COMP30027 Report

1. Introduction

As the world progresses toward driverless cars, reliable traffic sign detection is crucial. If this technology is not done well and carefully, it will cost human lives. This means that the system must be both fast and accurate when facing a traffic sign, and this is also the intention of this report. We will be exploring 2-3 models that were analysed on 3 different datasets. The models include the simple KNN, Random Forest, and SVM.

2. Methodology

2.1 Dataset and provided features

The classic German Traffic Sign Recognition Benchmark, or GTSRB, was used for this classification module along with the provided pre-processed data. The training data file consists of 5488 instances in the form of images. The training file contains four additional CSVs: train_metadata.csv, color_histogram.csv, hog_pca.csv, and additional_feature.csv. A testing data file was also provided. This file has fewer instances, 2353 instances only, and doesn't include labels. The testing file also gives us four CVSs like training: test_metadata.csv, color_histogram.csv, hog_pca.csv, and additional_features.csv. image size varies, from 15x15 to 250x250 pixels. There was a total of 43 different classes in the dataset.

In our model, we use only three of the four provided feature CSVs: the meta datasets, color_histogram, and hog_pca.

2.1.1 Features dataset

For each of the training and testing data files, three additional comma-separated values files were given. Some of the models will be using color_histogram.csv and Histogram of Oriented Gradients or HOG_PCA as the training dataset for two of our models. The CSV color_histogram.csv contains L2-normalised frequency counts of pixel intensity in RGB space with no extra scaling. Meanwhile, the HOG_PCA dataset has the top 20 principal components from the original HOG features on a 64x64 grayscale using 9 orientations.

2.2 Data pre-processing

In addition to the provided datasets, a Histogram of Oriented Gradients was utilised for feature extraction. The images were resized to 64x128 pixels to ensure consistency, as the image sizes varied from 15x15 to 250x250 pixels. Next, the resized images were turned into a column vector of HOG feature descriptions and flattened to a 1D array for further usage. The pre-processing resulted in a high-dimensional fattened vector. The dataset resulting from this pre-processing will be called HOG. Normally, the images will be converted to grayscale as well, but color is an important factor for this classification model, so it will be kept colorful. Another reason for keeping the data in color is that a HOG_PCA was given, and we want to explore a higher-dimensional dataset.

2.3 Classifier Model

For each of our models, we will use a cross-validation split of 2 and 5 for a 20% and 50% split between the training and validation sets.

2.3.1 Support Vector Machine (SVM)

The main classifier model used in this project was Support Vector Machine due to its strong performance on high-dimensional data. It was used over three different datasets: color_histogram.csv, HOG_PCA.csv, and HOG.

2.3.2 Random Forest

The random forest model was chosen as it is a more scalable model and can determine

important features, which is not possible with SVM, enhancing the robustness. This model was trained over the HOG dataset.

2.3.3 KNN

Alongside two other more complex models, KNN was used due to its simplicity, not needing many assumptions, and is effective when handling high-dimensional datasets.

