

Glass Classification Using Machine Learning

Tuan Anh Le
Matrikelnummer: 1586277

Table of Contents

I.	Abstract.....	2
II.	Introduction.....	2
III.	Methodology	2
1.	Utilizing python libraries.....	2
IV.	Dataset	2
1.	Data Preprocessing	2
V.	UML Diagrams	3
VI.	Classification Model.....	4
1.	Algorithm Selection	4
2.	Hyperparameter Tuning	5
3.	Ensemble Approach	5
VII.	Evaluation Metrics.....	6
1.	Before implementing majority vote:.....	6
a)	Evaluating Individual Models:.....	6
2.	After implementing majority vote:	6
a)	Results.....	7
b)	Discussion of Results	8
VIII.	User Interface	8
IX.	Discussion and Future Work	9
1.	Challenges.....	9
2.	Future Enhancements	9
X.	Conclusion	9
XI.	References.....	9

I. Abstract

This project creates a machine learning-based system to classify different types of glass based on their chemical composition. From the UGI Glass Identification dataset, the model was trained using Support Vector Machines (SVM). To improve the accuracy of the model, a method which implements voting by majority is used. This project also provides performance evaluation and a simple Graphical User Interface (GUI).

II. Introduction

A Support Vector Machine (SVM) is a powerful machine learning algorithm widely used for both linear and nonlinear classification with kernel trick. SVMs are particularly effective because they focus on finding the **maximum separating hyperplane** between the different classes in the target feature.

III. Methodology

1. Utilizing python libraries

This project uses the an algorithm from sklearn.svm, C-Support Vector Classification (SVC).

There are two parameters in SVC that should be taken into account:

- C: float, default=1.0 - Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive.
- Kernel: {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf' - Specifies the kernel type to be used in the algorithm.
- Gamma: {'scale', 'auto'} or float, default='scale' - Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

IV. Dataset

The UCI Glass Identification dataset was used, containing 214 samples, each has 9 attributes: RI (Reflection Index), Na, Mg, Al, Si, K, Ca, Ba, Fe; representing the chemical composition of glass.

This dataset has 6 labels: 1, 2, 3, 5, 6, 7.

1. Data Preprocessing

a) Loading and Inspection:

- Data was extracted from the csv file.

b) **Normalization:**

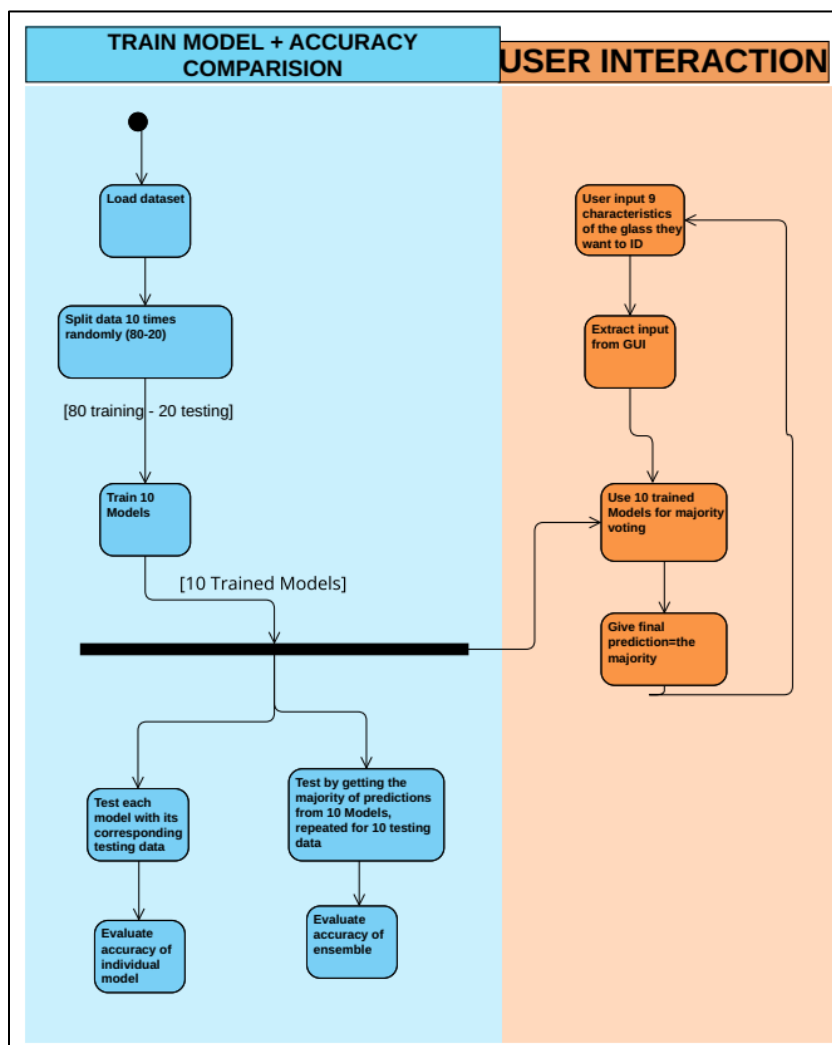
- Features were normalized using StandardScaler to ensure uniform scaling. (this is done inside training function)

