Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Broadband Infocommunications and Electromagnetic Theory

Radar Cross Section (RCS) Histogram Classification using Machine Learning Algorithms

Author
Srđan Ćosić

Contents

Chapter 1

Introduction

1.1 Radar Cross Section (RCS)

Object detection and classification accuracy play a crucial part in today's society, as they allow the possibility of locating and identifying objects in a given space. Radar (RAdio Detection And Ranging) is a system of finding the position and velocity of an object by bouncing a radio wave off it and analyzing the reflected wave [9]. By measuring the time interval between emitting and receiving the signal, it is possible to determine the distance between the object and the radar. As there are many factors to consider, such as the size of the object, its shape, and its material composition, among many others, there is a need to define a concrete metric for measuring the visibility of an object.
Radar cross section (RCS) is equivalent to the measure of the target's ability to reflect radar signals in the direction of the radar receiver. The strike of a radiated power from a radar system, on a target, causes re-radiation in all directions. This reflected signal contains valuable information for the classification and identification of the targets, including the target's size, shape, material composition and orientation.
To quantify how objects are visible, radar cross section theory was introduced, which is the ratio of reflected power back to the radar to the incident power density of a target [7]. It is written as:

$$\sigma = 4\pi R^2 \lim_{R\to\infty} (\frac{P_{Dr}}{P_{Di}}) \tag{1.1}$$

Calculating the RCS on complex targets is challenging, thus approximate methods are used. The most common approximate methods are: Geometrical Optics (GO), Physical Optics (PO), Geometrical Theory of Diffraction (GTD), Physical Theory of Diffraction (PTD), and Method of Equivalent Currents (MEC).

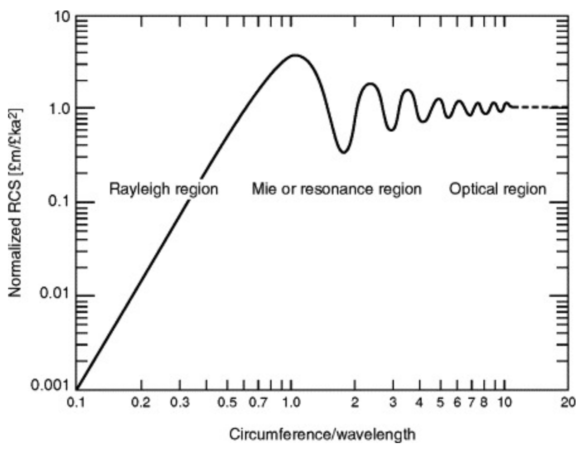

Figure 1.1: The three regions of RCS as a function of frequency [8]

The radar cross section as a function of frequency may generally be broken into three regions. These regions are referred to as Rayleigh, resonance, and optical region (See Figure 1.1) The optical region will be the focus of this project, where the frequency of the incident energy is quite high. Thus, the detection criteria in Physical Optics is that the wavelength must be significantly smaller than the dimensions of the object. Other very important parameters include the incident angle, which is the angle the radar signal hits the target, and the distance between the target and the antenna.

1.2 Introduction to the Dataset

Six differently shaped objects are hit by radar waves. The targets are made out of a good conducting material, and their size is significantly greater than the wavelength. The shapes in question are triangle, square, circle, and their combinations(triangle and square, triangle and circle, and square and triangle) (See Figures 1.3 and 1.4). The applied parameters are the following:

Parameter	**Value**
Surface Area	$36\,m^2$
Frequency(ω)	$24\,GHz$
Wavelength(λ)	$12.5\,mm$
Distance to Objects	$1\,m$ to $100\,m$
Incident Angle(θ)	$-1°$ to $1°$

Table 1.1: Radar Parameters

Within the range of the incident angle, using the Physical Optics approximation, the radar cross section value fluctuates (See Figure 1.2).

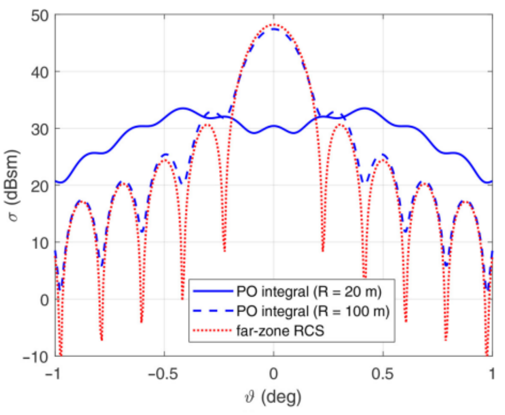

Figure 1.2: RCS with respect to the incident angle [4]

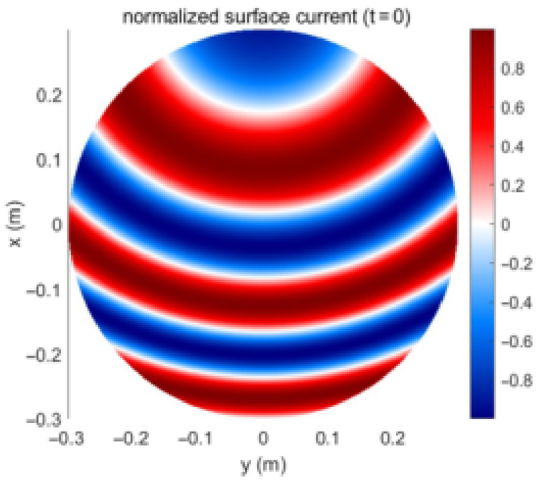

Figure 1.3: A simple circular target and the distribution of the surface current density at a certain position of the antenna [4]

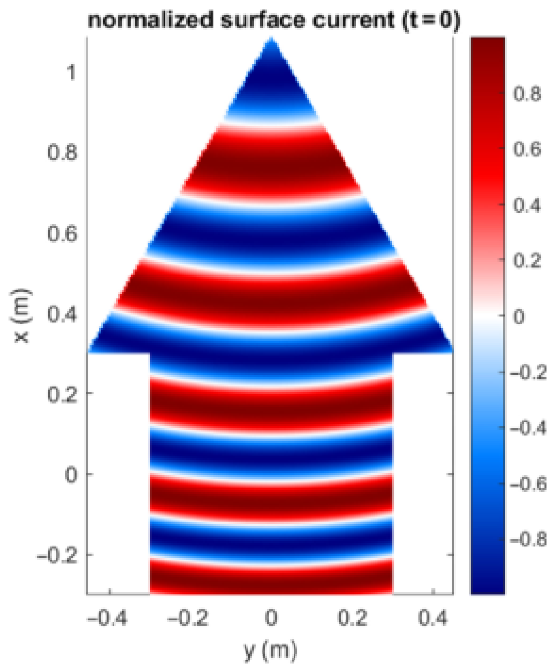

Figure 1.4: A complex target consisting of a triangle and a square, and the distribution of the surface current density at a certain position of the antenna [4]

The apparent varying distribution of RCS values based on parameters like the incident angle hides certain valuable statistics. These statistics will be represented as histograms. For example, the distribution of the histogram depends on the distance between the target and the radar antenna, and it is apparent that it stabilizes once the distance gets close to 100m (See Figure 1.5). These insights are valuable as it means each bin of a histogram is a feature that can be used for classifying the shapes.

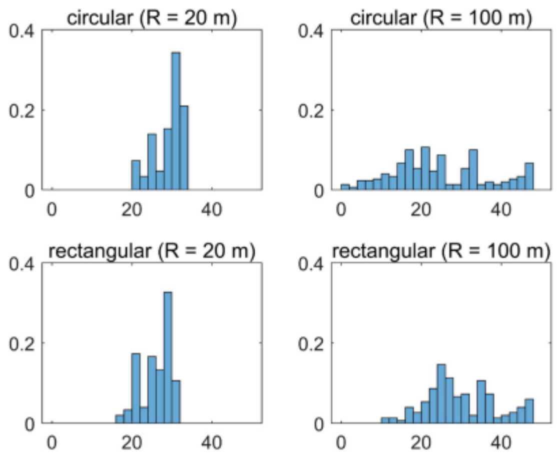

Figure 1.5: Histograms of the RCS of circular and rectangular plates at different R distances [4]

In summation, the dataset comprises 6 classes, 100 features in the form of histograms, with a total of 6000 observations.

Chapter 2

Theoretical Fundamentals of Supervised ML Classification algorithms

Machine learning is a subset of artificial intelligence focused on building computer algorithms that are capable of learning from data, by mimicking the way humans think, thus allowing computers to acquire knowledge, adjust their behavior, and improve their performance autonomously without needing explicit instructions. While machine learning has a wide range of applications, the increasingly digitized world and the large amount of data that comes with it make it especially useful in classifying data.

