University of Tehran

School of Electrical and Computer Engineering

Machine Learning

Homework 4: Support Vector Machines and Ensemble Learning

Author:
Alireza Javid

Student Number:
810198375

Contents

1 Problem 1: Basics of SVM

(a) A linear discriminant function can be written as $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ where $\mathbf{w}$ is the weight vector and w_0 is the bias. When $g(\mathbf{x})$ is linear, the decision surface is a hyperplane whose orientation is determined by the normal vector $\mathbf{w}$ (normal to the hyperplane) and location is determined by the bias w_0.

(b) SVM finds an optimal hyperplane that maximizes the margin between the classes while minimizing the classification errors. When training data contains noise, SVM can still separate the classes by applying several techniques:

- Regularization: SVM applies a regularization parameter (C) that controls the trade-off between maximizing the margin and allowing misclassifications. By adjusting the value of C, SVM can handle noisy data more effectively. A larger C value allows for fewer misclassifications, which may lead to overfitting noisy data, while a smaller C value allows for more misclassifications and may result in a wider margin.
- Kernel Trick: SVM can utilize different kernel functions to transform the input data into a higher-dimensional space where it becomes easier to separate noisy classes. Non-linear kernels such as the polynomial or radial basis function (RBF) kernel can help capture complex relationships and handle noisy data more effectively.
- Outlier Removal: SVM can be combined with outlier detection techniques to identify and remove noisy data points that significantly deviate from the general pattern. By removing outliers, the SVM model can focus on learning the underlying structure of the data without being influenced by extreme noise.
- Cross-Validation: By using cross-validation techniques such as k-fold cross-validation, SVM can estimate the performance of the model on unseen data. This helps in evaluating the model's ability to generalize and handle noise. Cross-validation can also aid in tuning the hyperparameters of SVM to find the best configuration for handling noisy data.

These techniques help SVM find an optimal hyperplane that separates the classes while minimizing the impact of noise on the classification decision.

(c) Kernel functions are used to indirectly evaluate a mapping function $\phi(\cdot)$ that transforms input data $x \in \mathbb{R}^d$ to a higher-dimensional space Q. By utilizing kernel functions, we can perform calculations in the original input space without explicitly computing the transformed feature vectors in Q. This approach allows us to efficiently handle complex, nonlinear relationships within the data.

The width of a margin is $\frac{2}{||\mathbf{w}||}$ and we have

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(x_i) \phi(x_j)^T$$

So we can say $\phi_2 = [2x, 2x^2]$ has greater norm 2 and has ***greater*** margin.

2 Problem 2: Kernel validations

$K(x,y)$ is a valid (Mercer) kernel, if the corresponding kernel matrix is symmetric positive semi-definite.

(a)

$$K(x,y) = \exp\left(-\frac{||x-y||^2}{\sigma^2}\right) = \exp\left(-\frac{(x-y)^T(x-y)}{\sigma^2}\right)$$

$$= \exp\left(-\frac{||x||^2+||y||^2-2x^Ty}{\sigma^2}\right) = \exp\left(-\frac{||x||^2}{\sigma^2}\right)\exp\left(-\frac{||y||^2}{\sigma^2}\right)\exp\left(2\frac{x^Ty}{\sigma^2}\right)$$

$$= \left[\exp\left(-\frac{||x||^2}{\sigma^2}\right)\exp\left(\sqrt{2}\frac{x^T}{\sigma^2}\right)\right]\left[\exp\left(-\frac{||y||^2}{\sigma^2}\right)\exp\left(\sqrt{2}\frac{y}{\sigma^2}\right)\right]$$

$$= \phi(\mathbf{x})^T\phi(\mathbf{y}) = \langle \mathbf{x}, \mathbf{y}\rangle$$

Where $\phi(\mathbf{x}) = \exp\left(-\frac{||x||^2}{\sigma^2}\right)\exp\left(\sqrt{2}\frac{x}{\sigma^2}\right)$

(b) Suppose $k_1(x,y) = \langle \phi^{(1)}(x), \phi^{(1)}(y)\rangle$ and $k_2(x,y) = \langle \phi^{(2)}(x), \phi^{(2)}(y\rangle$

- if we have $a > 0$ and $b > 0$ we can define $\phi(x) = [\sqrt{a}\phi^{(1)}(x)\ \sqrt{b}\phi^{(2)}(x)]$. Now we have:

$$K(x,y) = \langle \phi(x), \phi(y)\rangle = ak_1(x,y) + bk_2(x,y)$$

- As we see, to write kernel function as a product of two functions we must have $a > 0$, $b > 0$, and for other a and b, $K(x,y)$ is not a kernel function.

(c) We can write any valid kernel function as $K(x,y) = f(x)\phi(x)^T\phi(y)f(y)$. If we assign $\phi(x) = \phi(y) = I$ we have:

$$K(x,y) = f(x)f(y)$$

Which is a valid kernel.

