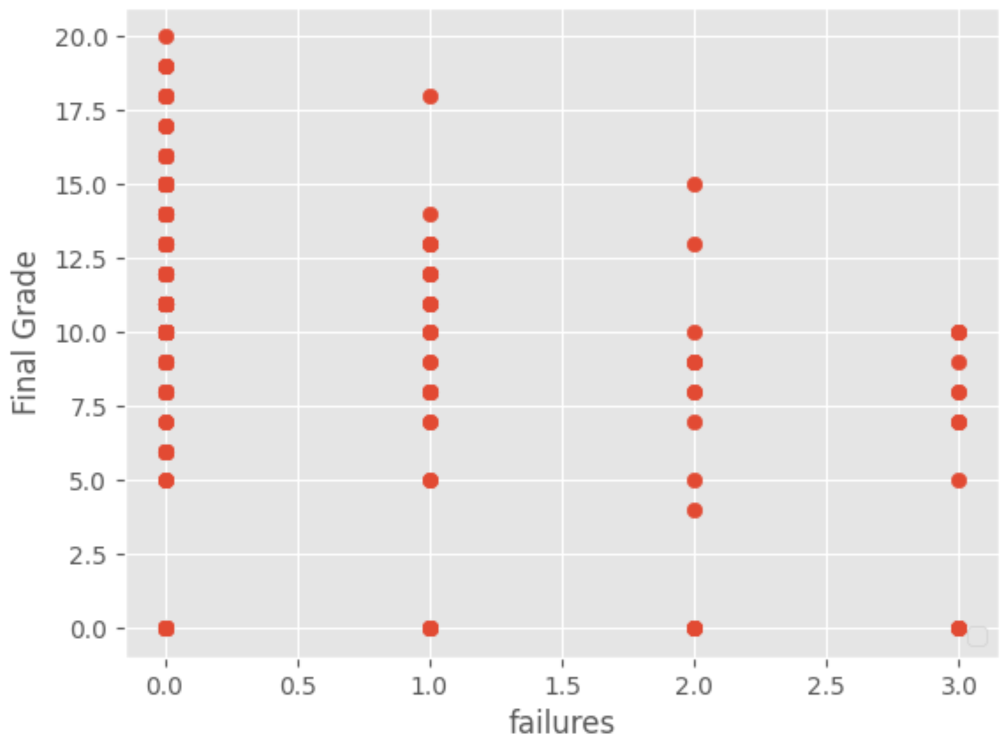
For the experiment purpose, we can replace the activation function with a continuous function such as the sigmoid function. Results might not be that good as compared to other regression-specific algorithms.

```
import pandas as pd
import numpy as np
import sklearn
from sklearn import linear_model
from sklearn.utils import shuffle
import matplotlib.pyplot as plt
from matplotlib import style
import pickle
style.use("ggplot")
```

```
data = pd.read_csv("dataset/student-mat.csv", sep=";")

predict = “G3"

data = data[[ "G1", "G2", "G3", "studytime", "failures", "absences"]]
data = shuffle(data)

x = np.array(data.drop([predict], 1))
y = np.array(data[predict])

x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(x, y, test_size = 0.1)

# Train model multiple times to find the highest accuracy
best = 0
for _ in range(200):
    x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(x, y, test_size = 0.1)
    linear = linear_model.LinearRegression()
    linear.fit(x_train, y_train)
    acc = linear.score(x_test, y_test)
    print("Accuracy: " + str(acc))
    if (acc > best):
        best = acc
        with open("stdgrd.pickle", "wb") as f:
            pickle.dump(linear, f)
print("Highest Accuracy:", best)

pickle_in = open("stdgrd.pickle", "rb")
linear = pickle.load(pickle_in)

predictions = linear.predict(x_test)

for x in range(len(predictions)):
    print("Predicted Final grade:", predictions[x], "Data:", x_test[x], "Final grade:", y_test[x])

# Create visualisation of the model
plot = "failures"
plt.scatter(data[plot], data["G3"])
plt.legend(loc=4)
plt.xlabel(plot)
plt.ylabel("Final Grade")
plt.show()
```