3. Result

3.1. Dataset

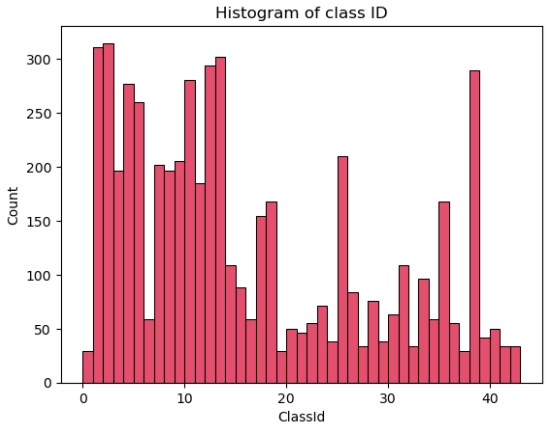

Figure 1- Histogram of the distribution of 43 classes in the training dataset

Figure 2- Image 119, class 0

Figure 3- Image 003, class 3

Figure 4- Image 141, class 39

Figure 5- Image 072, class 12

Figure 6- Image 002, class 13

3.2. SVM

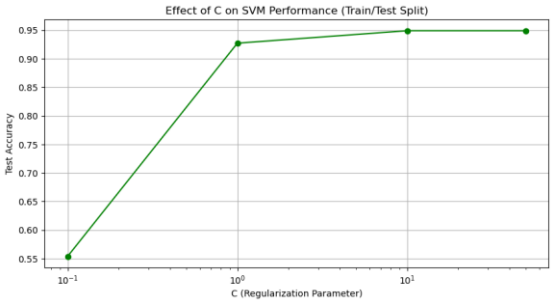

Figure 7-Effect of C on SVM performance on HOG_PCA

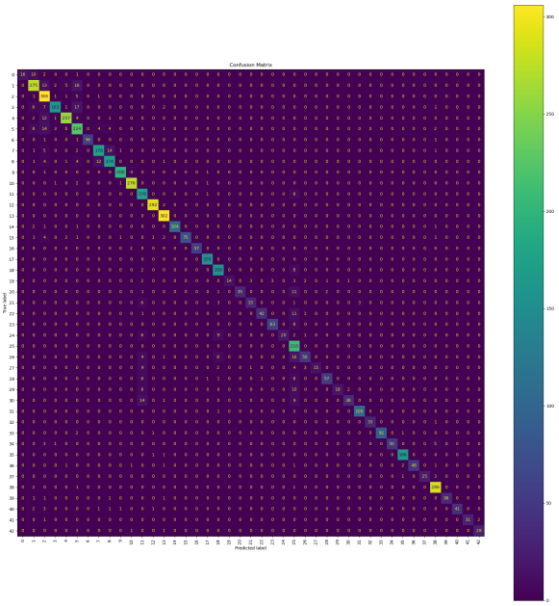

Figure 8- Confusion matrix of SVM on cross-validation with cv=2 and C=10, on HOG dataset

