AKDENIZ UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

Machine Learning – Homework 2

Temmuz Burak Yavuzer

20160808026

Purpose 1: You are expected to implement a python code for the basic perceptron and test it on the given "iris.csv" file. Train it for several epochs and calculate the accuracy.

Before all that we have to understand the *General Explanation of the Model* that we are going to use.

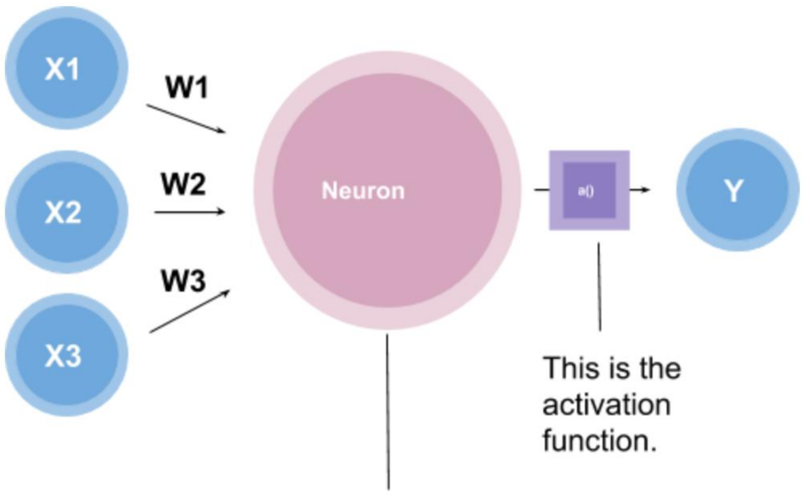
The Perceptron Algorithm

Is the simplest type of artificial neural network.

It is a model of a single neuron that can be used for two-class classification problems and provides the foundation for later developing much larger networks.

The Perceptron is inspired by the information processing of a single neural cell called a neuron.

A neuron accepts input signals via its dendrites, which pass the electrical signal down to the cell body.



$$\sum_{n=0}^{2} x_n w_n$$

There are different kinds of activation functions that exist, for example:

1. **Hyperbolic Tangent:** used to output a number from -1 to 1.
2. **Logistic Function:** used to output a number from 0 to 1.

etc.

Note: Activation functions also allow for non-linear classification.

Since the range we are looking for is between 0 and 1, we will be using a Logistic Function to achieve this.

Logistic Functions

Logistical functions have the formula,

$$g(z) = \frac{1}{1 + e^{-z}}$$

This will allow us to output numbers that are between 0 and 1 which is exactly what we need to build our perceptron.

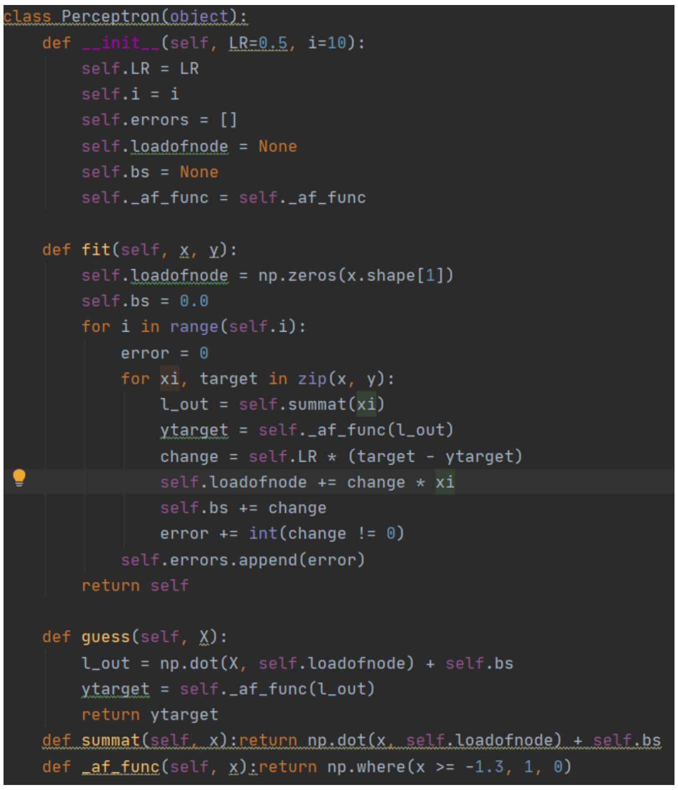
The bias is a threshold the perceptron must reach before the output is produced. So the final neuron equation looks like:

$$Y = \sum(weight * input) + bias$$

In a similar way, the Perceptron receives input signals from examples of training data that we weight and combined in a linear equation called the activation.