(d) We know $k_1(x,y)$ is a valid kernel function so we can write $k_1(x,y) = \langle \phi^{(1)}(x), \phi^{(1)}(y)\rangle$. We can demonstrate $K(x,y)$ is also a valid kernel as follow:

$$K(x,y) = k_1(g(x), g(y)) = \langle \phi(g(x)), \phi(g(y))\rangle = \langle \phi^{'}(x), \phi^{'}(y)\rangle$$

(e)

$$K(x,y) = k_1(x,y)k_2(x,y) = \left\langle \phi^{(1)}(x), \phi^{(1)}(y)\right\rangle \cdot \left\langle \phi^{(2)}(x), \phi^{(2)}(y)\right\rangle$$

$$= \left(\sum_i \phi_i^{(1)}(x)\phi_i^{(1)}(y)\right) \cdot \left(\sum_j \phi_j^{(2)}(x)\phi_j^{(1)}(y)\right) =$$

$$\sum_{i,j=1}^{d} \left(\phi_i^{(1)}(x)\phi_j^{(2)}(x)\right) \cdot \left(\phi_i^{(1)}(y)\phi_j^{(2)}(y)\right) = \left\langle \phi'(x), \phi'(y)\right\rangle$$

3 Problem 3: Soft-SVM Class-Separability

We define the optimization problem for soft margin SVM as below:

$$\min \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{2}\sum_{i=1}^{N}\varepsilon_i^2$$

$$s.t.\ \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \varepsilon_i$$

As we can see, we have a quadratic term of ε_i. We define:

$$\hat{\mathbf{w}}^T = [\mathbf{w}_i^T\ ,\ \sqrt{C}\varepsilon^T] \quad \hat{x_i}^T = [\mathbf{x}_i^T\ ,\ \frac{1}{\sqrt{C}}{e_i}^T]$$

where e_i represents an N-dimensional vector where all elements are 0 except for the ith dimension, which is equal to 1. Now we can rewrite the equation as below:

$$\min \frac{1}{2}||\hat{\mathbf{w}}||^2$$

$$s.t.\ \hat{\mathbf{w}}^T\hat{x_i} + b \geq 1$$

4 Problem 4: SVM on Iris Dataset

(a) The Iris dataset is a classic machine learning dataset that contains measurements of four features (sepal length, sepal width, petal length, and petal width) of three different species of Iris flowers (Setosa, Versicolor, and Virginica). It is commonly used for classification tasks in machine learning.

When using the linear kernel in Support Vector Machines (SVM) for classification, it means that the SVM algorithm employs a linear decision boundary to separate the different classes of Iris flowers based on the input feature measurements. The linear kernel assumes that the classes can be effectively separated by a straight line in the feature space. The linear kernel is well-suited for datasets with linearly separable classes, making it a suitable choice for the Iris dataset due to the relatively distinct boundaries between the species.

Our result for this dataset is:

```
Correlation Matrix:
[[ 1.         -0.16186599]
 [-0.16186599  1.        ]]

F1-Score: 0.89923273657289
Precision: 0.9013888888888889
Recall: 0.9
Accuracy : 0.9
```

Figure 1: Model measures

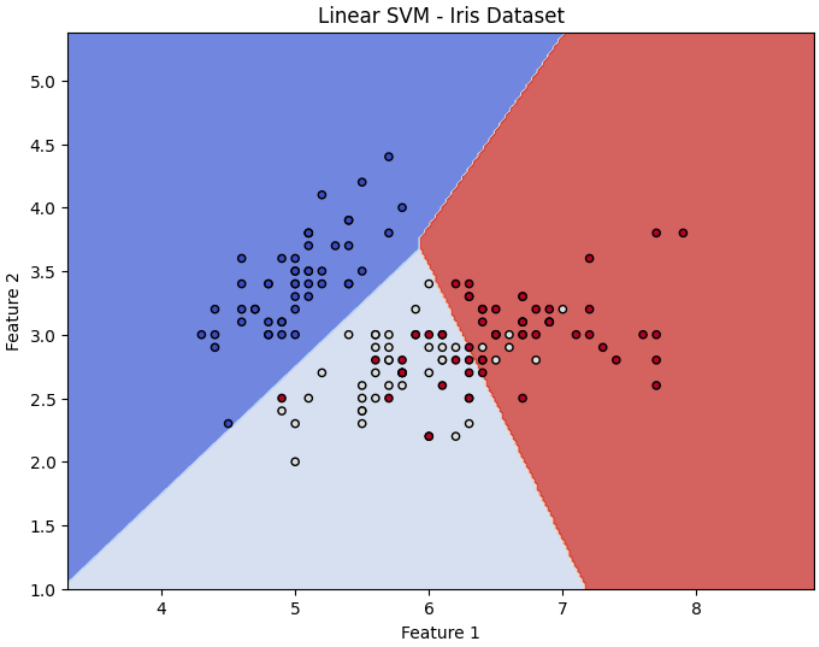


Figure 2: SVM decision boundaries with linear kernel

(b) As we see in the following part, the linear kernel in SVM effectively separates Iris flower species based on "Petal Width" and "Petal Length," and improves classification accuracy. It finds an optimal line that discriminates each class and capitalizes on distinct boundaries present in the Iris dataset.