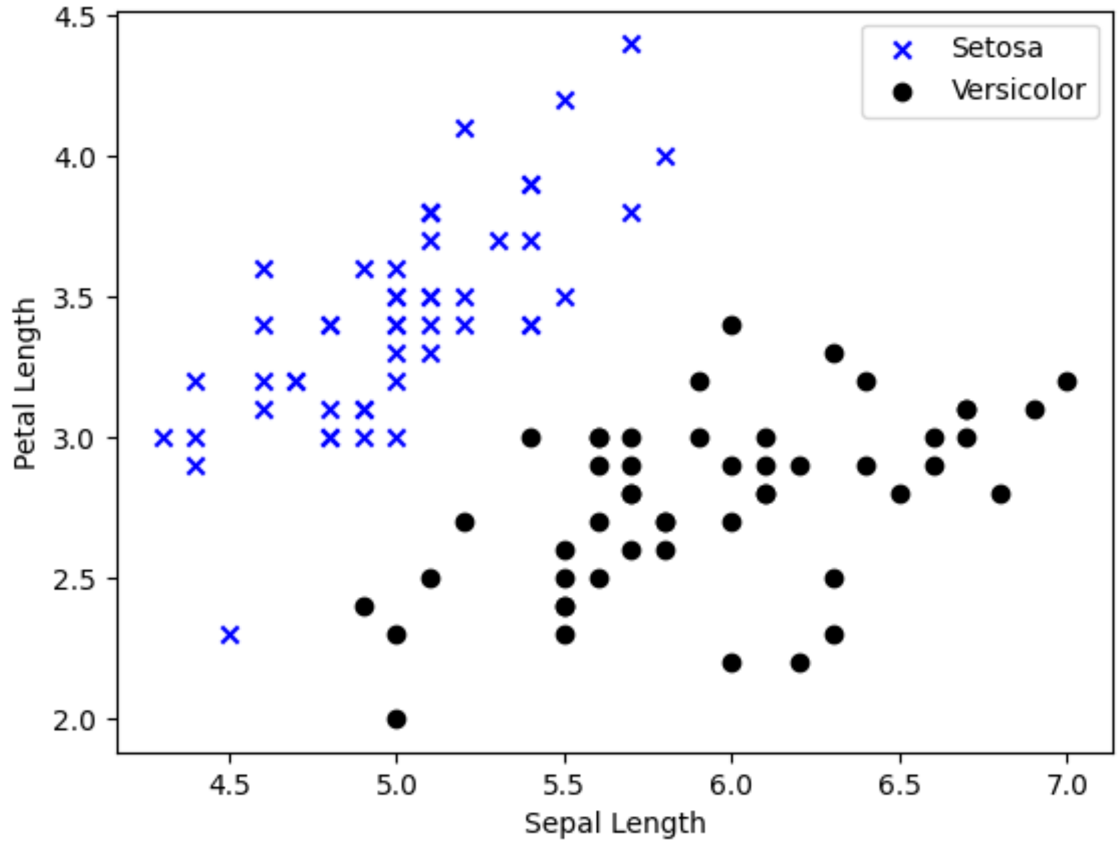

PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help
Thu 5:41:15 PM
AssignmentML – LR.py
AssignmentML LR.py
AssignmentML ~/Pycharm
dataset
50_Startups.csv
bottle.csv
58 plt.ylabel("Final Grade")
59 plt.show()
Terminal: Local
x = np.array(data.drop([predict], 1))
Accuracy: 0.8381648063482231
Accuracy: 0.825693764262399
Accuracy: 0.8977788226489285
Accuracy: 0.8138835415922578
Accuracy: 0.6706810914713704
Accuracy: 0.6094529145359895
Accuracy: 0.703417601700127
Accuracy: 0.794780509705711
Accuracy: 0.896614223785734
Accuracy: 0.8529586495416979
Accuracy: 0.8234960462753599
Accuracy: 0.7377939818279771
Accuracy: 0.6940818765081767
Accuracy: 0.7030557051692478
Accuracy: 0.7051082608501804
Accuracy: 0.8753850876629051
Accuracy: 0.7876694948850729
Accuracy: 0.7797241988490822
Accuracy: 0.8624807210274585
Accuracy: 0.800326507128438
Accuracy: 0.9266859749707146
Accuracy: 0.8501357042624119
Accuracy: 0.8657963496953005
Accuracy: 0.8538936021065926
Accuracy: 0.8023490718523932
Accuracy: 0.8196699396856324
Accuracy: 0.8800471369763636
TODO Problems Terminal R Jobs Python Packages R Console Python Console Event Log
PEP 8: W292 no newline at end of file
59:11 (1683 chars, 58 line breaks) LF UTF-8 4 spaces Python 3.8 (AssignmentML)


PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help
Thu 5:41:30 PM
AssignmentML – LR.py
AssignmentML LR.py
AssignmentML ~/Pycharm
dataset
50_Startups.csv
bottle.csv
58 plt.ylabel("Final Grade")
59 plt.show()
Terminal: Local
Highest Accuracy: 0.9555318201570719
Predicted Final grade: 18.46948917428283 Data: [19 18 3 0 0] Final grade: 19
Predicted Final grade: 8.40492602515802 Data: [9 9 2 0 2] Final grade: 10
Predicted Final grade: 3.817961117158635 Data: [6 5 1 1 0] Final grade: 0
Predicted Final grade: 13.50777390582875 Data: [15 13 2 0 9] Final grade: 15
Predicted Final grade: 16.202606942902044 Data: [15 16 2 0 2] Final grade: 16
Predicted Final grade: 10.978207787351735 Data: [10 11 1 0 8] Final grade: 10
Predicted Final grade: 10.753395620601097 Data: [11 11 2 0 4] Final grade: 11
Predicted Final grade: -1.9245113859531435 Data: [5 0 1 3 0] Final grade: 0
Predicted Final grade: 18.248572956446264 Data: [16 18 2 0 0] Final grade: 19
Predicted Final grade: 11.693333450050881 Data: [14 12 2 1 0] Final grade: 12
Predicted Final grade: 7.875518086036827 Data: [10 8 2 0 10] Final grade: 8
Predicted Final grade: 12.811375445738745 Data: [14 12 2 0 20] Final grade: 13
Predicted Final grade: 14.062895537687442 Data: [13 14 1 0 0] Final grade: 14
Predicted Final grade: 12.016929524143222 Data: [12 12 1 0 2] Final grade: 11
Predicted Final grade: 12.063143479252684 Data: [11 12 2 0 12] Final grade: 11
Predicted Final grade: 13.447811342547086 Data: [15 14 3 2 4] Final grade: 15
Predicted Final grade: 12.80130960860235 Data: [10 13 1 0 4] Final grade: 14
Predicted Final grade: 18.35642435018178 Data: [18 18 4 0 6] Final grade: 18
Predicted Final grade: 9.466095998778506 Data: [9 10 3 0 9] Final grade: 9
Predicted Final grade: 5.463463077510659 Data: [7 6 2 0 10] Final grade: 6
Predicted Final grade: 4.644834122507369 Data: [8 6 2 2 2] Final grade: 5
Predicted Final grade: 9.553447130142061 Data: [11 10 3 0 4] Final grade: 10
Predicted Final grade: 9.825790733453623 Data: [12 10 2 0 2] Final grade: 11
Predicted Final grade: 3.40044473670169 Data: [6 5 3 1 0] Final grade: 0
Predicted Final grade: 17.625419625654683 Data: [16 17 1 0 4] Final grade: 18
Predicted Final grade: 13.882349351589228 Data: [15 14 4 0 4] Final grade: 14
Predicted Final grade: 15.291056032276735 Data: [15 15 2 0 4] Final grade: 15
TODO Problems Terminal R Jobs Python Packages R Console Python Console Event Log
PEP 8: W292 no newline at end of file
59:11 (1683 chars, 58 line breaks) LF UTF-8 4 spaces Python 3.8 (AssignmentML)