The subcategory of machine learning that handles classification tasks is supervised machine learning. A supervised machine-learning classification technique involves training a model to make predictions about the class inputs based on labeled training data.

Among the plethora of classification algorithms, the following three will be used:

- K-Nearest Neighbors Classification Model
- Decision Tree Classification Model
- Support Vector Machine

2.1 K-Nearest Neighbors Classification Model

2.1.1 Historical Background and Evolution of KNN

The K-Nearest Neighbors (KNN) algorithm was introduced in 1951 by mathematicians Evelyn Fix and Joseph Hodges [5], as a nonparametric method for classification and regression. In 1967 mathematicians Thomas Cover and Peter Hart revealed the upper bound error rate of the algorithm, which served as a key to assessing the effectiveness of different optimization strategies. This allowed researchers to develop new approaches for improving the accuracy of different classification algorithms, including KNN. Over the years, KNN has evolved significantly, now a fundamental pillar in the field of machine learning, with various adaptations enhancing its basic form to address specific data types and application scenarios.

2.1.2 Basic Principles of KNN

A new observation is classified based on the majority class of its K nearest neighbors in the feature space. The algorithm is an example of a lazy learner, where the data is processed during the training phase.

2.1.3 Mathematical Formulation of KNN

The metric that calculates the distance between data points significantly impacts the performance of the algorithm, as it affects how the similarity between data points will be calculated. More prominent distance metrics include:

Euclidean distance(the L2 distance), representing the straight line distance between two points in Euclidean space:

$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.1}$$

Manhattan distance(the L1 distance), calculated as the sum of the absolute differences between the coordinates of the two points:

$$d = \sum_{i=1}^{n} |x_i - y_i| \tag{2.2}$$

Chebyshev distance, representing the maximum absolute difference between the coordinates of two points along any dimension:

$$d = \max_{i=1}^{n} |x_i - y_i| \tag{2.3}$$

Minkowski distance, representing a generalization of the L1 and L2 distances:

$$d = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}} \tag{2.4}$$

2.1.4 Algorithmic Steps in KNN

For an observed data point, the algorithm calculates the distances to all other data points so it can discern the nearest neighbors. By performing a simple majority vote using the classes of its nearest neighbors, the new data point is categorized.

The application of the KNN algorithm can be summarized in the following steps:

1. Splitting the dataset into training and validation sets

2. Selecting the value of the K parameter

3. For each point in the test data:

  3.1 Calculating the distance from the observed data point to every other data point
  3.2 Determining the nearest neighbors based on the distances
  3.3 Compiling the categories of the nearest neighbours
  3.4 Using a simple majority voting of the categories to determine the category of the observed data point

2.1.5 Parameter Selection and Its Impact

The most important parameter to consider is clearly the K parameter or the number of neighbors considered for the classification. Lowering the value of K, makes the algorithm more sensitive to noise, as each nearest neighbor has a higher influence on the final result. Raising the value, on the other hand, smooths the decision boundaries between classes, leading to a decrease in complexity which may cause underfitting.

2.1.6 Complexity Analysis

Since the distances are calculated between the new observation and all the other observations, the algorithm is computationally expensive when dealing with large datasets. The algorithm is a lazy learner, it stores the entire dataset in memory, making it space-intensive, and the time complexity of classifying a new data point is proportional to the size of the training set.

2.1.7 Advantages and Limitations

Although one of the simpler machine learning algorithms, KNN has found its application in various fields, such as handwriting detection, image, speech, and video recognition etc. Despite its widespread use, KNN does have its limitations when used on large datasets, such as its slow speed and memory issues.

2.1.8 Conclusion

Although not without its drawbacks, KNN is a powerful machine learning algorithm, with its relatively straightforward implementation and overall good performance.

2.2 Decision Tree Classification Model

2.2.1 Historical Background and Evolution of Decision Trees

Due to its intuitive nature, the algorithm had for a long time found uses in psychology to model human learning, and it wasn't until the middle of the 20th century that the algorithm found practical uses in supervised machine learning problems. The first groundwork for future theoretical development was made by Sir Ronald Aylmer Fisher's

Iris flower data set, used as an example of discriminant analysis. In 1963 James Morgan and John Sanquist published the first regression tree model. Robert Messenger and Lewis Mandell published the first classification tree in 1972, called the Theta Automatic Interaction Detector (THAID). These crucial milestones allowed subsequent studies to refine the model and expand the applicable use of the algorithm to make it a cornerstone of machine learning as it is today.

2.2.2 Basic Principles of the Decision Trees

The algorithm classifies data based on a hierarchical structure of decision rules. It's an example of an eager learning algorithm, as the model is built and able to make predictions after the training phase is completed.

2.2.3 Mathematical Formulation of Decision Trees

The algorithm is based on partitioning data, therefore selecting a suitable metric that will measure the effectiveness of each possible split is crucial for ensuring the highest possible accuracy. Most commonly used metrics are the following:

Gini impurity: A measure of how often a randomly chosen element would be incorrectly identified. The impurity range spans from 0 (all elements belonging to the same class) to 0.5 (equal distribution among classes). The formula is given by:

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2 \tag{2.5}$$

G_i is the Gini impurity of the ith node.
$p_{i,k}$ is the ratio of class k instances among the training instances in the ith node.

Entropy: A measure of information disorder or impurity in a set. The entropy ranges between 0 (pure nodes) and 1 (the maximum disorder). The formula is given by:

$$H_i = -\sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^{n} p_{i,k} \log_2(p_{i,k}) \tag{2.6}$$

H_i is the Entropy of the ith node.
$p_{i,k}$ is the ratio of class k instances among the training instances in the ith node.

2.2.4 Algorithmic Steps in Decision Trees

The classification involves making a decision at each internal node based on a required feature of the observation, and it stops when a leaf node is reached.
The application of the decision tree algorithm can be summarized in the following steps:

1. Splitting the dataset into training and validation sets
2. Selecting the values of its parameters
3. For the training phase:

    3.1 Starts with a root node containing the entire training dataset
    3.2 The dataset is split based on the selected tree-splitting criteria
    3.3 Where splitting takes place, an internal node is created
    3.4 Creating leaf nodes after further splitting is not possible

4. Classification by the new observation traversing the previously generated tree

2.2.5 Parameter Selection and Its Impact

The key to proper classification in the algorithm involves building the best tree for a given dataset. As a result, tuning the parameters that dictate the tree-building process is fundamental for ensuring that the tree will be the best-performing one. Some key parameters include:

max_depth: Specifies the length of the tree. A tree that is too big is also too complex, thus potentially capturing noise [6].
min_samples_split: Defines the minimum number of samples a dataset needs to have during the tree-building process to be split. If unable to split, a leaf node will be created in its place.
min_samples_leaf: The minimum number of samples dataset needs to have during the training phase for a leaf node to be created.

2.2.6 Complexity Analysis

As it is an eager algorithm, storing the entire dataset in memory is not required. Instead, the space complexity depends on the depth of the tree, making it more space-efficient than the KNN algorithm. The time complexity is determined by the time it takes for an observation to traverse the tree, having a logarithmic relationship with the number of nodes.

2.2.7 Advantages and Limitations

Decision trees can be built to be very simple to overly complex, and this versatility is what lends itself to being used in different fields. The algorithm can be found in telecommunications for network fault diagnosis, manufacturing for quality control, and as a part of recommendation system algorithms, among many others. Decision trees are a type of greedy algorithm, as they find the best local solutions at each stage, which may lead to a final solution that is not optimal [6].

2.2.8 Conclusion

The decision tree algorithm's efficiency and versatility is the reason it finds frequent use in many technological fields. The algorithm has its limitations, however, such as its greedy nature causing it to be prone to overfitting. Techniques that bolster the performance are being refined, such as its classifier variant in the form of random forests that use ensemble methods.

2.3 Support Vector Machine

2.3.1 Historical Background and Evolution of Support Vector Machine

The Support Vector Machine algorithm was invented in 1963 by Vladimir Vapnik and Alexey Chervonenkis, with its early application being fairly limited. One of the most important milestones in the research of the algorithm was the addition of the kernel trick [6]. Introduced in 1992, it allowed the algorithm to easily handle datasets whose features have a nonlinear relationship. This development allowed the algorithm to become a standard in many machine-learning problems.