(c)
- RBF (Radial Basis Function) Kernel: The RBF kernel is a popular choice for SVMs due to its ability to handle non-linear decision boundaries. It uses a Gaussian-like function to measure the similarity between data points. RBF kernels are well-suited for datasets with complex and non-linear relationships between features, making them suitable for a wide range of datasets.
- Linear Kernel: The linear kernel is a simple and efficient choice for SVMs. It uses a linear decision boundary to separate classes, assuming that the data is linearly separable. Linear kernels work well when the data can be effectively separated by a straight line, making them suitable for datasets with linear relationships between features.
- Polynomial Kernel: The polynomial kernel allows SVMs to capture non-linear relationships between features by introducing polynomial terms. It transforms the data into a higher-dimensional space, enabling the creation of non-linear decision boundaries. Polynomial kernels are suitable for datasets with curved or non-linear boundaries, but they require careful tuning of the degree parameter to balance underfitting and overfitting.

Figure 3 shows the good performance of SVM in this problem.

----------Train Data----------
RBF Kernel:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	40
1	0.93	0.93	0.93	41
2	0.92	0.92	0.92	39
accuracy			0.95	120
macro avg	0.95	0.95	0.95	120
weighted avg	0.95	0.95	0.95	120

Linear Kernel:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	40
1	0.93	0.93	0.93	41
2	0.92	0.92	0.92	39
accuracy			0.95	120
macro avg	0.95	0.95	0.95	120
weighted avg	0.95	0.95	0.95	120

Polynomial Kernel:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	40
1	0.87	1.00	0.93	41
2	1.00	0.85	0.92	39
accuracy			0.95	120
macro avg	0.96	0.95	0.95	120
weighted avg	0.96	0.95	0.95	120

(a) Classification report for train data

----------Test Data----------
RBF Kernel:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

Linear Kernel:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

Polynomial Kernel:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

(b) Classification report for Test data

Figure 3: Classification report for Iris dataset

Figure 4 illustrates the RBF kernel has more smooth decision boundary compared to the linear kernel.

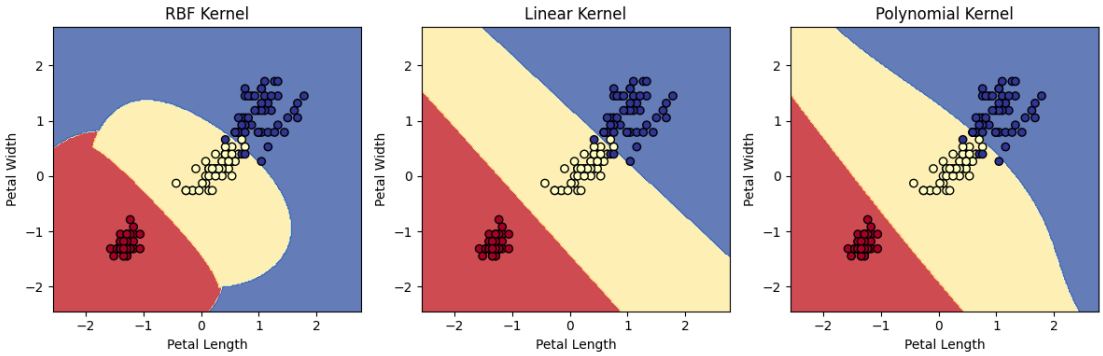


Figure 4: SVM decision boundaries with different kernels

(d)

- Regularization: Regularization is a technique used in Support Vector Machines (SVM) to prevent overfitting and improve the model's generalization ability. It adds a penalty term to the SVM objective function, which balances the trade-off between fitting the training data well and keeping the model's complexity in check. The regularization hyperparameter, commonly denoted as C, controls the strength of regularization. A higher value of C leads to less regularization, allowing the model to fit the training data more closely, potentially increasing the risk of overfitting.
- Gamma: In SVM, the gamma hyperparameter determines the influence of a single training example on the decision boundary. It defines the width of the Gaussian Radial Basis Function (RBF) kernel. A small gamma value leads to a wider kernel, considering more distant data points in the decision boundary calculation. Conversely, a high gamma value creates a narrower kernel, giving more weight to nearby points. High gamma values can result in more complex and intricate decision boundaries, potentially leading to overfitting if not properly tuned.

We used four different sets of γ and C to show the change in decision boundaries. Our accuracy also decreases in these four cases.