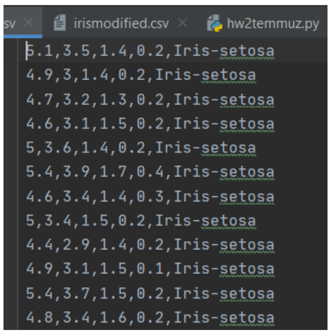
The activation is then transformed into an output value or prediction using a transfer function, such as the step transfer function.

Implementation of the Perception Model via Python

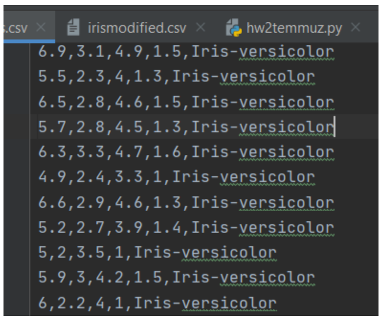


```python
class Perceptron(object):
    def __init__(self, LR=0.5, i=10):
        self.LR = LR
        self.i = i
        self.errors = []
        self.loadofnode = None
        self.bs = None
        self._af_func = self._af_func

    def fit(self, x, y):
        self.loadofnode = np.zeros(x.shape[1])
        self.bs = 0.0
        for i in range(self.i):
            error = 0
            for xi, target in zip(x, y):
                l_out = self.summat(xi)
                ytarget = self._af_func(l_out)
                change = self.LR * (target - ytarget)
                self.loadofnode += change * xi
                self.bs += change
                error += int(change != 0)
            self.errors.append(error)
        return self

    def guess(self, X):
        l_out = np.dot(X, self.loadofnode) + self.bs
        ytarget = self._af_func(l_out)
        return ytarget
    def summat(self, x):return np.dot(x, self.loadofnode) + self.bs
    def _af_func(self, x):return np.where(x >= -1.3, 1, 0)
```

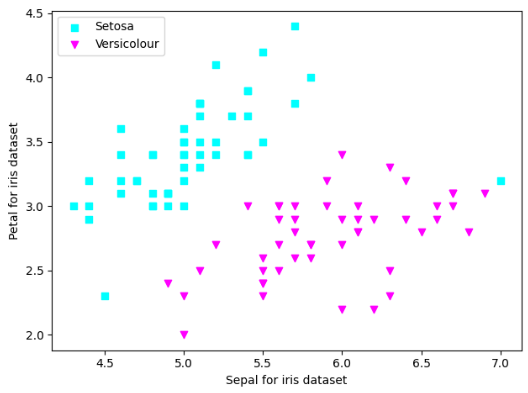
As we learned, perception model is based on main perception class with a 2 function fit and guess function to fitting the giving data and guessing the given input whether it is Iris Setosa or Iris Versicolor. Predict function returns the value of 0 for setosa 1 for versicolor with the help of the function that we talked about in the first part of the article. As you remember it is called as activation function. Summat function is short version of sum of matrix.It is obviously summing the matrixs input corresponding their the load. Fit function is one of the important function on the project because it is the main function to train the model with test data. Basically test data is around 20% or 30% of the entire data.Parameter of fit function means, x is used for s is the number of samples, f is for number of features.This means that we have SxF matrix. Y is for the type of the flower. This function basically , if (target – ytarget)xLearningRate result is equal to 0 then guessing is correct. Other wise it has to change the load of array and try again.