2.3.2 Basic Principles of Support Vector Machines

The model works by separating observations by drawing a boundary between them, as far as possible from the observations. It is a type of eager learning algorithm, thus the model does not have to save the entire dataset in memory to make predictions. For finding the optimal solution, the model uses the data points closest to the boundary which are called support vectors.

2.3.3 Mathematical Formulation of Support Vector Machines

In the case of non-linearly separable data, transforming the original space of the observations into one with a higher dimension makes it possible to draw decision boundaries between them. This is accomplished using different kernel functions [1] including:

Radial Basis Function (RBF), commonly used for non-linear separation, uses a Gaussian function. The formula is given by:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \tag{2.7}$$

γ: the kernel coefficient
$\|\mathbf{x}_i - \mathbf{x}_j\|^2$: the squared Euclidean distance between the two input vectors
σ: controls the width of the bell-shaped curve.

Sigmoid Function, using a function that maps input values to a value between 0 and 1. The formula is given by:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\alpha \mathbf{x}_i^T \mathbf{x}_j + h) \tag{2.8}$$

α: is the scaling parameter
$\mathbf{x}_i^T \mathbf{x}_j$: dot product of the input vectors.
h: a constant.

2.3.4 Algorithmic Steps in Support Vector Machines

New observations are classified based on their mapped position in the transformed feature space.
The application of the SVM algorithm can be summarized in the following steps:

1. Splitting the dataset into training and validation sets

2. Selecting the values of its parameters
3. Applying a kernel function to transform the feature space
4. Defining a potential decision boundary
5. Maximizing the margin using support vectors
6. Classification based on the position of an observation in the feature space relative to the hyperplane

2.3.5 Parameter Selection and Its Impact

Parameters that affect the shape of the decision boundary have a significant impact on the performance of the mode. Some key parameters include:
C, also known as the regularization parameter, controls the importance of maximizing the margin and minimizing error. Higher C values make the margin smaller, potentially leading to poor generalization.
gamma, defines the amount of curvature of the decision boundary by tuning the weight of each observation.

2.3.6 Complexity Analysis

The computational complexity of the Support Vector Machine algorithm exhibits significant variability, depending on many factors, such as the number of support vectors, the applied kernel function, the linear or non-linear relationship of observations, etc. The algorithm can be exceptionally computationally expensive when dealing with large datasets, due to the complexity of the kernel function calculations.

2.3.7 Advantages and Limitations

The kernel trick allows Support Vector Machines to perform well when handling high-dimensional data. This ability is the reason Support Vector Machines can be found in domains where data is particularly complex, such as bioinformatics, for protein and cancer classification, image and face detection, and remote sensing among others. It possesses a regularization parameter, which allows the algorithm to generalize well to new data. The biggest downside of the algorithm is its bad scaling with the size of the dataset.

2.3.8 Conclusion

Support Vector Machine Algorithm is recognized as an integral component of machine learning, due to its performance on both simple and complex datasets. Its application, although already widespread, will continue growing as further studies enhance the algorithm's performance.

Chapter 3

Analysis of basic Supervised ML Classification algorithms

3.1 Building basic classifiers

Three different machine learning models will be created for classifying simple flat shapes based on their histogram representation. Various machine learning techniques for enhancing the performance will be applied, including Scaling, Hyperparameter Tuning, and Ensemble Learning Techniques, and their impact on the performance of the model will be presented and discussed.
The dataset contains 1000 observations, and it will be partitioned so 70% will be allocated for the training set, and the rest will be used for the validation set. The training set will be used for building the models, and the validation set will be used for evaluating their performance.
It is first necessary to examine the baseline performances of the models, which will be accomplished using default values. Accuracy will be used as a metric for performance evaluation. It is the ratio of how many correct predictions the model made to the total number of predictions. The results after building and testing the models are the following:

Model	**Accuracy (%)**
KNN	76.62
CART	70.31
SVM	67.69

Table 3.1: Basic classification model accuracy

From the basic classifiers, KNN algorithm is the most accurate.

3.2 Model Performance Improvement

The machine learning algorithms process features of a dataset as if they are on the same scale. Thus, its necessary to apply scaling on the dataset, as it ensures that one feature does not impact the performance more than the others. One of the most effective scaling techniques is called standardization. It transforms data so it has a standard Gaussian distribution with a mean of 0 and a standard deviation of 1.

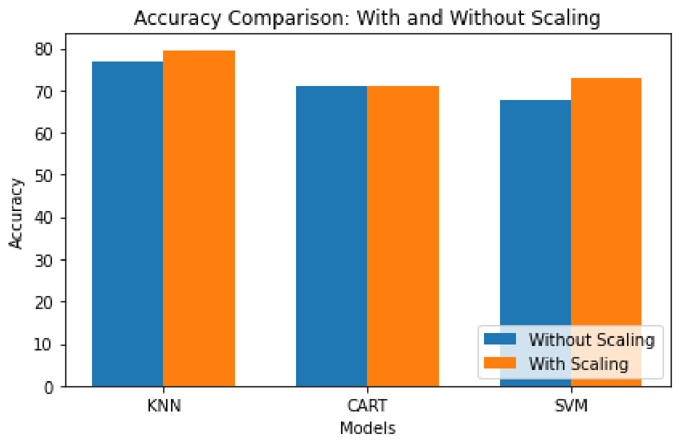

Figure 3.1: Comparison of accuracy before and after standardization

Parameters that affect the learning process greatly, but have to be preset by the user, are called hyperparameters. As they control the learning process, tuning them is crucial for optimizing performance. One popular tuning method is called the grid search. Grid search systematically explores all possible combinations of parameter values within a predefined grid and picks the best ones.

The impact of different hyperparameter values on the accuracy of the K-Nearest Neighbors Classification Model can be read from the following plots:

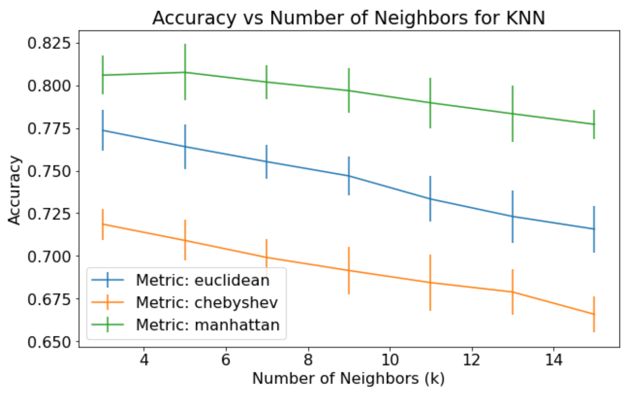

Figure 3.2: Line plot depicting the impact of different K values and distance metrics on accuracy

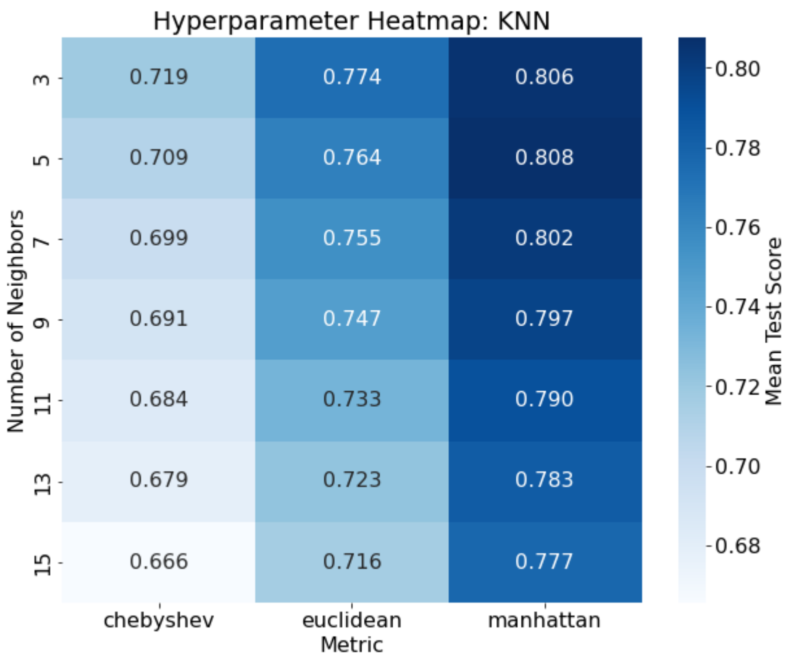

Figure 3.3: Hyperparemeter tuning plot depicting the impact of different K values and distance metrics on the accuracy

It can be concluded that lower neighbor values (k=5) and the Manhattan distance metric yield the highest accuracy.