c) **Train-Test Split:**

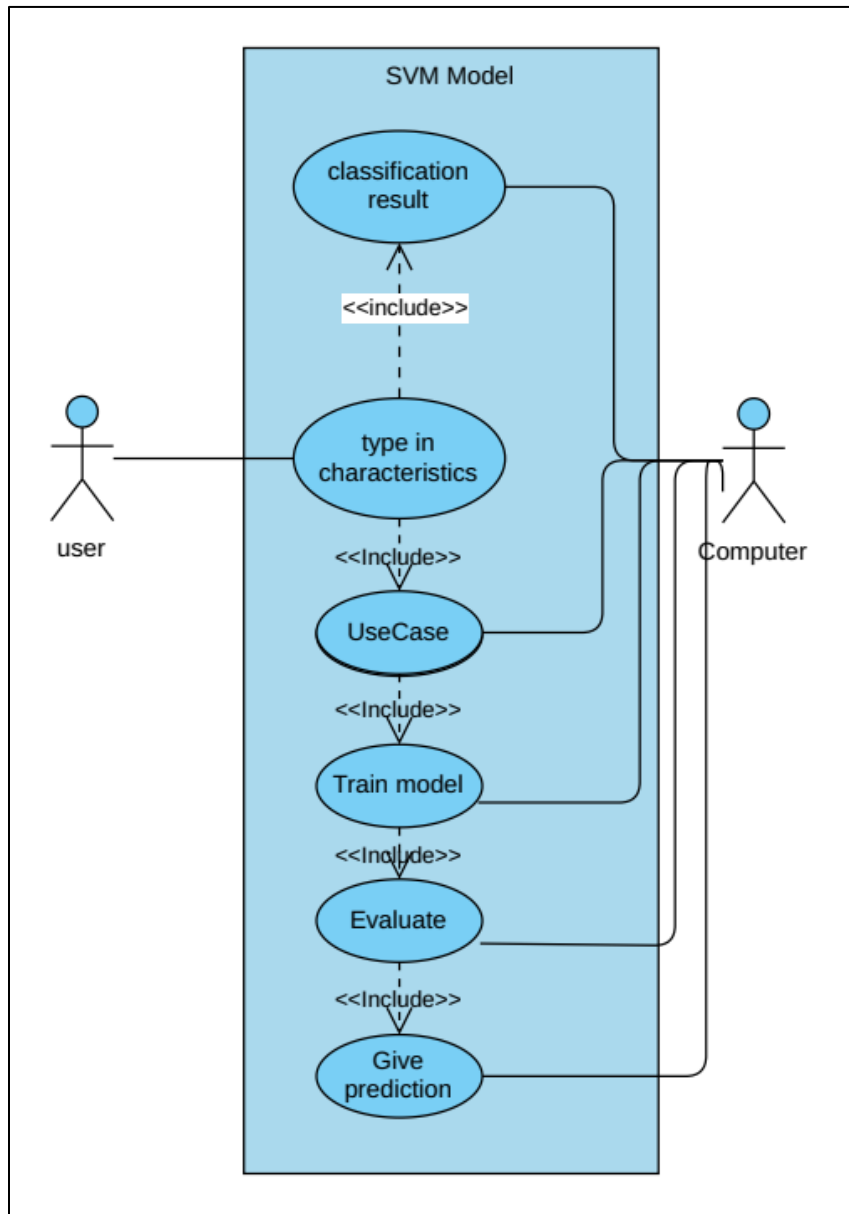
- The dataset was split into 80% training and 20% testing ratio, maintaining class proportions using stratification.

V. UML Diagrams

- **Activity Diagram:**



- Use case diagram:



VI. Classification Model

1. Algorithm Selection

- The SVM model with a radial basis function (RBF) kernel was chosen for its ability to handle non-linear relationships in the data.

2. Hyperparameter Tuning

- Through a manual process of trials and errors, the two optimal parameters were found by:
 - C (regularization parameter) tested with values [1, 10, 20, 30, ..., 100].
 - Gamma (kernel coefficient) tested with values [1, 0.9, 0.8, ..., 0.1].
- ➔ Optimal parameters: C=100, gamma=0.1

3. Ensemble Approach

- Nevertheless, the accuracy of the SVM model alone is found to be quite low, around 70%. In the effort of increasing the accuracy, an approach by ensembling was implemented, inspired by Bootstrap Aggregation (bagging). With bagging multiple models are created from subsets of dataset, then the final decision is found by Random Forest Tree. However, the method is too complicated for our task.
- Therefore, I came up with the idea of using *majority voting*. In this project, 10 SVMs were created by splitting the dataset (with ratio 80-20) 10 times. Each SVM is found to have accuracy of around 70%. Now the process of majority voting comes into play. For each testing data batch, all 10 SVMs are involved in predicting the label; then the majority of the predictions is taken as the final prediction. This way the accuracy is improved to 95%.

