# **Assignment 1 – Classification using Scikit-learn**

**Student Name**: Ya Li Lin (Kelly) **Student ID**: 22721291 **Programme**: 4BCT

## **Algorithm 1 – Logistic Regression**

Logistic Regression is a well-established supervised learning algorithm used for binaryclassification problems [1]. Rather than predicting continuous values, it estimates the probability that something belongs to one of two possible classes, such as “fire” or “no fire” [2]. It achieves this by using a mathematical function that converts predictions into probabilities between 0 and 1. Because of its simplicity and clear interpretability, Logistic Regression is often used as a starting point before applying more complex machine learning techniques [2], [4].

**Detailed Description of Algorithm 1.**

Logistic Regression works by combining several input features into a single weighted sum, then passing that sum through a sigmoidfunction:

The sigmoid curve compresses all values into a range between 0 and 1, allowing them to be interpreted as probabilities [2], [4]. Predictions with probabilities of 0.5 or higher are classified as “fire”, while those below 0.5 are labelled “no fire” [3]. The model adjusts its weights to minimise error during training and often includes regularisation to prevent overfitting, which happens when a model fits the training data too closely [2]. Logistic Regression is fast, easy to train, and helps identify which input features are most influential. However, it assumes a linearrelationship between variables and cannot easily capture complex, non-linear interactions [5].

A screenshot of a graph

AI-generated content may be incorrect.

*Figure 1. Logistic Regression decision boundary showing a linear separation between two classes [1].*

**Why I choose this algorithm.**

I chose Logistic Regression because it’s straightforward, transparent, and provides an excellent baseline for comparison [1]. It lets me see which environmental factors, such as humidity or temperature, contribute most to wildfire risk. The model is easy to interpret, quick to train, and requires minimal computational power. Using Logistic Regression first also makes it easier to understand how a linear model performs before moving on to more complex approaches like Support Vector Machines (SVMs). This helps build up the analysis step by step [6].

**Hyperparameter Details for Tuning.**

***C****:* Inverse of regularisation strength. A small C applies stronger regularisation, simplifying the model but risking underfitting. A large C reduces regularisation, fitting more closely to training data but with a risk of overfitting [3], [4].

***Penalty****:* Determines the type of regularisation. L2 is most common, while “none” removes regularisation entirely [3].

## **Algorithm 2 – Support Vector Machines (SVM) with RBF Kernel**

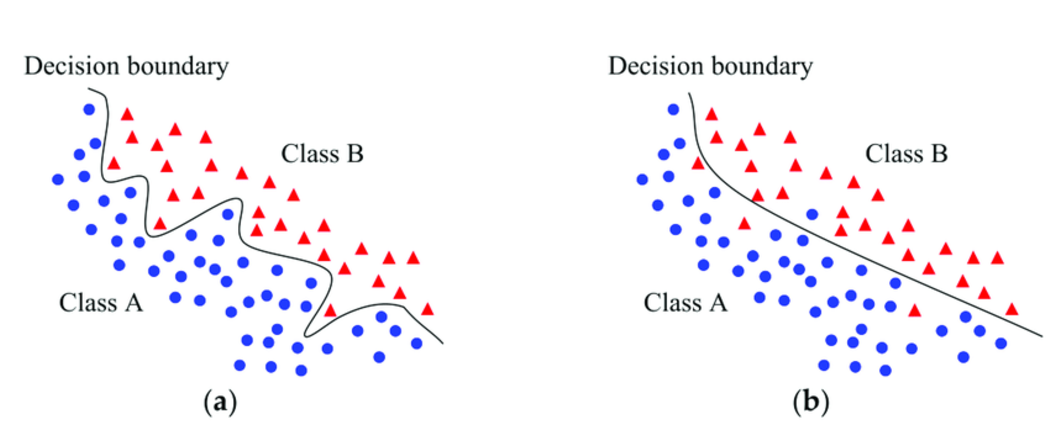
Support Vector Machines (SVMs) are supervised learning algorithms that classify data by drawing the best possible boundary between two classes [1]. The idea is to maximise the margin between data points on either side of the line, giving the model more confidence in its predictions. When data cannot be separated with a straight line, an RBFkernel can be applied to create a flexible, curved boundary. This makes SVMs particularly effective for capturing complex, non-linear relationships [2], [4].