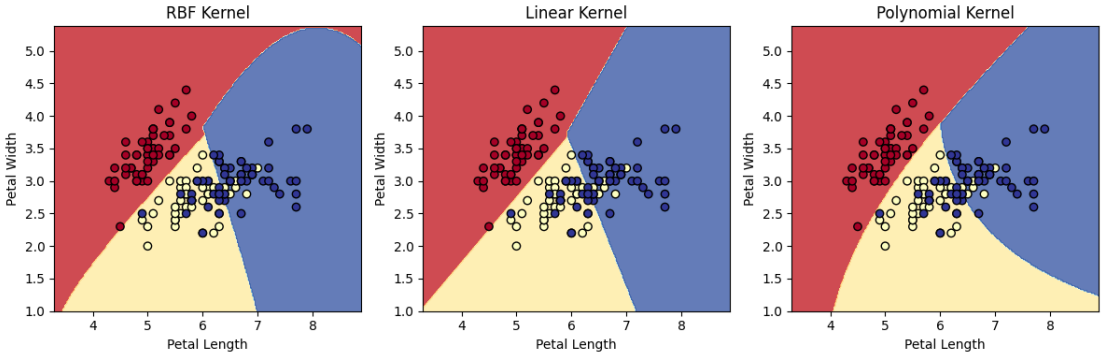


Figure 5: SVM decision boundaries with $C = 1, \gamma = 100$

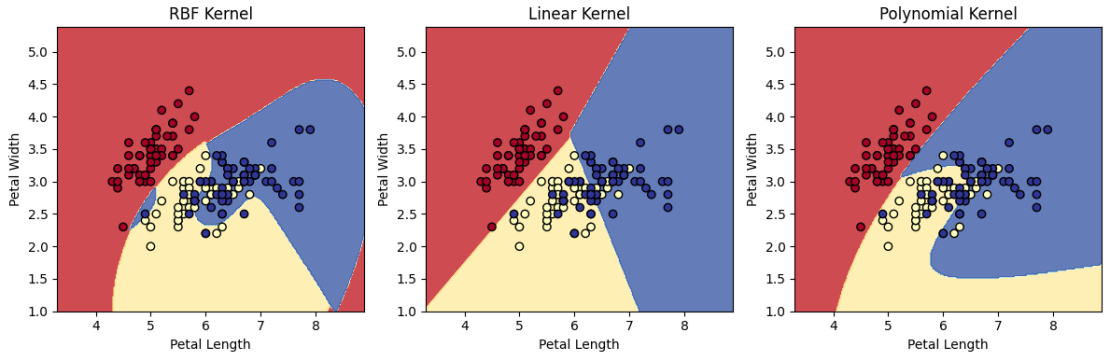


Figure 6: SVM decision boundaries with $C = 10000, \gamma = 0.1$

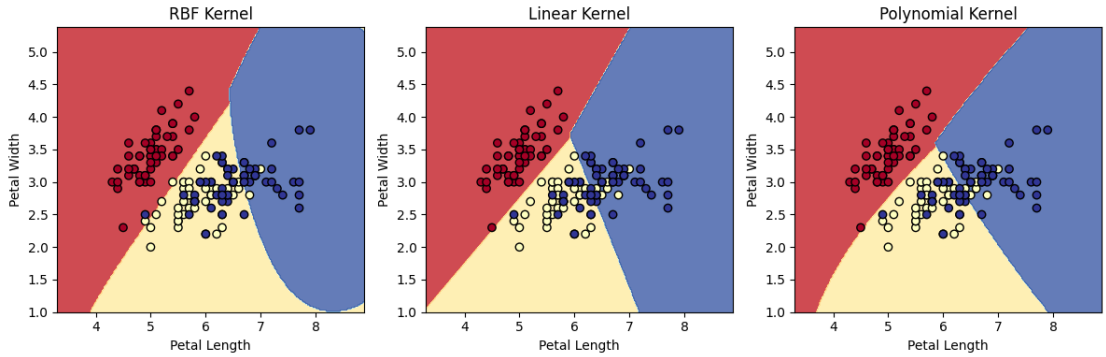


Figure 7: SVM decision boundaries with $C = 0.1, \gamma = 0.01$

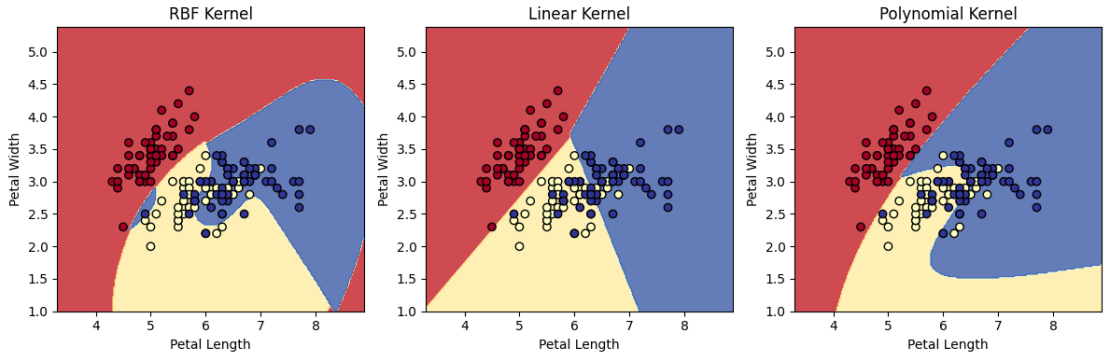


Figure 8: SVM decision boundaries with $C = 10000, \gamma = 100$

As we said before a high value of C leads to memorizing the train data and overfitting and high gamma values can result in more complex boundaries and again overfitting.

