11.

The result of the experiment is shown in the following figure:

```
Results Table
------------------------------------------------
Margin for each combination:
      gamma=0.1  gamma=1.0  gamma=10.0
C
0.1    0.041274   0.090780    0.090793
1.0    0.018166   0.009078    0.009079
10.0   0.017794   0.008984    0.008982
```

Figure 1: result table

optimal parameter values

From the figure, we obtain that the largest margin is achieved when the value of γ is 10, and the value of C is 0.1.

perspective from C

As explained in the previous question, the value of C represents a trade-off between the margin and the training error.

For example, when C is large, to minimize the primal problem, we need to prioritize fitting the training data than minimizing the length of the weight vector, so that $||\mathbf{w}||$ may be larger, and the margin is smaller.

We can see this in our resulting table, for each given γ, when C gets larger, the margin is smaller.

perspective from γ

In this problem, we're asked to use the rbf kernel, which is defined as:

$$K(x, x') = \exp(-\gamma||x - x'||^2)$$

The meaning of this kernel is that it measures the similarity between two points x and x' by considering the distance between them, and γ controls the standard for how we consider two points to be similar.

For example, when γ is large, the exponential function decreases rapidly as the distance between x, x' becomes larger. This means that two points need to be very close to each other to be considered similar.

Why this will affect the decision boundary is because for a large γ, $\mathbf{K}(x, x')$ is very small even if x and x' are sligtly distant to each other, so the influence of a single training point will be limited to a small region around it. Therefore, the decision boundary will be influenced by only a few points around it, so the boundary is more intricate.

On the other hand, when γ is small, even for more distant points, they might also affect the decision boundary, so the boundary is smoother.

To conclude, a large γ increases the complexity of the model, and results in a smaller margin, and this conclusion also matches our result.

snapshot of code

The snapshots of my code are shown below: > The beginning of the code is the same as the previous question, so I will only show the part after prepocessing (after relabeling).

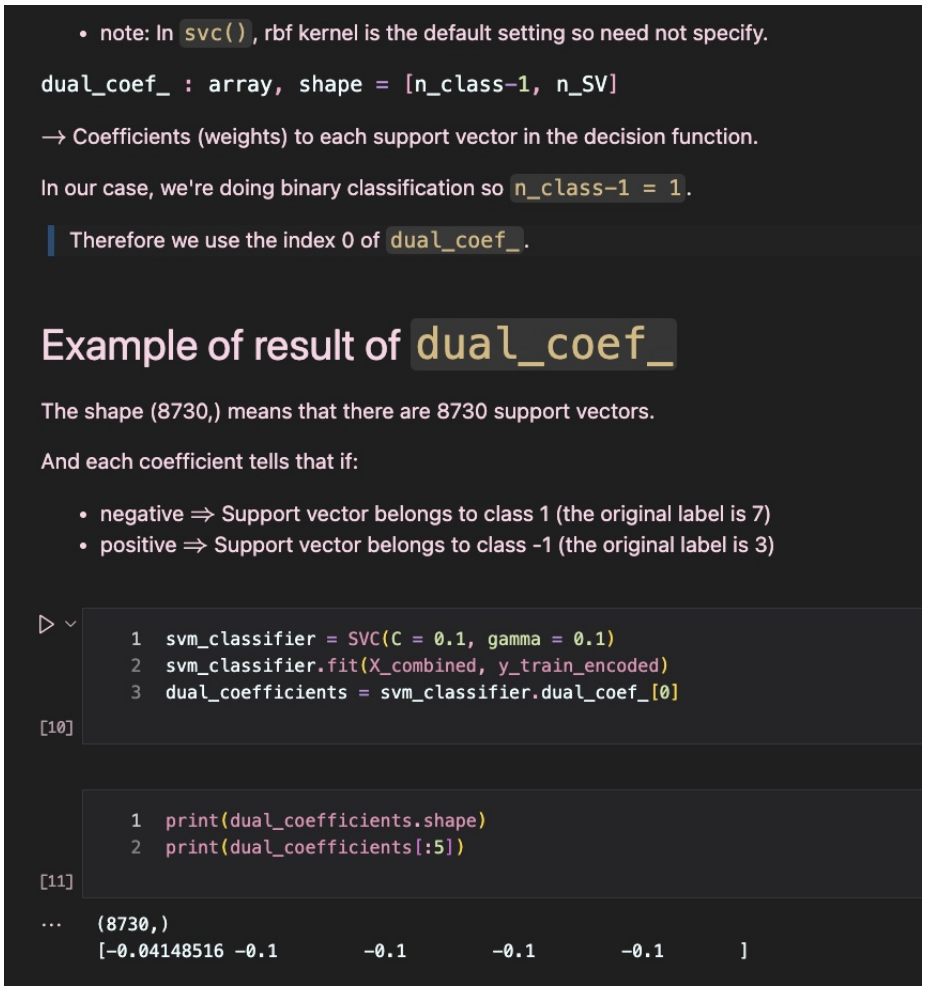

Figure 2: code snapshot 1

The weight vector is defined as:

> see Lecture 4 slide 10

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{z}_n$$

And

$$\begin{aligned} ||\mathbf{w}||^2 &= \mathbf{w}^T \mathbf{w} \\ &= \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m K(\mathbf{x}_n, \mathbf{x}_m) \\ &= \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \exp(-\gamma ||\mathbf{x}_n - \mathbf{x}_m||^2) \end{aligned}$$

```python
# Use this list to store the result of form (C, gamma, margin)
result = []

for C in [0.1, 1, 10]:
    for gamma in [0.1, 1, 10]:
        svm_classifier = SVC(C = C, gamma = gamma)
        svm_classifier.fit(X_combined, y_train_encoded)

        support_vectors = svm_classifier.support_vectors_
        dual_coefficients = svm_classifier.dual_coef_[0]
        support_labels = y_train_encoded[svm_classifier.support_]

        w_norm_squared = 0
        num_support_vectors = len(support_vectors)

        for i in range(num_support_vectors):
            for j in range(num_support_vectors):
                w_norm_squared += (
                    dual_coefficients[i] * dual_coefficients[j]
                    * support_labels[i] * support_labels[j]
                    * np.exp(-gamma * np.linalg.norm(support_vectors[i] - support_vectors[j])**2)
                )

        margin = 1.0 / np.sqrt(w_norm_squared)
        result.append((C, gamma, margin))
```

[17]

Figure 3: code snapshot 2