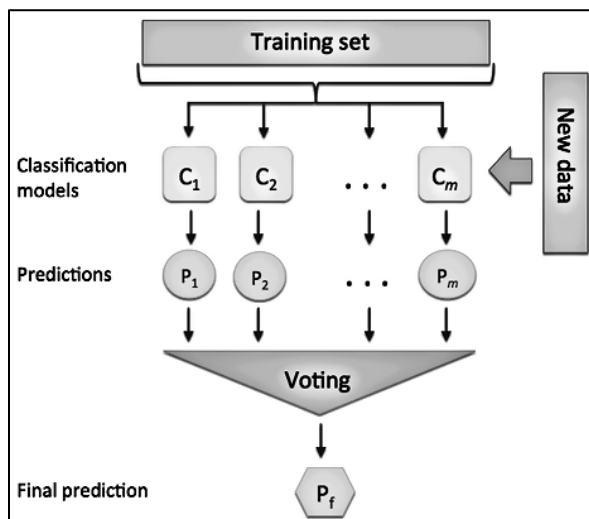


Figure: graphical representation of majority voting

- For instance, Evaluating Test Data from Model 1 (test data from first data split, this batch is used to train Model 1)

Model 1: Predicted [1 3 2 2 1 1 1 5 2 2 1 5 7 5 2 1 2 3 1 2 2 7 1 2 7 2 2 2 2 1 6 2 7 2 1 7 3 5 2 2 1 2 1]

Model 2: Predicted [3 1 2 2 1 1 1 6 2 2 1 5 7 5 2 1 3 1 1 5 1 7 1 2 7 2 2 7 2 1 6 2 7 3 1 7 2 2 2 2 1 1 1]

Model 3: Predicted [3 3 2 2 3 1 1 6 2 2 1 5 7 5 2 1 3 1 1 7 1 7 1 2 7 2 2 7 2 1 6 2 7 2 1 7 2 2 2 2 1 1 1]

Model 4: Predicted [3 3 2 2 3 1 1 5 2 2 1 2 7 5 2 1 3 1 1 7 1 7 1 2 7 2 2 2 2 1 6 2 7 2 1 7 2 2 2 2 1 1 1]

Model 5: Predicted [3 1 2 2 3 1 1 6 2 2 1 5 7 5 2 1 3 1 1 5 1 7 1 2 7 2 2 7 2 1 6 2 7 2 1 7 2 2 2 2 1 1 1]

Model 6: Predicted [3 1 2 2 3 1 1 6 2 2 1 2 7 5 2 1 3 1 1 5 1 7 2 2 7 2 2 7 2 1 6 2 7 2 1 7 1 2 2 2 1 1 1]

Model 7: Predicted [3 1 2 2 3 1 1 6 2 2 1 5 7 5 2 1 2 1 2 7 2 7 1 2 7 2 2 7 2 1 6 2 7 2 1 7 2 2 2 2 1 2 1]

Model 8: Predicted [3 1 2 2 3 1 1 6 2 2 1 5 7 5 2 1 3 1 1 5 1 7 1 2 7 2 2 2 2 1 6 2 7 2 1 7 1 2 2 2 1 1 1]

Model 9: Predicted [3 1 2 2 3 1 1 6 2 2 1 5 7 5 2 2 3 1 1 7 1 7 1 2 7 2 2 2 2 1 6 2 7 2 1 7 2 2 2 2 1 2 1]

Model 10: Predicted [3 1 2 2 3 1 1 6 2 2 1 5 7 5 2 1 3 1 1 5 1 7 1 2 7 2 2 7 2 1 6 2 7 2 1 7 2 2 2 2 1 1 1]

True Labels Test 1: [3, 1, 2, 2, 3, 1, 1, 6, 2, 2, 1, 5, 7, 5, 2, 1, 3, 1, 1, 5, 1, 7, 1, 2, 7, 2, 2, 7, 2, 1, 6, 2, 7, 2, 1, 7, 2, 2, 2, 1, 1, 1]

Final Voted Predict: [3 1 2 2 3 1 1 6 2 2 1 5 7 5 2 1 3 1 1 5 1 7 1 2 7 2 2 7 2 1 6 2 7 2 1 7 2 2 2 2 1 1 1]

- ➔ It can be seen that for the first test, if there was no majority voting, model 1 would have guessed the incorrect label (1 instead of 3)

VII. Evaluation Metrics

- The metrics are calculated by using `accuracy_score`, `classification_report`, `confusion_matrix` from `sklearn.metrics`.