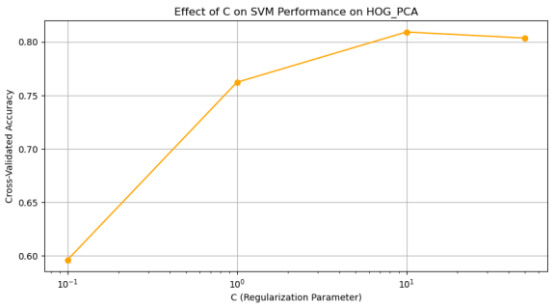

Figure 9- Effect of C on SVM performance on HOG_PCA with scaler

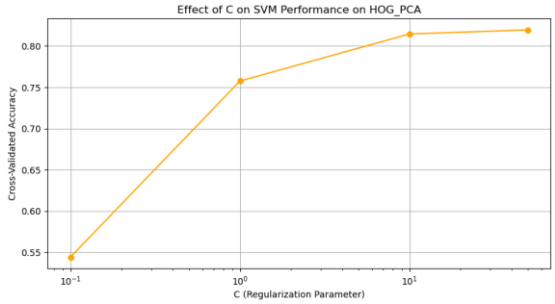

Figure 10-The SVM on HOG_PCA without scaler

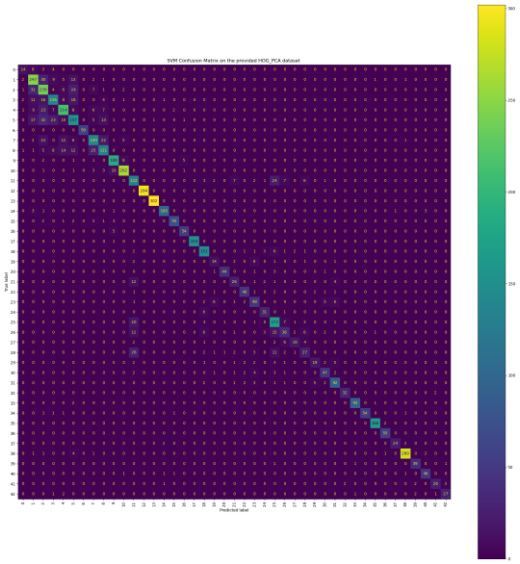

Figure 11- Confusion matrix for SVM on HOG_PCA dataset with C=10 and no scaler

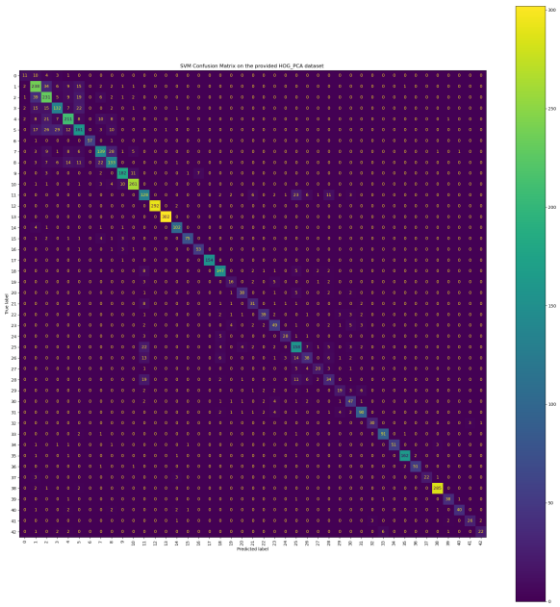

Figure 12- Confusion matrix for SVM on HOG_PCA dataset with C=10

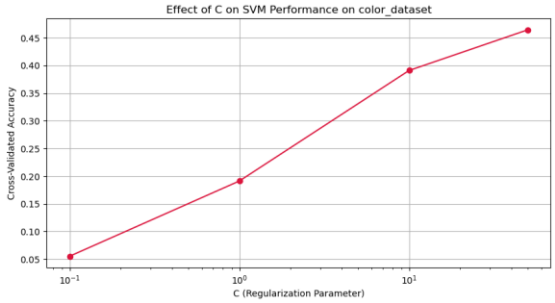

Figure 13- Effect of C on SVM performance on color_histogram

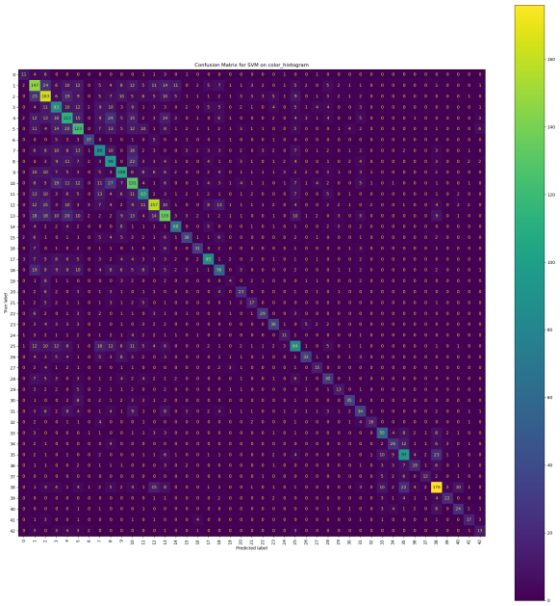

Figure 14- Confusion matrix for SVM on color_histogram with C = 100 and without scaler

	HOG	HOG_PCA	Color_hist
Mean Accuracy	0.908	0.823	0.473
Mean Precision	0.947	0.824	0.510
Mean Recall	0.842	0.794	0.454
F1	0.878	0.804	0.473
Kaggle Accuracy	0.945	0.314	0.621

Table 1- Performance metric of SVM on HOG_HOC_PCA, and color_histogram after

3.3. Random Forest

	HOG	HOG_PCA	Color_hist

Mean Accuracy	0.916	0.723	0.476
Mean Precision	0.946	0.774	0.646
Mean Recall	0.885	0.629	0.411
F1	0.909	0.664	0.472
Kaggle Accuracy	0.936	0.375	0.598

Table 2- Performance of Random Forest on HOG, HOG_PCA, and color_histogram

	HOG_PCA	Color_hist
Mean Accuracy	0.716	0.458
Mean Precision	0.768	0.619
Mean Recall	0.629	0.392
F1	0.664	0.449
Kaggle Accuracy	0.378	0.595

Table 3- Performance of Random Forest on HOG, and color_histogram with default setting

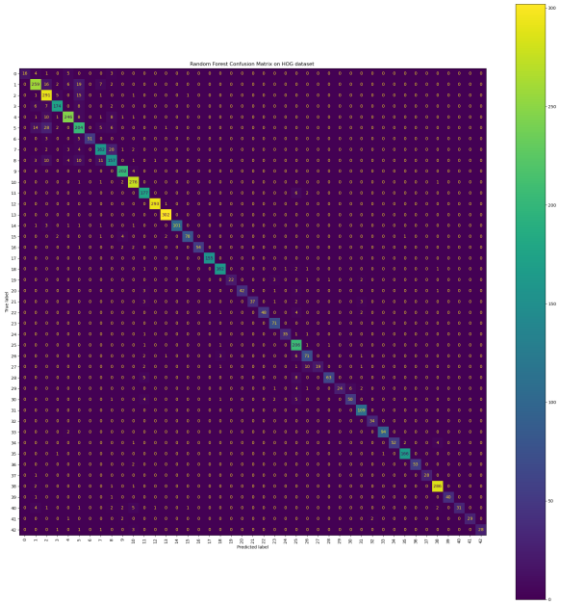

Figure 15- Confusion matrix for Random Forest on HOG dataset.