(e) Using the GridSearchCV, the best value for hyperparameters is $C = 100, \gamma = 1$ and it improves our results slightly. The results are:

```
SVC(C=100, gamma=1)
----------Train Data----------
RBF Kernel:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       0.95      0.98      0.96        41
           2       0.97      0.95      0.96        39

    accuracy                           0.97       120
   macro avg       0.98      0.97      0.97       120
weighted avg       0.98      0.97      0.97       120

----------Test Data----------
RBF Kernel:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

(a) Classification report for RBF kernel

```
SVC(C=100, gamma=1, kernel='linear')
----------Train Data----------
Linear Kernel:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       0.95      0.93      0.94        41
           2       0.93      0.95      0.94        39

    accuracy                           0.96       120
   macro avg       0.96      0.96      0.96       120
weighted avg       0.96      0.96      0.96       120

----------Test Data----------
Linear Kernel:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

(b) Classification report for linear kernel

```
SVC(C=100, gamma=1, kernel='poly')
----------Train Data----------
Polynomial Kernel:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       0.93      0.98      0.95        41
           2       0.97      0.92      0.95        39

    accuracy                           0.97       120
   macro avg       0.97      0.97      0.97       120
weighted avg       0.97      0.97      0.97       120

----------Test Data----------
Polynomial Kernel:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

(c) Classification report for polynomial kernel

Figure 9: Classification report for different kernels

(f) In the OvO approach, a binary SVM classifier is trained for each pair of classes. For a problem with N classes, $\frac{N(N-1)}{2}$ binary classifiers are trained. OvO can be computationally expensive for large datasets since it requires training multiple classifiers, but it can handle imbalanced class distributions effectively.

In the OvR approach, a binary SVM classifier is trained for each class, treating it as the positive class and the remaining classes as the negative class. OvR is computationally efficient since it requires training only N binary classifiers for N classes, making it more suitable for large datasets. However, OvR can struggle with imbalanced datasets or when classes overlap significantly.

In the results below, we can see that in a linear kernel, the OvO approach gives a better model. Using RBF and polynomial kernels shows similar results for OvO and OvR because our data is already linearly separable.

Kernel = linear_o_v_r

----------Train Data----------

	precision	recall	f1-score	support
0	1.00	1.00	1.00	37
1	0.94	0.91	0.92	33
2	0.92	0.94	0.93	35
accuracy			0.95	105
macro avg	0.95	0.95	0.95	105
weighted avg	0.95	0.95	0.95	105

----------Test Data----------

	precision	recall	f1-score	support
0	1.00	0.92	0.96	13
1	0.88	0.88	0.88	17
2	0.88	0.93	0.90	15
accuracy			0.91	45
macro avg	0.92	0.91	0.92	45
weighted avg	0.91	0.91	0.91	45

(a) Classification report for linear kernel OvR

Kernel = linear_o_v_o

----------Train Data----------

	precision	recall	f1-score	support
0	1.00	1.00	1.00	37
1	0.97	0.94	0.95	33
2	0.94	0.97	0.96	35
accuracy			0.97	105
macro avg	0.97	0.97	0.97	105
weighted avg	0.97	0.97	0.97	105

----------Test Data----------

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	1.00	0.94	0.97	17
2	0.94	1.00	0.97	15
accuracy			0.98	45
macro avg	0.98	0.98	0.98	45
weighted avg	0.98	0.98	0.98	45

(b) Classification report for linear kernel OvO

Figure 10: Classification report for the linear kernel with OvR and OvO approaches

Kernel = rbf_o_v_r

----------Train Data----------

	precision	recall	f1-score	support
0	1.00	1.00	1.00	37
1	1.00	1.00	1.00	33
2	1.00	1.00	1.00	35
accuracy			1.00	105
macro avg	1.00	1.00	1.00	105
weighted avg	1.00	1.00	1.00	105

----------Test Data----------

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	0.89	0.94	0.91	17
2	0.93	0.87	0.90	15
accuracy			0.93	45
macro avg	0.94	0.94	0.94	45
weighted avg	0.93	0.93	0.93	45

(a) Classification report for RBF kernel OvR

Kernel = rbf_o_v_o

----------Train Data----------

	precision	recall	f1-score	support
0	1.00	1.00	1.00	37
1	1.00	1.00	1.00	33
2	1.00	1.00	1.00	35
accuracy			1.00	105
macro avg	1.00	1.00	1.00	105
weighted avg	1.00	1.00	1.00	105

----------Test Data----------

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	0.89	0.94	0.91	17
2	0.93	0.87	0.90	15
accuracy			0.93	45
macro avg	0.94	0.94	0.94	45
weighted avg	0.93	0.93	0.93	45

(b) Classification report for RBF kernel OvO

Figure 11: Classification report for RBF kernel with OvR and OvO approaches

Kernel = poly_o_v_r

----------Train Data----------

	precision	recall	f1-score	support
0	1.00	1.00	1.00	37
1	0.97	0.94	0.95	33