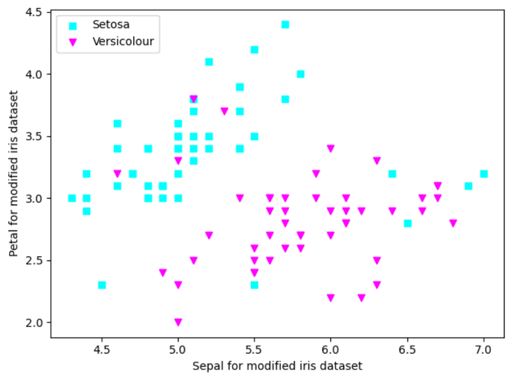


```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
```

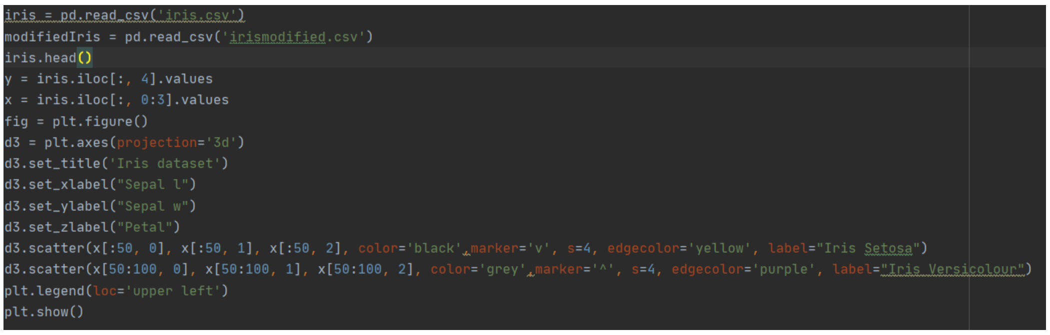


```
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4,1.3,Iris-versicolor
6.5,2.8,4.6,1.5,Iris-versicolor
5.7,2.8,4.5,1.3,Iris-versicolor
6.3,3.3,4.7,1.6,Iris-versicolor
4.9,2.4,3.3,1,Iris-versicolor
6.6,2.9,4.6,1.3,Iris-versicolor
5.2,2.7,3.9,1.4,Iris-versicolor
5,2,3.5,1,Iris-versicolor
5.9,3,4.2,1.5,Iris-versicolor
6,2.2,4,1,Iris-versicolor
```

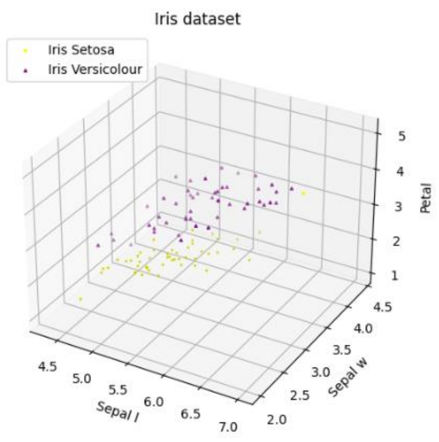




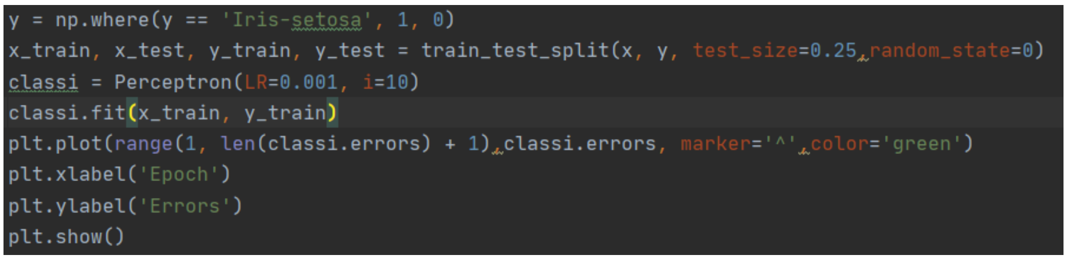
This is the our dataset that we are using. It has 4 attributes and classified as Iris Setosa and Iris Versicolor. First graph is for representing the Iris Dataset while the second graph is for the represending the Modified Dataset. We can easily detect the errors that should be change by just looking at those graphs.



```
iris = pd.read_csv('iris.csv')
modifiedIris = pd.read_csv('irismodified.csv')
iris.head()
y = iris.iloc[:, 4].values
x = iris.iloc[:, 0:3].values
fig = plt.figure()
d3 = plt.axes(projection='3d')
d3.set_title('Iris dataset')
d3.set_xlabel("Sepal l")
d3.set_ylabel("Sepal w")
d3.set_zlabel("Petal")
d3.scatter(x[:50, 0], x[:50, 1], x[:50, 2], color='black',marker='v', s=4, edgecolor='yellow', label="Iris Setosa")
d3.scatter(x[50:100, 0], x[50:100, 1], x[50:100, 2], color='grey',marker='^', s=4, edgecolor='purple', label="Iris Versicolour")
plt.legend(loc='upper left')
plt.show()
```

