UBC Master of Data Science

DATA 572: Supervised Machine Learning

Project Report

February 7, 2024

Kiran John

Jason Samuel Suwito

Abstract

The introduction of machine learning models has reshaped the landscape of many industries, assisting in tasks such as process optimization, predictions, and classification. In recent years, milk producers have taken an interest in how they can effectively leverage classification-based models to determine the quality of their milk. In this project, we explore the use of several machine learning methods to classify milk into three grades: low (bad), medium (moderate), or high (good). The dataset we use is a collection of 1059 manually collected observations of milk associated with seven key features: pH, Temperature, Taste, Odor, Fat, Turbidity, and Colour with the corresponding quality grade. With our analysis, we compare four supervised learning methods and one unsupervised learning method to evaluate which model can predict milk quality with the highest degree of accuracy. Our best-performing method provides a gateway for machine learning in the milk industry, providing a tool to reliably monitor and improve milk production quality. The analysis in this report is accessible in https://github.com/JKiran4/Data_572_Project.

1 Introduction

Advances in artificial intelligence and machine learning have revolutionized the way people see and interact with the world, offering capabilities to improve any industry. Classification is one of the most prevalent uses of machine learning, allowing for categorizing data into pre-defined labels. Classification models can capture complex patterns in data, achieve high efficiency in processing large datasets, and contain significantly less bias compared to human actors. It has enabled innovations in many areas which include fraud detection, medical diagnosis, and malware detection. Despite its widespread use, there are still industries that are only just beginning to adopt machine learning. For the milk industry, producers are seeking methods to classify the quality of their milk to improve production.

In our analysis, we aim to answer one key question: **Out of four supervised learning methods and one unsupervised learning method, which classification model can predict milk quality with the highest**

degree of accuracy? To guide our experiments, we use a milk dataset from Kaggle, consisting of entries for milk samples which are made up of seven features (pH, *Temperature*, *Taste*, *Odor*, *Fat*, *Turbidity*, and *Colour*) and the associated milk quality (low, medium, or high). Since classification is the desired approach for tackling our research question, we chose five models: Logistic Regression, K-Nearest Neighbours (KNN), Decision Tree, Random Forest, and Multilayer Perceptron (MLP). To ensure robust results, we undergo rigorous data processing, sampling, and hyperparameter tuning while using multiple evaluation metrics to compare our best-performing models. With this paper, we hope to achieve two key outcomes:

1. Boost the adoption of machine learning for the milk (and other food-related) industries.
2. Provide knowledge on the benefits and downsides of our chosen classification models.

2 Methodology

In this section, we provide in-depth explanations for each step we undertook in the project.

2.1 Data Exploration

Before beginning our primary analysis, we aim to explore various facets of the milk quality data to identify potential quality issues. The dataset itself is extremely clean; it consists of 1059 rows, without any missing values or misinput entries. Our class (target variable) distribution is slightly imbalanced, with 256 entries of milk classified as high quality, 374 as medium, and the remaining 429 as low quality. There are three quantitative variables: *Temperature* - ranging from 34°C to 90°C, *pH* - Ranging from 3.0 to 9.5, and *Colour* - Ranging from 240 to 255. As we observe in Appendix A, the distributions for each variable differ from each other. The *Temperature* distribution exhibits a right skew, the *pH* distribution is centered around 6.0 to 7.0 but has outliers, and the *Colour* distribution is left-skewed.

There are also four qualitative variables: *Taste*, *Odor*, *Fat*, and *Turbidity* which are all binary values. For *Taste* and *Odor*, 0 corresponds to ‘Bad’ and 1 is ‘Good’. For *Fat* and *Turbidity*, 0 is considered ‘Low’ while

1 is ‘High’. Once again, the distributions for these variables are seen in Appendix B. Unlike the quantitative features, most qualitative variables exhibit balanced distributions except for *Fat*. Regardless, the knowledge of imbalances within the predictors will assist us when we conduct our processing.

Our final task in exploration is to assess multicollinearity within our features. Multicollinearity occurs when there are strong relationships between predictors in a dataset and can, as a result, heavily bias the model or skew results. However, as we observe in Appendix C, none of the variables exceed a correlation magnitude of 0.5, which indicates that multicollinearity is not likely to be present within our dataset. However, as we observe in Appendix C, none of the variables exceed a magnitude of 0.5 in correlation, which indicates that multicollinearity is not likely to be present within our dataset.

The last key observation we discovered involves duplicates. We found that 92% (976 entries) in the dataset were duplicates. Upon further investigation, these were not intentional but occurred due to the limited combination of features and classes. Though this is the case, having a dataset free of duplicate values proves to be useful in our modelling phase.

2.2 Data Processing

Within this step, we undergo some simple procedures to address the concerns we identified in our exploration while making minor changes to the dataset for ease of use in our modelling. First, we fix any inconsistencies within column names to avoid input errors. We then encode our target class, milk quality, into numerical categories so our classification models can run. Lastly, we standardize our predictors to deal with distribution issues we identified earlier before splitting the dataset into training and testing sets with a fixed seed. We use a 70/30 split due to the smaller size of the dataset and our interest in testing the classification models’ performance on a greater amount of unseen data. To provide additional validation for our models, we create a separate dataset for model evaluation (with the same properties as our main dataset) but remove all duplicate values.

2.3 Classification Models

Our model choice for this analysis is very intentional; we use models in order of increasing complexity to understand whether a simpler model is more beneficial in the context of our dataset. Though advanced classification methods can capture complexities within the dataset, they perform equally as well or even worse than traditional methods when the data is easier to understand.

The first model we use is logistic regression, which uses a sigmoid function on a linear combination of inputs to produce a probability score, classifying data based on a decision threshold. We then move up to using the k-nearest neighbours (KNN) model, a distance-based algorithm that classifies data by finding the 'K' closest points (neighbors) and assigning the most common category among them. Following KNN, we explore some tree-based methods, starting with decision trees. Decision trees split data into branches based on feature conditions, making a series of binary decisions until it reaches a final classification. An extension of this is the random forest, which uses an ensemble of multiple decision trees to reduce overfitting and improve performance. Finally, we broaden the range of our study with an unsupervised learning technique, the multilayer perceptron (MLP). MLPs are a type of artificial neural network that consists of several layers of neurons that attempt to capture the data with the assistance of activation functions.

2.4 Model Evaluation

For each model, we attempt to apply a similar framework to achieve consistency between our results:

1. Run the model with default parameters without additional sampling techniques.
2. Use grid search cross-validation to evaluate multiple parameter values and optimize for the best possible model.

3. Create a classification report and confusion matrix and return the accuracy on the testing data.
4. Run the model with the optimized hyperparameters on the testing data without duplicate values (and return the classification report, confusion matrix, and accuracy again).

We mentioned previously that having a separate dataset without duplicates would help us validate our models. The primary reason for this is that a model which can achieve high accuracy on a complete dataset and a sparse dataset can prove its resiliency to unseen data. Thus, our key evaluation metrics are as follows: the test dataset accuracy (with and without duplicates), the classification report outputs (with and without duplicates), and a confusion matrix (with and without duplicates).

3 Results and Analysis

3.1 Logistic Regression

Our simplest model, Logistic Regression, yields a satisfactory accuracy for both the test data with duplicates and the test data without duplicates. After hyperparameter tuning via Grid Search Cross Validation (Grid Search CV), the model's best performance is as follows:

Parameters Tuned: 'C': 1, 'max_iter': 500, 'solver': 'lbfgs'		
Accuracy (Test Data)	**Accuracy (Test Data without Duplicates)**	
0.850804792989359	0.725	
Precision	**Recall**	**F1 Score**
0.87	0.86	0.87
Label	**Classified**	**Misclassified**
0 'low'	89.2%	10.8%
1 'medium'	93.6%	6.4%
2 'high'	75.5%	24.5%

Table 1. Logistic Regression Summary of Results

Logistic regression yields a higher accuracy for the entire test data compared to the one without duplicates. Its precision, recall, and f-1 score for the entire test data also yields a satisfactory value of around 0.87. This is expected since with the presence of duplicates, there is a high possibility of data leakage among the training and testing split, meaning that it would be very easy for the model to classify

it since the model has seen the data before. In terms of specific class misclassifications, 'high' milk grade is the most misclassified. This is expected since there is less milk with a 'high' grade compared to 'low' and 'medium'. Hence, the model would tend to struggle more in predicting which milk would have the 'high' grade. While the result of logistic regression is satisfactory, theoretically, logistic regression is not designed for non-binary classifiers. Since there are 3 different classes for the target variable, 'grade', and these classes are imbalanced, bias would be introduced. This is shown in the exploratory data analysis and indicates that logistic regression may not be the best choice for this dataset. Hence, KNN is tested next as it is more flexible and does not rely on specific distribution assumptions in the data.

3.2 KNN

KNN's hyperparameter tuning is approached differently from other models in this experiment. The Elbow Curve is used to tune the main parameter which is 'K' or the number of neighbors to consider when classifying the milk grades. As shown in Appendix E, 'k' = 1 to 'k' = 8 yields a very accurate result with accuracy scores above 0.98. This is expected due to the existence of duplicate data as justified in sections 2.1 and 2.4. With only approximately 8% of the data being unique combinations, it is most likely that there are multiple occurrences of data leakage between the training and testing sets. This means that the model does not need to search beyond 'k=1' to find the correct prediction in the testing set since it has seen the data before in the training set. To further justify which 'k' should be used for a more robust model, the model was tested on unseen data using the non-duplicate data for training and testing. Appendix F shows that 'k' = 3 performs the best. Hence, 'k' = 3 was used for its high performance in the testing set for both datasets, with and without duplicates. The result of KNN with its best parameter is shown below:

Parameters Tuned: ‘n_neighbors’ : 3		
Accuracy (Test Data)	**Accuracy (Test Data without Duplicates)**	
0.9952830188679244	0.7470588235294118	
Precision	**Recall**	**F1 Score**
0.99	0.99	0.99
Label	**Classified**	**Misclassified**
0 ‘low’	100%	0%
1 ‘medium’	100%	0%
2 ‘high’	75.5%	24.5%

Table 2. KNN Summary of Results

KNN yields a very high accuracy for the test data at approximately 0.995. Its precision, recall, and f-1 score also yield a very satisfactory value of around 0.99. This means that the data is well separated between ‘grades’ and similar combinations are close together. Similar to Logistic Regression, most of the misclassifications happen in the ‘high’ milk grade while none of them are misclassified in the ‘medium’ and ‘low’ milk grade. This is expected since ‘high’ milk grade has the lowest proportion of data compared to ‘medium’ and ‘low’. While KNN yields a very high accuracy score for test data, its accuracy drops drastically to 0.747 when duplicates are removed. This means that KNN relies heavily on duplicates or similar samples, rather than learning generalizable patterns. Hence a Decision Tree is tested since it learns rules rather than memorizing instances and may handle class imbalances better.

3.3 Decision Tree

After hyperparameter tuning via Grid Search CV, the best performance of Decision Tree Model is as follows:

Parameters Tuned: 'C': 1, 'max_iter': 500, 'solver': 'lbfgs'		
Accuracy (Test Data)	**Accuracy (Test Data without Duplicates)**	
0.9943262094250201	0.8073529411764706	
Precision	**Recall**	**F1 Score**
0.87	0.86	0.87
Label	**Classified**	**Misclassified**
0 ‘low’	100%	0%
1 ‘medium’	100%	0%
2 ‘high’	97.6%	2.4%

Table 3. Decision Tree Summary of Results

Like KNN, the Decision Tree yields a very high accuracy for the test data at approximately 0.994. Its precision, recall, and f-1 score also yield a very satisfactory value of around 0.99. When duplicates are removed from the data, the accuracy drops slightly to 0.81 which is significantly better than KNN's performance in non-duplicated data. This is expected since the Decision Tree creates partitions on the feature space based on learned rules or characteristics. Hence, a Decision Tree would inherently be better at generalizing patterns compared to KNN. As with the previous models, most misclassifications happen in the 'high' milk grade which is expected since the 'high' milk grade has the lowest number of data compared to 'medium' and 'low'. Since the Decision Tree performs the best out of the first three models, Random Forest, which is a more advanced version of the Decision Tree, is tested.

3.4 Random Forest

Random Forests use bagging to aggregate a collection of trees as opposed to a single decision tree. Some additional parameters in the Random Forest include the option to bootstrap samples and the number of estimators used in the model. After tuning, the best performance of the tree is as follows:

Parameters Tuned: 'bootstrap': True, 'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50		
Accuracy (Test Data)	**Accuracy (Test Data without Duplicates)**	
0.9905660377358491	0.64	
Precision	**Recall**	**F1 Score**
0.99	0.99	0.99
Label	**Classified**	**Misclassified**
0 'low'	100%	0%
1 'medium'	99%	1%
2 'high'	97.6%	2.4%

Table 4. Random Forest Summary of Results

Like KNN and Decision Tree, Random Forest yields a very high accuracy for the test data at approximately 0.991. Its precision, recall, and f-1 score also yield a very satisfactory value of around 0.99. When duplicates are removed from the data, the accuracy drops significantly to 0.64. This significant decrease in accuracy indicates that Random Forest overfits the data significantly. This is expected since the dataset is simplistic

in nature due to the limited number of combinations between features. This means that complex machine learning algorithms like Random Forest tend to overanalyze the data and overfit underlying patterns. In terms of classification, its performance is very similar to Decision Tree which is also expected since Random Forest uses trees to classify the data.

3.5 Multilayer Perceptron

Our final model is an unsupervised learning method, which ranks the highest in terms of complexity and interpretability out of all the models we tested. Multilayer Perceptron (MLP) is a model derived from neural networks, specifically an artificial neural network that is feedforward (information moves in one direction from the input to the output). Due to the difficulty in applying a cross-validation grid and lack of model improvement with minor model tweaking, parameter tuning was not conducted for MLP. Regardless, we can observe the model performance below:

Accuracy (Test Data)		**Accuracy (Test Data without Duplicates)**
0.9905660377358491		0.6399999856948853
Precision	**Recall**	**F1 Score**
0.99	0.99	0.99
Label	**Classified**	**Misclassified**
0 ‘low’	100%	0%
1 ‘medium’	99%	1%
2 ‘high’	97.6%	2.4%

Table 5. Multiplayer Perceptron Summary of Results

Like Random Forest, MLP yields a very high accuracy for the test data at approximately 0.991. Its precision, recall, and f-1 score also yield a very satisfactory value of around 0.99. When duplicates are removed from the data, the accuracy drops significantly to 0.64. This significant decrease in accuracy emphasizes that complex models like Random Forest and MLP overfit the data significantly.

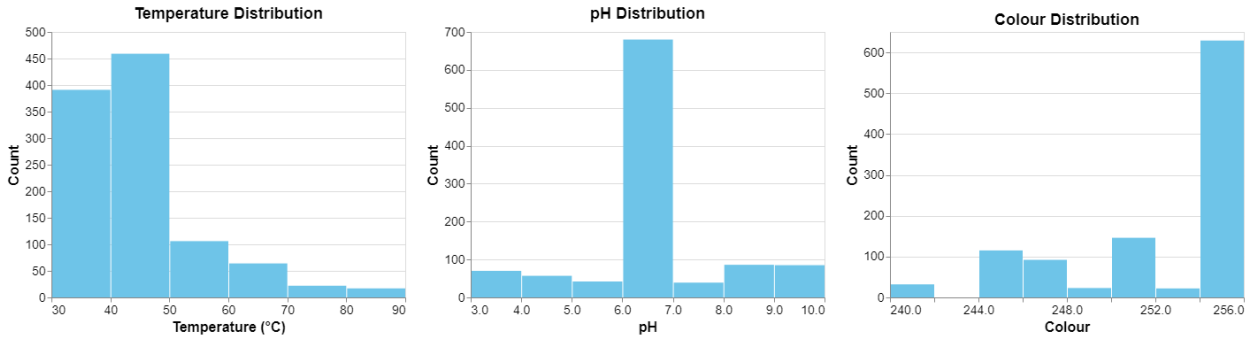
3.6 Discussion

Across all the models tested, a consistent trend was observed where the accuracy drops when duplicates are removed from the data. This trend strongly suggests the presence of data leakage between the training and testing documents. This leakage is likely to be caused by the significant number of duplicates that exist in the data. Since these duplicates are natural, meaning that they are not intentionally created, evaluating the model on the full dataset is still a valid approach. However, when the model is applied to other datasets that may have fewer duplicates, its performance may not be as high as the ones in this experiment. Another notable trend is the frequent misclassification of the 'high' milk grade class. This aligns with the discoveries in the exploratory data analysis where the milk grade variable has imbalance classes with 'high' having less data in comparison to other milk grades.

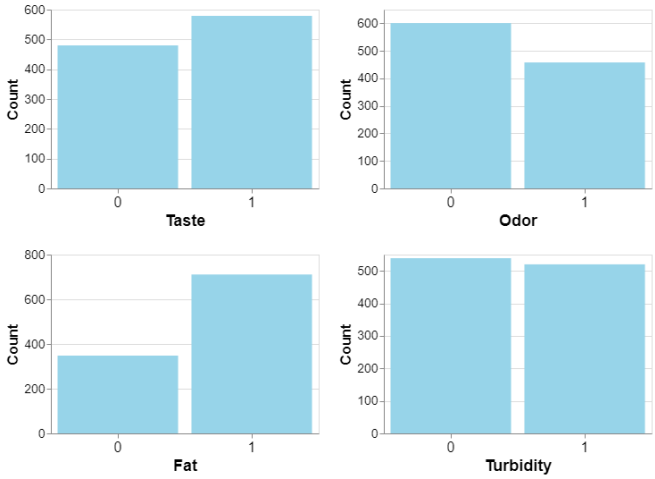
4 Conclusion

Milk classification, given the limited number of feature combinations, appears to be a relatively simple problem. As a result, simple models such as KNN and Decision Trees are sufficient for achieving high accuracy. In this experiment, KNN and Decision Tree perform similarly on the full dataset. However, when duplicates were removed, the Decision Tree outperformed all other models tested. This suggests that the Decision Tree is the most well-rounded model for this specific dataset and task. While it may be tempting to use an advanced machine learning model to solve all problems, this study proves that complexity does not translate to higher performance. Despite the excellent accuracy achieved by the models, there are still areas for improvement. One possible extension would be to refine the sampling techniques to minimize data leakage. Additionally, expanding hyperparameter tuning options could help optimize model performance better. Overall, milk classification is a task that can be efficiently and accurately handled using simple machine learning models. With proper implementation of machine learning techniques, classification tasks can be significantly improved in both accuracy and efficiency.

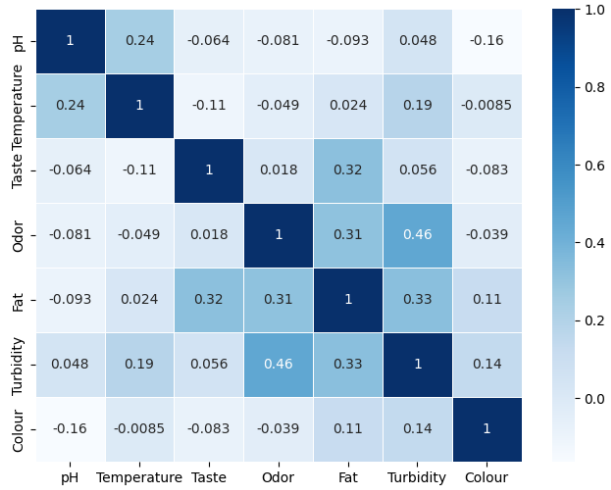
Appendix



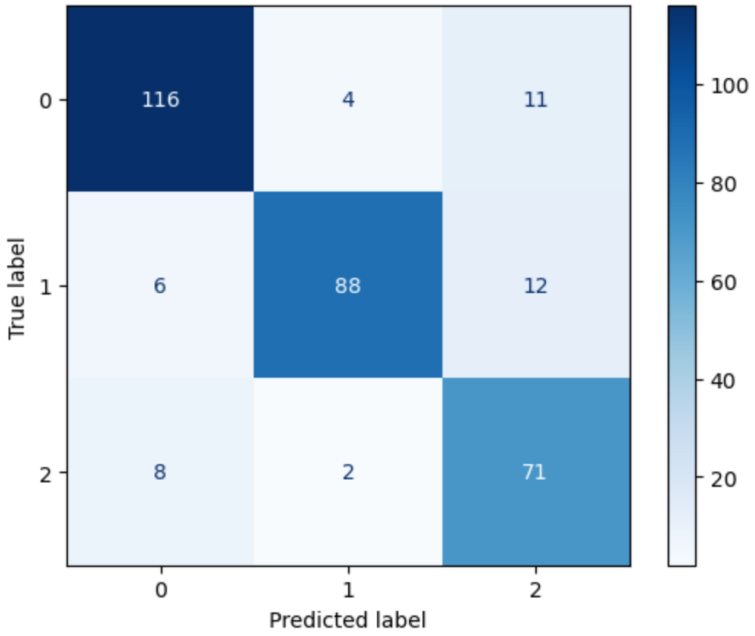
Appendix A. Distribution of Temperature, pH, and Colour



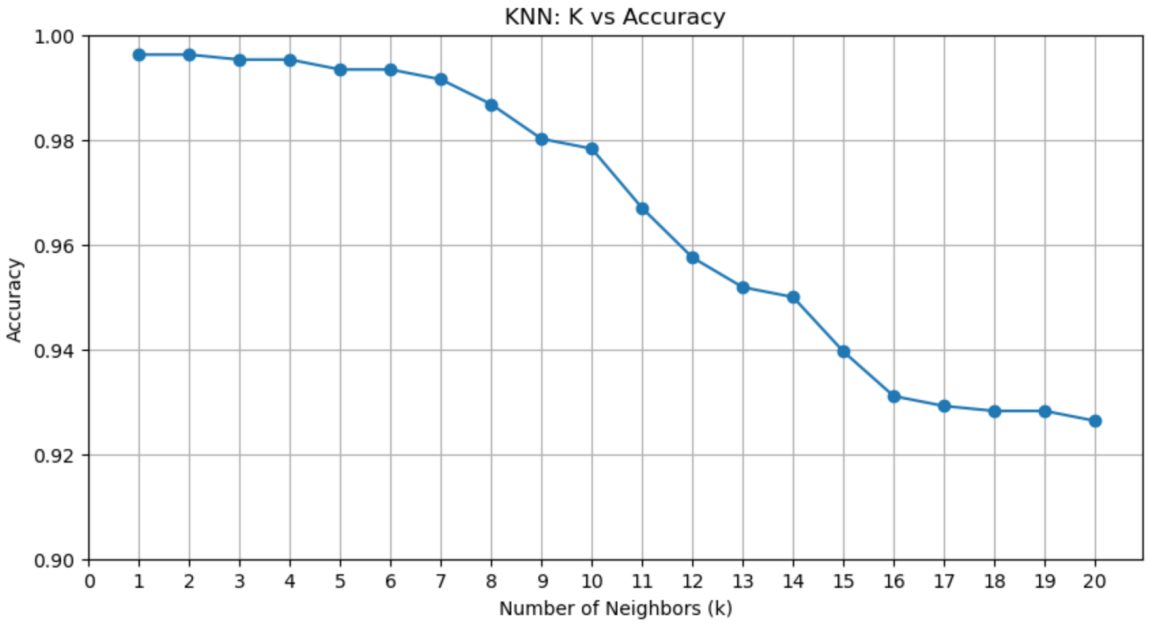
Appendix B. Distribution of Taste, Odor, Fat, and Turbidity



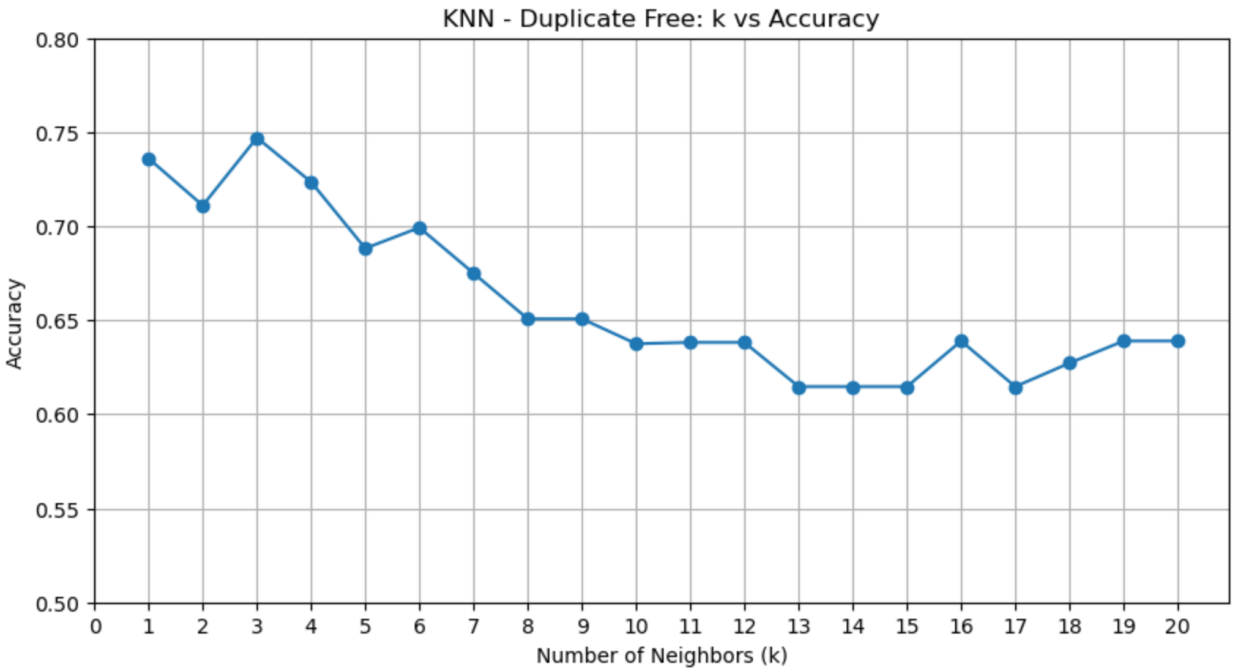
Appendix C. Correlation Matrix of Variables



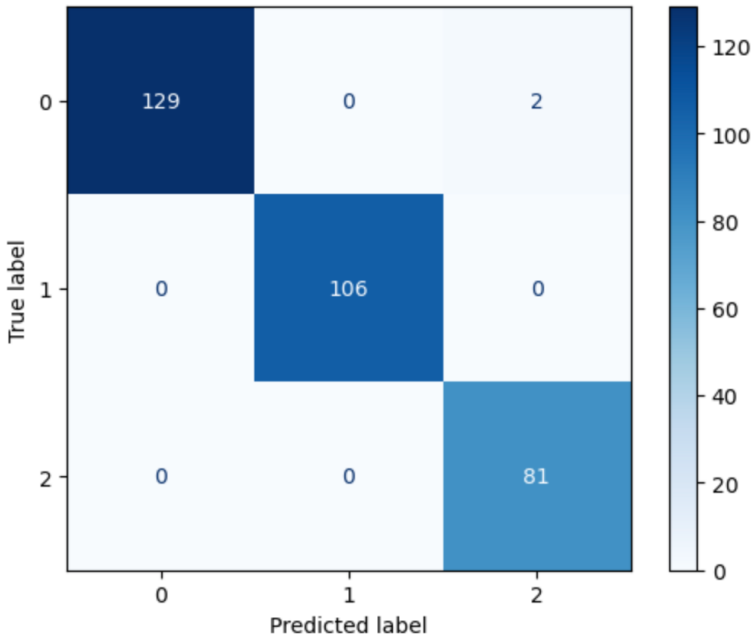
Appendix D. Logistic Regression Confusion Matrix after Hyperparameter Tuning



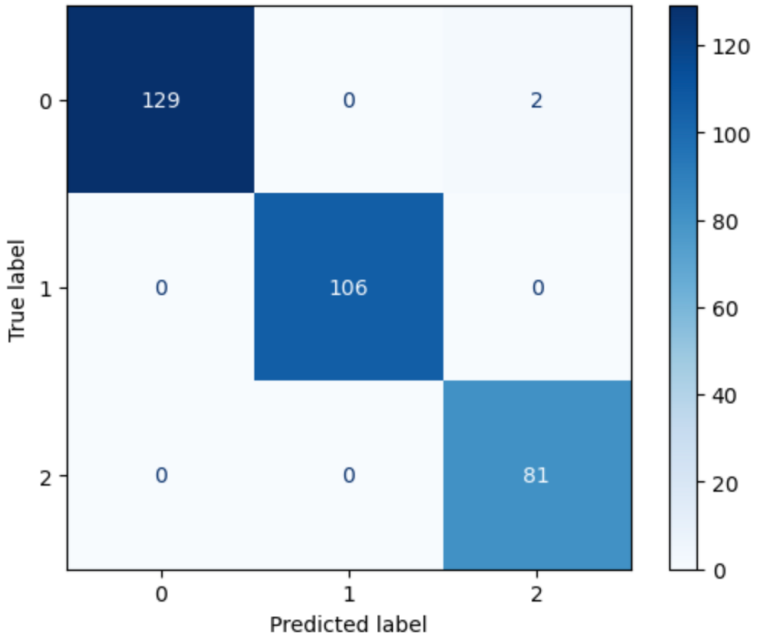
Appendix E. KNN Elbow Plot for Full Dataset



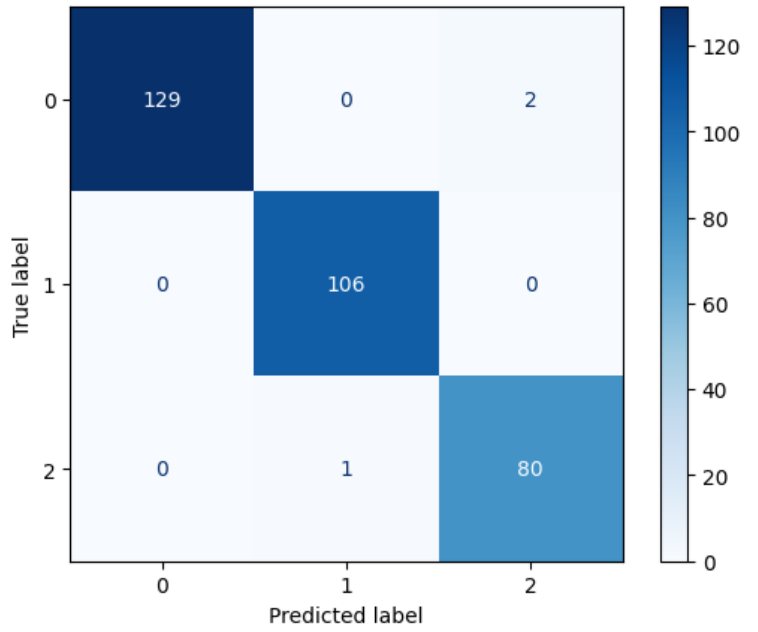
Appendix F. KNN Elbow Plot for Duplicate Free Dataset.



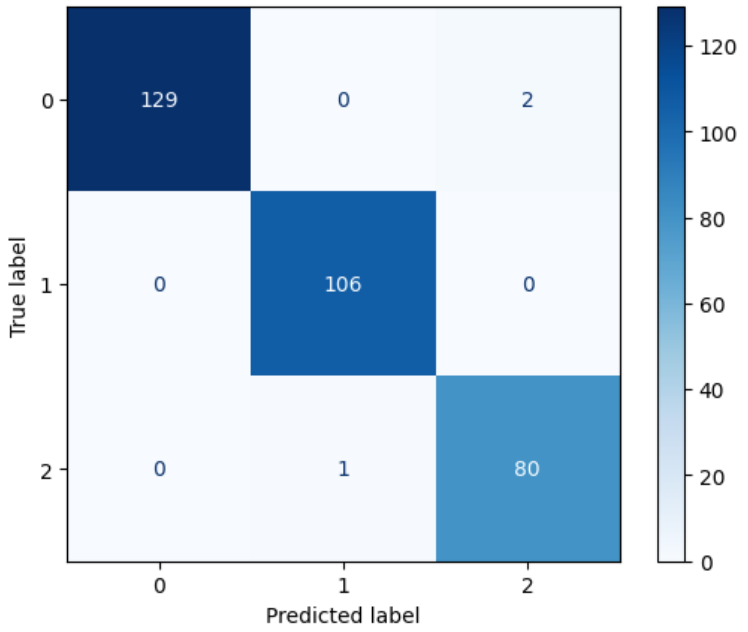
Appendix G. KNN Confusion Matrix after Hyperparameter Tuning



Appendix H. Decision Tree Confusion Matrix after Hyperparameter Tuning



Appendix I. Random Forest Confusion Matrix after Hyperparameter Tuning



Appendix J. MLP Confusion Matrix after Hyperparameter Tuning