The impact of different hyperparameter values on the accuracy of the Decision Tree Classification Model can be read from the following plots:

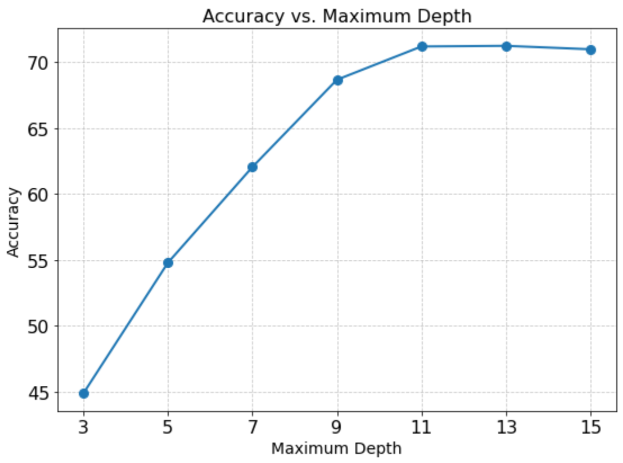

Figure 3.4: Line plot depicting the impact of different maximum depth values on accuracy

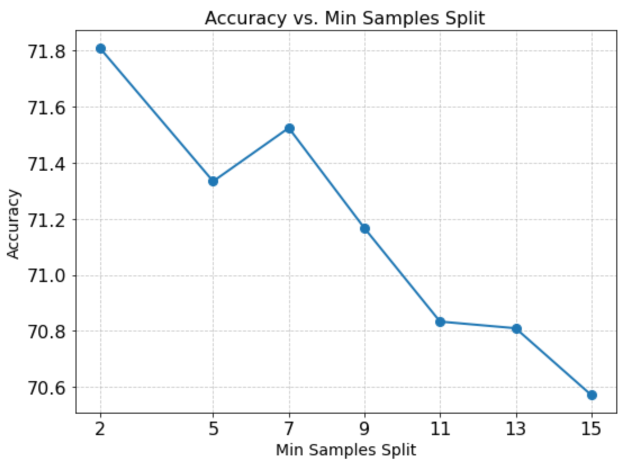

Figure 3.5: Line plot depicting the impact of different min samples split values on accuracy

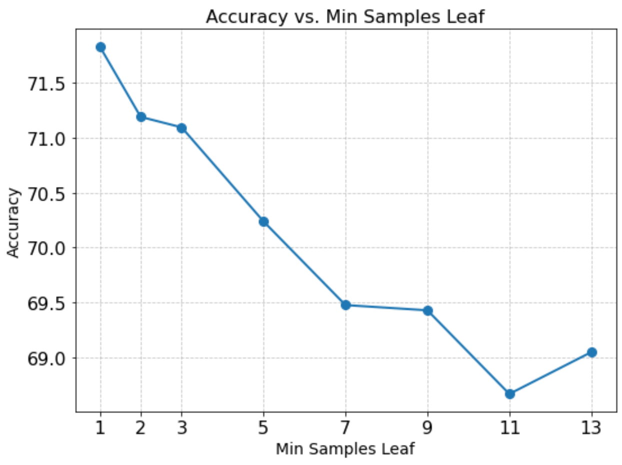

Figure 3.6: Line plot depicting the impact of different min samples leaf values on accuracy

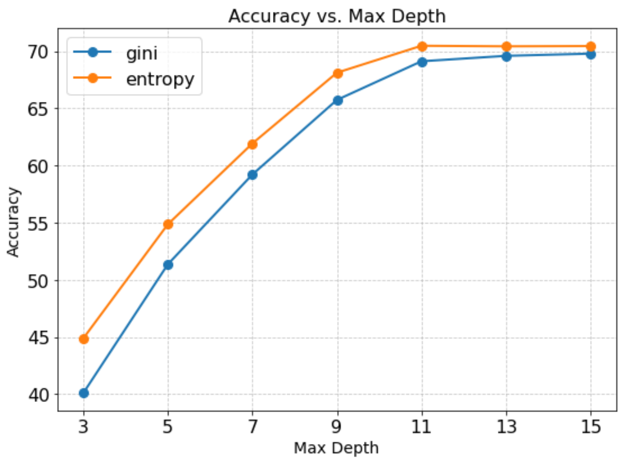

Figure 3.7: Line plot depicting the impact of different criterion types on accuracy

It is evident from the figures that a max_depth value of 11, along with smaller values for min_samples_split and min_samples_ leaf yield higher accuracy.

The impact of different hyperparameter values on the accuracy of the Support Vector Machine Model can be read from the following plots:

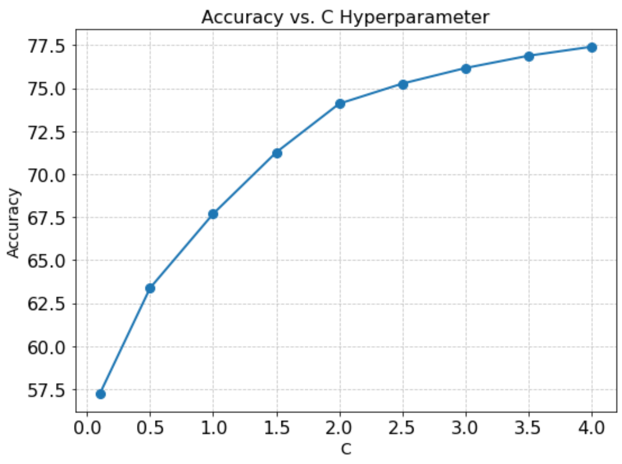

Figure 3.8: Line plot depicting the impact of different C values on accuracy

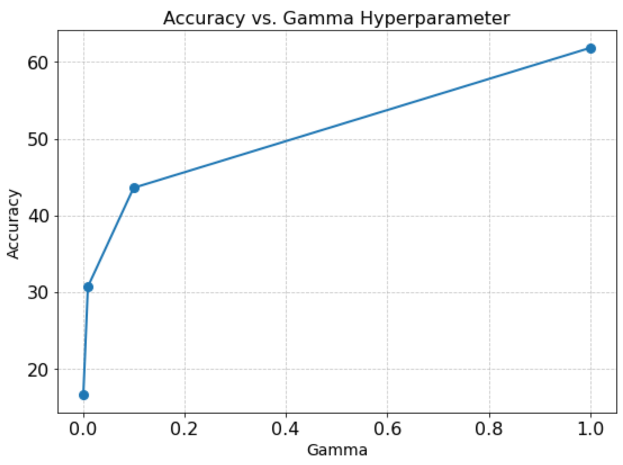

Figure 3.9: Line plot depicting the impact of different gamma values on accuracy

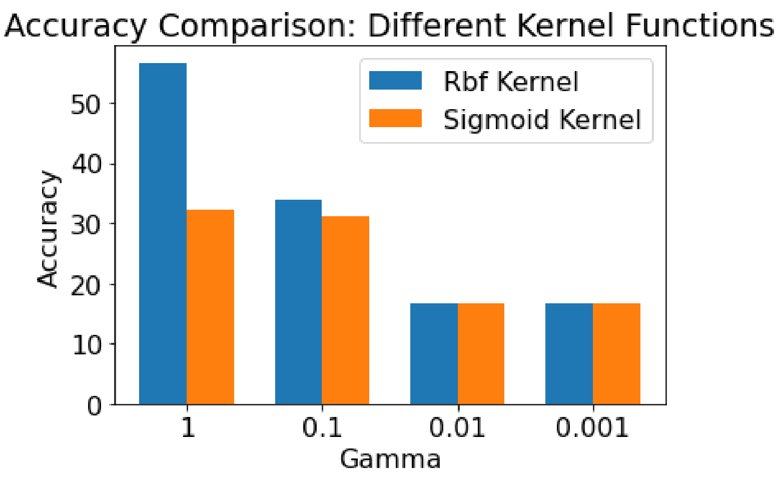

Figure 3.10: Comparison of accuracy of different kernel functions

Using the RBF kernel function, along with higher values for the regularization parameter C and gamma give better accuracy.

Model	Default Accuracy (%)	Tuned Accuracy (%)
KNN	76.62	81.26
CART	70.31	70.76
SVM	67.69	78. 52

Table 3.2: Accuracy before and after model performance improvement

Feature scaling and hyperparameter tuning were successful. A clear increase in the performance of the models can be observed (See Table 3.2).