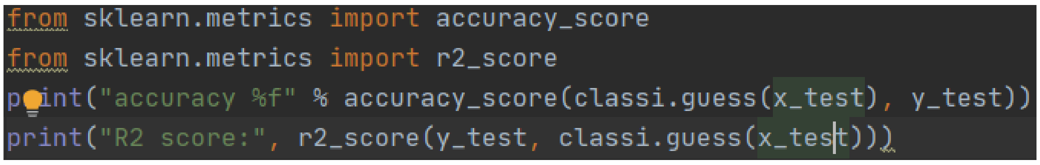


This part is basically for reading the csv file. Getting a general knowledge about the dataset using 3d plot by using matplotlib python library

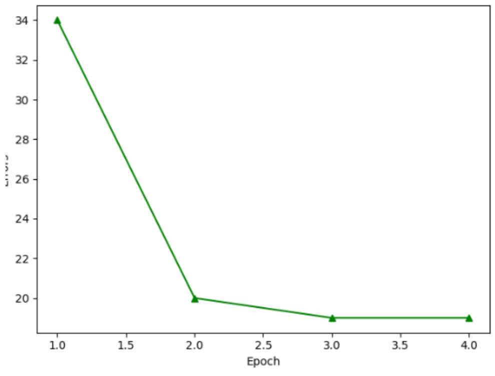


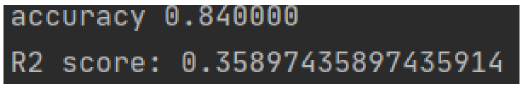
```
y = np.where(y == 'Iris-setosa', 1, 0)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25,random_state=0)
classi = Perceptron(LR=0.001, i=10)
classi.fit(x_train, y_train)
plt.plot(range(1, len(classi.errors) + 1),classi.errors, marker='^',color='green')
plt.xlabel('Epoch')
plt.ylabel('Errors')
plt.show()
```

We got the information that we want from the previous steps.We were able to see the problematic part of the dataset. Now, It is time to splitting the data into train and test part with the model. As you can see, we choosed 25% of the data as a test size. As a result we are able to we are able to see the errors by each epoch.With the help of the bottom of the code, we are able to see the accuracy. First part is the result of 4 iteration, Second part is the result of 100 iteration. This means that when we increase the iteration, we can train better the data and get much better results.

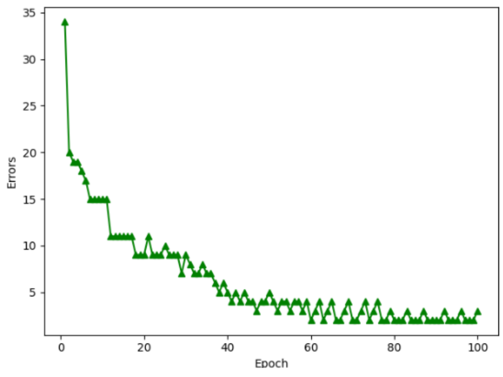


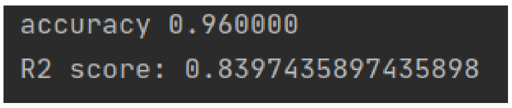
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score
print("accuracy %f" % accuracy_score(classi.guess(x_test), y_test))
print("R2 score:", r2_score(y_test, classi.guess(x_test)))
```