1. Before implementing majority vote:

a) Evaluating Individual Models:

- Model 1 Accuracy: 72.09%
- Model 2 Accuracy: 69.77%
- Model 3 Accuracy: 74.42%
- Model 4 Accuracy: 62.79%
- Model 5 Accuracy: 67.44%
- Model 6 Accuracy: 74.42%
- Model 7 Accuracy: 60.47%
- Model 8 Accuracy: 72.09%
- Model 9 Accuracy: 74.42%
- Model 10 Accuracy: 72.09%

2. After implementing majority vote:

- Accuracy:** Measures overall correct predictions.
- Precision:** Evaluates false positive rate.

- **Recall:** Indicates true positive rate.
- **F1-Score:** Harmonic mean of precision and recall.
- **Confusion Matrix:** Provides detailed insights into class-specific predictions.

a) Results

The model achieved the following metrics:

- **Accuracy:** 95.35%
- **Precision:** 96%
- **Recall:** 95%
- **F1-Score:** 95%

Classification Report Table

Class	Precision	Recall	F1-Score	Support
1	0.92	0.96	0.94	140
2	0.98	0.91	0.94	150
3	0.86	1.00	0.92	30
5	1.00	1.00	1.00	30
6	1.00	1.00	1.00	20
7	1.00	1.00	1.00	60

Confusion Matrix Table

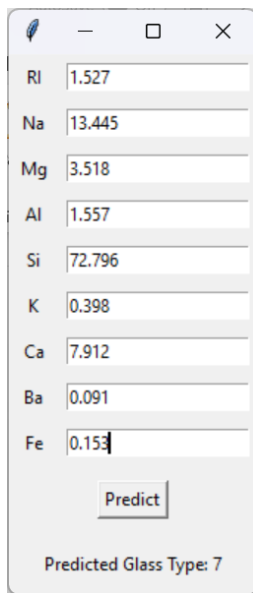
Actual \ Predicted	Class 1	Class 2	Class 3	Class 5	Class 6	Class 7
Class 1	134	3	3	0	0	0
Class 2	12	136	2	0	0	0
Class 3	0	0	30	0	0	0
Class 5	0	0	0	30	0	0
Class 6	0	0	0	0	20	0
Class 7	0	0	0	0	0	60

b) Discussion of Results

- Most classes were predicted accurately, with minor misclassifications between Classes 1 and 2.
- Smaller classes (e.g., Class 6) showed perfect classification, likely due to their distinct chemical compositions.

VIII. User Interface

- A graphical user interface (GUI) was developed using tkinter. The interface allows users to input chemical compositions and receive predictions with confidence scores. The GUI scales inputs using pre-fitted scalers for consistency.
- User's input is extracted, for example:



The image shows a tkinter GUI window with a title bar containing a feather icon, a minus sign, a square icon, and a close button. The window contains ten input fields, each with a label and a value: RI (1.527), Na (13.445), Mg (3.518), Al (1.557), Si (72.796), K (0.398), Ca (7.912), Ba (0.091), and Fe (0.153). Below the input fields is a 'Predict' button. At the bottom of the window, it displays 'Predicted Glass Type: 7'.

Figure: manually inputting to GUI window

Inputs: [[1.527, 13.445, 3.518, 1.557, 72.796, 0.398, 7.912, 0.091, 0.153]]

Model 1: Prediction 7

Model 2: Prediction 7

Model 3: Prediction 7

Model 4: Prediction 7

Model 5: Prediction 7

Model 6: Prediction 2

Model 7: Prediction 7

Model 8: Prediction 2

Model 9: Prediction 2

Model 10: Prediction 2

➔ Depending on different inputs, there could be different predictions between each models.

IX. Discussion and Future Work

1. Challenges

- **Imbalanced Dataset:** The smaller number of samples in some classes required careful handling to avoid biases.
- **Parameter Sensitivity:** Hyperparameter tuning was computationally intensive.

2. Future Enhancements

- Incorporate additional classifiers like Neural Networks or Random Forests.
- Address data imbalance using techniques like SMOTE.
- Extend the GUI to display model evaluation metrics and provide visualization of feature importance.

X. Conclusion

This study demonstrates the effectiveness of SVMs for glass classification, achieving higher accuracy by majority voting. This was clearly illustrated in example in section V.3. However, sometimes the voting can cause the prediction from right to wrong, but overall they are very extreme cases.

XI. References

[Support Vector Machine \(SVM\) Algorithm - GeeksforGeeks](#)

[chatgpt.com](#)

[SVC — scikit-learn 1.6.1 documentation](#)

[UML Activity Diagram Tutorial | Lucidchart](#)