Problem Set 2

Problem 1

Part 1:

1. Code provided in submission
2. Training Data:

C Value	Accuracy
.001	92.8%
.01	92.8%
.1	92.8%
1	92.8%
10	93.1%
100	93.5%
1000	93.4%

3. Validation Data:

C Value	Accuracy
.001	93.1%
.01	93.1%
.1	93.1%
1	93.1%
10	93.9%
100	93.9%
1000	93.9%

4. Test Data:

C Value	Accuracy
.001	90.8%
.01	90.8%
.1	90.8%
1	90.6%
10	91.6%

C Value	Accuracy
100	92%
1000	92%

Part 2:

1. Code provided in submission
2. Training Data:

	sigma = 0.001	sigma = 0.01	sigma = 0.1	sigma = 1	sigma = 10	sigma = 100
c = 0.001	0.6	0.6	0.6	0.59	0.51	0.63
c = 0.01	0.61	0.61	0.61	0.6	0.68	0.6
c = 0.1	0.62	0.63	0.64	0.65	0.74	0.56
c = 1	0.99	0.99	0.99	0.99	0.83	0.56
c = 10	0.99	0.99	0.99	0.99	0.95	0.7
c = 100	0.99	0.99	0.99	0.99	0.99	0.8
c = 1000	0.99	0.99	0.99	0.99	0.99	0.53

3. Validation Data:

	sigma = 0.001	sigma = 0.01	sigma = 0.1	sigma = 1	sigma = 10	sigma = 100
c = 0.001	0.61	0.61	0.61	0.61	0.61	0.7
c = 0.01	0.61	0.61	0.61	0.61	0.71	0.71
c = 0.1	0.62	0.66	0.62	0.63	0.77	0.72
c = 1	0.65	0.66	0.7	0.72	0.84	0.74
c = 10	0.65	0.66	0.7	0.72	0.88	0.83
c = 100	0.65	0.66	0.7	0.72	0.88	0.91
c = 1000	0.65	0.66	0.7	0.7	0.88	0.64

4. Test Data:

	sigma = 0.001	sigma = 0.01	sigma = 0.1	sigma = 1	sigma = 10	sigma = 100
c = 0.001	0.6	0.6	0.6	0.6	0.6	0.7
c = 0.01	0.6	0.6	0.6	0.6	0.66	0.7
c = 0.1	0.6	0.6	0.61	0.62	0.73	0.7
c = 1	0.65	0.67	0.68	0.72	0.8	0.72
c = 10	0.65	0.67	0.68	0.72	0.85	0.81
c = 100	0.65	0.67	0.68	0.72	0.85	0.88
c = 1000	0.65	0.67	0.68	0.72	0.68	0.62

Part 3:

1. For this task, we should employ an SVM with a Gaussian kernel. Utilizing a validation set to prevent overfitting during training is particularly effective with the Gaussian kernel. This is because the Gaussian kernel can model more intricate and nonlinear relationships within the data, allowing us to achieve robust results.

Problem 2

1. We begin with the primal problem such that:

$$\text{Minimize:} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\exp(\xi_i)$$

$$\text{subject to:} \quad y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

   We can write the Lagrangian for this problem as:

$$L(w, b, \xi, \alpha) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\exp(\xi_i) + \sum_{i=1}^{N}\alpha_i[1 - \xi_i - y_i(w \cdot x_i + b)] - \sum_{i=1}^{N}\beta_i\xi_i$$

   Where α_i and β_i are the Lagrange multipliers for the inequality and non-negativity constraints.

   We need to calculate the partial derivatives for w, b and epsilon:

   $\frac{\partial \mathcal{L}}{\partial w} = w - \sum_{i=1}^{N}\alpha_i y_i x_i$

   $\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{N}\alpha_i y_i$

   $\frac{\partial \mathcal{L}}{\partial \xi_i} = C\exp(\xi_i) - \alpha_i - \beta_i$

   Substituting these back in, we get the dual to be:

$$\text{Maximize:} \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i, x_j) \text{ subject to: } \sum_{i=1}^{N}\alpha_i y_i = 0$$
$$\alpha_i \geq 0$$

2. Given a solution to the dual problem, one can construct a solution to the primal problem using the concept of complementary slackness. One must also ensure strong duality between the primal and the dual problem.

   Below is the approach one can take:

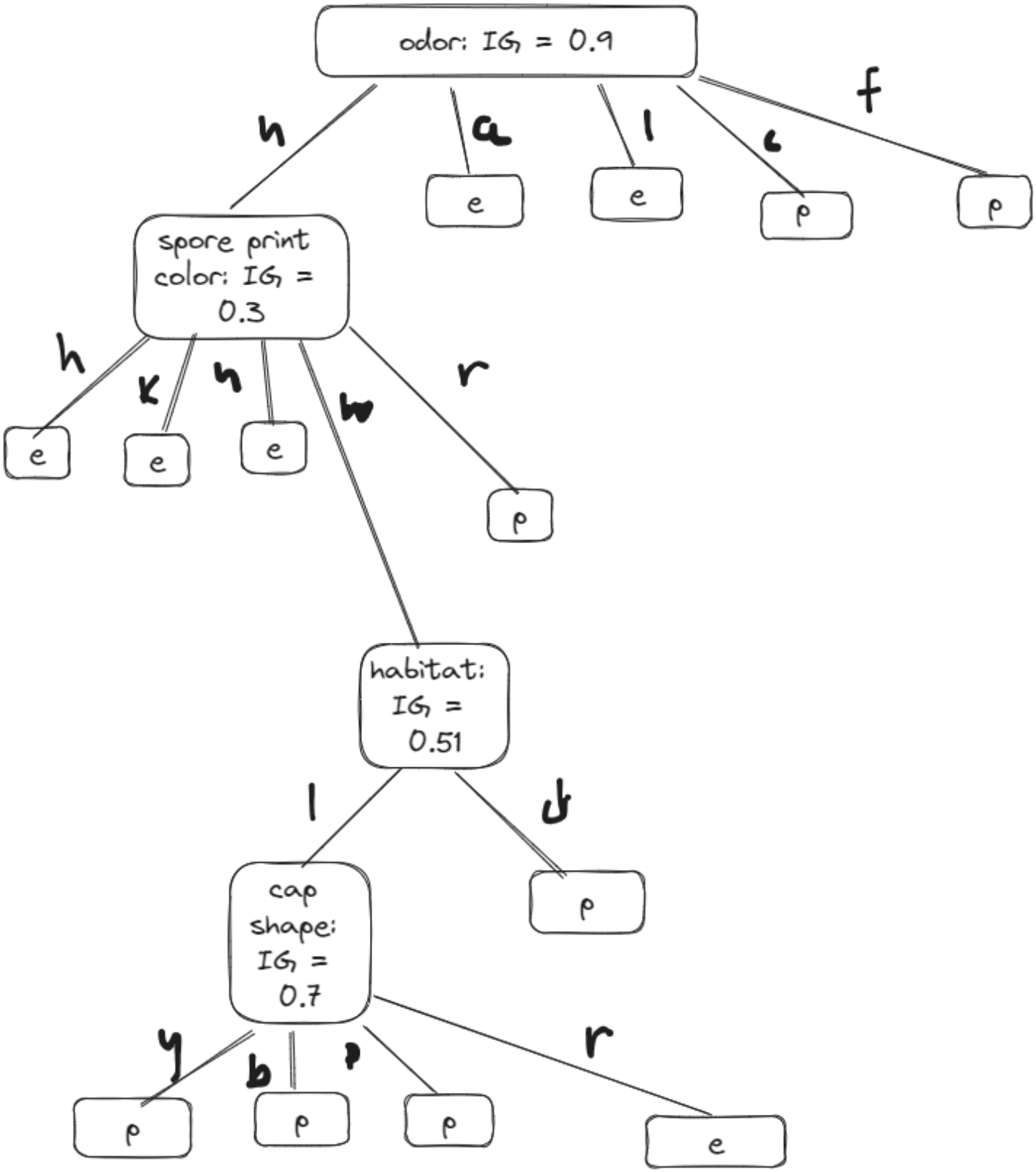
```
1. Find the optimal solution for the dual problem

2. Apply Complementary Slackness: Complementary slackness is a set of conditions that
provides a relationship between the primal and dual problems. It states that for each pair of
primal and dual variables, at least one must be zero. In other words, if a primal variable is
greater than zero, the corresponding dual variable must be zero, and vice versa.
```

```
3. Construct the Primal Solution: Utilize the dual variable values and complementary
slackness conditions to derive the optimal primal solution. This involves setting appropriate
primal variables to zero based on the values of the dual variables, ensuring the
complementary slackness conditions. Solve for the remaining primal variables to obtain the
complete primal solution.
```

3. Yes, the kernel trick can still be used in an SVM, even when there is an exponential penalty for when constraints are violated. The kernel trick is a method to map the input data into a higher-dimensional feature space without computing the transformed features. This allows linear algorithms like SVM to operate in this higher-dimensional space, effectively enabling the SVM to find non-linear decision boundaries in the original input space. The kernel trick does not depend on the penalty used to penalize violating constraints, so it does not matter that the penalty is exponential.
4. One can implement (sub)gradient descent by using a version of the hinge loss function. The modified primal objective function $J(w)$ is given by: $J(w) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\exp(\xi_i)$ where C is a hyperparameter controlling the trade-off between the margin and the amount of violation allowed. $[\nabla J(w) = w + C\sum_{i=1}^{n}\exp(\xi_i)\cdot\nabla\xi_i]$ where $\nabla\xi_i$ is the subgradient of the hinge loss. Using this new subgradient, you then update w and b with the learning rate and the subgradient to find the optimal solutions for both of those parameters. With subgradient descent, one would use a selection of points instead of the entire dataset to improve efficiency and convergence speed of the program.

Problem 3



1.
2. The tree is 98% accurate
3. The initial splits are crucial in a decision tree as they affect all subsequent splits. These splits are determined based on the training data. Different training/test splits result in different training data, which can lead to different initial splits in the decision tree. For example, if your training data has more instances of a particular class, the decision tree might prioritize splitting on features that help distinguish that class. If the training/test split changes, the composition of the training data changes, potentially leading to different initial splits.