```
accuracy 0.840000
R2 score: 0.35897435897435914
```



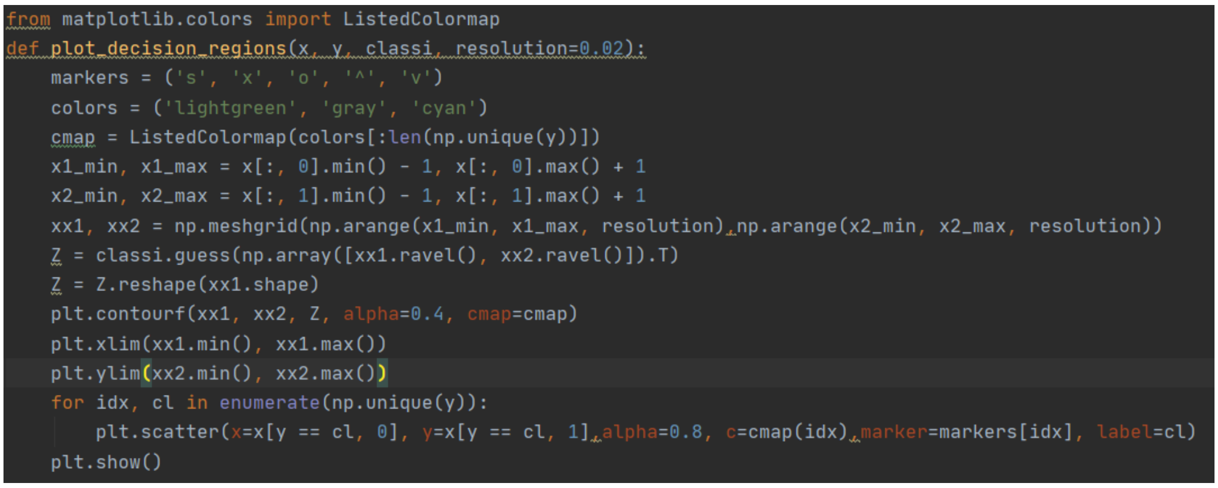


```
accuracy 0.960000
R2 score: 0.8397435897435898
```

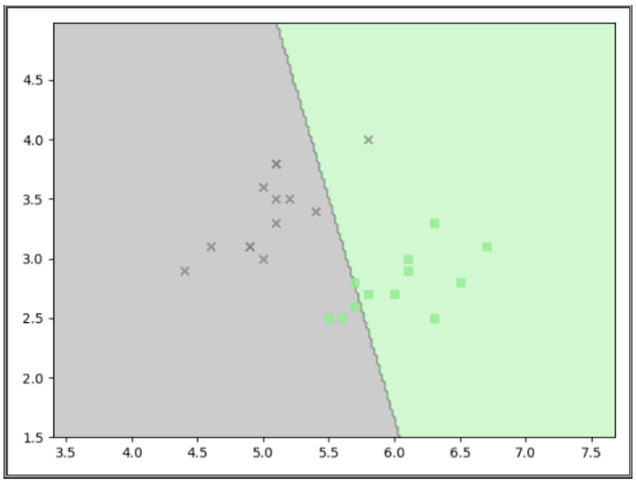
Purpose 2: You need to test your perceptron implementation on "irismodified.csv" Train it for several epochs and check if it can converge or not. If it can not converge, explain why and try to implement a solution with editing your perceptron code.

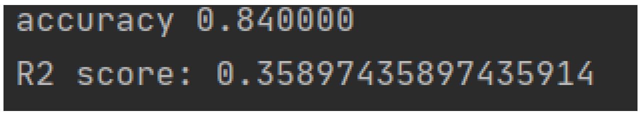
https://stackoverflow.com/questions/62119880/classifier-predict-in-python

This link is great for understanding the classifiers and creating fancy graphs like in below