2	0.94	0.97	0.96	35
accuracy			0.97	105
macro avg	0.97	0.97	0.97	105
weighted avg	0.97	0.97	0.97	105

----------Test Data----------

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	1.00	1.00	1.00	17
2	1.00	1.00	1.00	15
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

(a) Classification report for polynomial kernel OvR

Kernel = poly_o_v_o

----------Train Data----------

	precision	recall	f1-score	support
0	1.00	1.00	1.00	37
1	0.97	0.94	0.95	33
2	0.94	0.97	0.96	35
accuracy			0.97	105
macro avg	0.97	0.97	0.97	105
weighted avg	0.97	0.97	0.97	105

----------Test Data----------

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	0.94	1.00	0.97	17
2	1.00	0.93	0.97	15
accuracy			0.98	45
macro avg	0.98	0.98	0.98	45
weighted avg	0.98	0.98	0.98	45

(b) Classification report for polynomial kernel OvO

Figure 12: Classification report for the poly kernel with OvR and OvO approaches

You can see the code details in the attached file.

5 Problem 5: Kernels & Hard-SVM

(a)

$$||x-y||^2 = (x-y)^T(x-y) = x^Tx - x^Ty - y^Tx + y^Ty$$

As we know $x^Ty = \phi(x)^T\phi(y) = K(x,y)$.

$$||x-y||^2 = K(x,x) + K(y,y) - 2\times K(x,y) = 2 - 2\exp(\frac{-1}{2}||x-y||^2)$$

We know $\exp(.) > 0$ so we have:

$$||x-y||^2 = 2 - 2\exp(\frac{-1}{2}||x-y||^2) \leq 2$$
$$\rightarrow \quad ||x-y||^2 \leq 2$$

(b)

$$\begin{aligned}\left(\mathbf{x}^\top\mathbf{y}+1\right)^2 &= \left(\sum_{i=1}^{d} x_iy_i + 1\right)^2 \\ &= 1 + 2\sum_i x_iy_i + \sum_i\sum_j x_ix_jy_iy_j \\ &= 1 + \sum_i \left(\sqrt{2}x_i\right)\left(\sqrt{2}y_i\right) + \sum_i\sum_j \left(x_ix_j\right)\left(y_iy_j\right)\end{aligned}$$

Thus $K(x,y) = \langle\phi(x),\phi(y)\rangle$ with

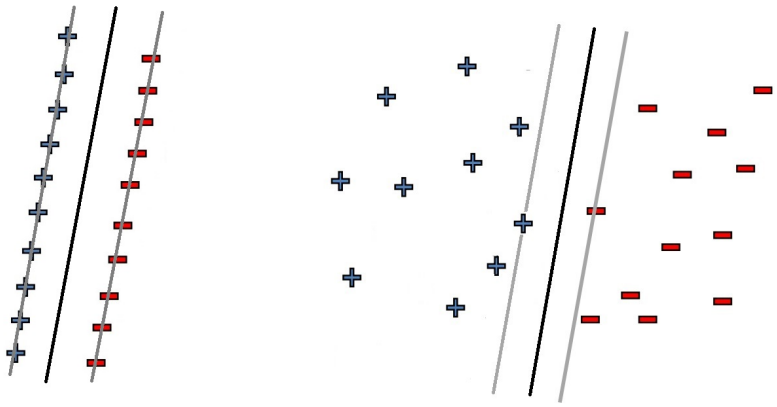
$$\phi(x) = [1, \sqrt{2}x_1 ...., \sqrt{2}x_n, x_1x_1, x_1x_2, ..., x_nx_n]$$

Note that we omit the duplicate terms like x_2x_1 and x_3x_1 (because we have already counted x_1x_2 and x_1x_3 and ...).

The dimension of the feature space associated to K is:

$$\underbrace{\frac{n^2-n}{2}}_{\text{duplicate terms}} + \underbrace{n}_{x_ix_i \text{ terms}} + \underbrace{n}_{\sqrt{2}x_i \text{ terms}} + 1 = \frac{1}{2}(n+1)(n+2)$$

(c) In a binary SVM, we at least need 2 SV to make a Hard linear SVM classification but we don't have any upper bound for support vectors and even all data could be support vectors. In the figure below we demonstrate this.



(a) All data is SV (b) Only two SV

Figure 13: Number of SV in Hard linear SVM classification

If a single point adds to the SVM model we have four different scenarios:

- The number of SVs does not change.

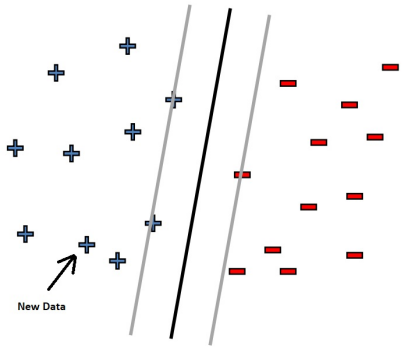


Figure 14: Change in Number of SV

- The number of SVs increased.

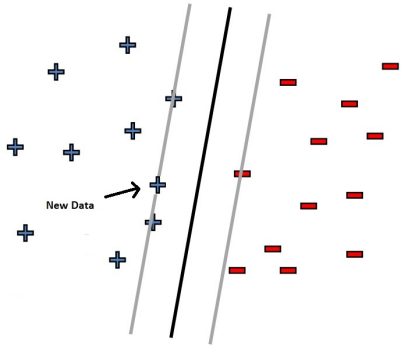


Figure 15: Increase in Number of SV

- The number of SVs decreased.

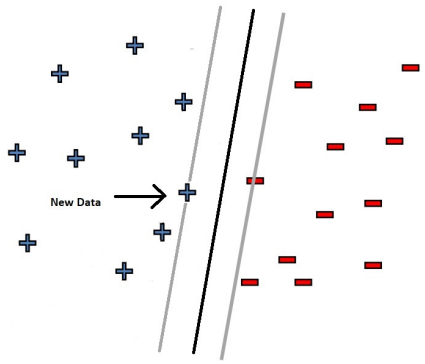


Figure 16: Decrease in Number of SV

- The data is no longer linearly separable and hard SVM does not work.

Figure 17: No longer linearly separable

6 Problem 6: Ensemble Learning

In this problem $p = 0.51$ and $1 - p = 0.49$ and we examine for different N.

(a) When we have $N = 5$ weak classifiers if more than 3 of them vote for the correct solution, our Bagging classifier makes the correct decision. So we can evaluate the probability of correct decision as below:

$$P_c(N) = \sum_{k=3}^{5} \binom{5}{k} (0.51)^k (0.49)^{5-k} = \binom{5}{3} (0.51)^3 (0.49)^2 +$$

$$\binom{5}{4} (0.51)^4 (0.49)^1 + \binom{5}{5} (0.51)^5 \approx 0.5187$$

(b) Just like in the previous case, when $N = 9$ if more than 5 of the weak classifiers vote for the correct solution, our Bagging classifier makes the correct decision.

$$P_c(N) = \sum_{k=5}^{9} \binom{9}{k} (0.51)^k (0.49)^{9-k} = \binom{9}{5} (0.51)^5 (0.49)^4 +$$

$$\binom{9}{6} (0.51)^6 (0.49)^3 + \binom{9}{7} (0.51)^7 (0.49)^2 + \binom{9}{8} (0.51)^8 (0.49) + (0.51)^9 \approx 0.5246$$

(c) We know if $p > 0.5$, $P_c(N)$ is monotonically increasing in N and $P_c(N) \to 1$ as $N \to \infty$

$$P_c(N) = \lim_{N \to \infty} \sum_{k=\frac{N+1}{2}}^{N} \binom{N}{k} (0.51)^k (0.49)^{N-k} = 1$$

In this scheme, we have assumed the decision of each classifier is taken independently of the others but as N increases the independence assumption becomes more and more unrealistic because their dataset becomes dependent on each other. However, generally increasing the number of classifiers increases the probability of a correct decision.

(d)

$$P_c(N) = \sum_{k=3}^{5} \binom{5}{k} (0.5)^k (0.5)^{5-k} = (0.5)^5 \sum_{k=3}^{5} \binom{5}{k} = (0.5)^5 (1 + 5 + 10) = 0.5$$

In contrast with previous results, if we have random classifiers, their ensemble with any N does not increase accuracy.

7 Problem 7: Support Vector Regression

(a) We have a little dataset in this part so we use GridSearchCV to evaluate the best hyperparameters to solve this regression problem. We use three kernels for this problem.

- RBF kernel: This kernel shows poor results for regression.

```
SVR(C=10000, gamma=0.1)
Mean Squared Error (MSE):
RBF Kernel: 84380660039.83066

R-squared Score:
RBF Kernel: -0.0460985162183134
```

Figure 18: Results for SVR with RBF kernel

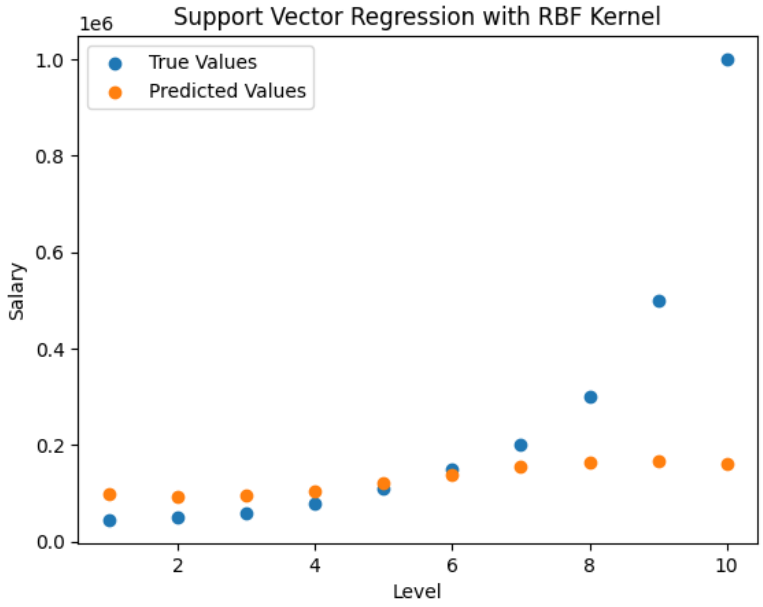


Figure 19: True values and predicted values of SVR with RBF kernel

- Linear kernel: This kernel has better results compared to the last one.