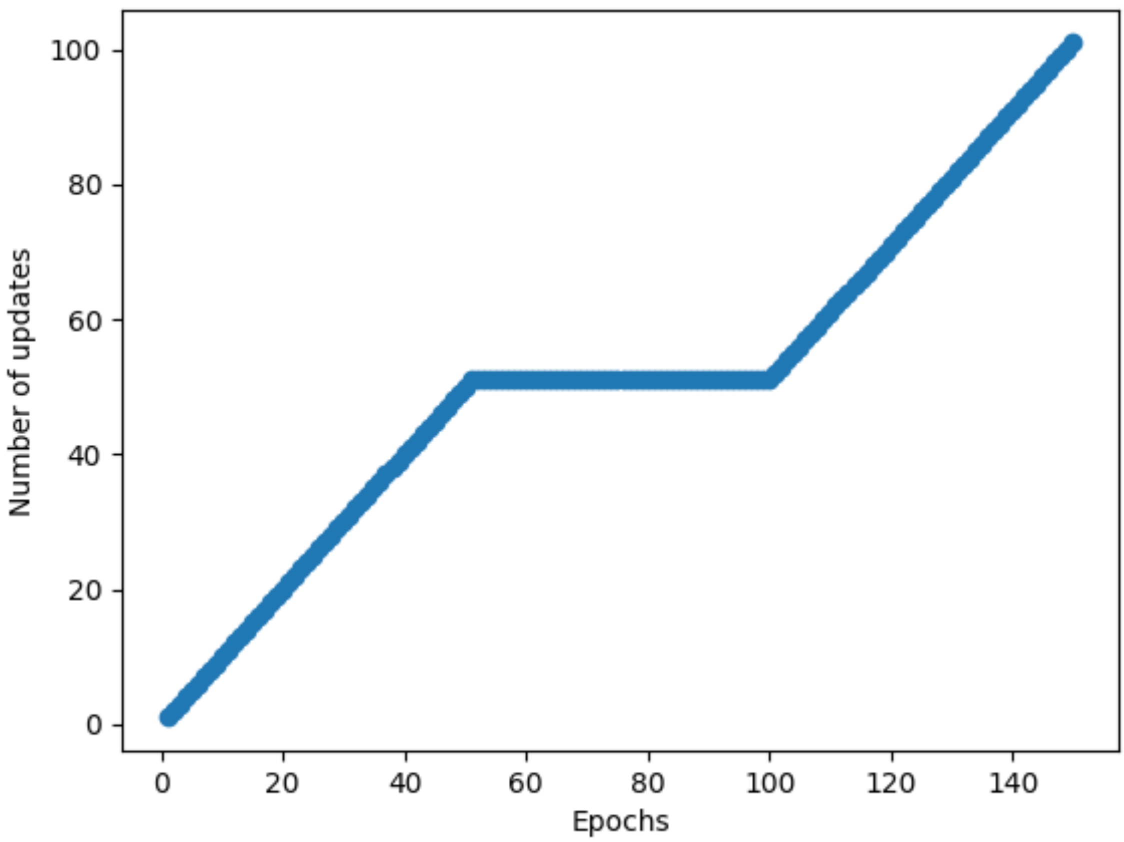


20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Final Grade
0.0
0.5
1.0
1.5
2.0
2.5
3.0
failures

Part 2



Setosa
Versicolor
Petal Length
Sepal Length
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0



Number of updates
Epochs
100
80
60
40
20
0
0
20
40
60
80
100
120
140

```python
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
import numpy as np

class Perceptron(object):
  def __init__(self, learning_rate=0.02, n_iter=50, random_state=1):
    self.learning_rate = learning_rate
    self.n_iter = n_iter
    self.random_state = random_state

  def fit(self, X, y):
    rand = np.random.RandomState(self.random_state)
    self.weights = rand.normal(loc=0.0, scale=0.01, size=1 +  X.shape[1])
    self.errors_ = []

    for _ in range(self.n_iter):
      errors = 0
      for x, target in zip(X, y):
        update = self.learning_rate * (target - self.predict(x))
        self.weights[1:] += update * x
        self.weights[0] += update
        errors += int(update != 0.0)
        self.errors_.append(errors)
      return self

  def net_input(self, X):
    z = np.dot(X, self.weights[1:]) + self.weights[0]
    return z

  def predict(self, X):
    return np.where(self.net_input(X) >= 0, 1, -1)

X,y = load_iris(return_X_y=True)
print(X,y)

plt.scatter(X[:50, 0], X[:50, 1],
            color='blue', marker='x', label='Setosa')
plt.scatter(X[50:100, 0], X[50:100, 1],
            color='black', marker='o', label='Versicolor')
```

```
plt.xlabel('Sepal Length')
plt.ylabel('Petal Length')
plt.legend(loc='upper right')
plt.show()

per = Perceptron(learning_rate=0.2, n_iter=50, random_state=1)
per.fit(X, y)
plt.plot(range(1, len(per.errors_) + 1), per.errors_, marker='o')
plt.xlabel('Epochs')
plt.ylabel('Number of updates')
plt.show()
```