**Detailed Description of Algorithm 2.**

SVMs work by identifying supportvectors, which are the key data points closest to the decision boundary [1]. The algorithm finds the boundary that separates the classes with the widest possible margin, which usually leads to better performance on unseen data. The RadialBasis Function (RBF) kernel allows SVMs to model curved, non-linear boundaries by transforming the data into a higher-dimensional space. This transformation is achieved through the kernel trick, calculated using:

*Figure 2. Radial Basis Function (RBF) kernel used in Support Vector Machines [4].*

Here, the parameter γ (gamma) controls how tightly the boundary curves around the data [2]. SVMs often achieve high accuracy but can take longer to train and are less interpretable than simpler models such as Logistic Regression [3], [4].



*Figure 3. SVM with RBF kernel showing non-linear decision boundary separating two classes [9].*

**Why I choose this algorithm.**

I selected the SVM with an RBF kernel because it handles non-linear and overlapping data very well [1], [2]. Wildfire patterns are unlikely to follow simple, straight-line relationships, so the flexibility of the RBF kernel is a major advantage. SVMs are also quite robust to outliers, which is useful for environmental data where some measurements may be noisy. Comparing the SVM with Logistic Regression allows me to explore both a linear and a non-linear approach, highlighting which method better captures patterns in the dataset [4].

**Hyperparameter Details for Tuning.**

***C****:* Balances margin width and accuracy. Higher C values fit the training data more closely, while lower C values allow a softer margin [2], [3].

***Gamma (γ):*** Defines how far the influence of a single point extends. A small γ gives a smooth, simple boundary; a large γ makes it complex [3], [4].

**Algorithm 1 - Logistic Regression - Model Training and Evaluation**

[TO DO: Complete implementation and testing]

**Data Preprocessing and Visualisation**

[TO DO: Load wildfire\_training.csv dataset]

[TO DO: Explore data distribution and class balance]

[TO DO: Handle any missing values or outliers]

[TO DO: Create visualization of feature distributions]

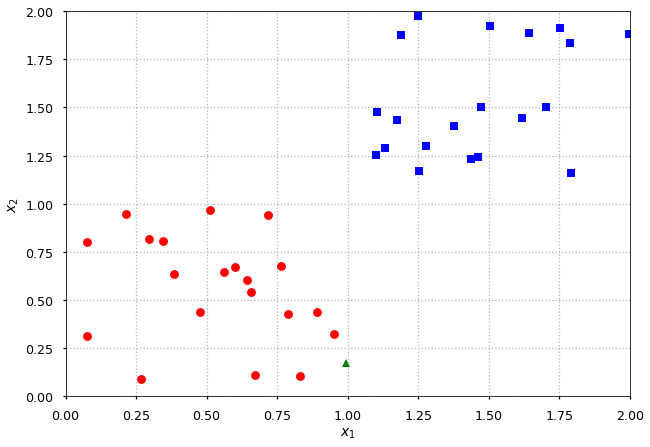
[TO DO: Generate correlation matrix for features]

Figure : Visualisation of the dataset before training

**Training and Evaluation Details**

[TO DO: Train Logistic Regression with default hyperparameters (C=1.0, penalty='l2')]

[TO DO: Record training accuracy and test accuracy]

[TO DO: Experiment with C values: [0.001, 0.01, 0.1, 1, 10, 100, 1000]]

[TO DO: Experiment with penalty values: ['l1', 'l2', 'none']]

[TO DO: Create table/graph showing accuracy vs hyperparameter values]

[TO DO: Generate confusion matrix and classification report]

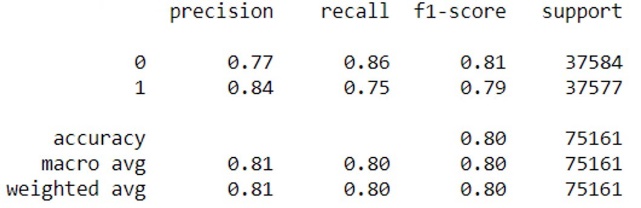
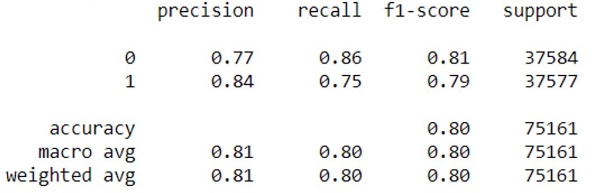


Figure : Summary of Results Achieved from Training and Testing

**Discussion of results**

[TO DO: Analyze which hyperparameter values performed best]

[TO DO: Discuss any patterns observed in the results]

[TO DO: Identify potential overfitting or underfitting]

**Algorithm 2 - Support Vector Machines (SVM) with RBF Kernel - Model Training and Evaluation**

[TO DO: Complete implementation and testing of SVM with RBF kernel]