```
SVR(C=10000, gamma=1, kernel='linear')
Mean Squared Error (MSE):
Linear Kernel: 40444521700.01319

R-squared Score:
Linear Kernel: 0.4985941788133459
```

Figure 20: Results for SVR with the linear kernel

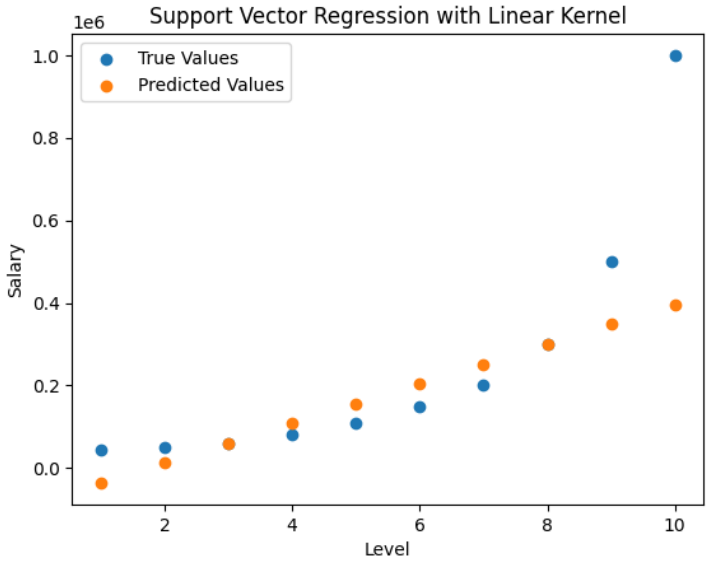


Figure 21: True values and predicted values of SVR with the linear kernel

- Polynomial kernel: This kernel has the best results for this problem. As we can see the true values can be modeled with good accuracy with a polynomial function.

```
SVR(C=1, gamma=1, kernel='poly')
Mean Squared Error (MSE):
Polynomial Kernel: 16483823630.377766

R-squared Score:
Polynomial Kernel: 0.7956438900430156
```

Figure 22: Results for SVR with the polynomial kernel

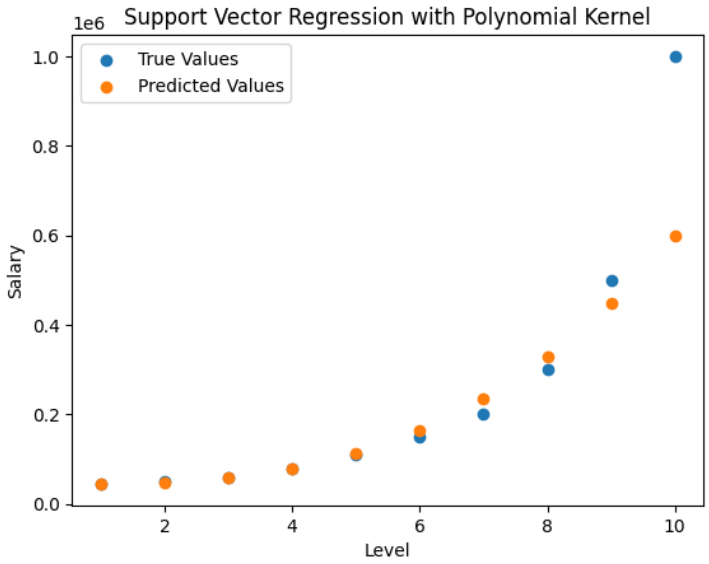


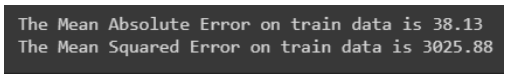
Figure 23: True values and predicted values of SVR with the polynomial kernel

(b) After loading the dataset, it is essential to perform data preprocessing and feature engineering steps to ensure the data is ready for modeling. We have many categorical and numerical columns and should perform any necessary data preprocessing and feature engineering steps, such as handling missing values, encoding categorical variables, and feature scaling.

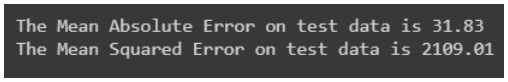
In this problem, we have used LabelEncoder for categorical columns due to the efficiency. If we use OneHotEncoder we may encounter many new columns which are not present in the training dataset and could make trouble for prediction. Recall that H2.csv has 2 times more data than H1.csv which we have used for training.

After dealing with categorical columns we use StandardScaler for the numerical values. We used a linear kernel for this part because of its efficiency.

The MSE and MAE metrics for this model are:



(a) MSE and MAE for the train data



(b) MSE and MAE for the test data

Figure 24: MSE and MAE metrics for the train and test data

We save the predicted values in *linear_kernel_prediction.csv* dataset attached to this file.

You can see the code details in the attached file.