3.3 Model Optimization

After performance-boosting techniques like scaling and hyperparameter tuning, a potential problem that might persist is the models generalizing poorly to unseen data.
To analyze the behavior of the model, it is important to check the model's degree of bias and variance.
Bias represents the error appearing when comparing the average predictions to the actual values. Variance represents changes in the model when new data is used.
Underfitting is the inability of the model to recognize patterns in the data, thus it is high bias and low variance. Overfitting, on the other hand, is when the model is too complex and thus makes decisions based on the noise, thus it is low bias and high variance.
In the previous section, it was concluded that the accuracy of the models is high, thus underfitting is not an issue.
For checking overfitting, learning and validation curves could be applied. Learning curves show model performance as the dataset increases, while validation curves show accuracy as a function of one of the hyperparameters. A good sign that there is overfitting, is if there is a large gap between the Training Score line and the Cross-Validation Score line on the learning curve.

Learning and validation curves of KNN:

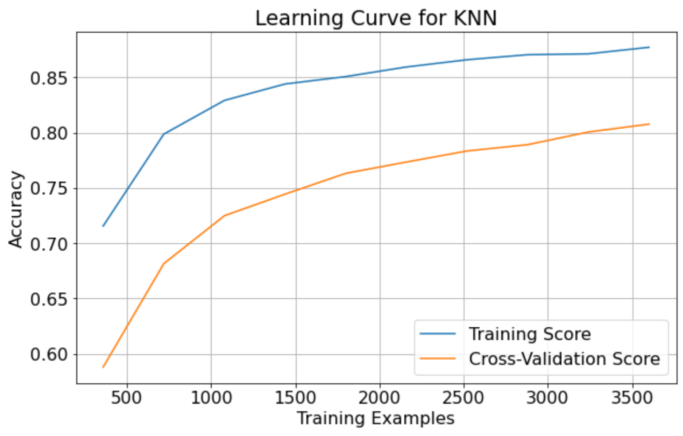

Figure 3.11: KNN classifier learning curve

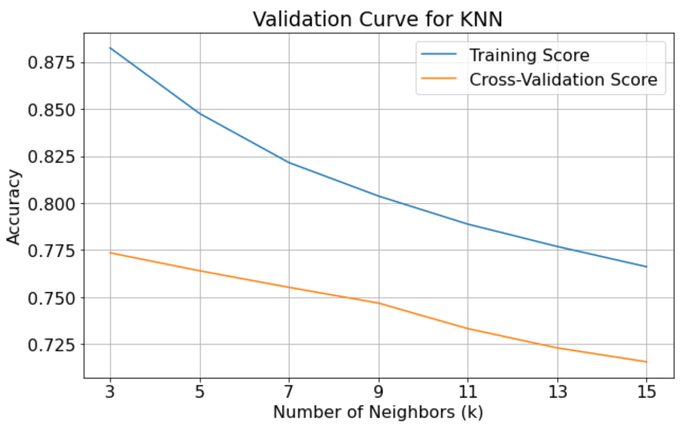

Figure 3.12: Validation curve for the K parameter

Learning and validation curves of Decision Tree Classifier:

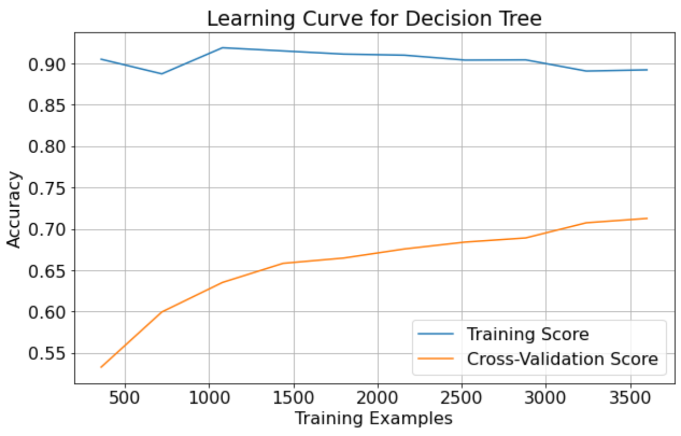

Figure 3.13: Decision Tree classifier learning curve

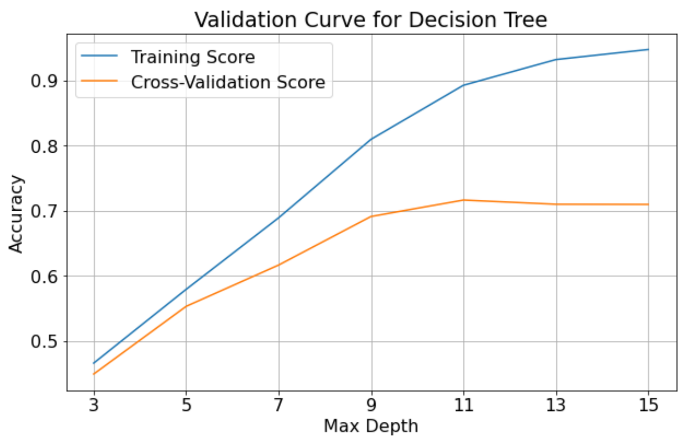

Figure 3.14: Validation curve for the maximum depth parameter

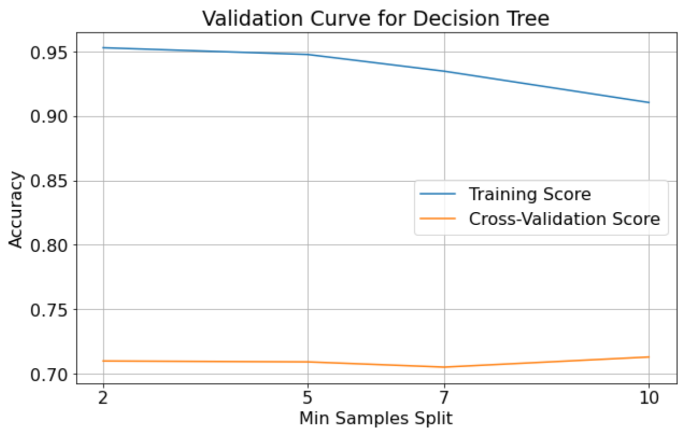

Figure 3.15: Validation curve for the min samples split parameter

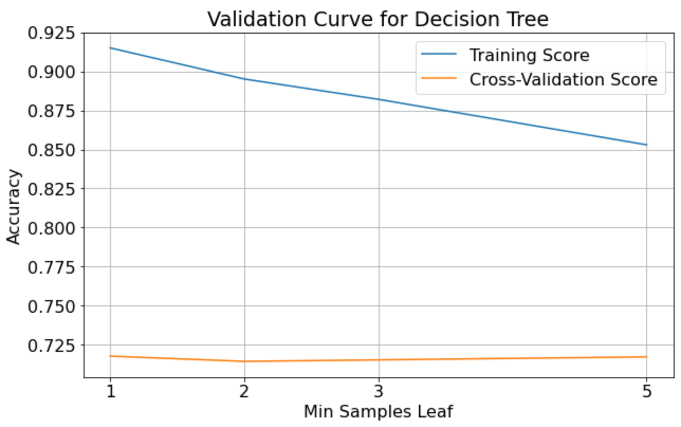

Figure 3.16: Validation curve for the min samples leaf parameter

Learning and validation curves of SVM:

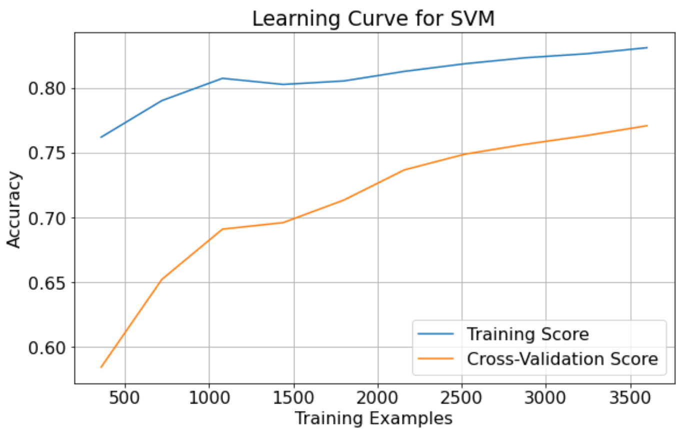

Figure 3.17: SVM classifier learning curve

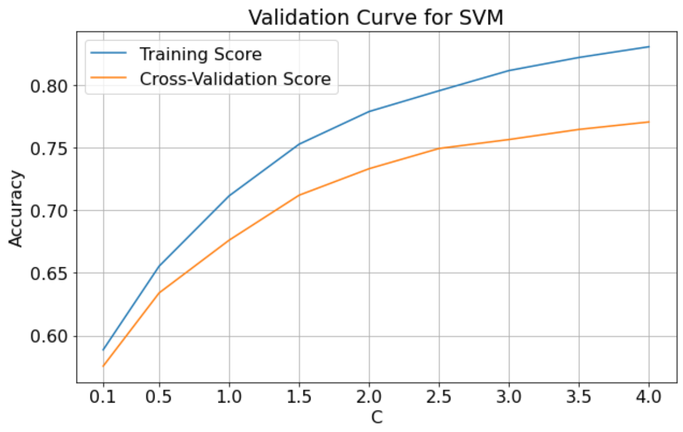

Figure 3.18: Validation curve for the C parameter

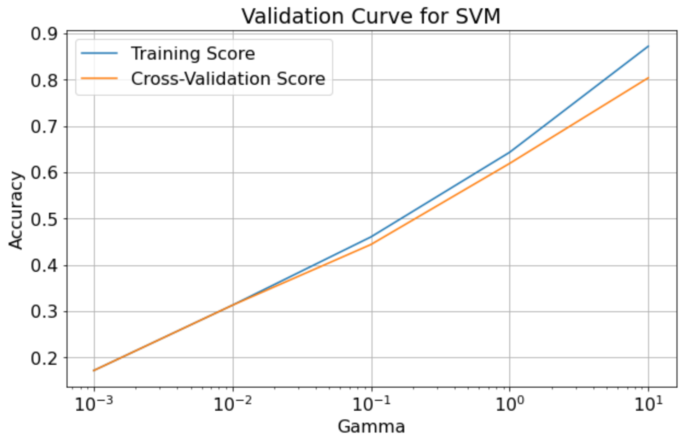

Figure 3.19: Validation curve for the gamma parameter

It is evident from the plots that the models are overfitting the data, thus it is beneficial to apply certain techniques to mitigate it. These include: adjusting the range of values for the grid search and employing ensemble learning techniques such as bagging [2]. In bagging, the technique of bootstrapping is employed to select multiple different sub-sets, with replacement, of the same dataset. Subsequently, multiple smaller models will be trained on these subsets. The predictions of these models are then grouped, using aggregation, and then by majority voting the best prediction will be chosen.
The parameter setting that yields the best accuracy score does not always yield the best overfitting score. Therefore it is necessary to assess the impact of each value in a specified range by subtracting validation and training accuracies (See Table 3.3).

3.3.1 Optimized KNN classifier

K Value	**Train Accuracy (%)**	**Validation Accuracy (%)**
3	88.74	77.67
5	85.62	77.50
7	82.60	76.22
9	81.10	75.17
11	80.05	73.50
13	78.48	72.94
15	77.79	72.72

Table 3.3: Comparison of training and validation accuracy for KNN classifier

Ideally, the value that provides a balance of high accuracy and low overfitting should be chosen. For this reason, the value of 7 seems the best fit.

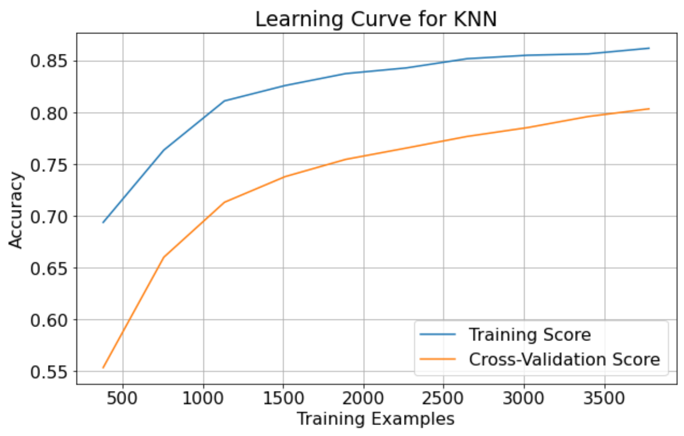

Figure 3.20: KNN classifier learning curve after optimization

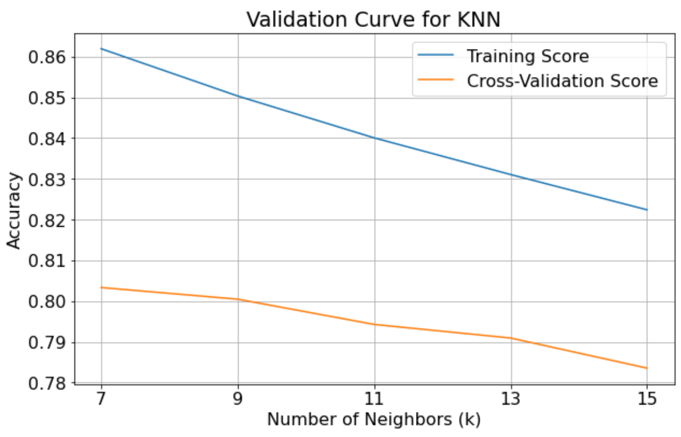

Figure 3.21: Validation curve for the K parameter after optimization

3.3.2 Optimized Decision Tree classifier

A lower value of the maximum tree depth parameter typically decreases overfitting, as it makes the decision tree less deep and, in turn, less complex. The value chosen by the grid search is reasonably high, therefore it should be set manually.

max_depth	Train Accuracy (%)	Validation Accuracy (%)
3	41.60	39.44
5	53.14	49.33
7	63.74	58.89
9	76.05	66.00
11	83.90	68.56
13	91.79	71.11
15	96.45	72.33

Table 3.4: Comparison of training and validation accuracy for Decision Tree classifier

A value that is not too high but also not too low seems like the best fit, therefore the value of 9 is selected.

Since multiple smaller trees are created during the ensemble learning process, the classifier is called a Random Forest classifier.

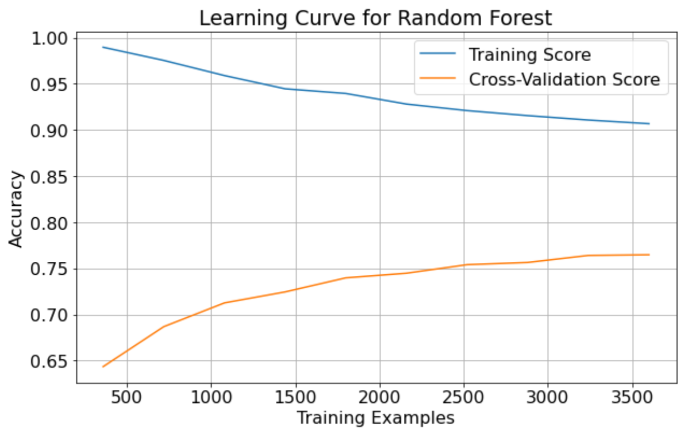

Figure 3.22: Random Forest classifier learning curve

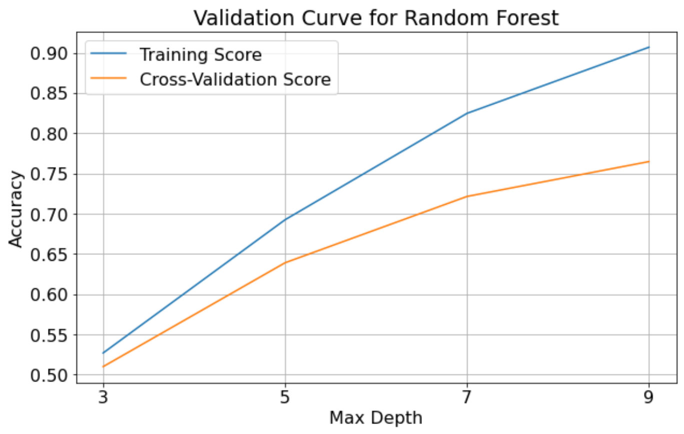

Figure 3.23: Validation curve for the maximum depth parameter after optimization

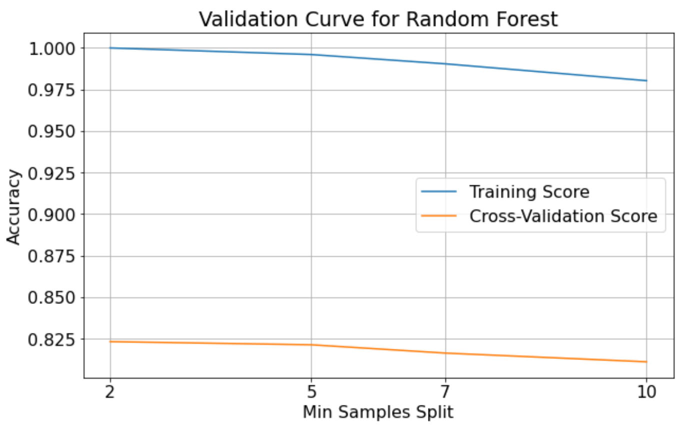

Figure 3.24: Validation curve for the min samples split parameter after optimization

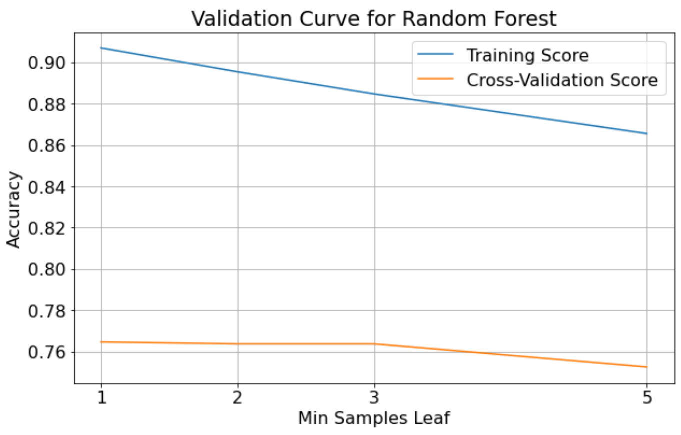

Figure 3.25: Validation curve for the min samples leaf parameter after optimization

3.3.3 Optimized SVM classifier

The key parameter for avoiding overfitting issues is the aptly named regularization parameter (C), as it controls the penalty for misclassified observations. Specifying the best C value means finding the appropriate trade-off of model complexity and potential noise capturing.

C Value	**Train Accuracy (%)**	**Validation Accuracy (%)**
0.1	59.43	57.33
0.5	66.24	62.44
1	72.17	67.72
1.5	76.69	72.72
2	78.55	73.78
2.5	80.29	75.00
3	81.88	75.89
3.5	82.83	76.50
4	83.57	77.44

Table 3.5: Comparison of training and validation accuracy for SVM classifier

The C value of 2.5 appears to be the best candidate for the model.

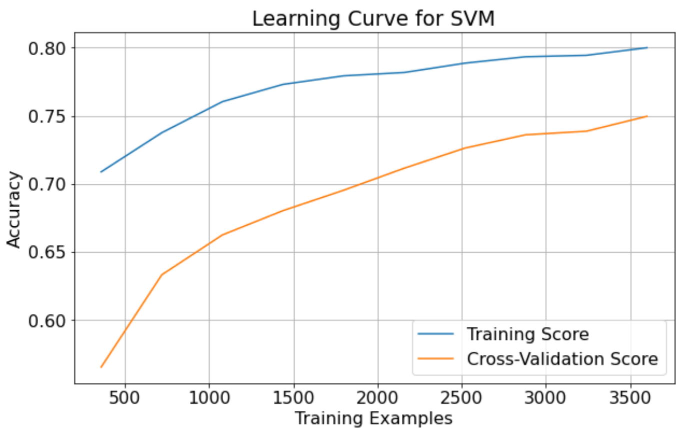

Figure 3.26: SVM classifier learning curve after optimization

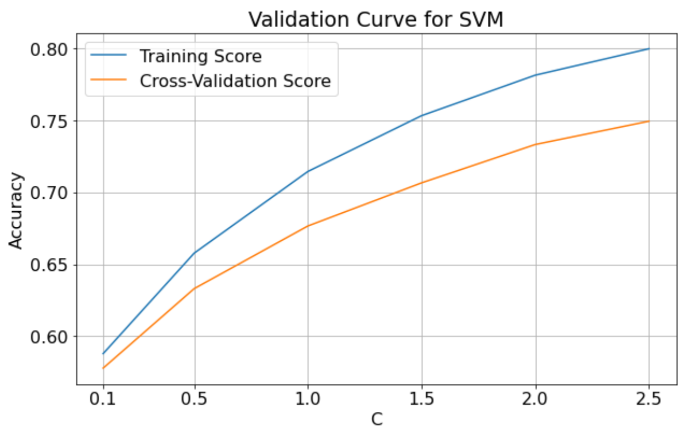

Figure 3.27: Validation curve for the C parameter after optimization

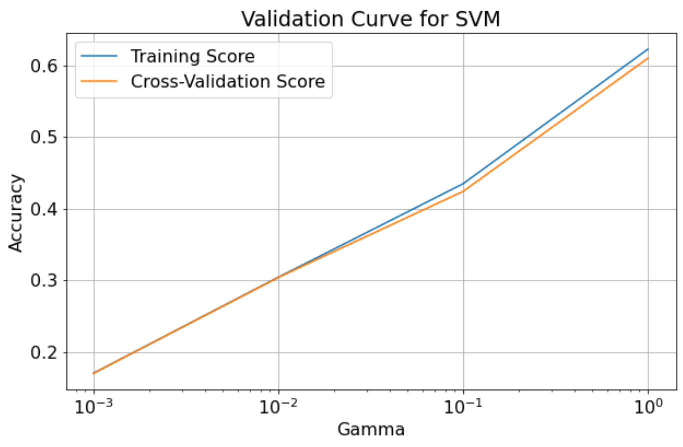

Figure 3.28: Validation curve for the gamma parameter after optimization

As the gap between the Training Score line and the Cross-Validation Score line is decreased among all three models, it can be concluded that overfitting was reduced.

3.3.4 Final performance estimation

The technique that will be used for assessing the final performances of the models is called cross-validation. Cross-validation works by splitting the training set into k-folds, then the model is trained k times using one of the folds as a validation set and the rest for training the model. The k number of results is then averaged to get one final value. The final result is less susceptible to variance during the initial test-training split and is a good way to assess model generalizability.

Model	Accuracy (%)
KNN	76.50
CART	77.79
SVM	76.26

Table 3.6: Final classification model accuracy

Confusion charts will be used to highlight how well each model predicts observations of their respective classes. The ones that are predicted that observations belong to are displayed on the x-axis, while on the y-axis are the real classes of the observations. The number of predictions is written inside the squares.

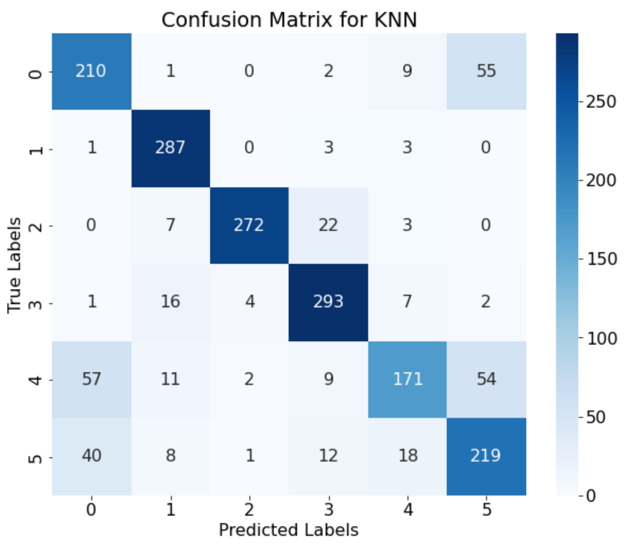

Figure 3.29: Confusion matrix for the KNN classifier

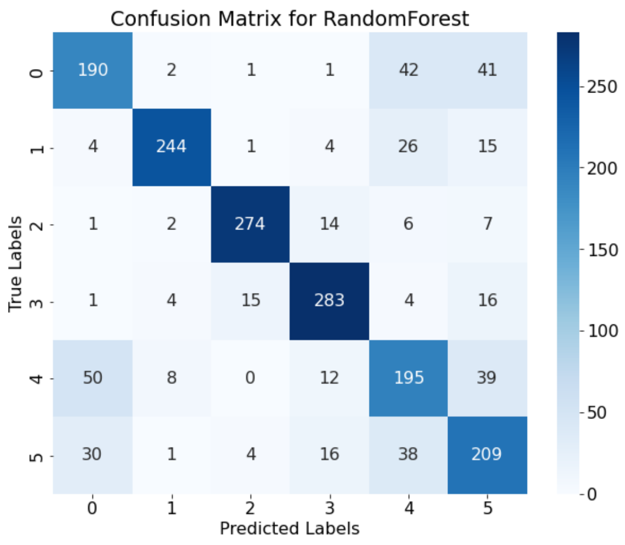

Figure 3.30: Confusion matrix for the Random Forest classifier

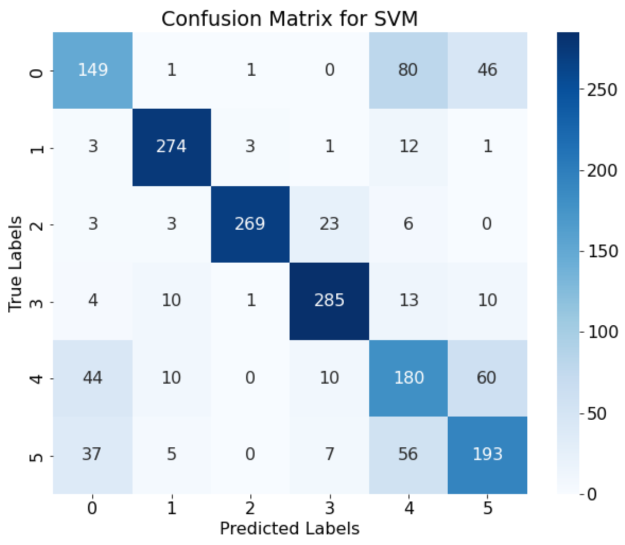

Figure 3.31: Confusion matrix for the SVM classifier

To graphically represent how well the models classify each specific class, Receiver Operating Characteristic Curve (ROC) are used [3].
False Positive Rate (FPR) represents the proportion of predictions incorrectly placed into the specified class, while in fact belonging to other classes (False Positives). It is plotted on the x-axis, and calculated as:

$$\text{False Positive Rate (FPR)} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

True Positive Rate (TPR), on the other hand, represents the ratio of correctly classified observations (True Positives). It is plotted on the x-axis, and calculated as:

$$\text{True Positive Rate (TPR)} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

The curve of lines that are closer to the top left corner is ideal. Another way to assess performance is to examine the Area under the ROC Curve (AOC-ROC). Values closer to 1 illustrate good performance.

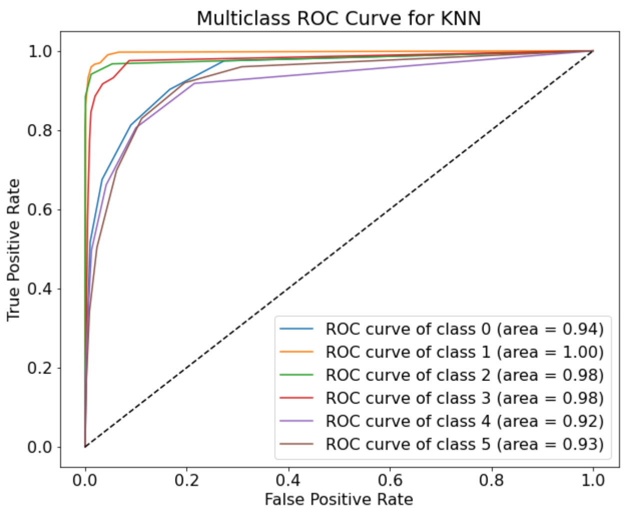

Figure 3.32: ROC for the KNN classifier

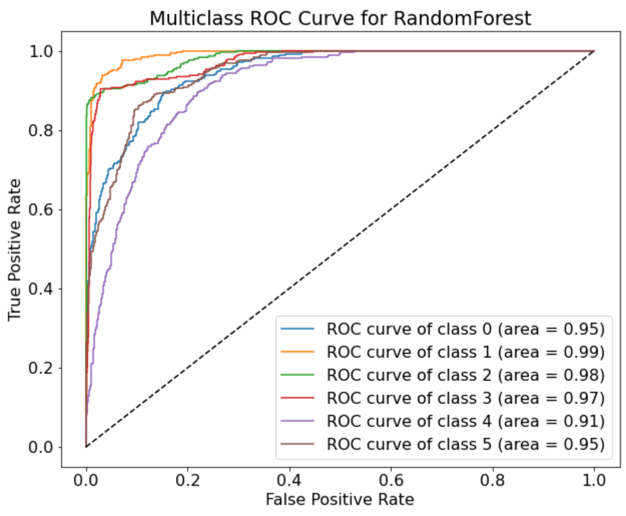

Figure 3.33: ROC for the Random Forest classifier

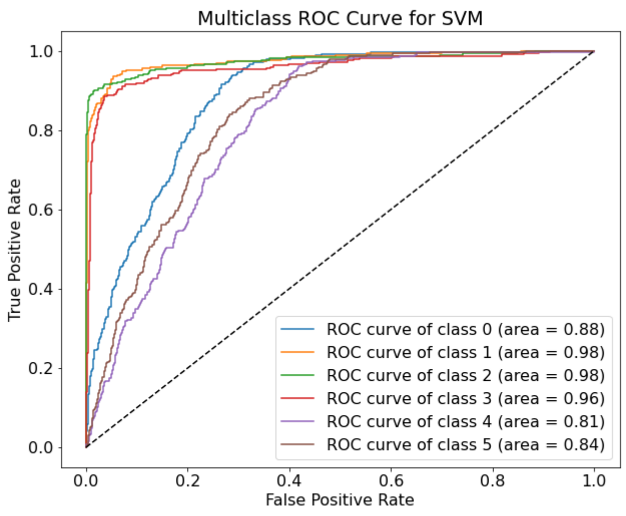

Figure 3.34: ROC for the SVN classifier

It can be deduced from the plots that all three models distinguish the six classes remarkably well. Three classes in each plot have a slightly lower performance, likely representing the three more complex shapes.

In conclusion, all three models perform well, the K-Nearest Neighbors Classification Model and the Decision Tree Classification Model being the most accurate for the dataset.

Chapter 4

Conclusion

Radar Cross Section (RCS) is a metric that holds significance on a global scale, and its importance has been described and the theoretical background explained. Using the principles of Physical Optics (PO), and the features derived from it in the form of histograms, six differently shaped objects were classified using supervised machine learning techniques. The shapes in question are a triangle, a circle, a square, a triangle combined with a circle, a triangle combined with a square, and a circle combined with a square. For the classification, three different algorithms have been used: the K-Nearest Neighbors Classification Model, the Decision Tree Classification Model, and the Support Vector Machine, with their theoretical foundations explored and applications described. After the basic classifiers have been built, the performance is enhanced using various techniques. Scaling was applied to guarantee that all features have an equal impact. Hyperparameter Tuning for selecting the most appropriate values for each parameter. Techniques for observing the signs of overfitting and reducing it were implemented, for example, ensemble learning methods, to ensure the models perform well on unseen data. Using numerous plots and tables, the impact of the procedures and the success of the final models have been accordingly visualized and presented. Additionally, cross-validation was incorporated to better assess accuracy. Finally, it can be concluded from the final results that the K-Nearest Neighbors Classification Model and the Decision Tree Classification Model are the most effective for classifying the dataset. For further research, deep learning architectures, for instance, Artificial Neural Networks (ANN), could be applied to classify the dataset.

Bibliography

[1] Christopher M. Bishop and Nasser M. Nasrabadi. *Pattern Recognition and Machine Learning.* Springer, 2006.

[2] Jason Brownlee. *Machine Learning Mastery With Python: Understand Your Data, Create Accurate Models and Work Projects End-To-End.* 2016.

[3] Andriy Burkov. *The Hundred-Page Machine Learning Book.* Andriy Burkov, 2019.

[4] Aysu Coskun and Sandor Bilicz. Target classification using radar cross section statistics of millimeter-wave scattering. In *Proc. of the 20th International IGTE Symposium*, page 79, 2022.

[5] Evelyn Fix and Joseph L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties, 1951.

[6] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow.* O'Reilly, 2019.

[7] Eugene F. Knott, John F. Schaeffer, and Michael T. Tulley. *Radar Cross Section.* SciTech Publishing, 2004.

[8] Hsueh-Jyh Li and Yean-Woei Kiang. Radar and inverse scattering. In WAI-KAI CHEN, editor, *The Electrical Engineering Handbook*, pages 671–690. Academic Press, 2005. ISBN 9780121709600. DOI: `10.1016/B978-012170960-0/50047-5`. URL `https://www.sciencedirect.com/science/article/pii/B9780121709600500475`.

[9] Bassem R. Mahafza and Atef Elsherbeni. *MATLAB Simulations for Radar Systems Design.* CRC Press, 2003.