**Data Preprocessing and Visualisation**

[TO DO: Load wildfire\_training.csv and wildfire\_test.csv datasets]

[TO DO: Check for missing values and handle if necessary]

[TO DO: Apply feature scaling/standardization - CRITICAL for SVM performance]

[TO DO: Note: SVMs are sensitive to feature scales, so StandardScaler or MinMaxScaler is essential]

[TO DO: Create visualization of scaled feature distributions]

[TO DO: Generate 2D scatter plots of key feature pairs colored by fire/no fire class]

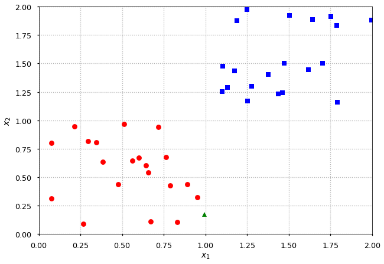
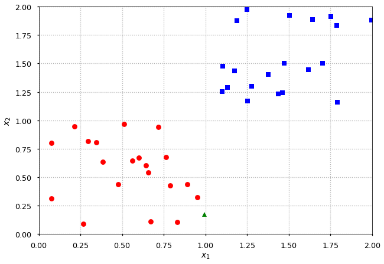
[TO DO: Document any data preprocessing steps applied]

Figure : Visualisation of the dataset before and after pre-processing was applied

**Training and Evaluation Details**

[TO DO: Train SVM with default hyperparameters (C=1.0, gamma='scale', kernel='rbf')]

[TO DO: Record training accuracy and test accuracy with default settings]

[TO DO: Experiment with C values: [0.1, 1, 10, 100, 1000]]

[TO DO: Experiment with gamma values: [0.001, 0.01, 0.1, 1, 10, 'scale', 'auto']]

[TO DO: Create grid showing accuracy for different C and gamma combinations]

[TO DO: Generate confusion matrix for best performing hyperparameter combination]

[TO DO: Create classification report showing precision, recall, F1-score]

[TO DO: Record training time for different hyperparameter values]

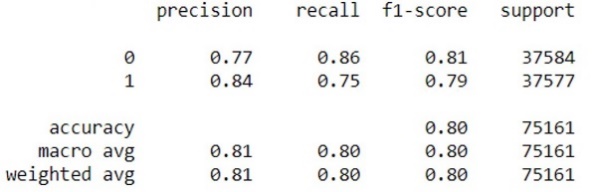
[TO DO: Note any convergence warnings or issues]

Figure : Summary of Results Achieved from Training and Testing

**Discussion of results**

[TO DO: Analyze which C and gamma combination performed best on test data]

[TO DO: Discuss trade-off between training accuracy and test accuracy]

[TO DO: Identify signs of overfitting (high training accuracy, lower test accuracy)]

[TO DO: Identify signs of underfitting (low accuracy on both training and test)]

[TO DO: Compare performance with very small gamma (smooth boundary) vs large gamma (complex boundary)]

[TO DO: Discuss impact of C parameter on margin width and support vector count]

[TO DO: Analyze any patterns in misclassified instances]

[TO DO: Consider computational cost vs accuracy trade-offs]

**Conclusions**

[TO DO: Write summary paragraph introducing the conclusions section]

[TO DO: Briefly restate the goal of comparing Logistic Regression and SVM for wildfire prediction].

**Key Findings**

[TO DO: State which algorithm achieved highest test accuracy overall]

[TO DO: Identify best hyperparameter combination for Logistic Regression (e.g., C=10, penalty='l2')]

[TO DO: Identify best hyperparameter combination for SVM (e.g., C=100, gamma=0.01)]

[TO DO: Note which algorithm was more sensitive to hyperparameter changes]

[TO DO: Discuss any significant differences in training vs test accuracy (overfitting indicators)]

[TO DO: Mention computational efficiency differences between the two algorithms]

[TO DO: Identify which features were most important for wildfire prediction (if applicable)]

**Comparative Analysis of Algorithm Performances**

[TO DO: Create comparison table showing:]

- Training accuracy: Logistic Regression vs SVM (default settings)

- Test accuracy: Logistic Regression vs SVM (default settings)

- Training accuracy: Logistic Regression vs SVM (optimized settings)

- Test accuracy: Logistic Regression vs SVM (optimized settings)

- Training time comparison

[TO DO: Create bar chart or line graph comparing performance metrics]

[TO DO: Discuss why one algorithm outperformed the other]

[TO DO: Analyze whether the non-linear SVM boundary provided advantage over linear Logistic Regression]

[TO DO: Discuss practical implications - which algorithm would you deploy in production and why?]

