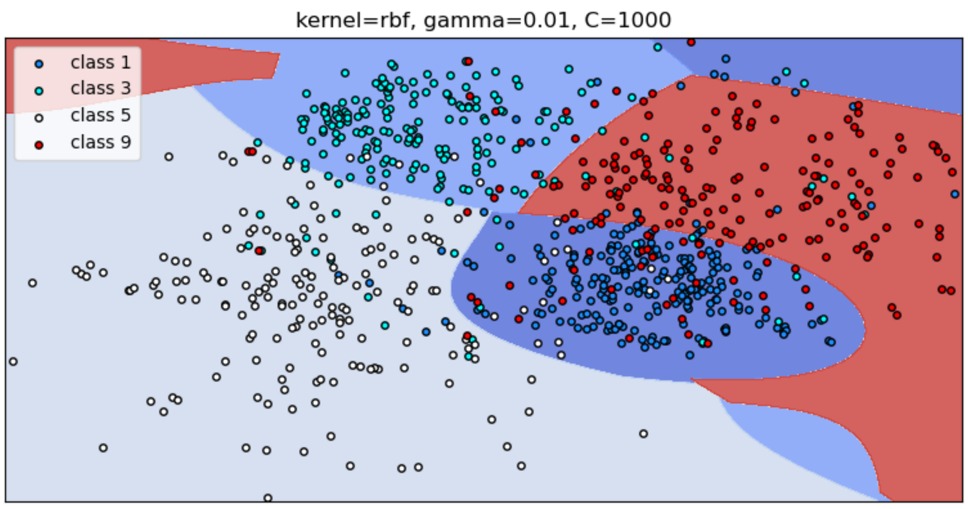
Exercise 1

Part 1

```
Tune the necessary hyperparameters by for instance grid search.
The original dataset is divided into training set(80%) and validation set(20%)
best parameters: {'C': 1000, 'gamma': 0.01, 'kernel': 'rbf'}
```

Part 2 Decision boundary



Exercise 2

Dataset sizes

Training: 10000   Test: 10000

```
train_set = np.loadtxt("mnist_train.csv", delimiter=",", skiprows=50001)  #
600001 rows = 1 label + 60000 samples
X = train_set[:, 1:]  # 10000 x 784
y = train_set[:, 0]  # 10000 x 1
test_set = np.loadtxt("mnist_test.csv", delimiter=",", skiprows=1)  # 10001 rows
= 1 label + 10000 samples
x_test = test_set[:, 1:]  # 10000 x 784
y_test = test_set[:, 0]  # 10000 x 1
```

Part 1

```
I used GridSearchCV
best parameters: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
svm Accuracy:  96.87 %
```

Part 2

```
svm one-vs-one accuracy:  96.87 %
my one-vs-all accuracy: 85.81 %
In terms of misclassification, the svm one-vs-one has higher accuracy than my
one-vs-one method.
Thus, svm one-vs-one performs better.
```