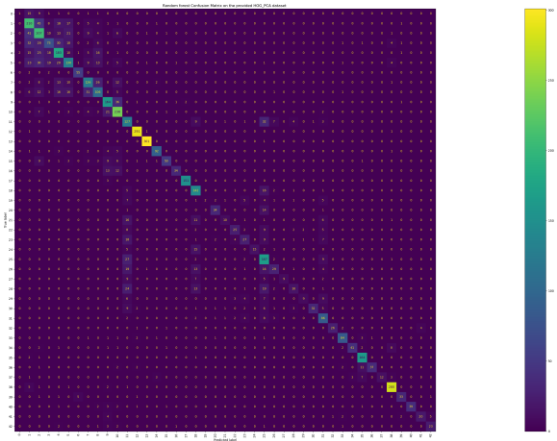

Figure 16- Confusion matrix for random forest on HOG_PCA dataset, default setting

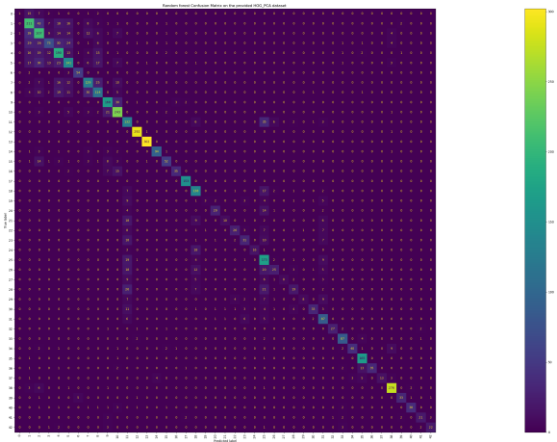

Figure 17- Confusion matrix for random forest on HOG_PCA dataset, new setting

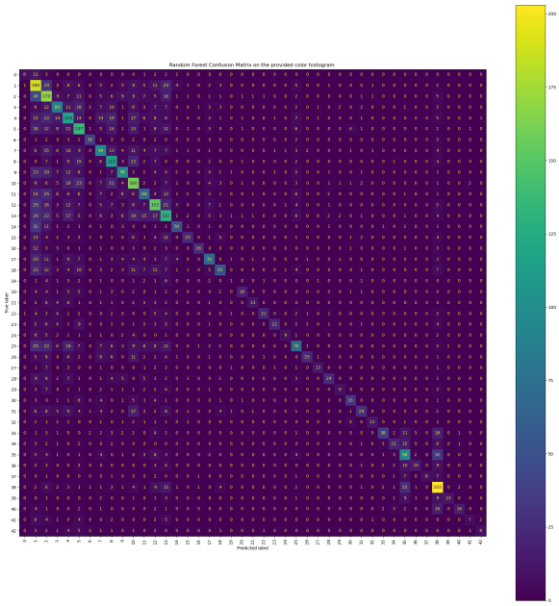

Figure 18- Confusion Matrix for random forest on color_histogram default setting

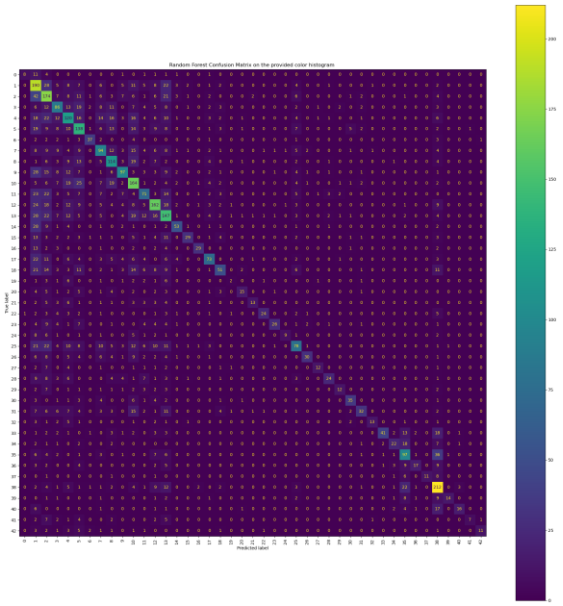

Figure 19- Confusion Matrix for random forest on color_histogram, new setting

Figure 20- Effect of n_neighbors on the accuracy of KNN on HOG_PCA

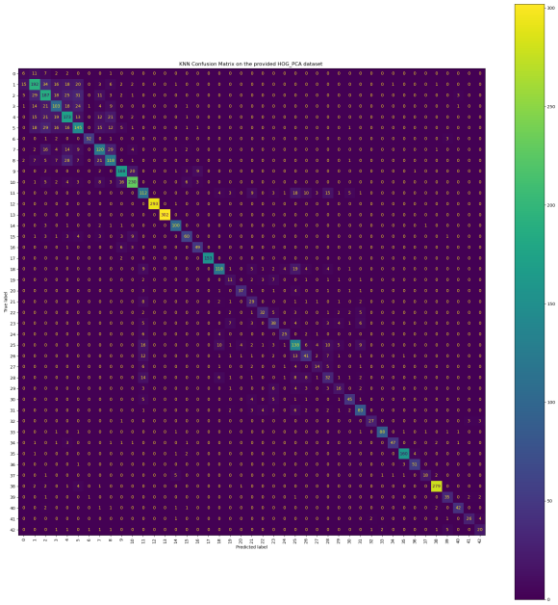

Figure 21- Confusion Matrix for KNN of k = 1 on HOG_PCA

3.4. KNN

	HOG_PCA	Color_hist
Mean Accuracy	0.732	0.476
Mean Precision	0.724	0.504
Mean Recall	0.696	0.469
F1	0.706	0.478
Kaggle Accuracy	0.317	0.616

Table 4- Performance of KNN on two different datasets

Figure 22- Effect of n-neighbours on KNN accuracy on color_histogram

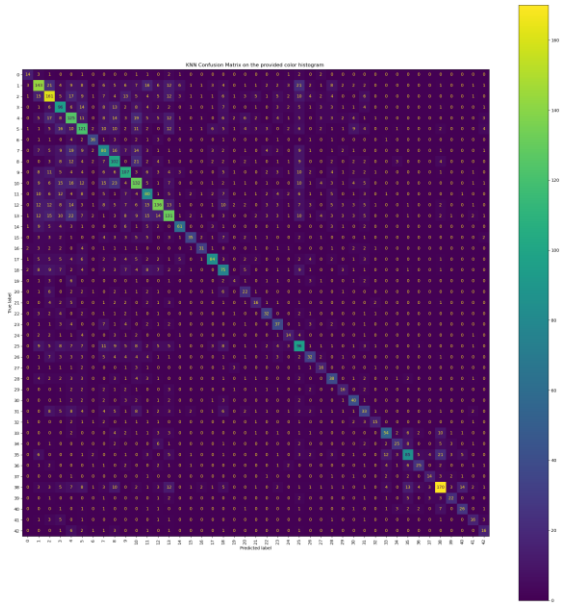

Figure 23- Confusion matrix for KNN with K = 1 on color_histogram

3.5. Overall

	SVM	Random Forest	KNN
Mean Accuracy	0.908	0.916	0.476
Mean Precision	0.947	0.946	0.504
Mean Recall	0.842	0.885	0.469
F1	0.878	0.909	0.478
Kaggle Accuracy	0.945	0.936	0.616

Table 5- Highest cross-validation accuracy on Kaggle, and the precision, recall, and F1-score on that train dataset with cv = 2, rounded to 3 decimal places, and testing result from Kaggle based on 50% of the test data.

4. Discussion and Critical Analysis

Figure 1 shows that the distribution of classes is not balanced. Most of the dataset is in classes 2-6, 8-14, 26, and 30, with 51.17% of the instances belonging to these classes. Surprisingly did not affect the performance of the models as the sample size is large enough. Most of the problems stemmed from the fact that traffic signs are very similar to each other.

4.1. SVM

Support Vector Machine was chosen as the primary algorithm for this project due to its consideration for the relationship between features and suitability for high-dimensional numerical data (Ehinger, 2025).