```python
from matplotlib.colors import ListedColormap
def plot_decision_regions(x, y, classi, resolution=0.02):
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
    x1_min, x1_max = x[:, 0].min() - 1, x[:, 0].max() + 1
    x2_min, x2_max = x[:, 1].min() - 1, x[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),np.arange(x2_min, x2_max, resolution))
    Z = classi.guess(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=x[y == cl, 0], y=x[y == cl, 1],alpha=0.8, c=cmap(idx),marker=markers[idx], label=cl)
    plt.show()
```

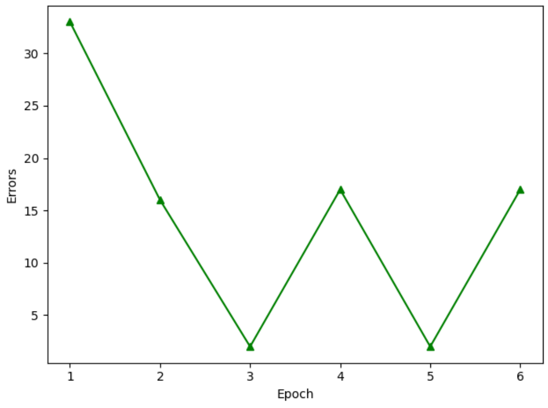


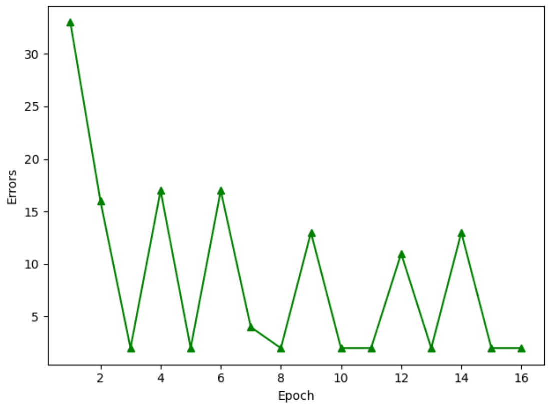


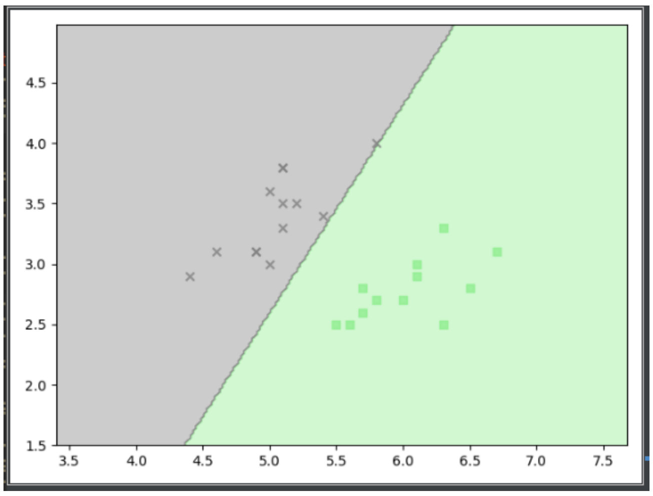
```
accuracy 0.840000
R2 score: 0.358974358974359 14
```

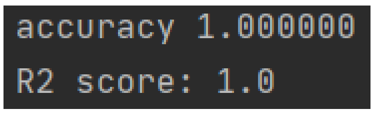
As you can see there are several dots that shouldn’t in the grey part, this shows that our predict not truly valid

Different Epoches









```
accuracy 1.000000
R2 score: 1.0
```

First of all, I tried to use same code for second purpose, Unfortunately I didn’t get the results that can be count as valid value and failed to guess the flower types. Changing the epochs number didn’t help to guess correctly too. At the end I decided to change the activation function. I get the much better results with this way.First r2 result was 0.35 on the other hand , I was able to get 1 as a R2 score and 1 as a accuracy.This shows that, changing the activation function effects

SOURCES

https://towardsdatascience.com/what-is-a-perceptron-basics-of-neural-networks-c4cfea20c590

https://stackoverflow.com/questions/62119880/classifier-predict-in-python

https://pythonmachinelearning.pro/perceptrons-the-first-neural-networks/

https://simonepeirone.it/posts/ml-01-perceptron/