Exercise 4

Part A

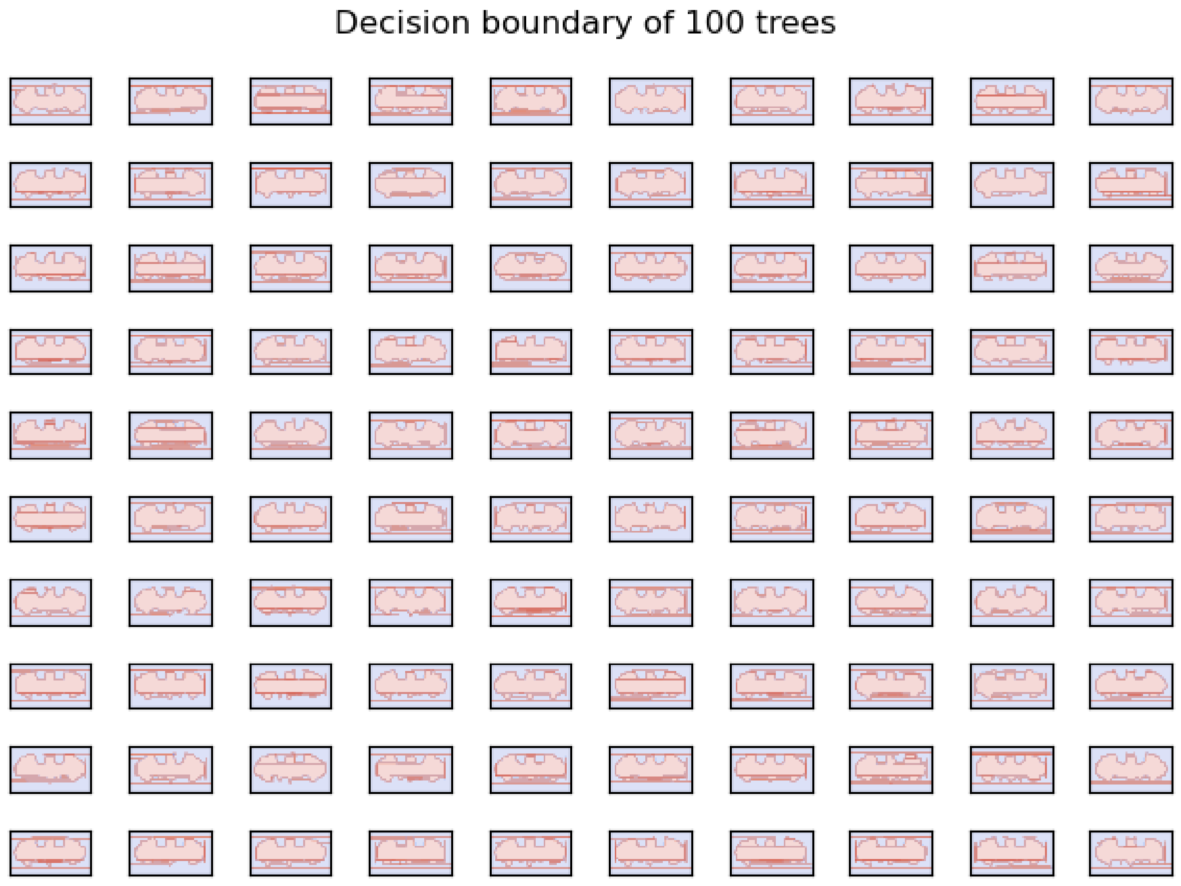
generalization errors: 108

Part B

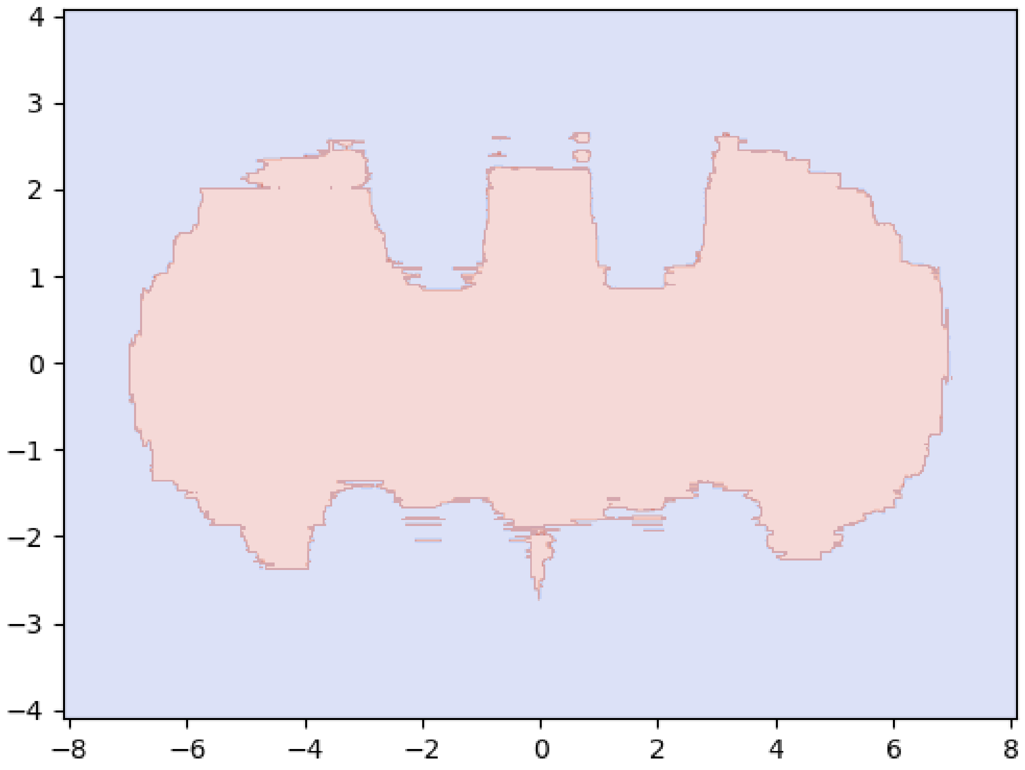
average generalization errors: 181

Part C

Decision boundary for 100 individual tree model



Decision boundary for Ensemble model

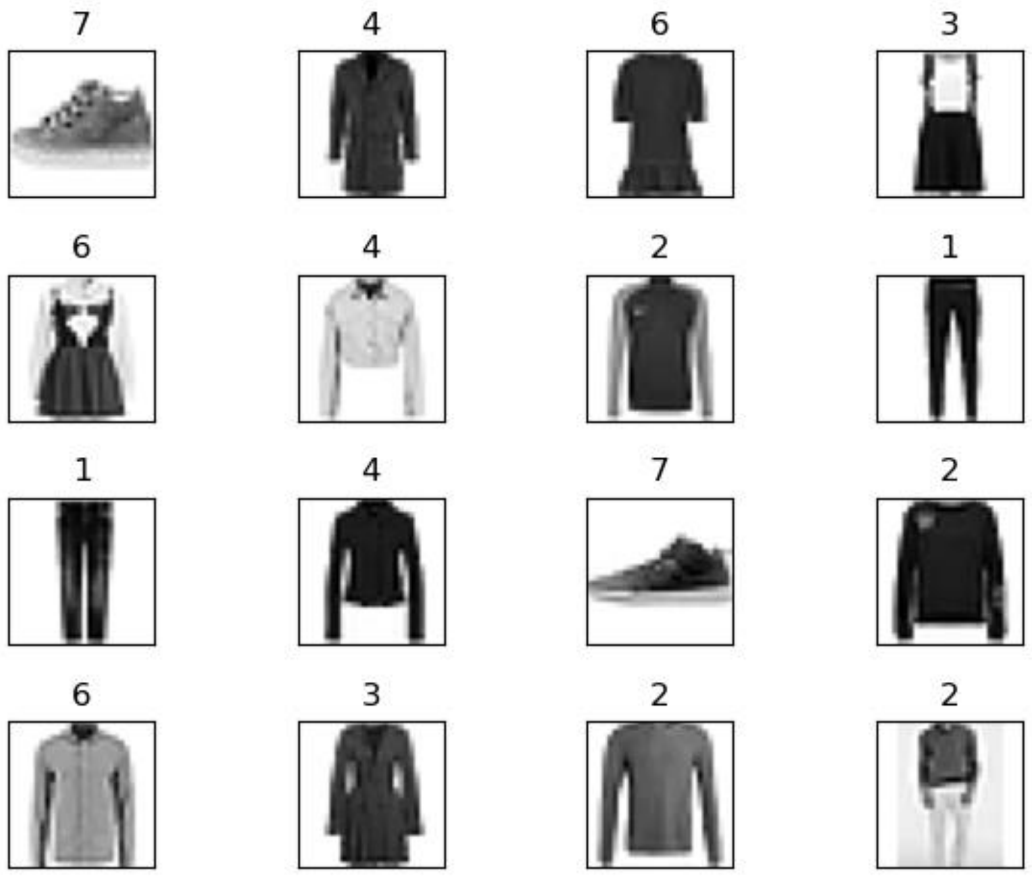


Part D

```
The results have been similar to the predictions, there was no surprises in the
results themselves,
However the biggest surprise was also the downside of the approach, the time
consumed to conduct
such an approach as abnormally long, this will not be feasible on most computers
with larger datasets,
it takes an abnormal amount of time.
```

Exercise 5

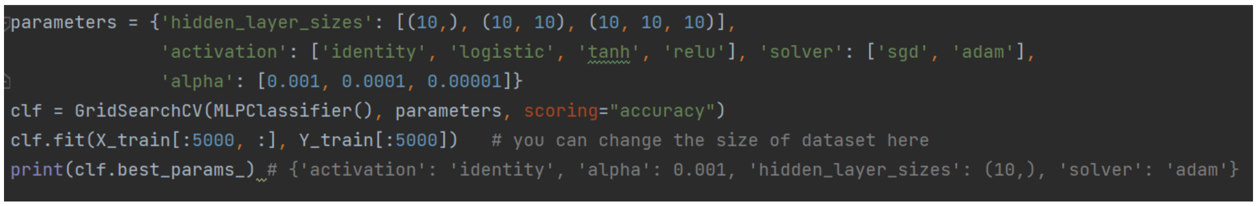
Part 1 Plot 16 random samples from the training set with the corresponding labels.