The hyperparameter C is crucial in controlling the trade-off between correctly classifying training instances and maximising the decision boundary's margin (European Information Technology Certification Institute, 2023). The values of C were selected based on the observed effect of C on SVM performance (Figures 7, 9, 10, and 13). A value of **C =10** was selected for the best accuracy for HOG and HOG_PCA, while **C=100** is needed for the color histogram features.
This model performed exceptionally well on the HOG dataset, achieving over **90%** accuracy on the Kaggle test set (Table 1). However, when applied to HOG_PCA, even with tuning, it achieved high accuracy during cross-validation but **performed poorly** on the Kaggle one. This discrepancy is likely due to the discarding of important information during dimensionality reduction. Additionally, the SVM did not perform well on the colored feature either. This is likely to be caused by the similarity between classes from 1 to 12. These classes shared the same color palette of red trim on a white background, and a black number or image in the middle.

4.2. Random Forest

Another classification model was **Random Forest**. While it was not the best choice, it offers a reduction from overfitting through averaging predictions from multiple decision trees, providing good generalization (Shafl, 2024). As shown in Table 2, the default Scikit-learn Random Forest gave **90.9% accuracy** and **94.6% precision** on HOG features. The high precision was important for the safety of

drivers, as missing a sign can cause a fatal accident. Although the recall was lower at 88.5%, the model remained reliable for traffic sign detection.

On the other hand, performance dropped on color histogram and HOG_PCA (Table 3. Hyperparameter tuning was used with min_samples_leaf = 1, min_samples_split = 2, and n_estimators = 200. This adjustment had significant increase Kaggle accuracy for HOG_PCA minimal effect on color_histogram (Table 2, Table 3). But overall, the tuning offers limited improvement for these two datasets.

4.3. KNN

KNN is one of the simplest classification algorithms to implement. However, it is highly susceptible to noise and performs poorly in high-dimensional space (Ehinger, 2025). This issue was evident in the confusion matrices (Figure 22 and Figure 24), which show that even with a k = 1, they failed to accurately classify classes 1-6 for HOG_PCA and 1-13 with color_histogram. It was suspected that the reason for this inaccuracy was the similarity between the classes (Figures 2, 3, and 6). The low accuracy, especially when testing (Table 4), was also due to the complexity of the distribution of the two datasets used: HOG_PCA has 19 attributes, while the color histogram contains 95. Having this many attributes in KNN can lead to the **curse of dimensionality**, making it harder for KNN to find any meaningful cluster or pattern in the data (Cornell University, 2018).

In theory, PCA could have helped with the reduction of dimension for this data. However, even with the usage of HOG_PCA, the nature of the dataset is too much for such a simple model. The choice of the default Euclidean distance did not affect this, as this was a problem of the model itself.

4.4. Model Comparison

Among the tested models, SVM achieved the best performance in both accuracy and precision, especially with the HOG dataset. Random Forest also showed good precision but struggled with less specific data. KNN performance was limited by the curse of dimensionality and a high percentage of noise in the data, despite the reduction effort

5. Conclusions

Overall, this study has successfully explored multiple machine learning approaches for traffic sign classification using diverse feature sets. The SVM model performed best while challenges like class similarity and high dimensionality affected others (Table 5). Any future work should address this limitation and provide methods for better accuracy.

6. References

Adam Shafl, 2024. *Random Forest Classification with Scikit-Learn* https://www.datacamp.com/tutorial/random-forests-classifier-python .

Conrnell University, 2018. *Lecture 2: k-nearest neighbors* https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html

European Information Technology Certification Institute, 2023. *What is the purpose of the C parameter in SVM? How does a smaller value of C affect the margin and misclassifications?* https://eitca.org/artificial-intelligence/eitc-ai-mlp-machine-learning-with-python/support-vector-machine/svm-parameters/examination-review-svm-parameters/what-is-the-purpose-of-the-c-parameter-in-svm-how-does-a-smaller-value-of-c-affect-the-margin-and-misclassifications/

Geeks for Geeks, 2025. K-*Nearest Neighbor(KNN) Algorithm.* https://www.geeksforgeeks.org/k-nearest-neighbours/

Geeks for Geeks, 2025. *How to choose the right distance metric in KNN?* https://www.geeksforgeeks.org/how-to-choose-the-right-distance-metric-in-knn/

Geeks for Geeks, 2025. *Random Forest Regression in Python.* *https://www.geeksforgeeks.org/random-forest-regression-in-python/*

The details of the theory were covered in the slides for Machine Learning COMP20027 (Kris Ehinger, personal communication, 3 April 2025)