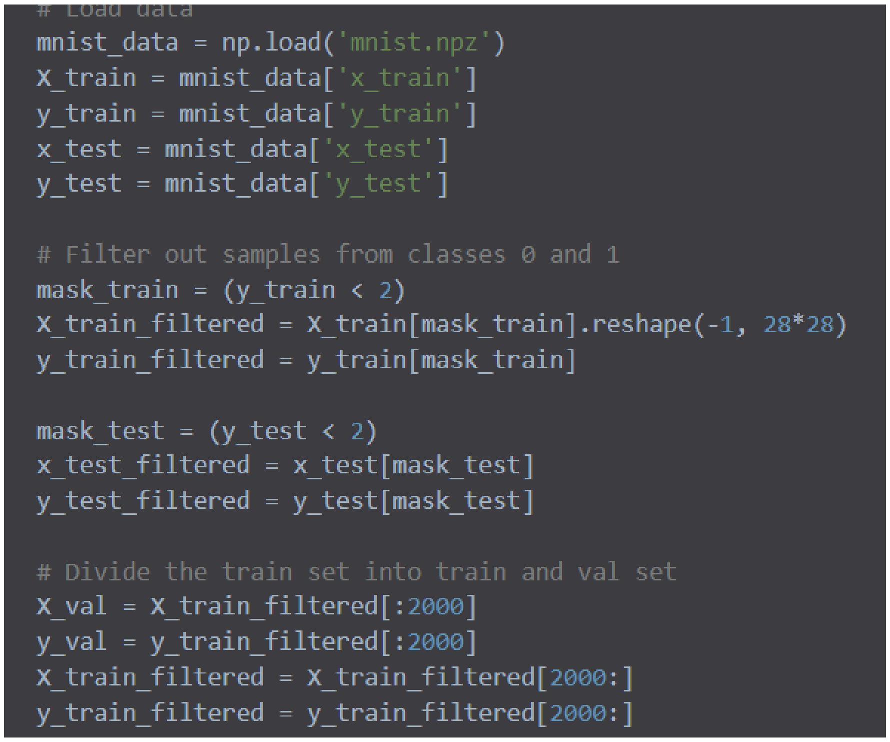
SML
Assignment 3

Vickey kumar
2021299

Question 1:
Use https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz MNIST dataset for this question and perform following tasks.
• It has all in all 60K train samples from 10 classes and 10K test samples.
10 classes are digits from 0-9. Labels or classes for all samples in train and test set is available.
We reduce the class buy only 0 ,1,

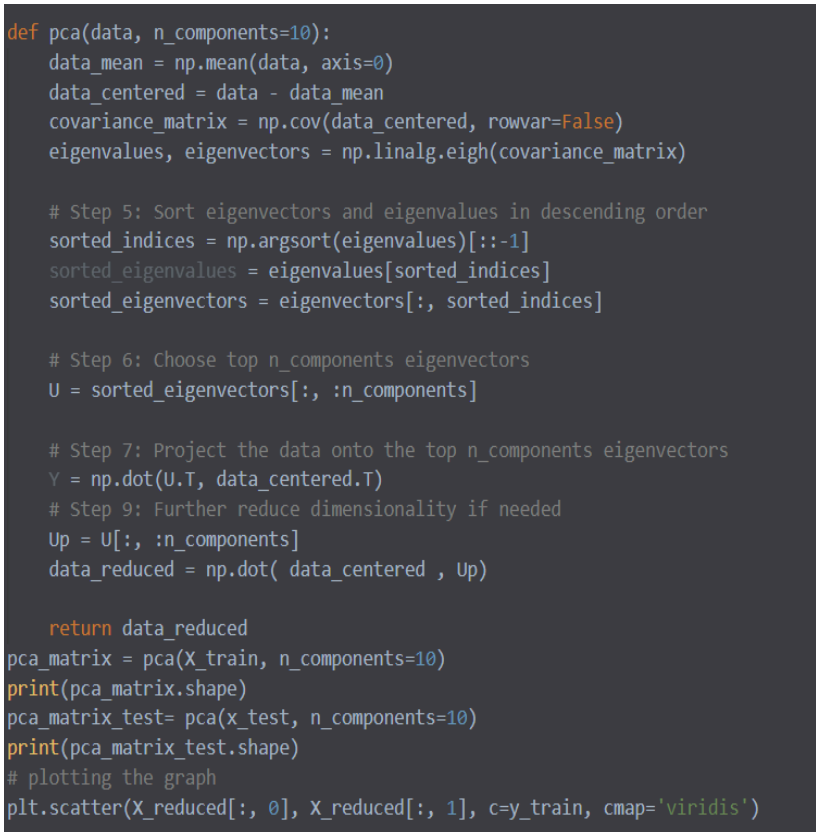
According to the question , i make a class list of “0’ , “1” and so on to “9”
To store the data of y_train which is labels and x_tain data ,



```
# Load data
mnist_data = np.load('mnist.npz')
X_train = mnist_data['x_train']
y_train = mnist_data['y_train']
x_test = mnist_data['x_test']
y_test = mnist_data['y_test']

# Filter out samples from classes 0 and 1
mask_train = (y_train < 2)
X_train_filtered = X_train[mask_train].reshape(-1, 28*28)
y_train_filtered = y_train[mask_train]

mask_test = (y_test < 2)
x_test_filtered = x_test[mask_test]
y_test_filtered = y_test[mask_test]

# Divide the train set into train and val set
X_val = X_train_filtered[:2000]
y_val = y_train_filtered[:2000]
X_train_filtered = X_train_filtered[2000:]
y_train_filtered = y_train_filtered[2000:]
```

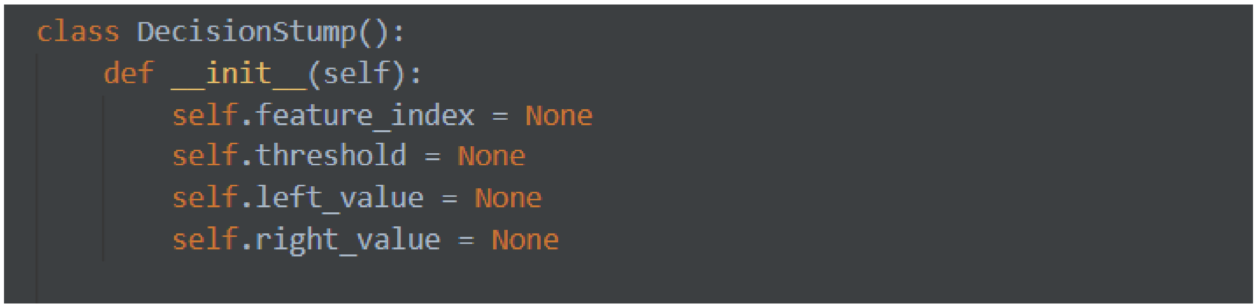
•

Now applied the pca on the 2 class 👍with p =5



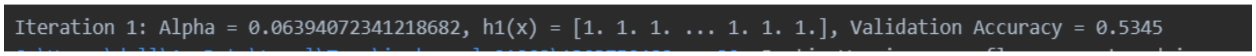
```
def pca(data, n_components=10):
    data_mean = np.mean(data, axis=0)
    data_centered = data - data_mean
    covariance_matrix = np.cov(data_centered, rowvar=False)
    eigenvalues, eigenvectors = np.linalg.eigh(covariance_matrix)

    # Step 5: Sort eigenvectors and eigenvalues in descending order
    sorted_indices = np.argsort(eigenvalues)[::-1]
    sorted_eigenvalues = eigenvalues[sorted_indices]
    sorted_eigenvectors = eigenvectors[:, sorted_indices]

    # Step 6: Choose top n_components eigenvectors
    U = sorted_eigenvectors[:, :n_components]

    # Step 7: Project the data onto the top n_components eigenvectors
    Y = np.dot(U.T, data_centered.T)
    # Step 9: Further reduce dimensionality if needed
    Up = U[:, :n_components]
    data_reduced = np.dot( data_centered , Up)

    return data_reduced
pca_matrix = pca(X_train, n_components=10)
print(pca_matrix.shape)
pca_matrix_test= pca(x_test, n_components=10)
print(pca_matrix_test.shape)
# plotting the graph
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=y_train, cmap='viridis')
```

Now , make the Descissiontreestump class to implement the stump



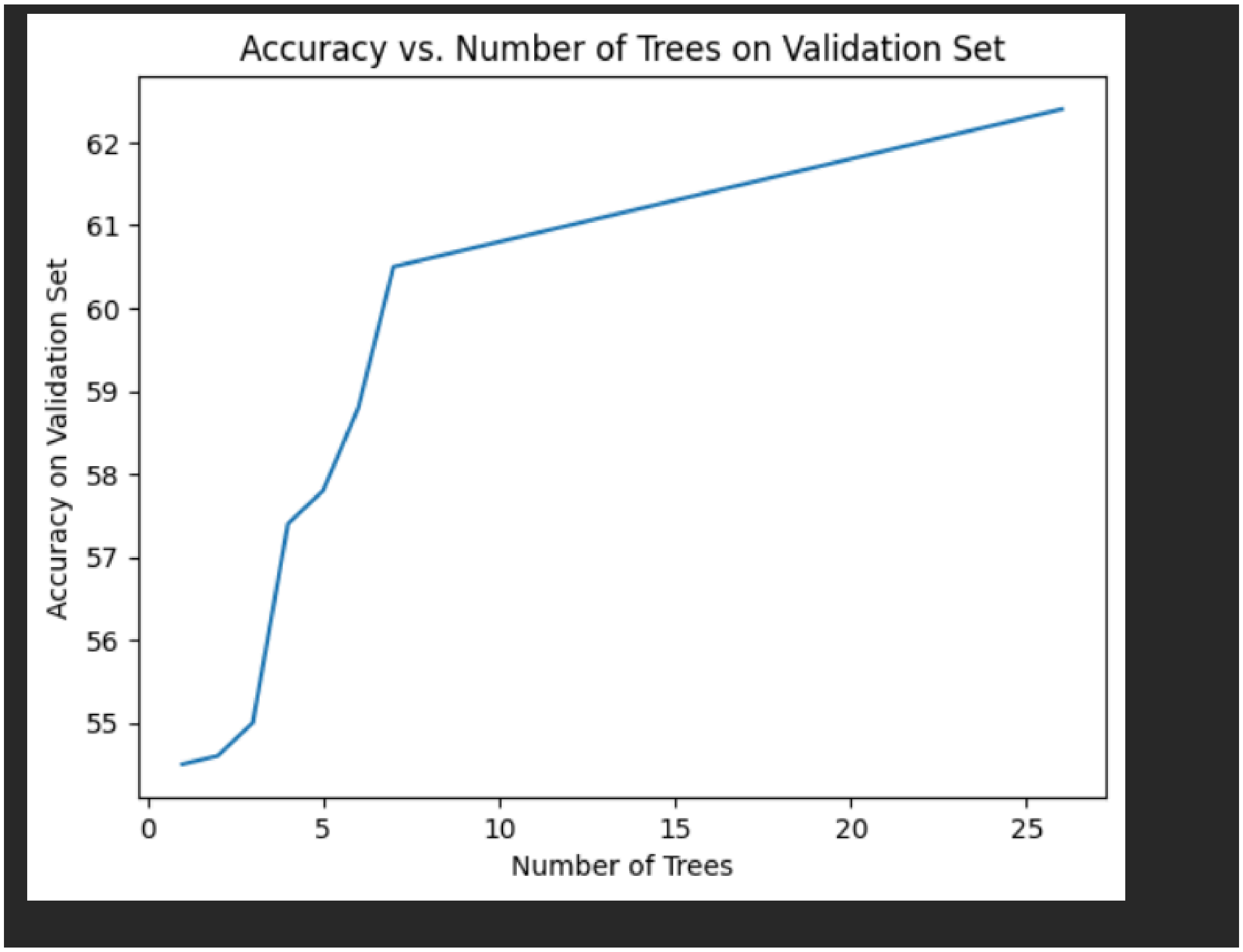
```
class DecisionStump():
    def __init__(self):
        self.feature_index = None
        self.threshold = None
        self.left_value = None
        self.right_value = None
```

After that we calculate the Compute α1 and update weights.



```
Iteration 1: Alpha = 0.06394072341218682, h1(x) = [1. 1. 1. ... 1. 1. 1.], Validation Accuracy = 0.5345
```

After that calculate for Compute α2 and update weights.
Similar for 300 as ask in question:
And finally make a graph for it with accuracy 53.4%


Accuracy vs. Number of Trees on Validation Set
Accuracy on Validation Set
Number of Trees
62
61
60
59
58
57
56
55
0
5
10
15
20
25

Question 2:
Now we implement gradient boosting algorithm:
Divide the train set into train and val set. Keep 1000 samples from each class for val. Note val should be used to evaluate the performance of the classifier. Must not be used in obtaining PCA matrix.

• Apply PCA and reduce the dimension to $p = 5$. You can use the train set of the two classes to obtain PCA matrix. For the remaining parts, use the reduced dimension dataset.

• Now learn a decision tree using the train set. You need to grow a decision stump. For each dimension, find the unique values and sort them

in ascending order. The splits to be evaluated will be midpoint of two consecutive unique values. Find the best split by minimizing SSR. Denote this as h1(x). [1]

• Compute residue using $y - .01h1(x)$.

• Now build another tree h2(x) using the train set but with updated labels. Note, now you have to update labels based on the way we update labels for absolute loss. That is the labels will be obtained as negative gradients. Compute residue using $y - .01h1(x) - .01h2(x)$. [1]

• Similarly grow 300 such stumps. Note, the labels are updated every iteration based on negative gradients.

• After every iteration find the MSE on val set and report. You should show a plot of MSE on val set vs. number of trees. Use the tree that gives lowest MSE and evaluate that tree on test set. Report test MSE. [1]