Part 2

I used GridSearchCV to find the best parameters

Best parameters: {'activation': 'identity', 'alpha': 0.001, 'hidden_layer_sizes': (10,), 'solver': 'adam'}



```
parameters = {'hidden_layer_sizes': [(10,), (10, 10), (10, 10, 10)],
              'activation': ['identity', 'logistic', 'tanh', 'relu'], 'solver': ['sgd', 'adam'],
              'alpha': [0.001, 0.0001, 0.00001]}
clf = GridSearchCV(MLPClassifier(), parameters, scoring="accuracy")
clf.fit(X_train[:5000, :], Y_train[:5000])   # you can change the size of dataset here
print(clf.best_params_) # {'activation': 'identity', 'alpha': 0.001, 'hidden_layer_sizes': (10,), 'solver': 'adam'}
```

GridSearch results with different dataset sizes:

5000: {'activation': 'identity', 'alpha': 0.001, 'hidden_layer_sizes': (10,), 'solver': 'adam'}

3000: {'activation': 'logistic', 'alpha': 0.001, 'hidden_layer_sizes': (10, 10), 'solver': 'sgd'}

2000: {'activation': 'tanh', 'alpha': 0.0001, 'hidden_layer_sizes': (10, 10, 10), 'solver': 'adam'}

1000: {'activation': 'identity', 'alpha': 0.0001, 'hidden_layer_sizes': (10,), 'solver': 'adam'}

I didn’t use larger dataset (greater than 5000) because my computer cannot handle this amount of computation and it would consume tremendous amount of time.

Part 3 Plot the confusion matrix.

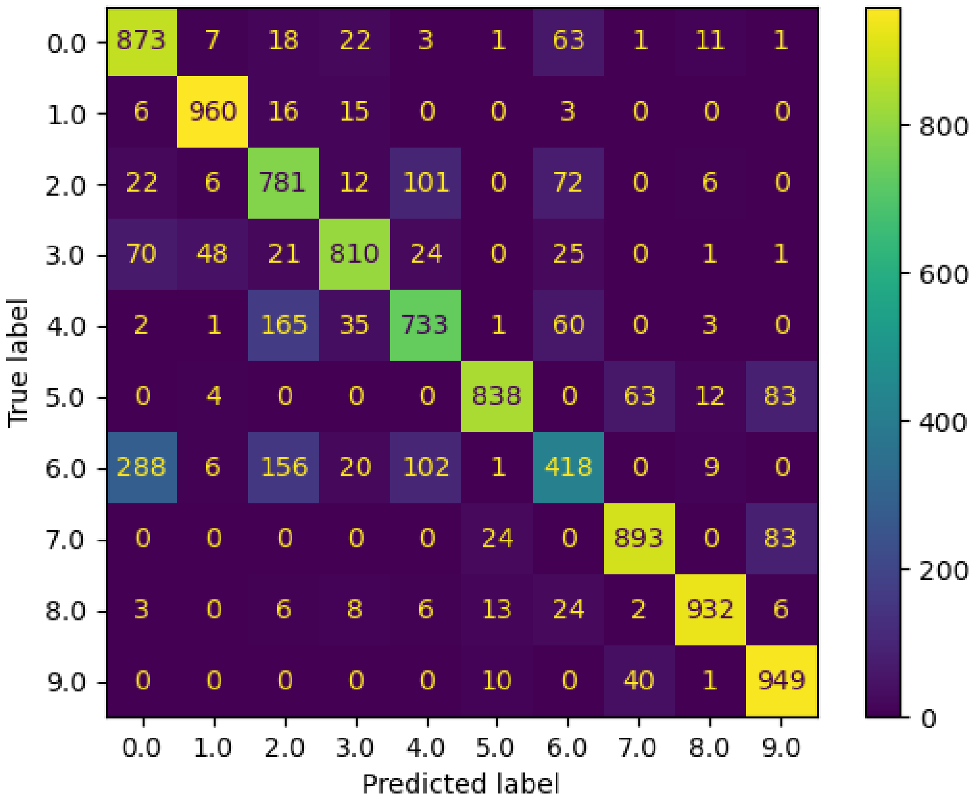
Categories 1 and 9 are the easy ones to classify, with correct number of predictions 944, 972 respectively.

Category 6 is the hard one to classify, with correct number of predictions 418.

Category 2 is often mixed with categories 4, 6.

Category 4 is often mixed with categories 6.

Category 6 is often mixed with categories 0, 2, 3, 4.


True label
Predicted label
0.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0
0.0 873 7 18 22 3 1 63 1 11 1
1.0 6 960 16 15 0 0 3 0 0 0
2.0 22 6 781 12 101 0 72 0 6 0
3.0 70 48 21 810 24 0 25 0 1 1
4.0 2 1 165 35 733 1 60 0 3 0
5.0 0 4 0 0 0 838 0 63 12 83
6.0 288 6 156 20 102 1 418 0 9 0
7.0 0 0 0 0 0 24 0 893 0 83
8.0 3 0 6 8 6 13 24 2 932 6
9.0 0 0 0 0 0 10 0 40 1 949
0
200
400
600
800