[TO DO: Note any trade-offs between accuracy, interpretability, and computational cost]

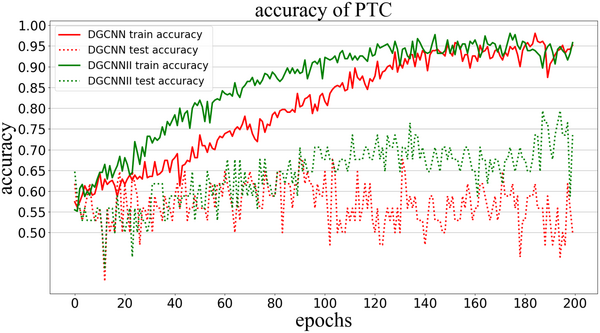
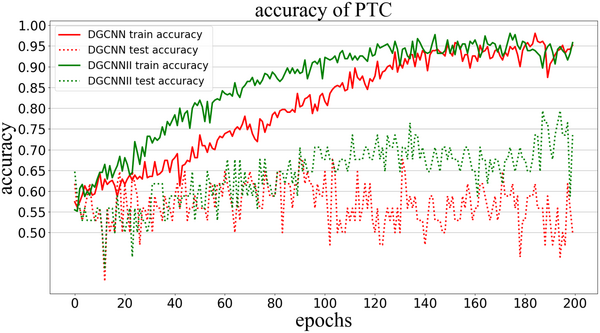
[TO DO: Remove placeholder Figure 7 or replace with actual comparison graph] 

Figure : Graphed comparison of results from the two algorithms

**Recommended Hyperparameter Valued based on Results**

*Algo 1*

*[TO DO: C = [insert optimal value] - justify why this value balanced bias-variance trade-off]*

*[TO DO: Penalty = [insert optimal value] - explain why this regularization type worked best]*

*Algo 2*

*[TO DO: C = [insert optimal value] - justify the margin width vs accuracy trade-off]*

*[TO DO: Gamma = [insert optimal value] - explain the boundary complexity chosen*

**Concluding Remarks**

[TO DO: Summarize main takeaway from the comparison]

[TO DO: State whether linear or non-linear approach was more suitable for wildfire data]

[TO DO: Mention limitations of the study (e.g., only tested 2 algorithms, limited hyperparameter ranges)]

[TO DO: Suggest future work - other algorithms to try, additional preprocessing techniques, feature engineering]

[TO DO: Final statement on the practical value of the findings for wildfire prediction]

**References**

1. [1] CT4101 Module Notes, “Classification,” School of Computer Science, University of Galway, 2025.
2. [2] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
3. [3] scikit-learn, “Logistic Regression,” [Online]. Available: <https://scikit-learn.org/stable/modules/linear_model.html>
4. [4] scikit-learn, “SVM with RBF kernel parameters,” [Online]. Available: <https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html>
5. [5] GeeksforGeeks, “Understanding Logistic Regression,” [Online]. Available: <https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/>
6. [6] D. W. Hosmer and S. Lemeshow, *Applied Logistic Regression*. Wiley, 2nd ed., 2000.
7. [7] GeeksforGeeks, “RBF SVM Parameters in scikit-learn,” [Online]. Available: <https://www.geeksforgeeks.org/python/rbf-svm-parameters-in-scikit-learn/>
8. [8] S. Mallat, “A tutorial on support vector machines for pattern recognition,” École Normale Supérieure, 1999. [Online]. Available: <https://www.di.ens.fr/~mallat/papiers/svmtutorial.pdf>
9. [9] FreeCodeCamp, "SVM Kernels – How to Tackle Nonlinear Data in Machine Learning," [Online]. Available: <https://www.freecodecamp.org/news/svm-kernels-how-to-tackle-nonlinear-data-in-machine-learning/>

**Report Update Log**

|  |  |  |
| --- | --- | --- |
| Date | Update Description | Time Spent |
| 20/09/25 | Researched Logistic Regression theory and applications | 3 hours |
| 23/09/25 | Wrote detailed description of Algorithm | 2 hours |
| 25/09/25 | Researched SVM with RBF kernel | 2 hours |
| 28/09/25 | Completed hyperparameter descriptions | 50 minutes |
| 29/09/25 | Structured TO DO sections for implementation | 2.5 hours |
| 30/09/25 | Reviewed scikit-learn documentation | 2 hours |
| 01/10/25 | Created report template structure | 1.5 hours |
| 04/10/25 | Finalized algorithm descriptions and references | 2 hours |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |