Milk Classification Report

DATA 572

Tim Pulfer, Jacob Rosen, Dhun Sheth

Table of Contents

1. Abstract

This study explores machine learning applications in predicting milk grade using a dataset of 1059 observations with seven predictors. Employing scikit-learn for preprocessing, feature selection, and model evaluation, we adopted a 75/25 train-test split for validation. Initial testing of seven models identified K-Nearest Neighbors, Decision Tree, and Random Forest as top performers, all achieving F1 scores of about 0.99. This concise approach demonstrates machine learning's capability to enhance dairy production efficiency by accurately predicting milk grade, highlighting the effectiveness of specific models and the importance of rigorous model evaluation.

2. Introduction

The following analysis was conducted to understand how machine learning techniques can be used in the dairy industry. The motivation is to predict milk grade based on the 7 predictors outlined below. By having a reliable model to predict grade, dairy farmers can better estimate the grade of milk that will be produced before reaching the end of the milk production process. This can help producers avoid failing to meet order demands by knowing if the milk they are producing will meet the grade requirements and also increase their production efficiency.

The dataset consists of 1059 manually collected observations with the following independent fields:

- **pH**
- **Temperature**
- **Taste**
- **Odor**
- **Fat**
- **Turbidity**
- **Color**
- **Grade** {low, medium, high}

pH and temperature consist of the actual recorded values but for taste, odor, fat, turbidity, and color - if optimal conditions were met then the field was given a value of 1, else 0.

Grade is the predictor variable that can take on classification values of low, medium, or high.

3. Methodology

We employ standard scalers to preprocess the dataset, ensuring that all features have a mean of zero and a standard deviation of one. This allows for a more equitable comparison between features, ensuring that features are not under-represented due to their individual scalar space. By applying standard scalers, we ensure that our predictive models are both accurate and robust.

Our approach involves leveraging scikit-learn's comprehensive suite of machine learning tools to preprocess data, select features, train models, and evaluate their performance. Initially, data preprocessing steps such as normalization and encoding categorical variables are applied to ensure the dataset is optimally prepared for modeling. Feature selection techniques provided by scikit-learn are then employed to identify the most significant predictors. Subsequently, a variety of machine learning algorithms available within scikit-learn, including but not limited to decision trees, support vector machines, and ensemble methods like random forests, are systematically trained and tested on the dataset. The models' performances are rigorously evaluated using cross-validation and metrics such as accuracy (F1 score) and precision, facilitating a comprehensive comparison to determine the most effective algorithm. This structured application of scikit-learn tools embodies a robust experimental design, aiming to achieve high-precision predictions through methodical analysis and evaluation.

We adopted a 75/25 train-test split ratio, where 25% of the dataset is reserved for testing and the remaining 75% is used for training the models. This split was facilitated using scikit-learn's train_test_split function, ensuring a random and representative distribution of data across both sets. This approach allows for a substantial training dataset to develop robust models while allocating a significant portion for testing to accurately assess model generalizability and performance on unseen data. This split ratio was chosen to balance the need for a large training dataset with the necessity of a sufficiently large test set to validate the models' predictive capabilities.

4. Experiment

Two experiments were conducted as part of this report. The first is to test the performance of seven different machine learning models using the sklearn default hyperparameters. Each model is trained with the same training subset from the dataset and tested using the same testing subset described above. The two models that performed the best were used in further experimentation testing hyperparameter tuning and optimization.

Experiment 1

Model	**Default Hyperparameters**
Logistic Regression	
Naive Bayes	
K-Nearest Neighbors	- N = 5
Support Vector Machine	- Kernel = ‘rbf’ - C = 1
Decision Tree	- Max_depth = None - Min_samples_split = 2 - Criterion = ‘entropy’ - Random_state = 0
Linear Discriminant Analysis	
Random Forest	- N_estimators = 100 - Max_depth = None - Min_samples_split = 2

Experiment 2

Model	Grid Search Parameters
Decision Tree	- Max_depth: [1,4,5,10,15,20] - Min_samples_split: [2,3,5]
K-Nearest Neighbors	- N_neighbors: [1:794]

5. Results

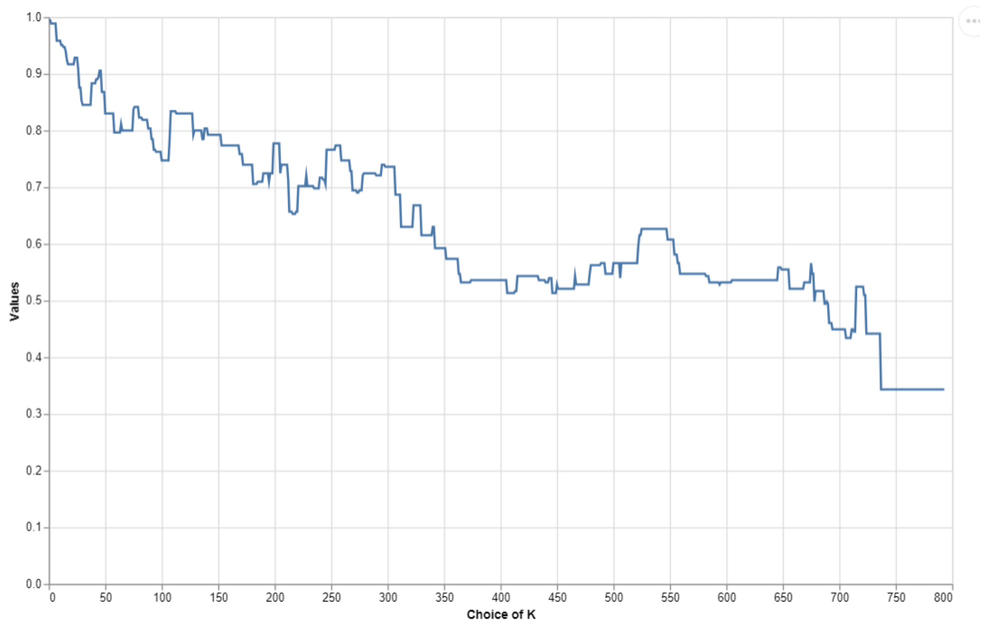
All models are evaluated based on the test set F1 score.

Experiment 1 Results

Model	F1 score on test set
Logistic Regression	0.826415
Naive Bayes	0.818868
K-Nearest Neighbors	0.988679
Support Vector Machine	0.932075
Decision Tree	0.992453
Linear Discriminant Analysis	0.701887
Random Forest	0.992453

Experiment 2 Results

Model	F1 score on test set
Decision Tree Best Parameters: {'max_depth': 10, 'min_samples_split': 2}	0.992453
K-Nearest Neighbors Best K: 1	0.996226



```
Best K to pick is 1 which gives an accuracy of 0.9962264150943396
```

Figure 1: Accuracy of the model fit with K varying from 1 to all data points in the training set

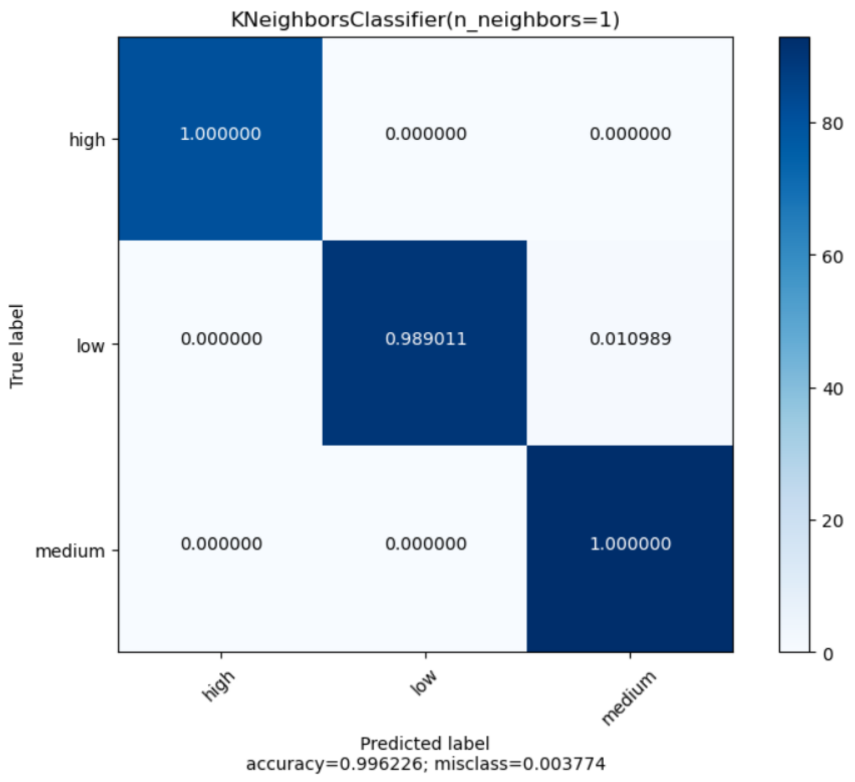


Figure 2: Misclassification matrix for the KNN model

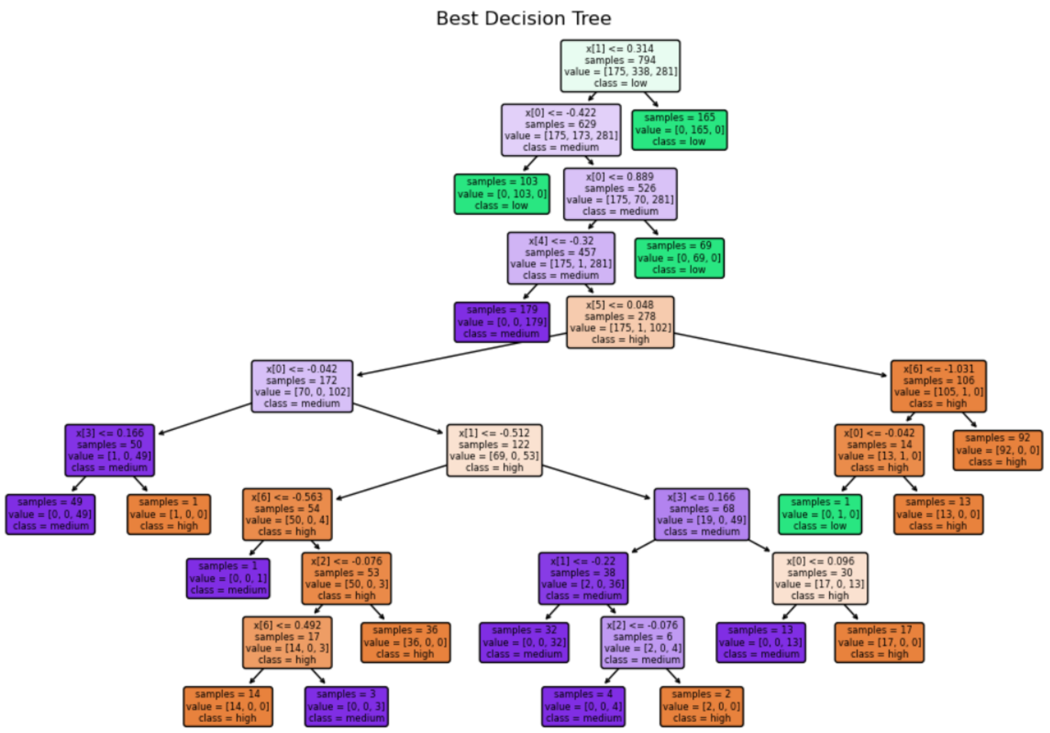


Figure 3: Best decision tree model after cross-validation and hyper-tuning parameters

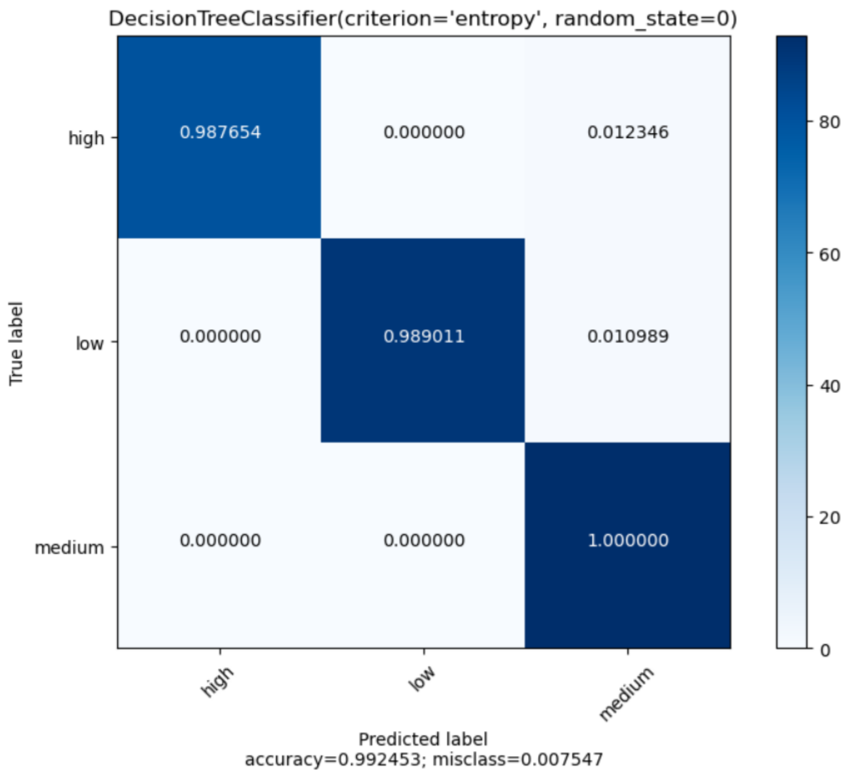


Figure 4: Misclassification matrix for the Decision Tree model

Discussion

The goal was to achieve the highest prediction accuracy and so the models that resulted in the highest F1 score during the initial assessment were chosen for additional parameter-tuning and cross-validation, to further increase their performance.

The decision tree model accuracy was not improved from the initial assessment because the optimal parameters for max_depth and min_samples_split were the default.

The default number of nearest neighbors to use is 5, however, after fine-tuning, the optimal K was found to be 1, which increased the KNN model accuracy from **0.988679** to **0.996226**.

Although it is difficult to picture the feature space for 7 features, the space is most likely very segmented depending on specific combinations of each feature. This segmentation may be a cause as to why models such as KNN, decision trees, and random forest performed so well - they can handle such feature spaces accurately, however, models like LDA and logistic regression struggle with such spaces and this can be seen in their poor performance.

6. Future Work

The findings from our investigation into the use of machine learning techniques for predicting milk grade have demonstrated significant potential to aid the dairy industry in enhancing production efficiency and meeting grade standards. The successful application of models such as K-Nearest Neighbors, Decision Tree, and Random Forest, all achieving high F1 scores, underscores the viability of machine learning models in this context. However, our research also highlights a critical area for future work: the need for more nuanced data. Currently, many of the predictors in our dataset are binary (0 or 1), limiting the depth of analysis and potentially oversimplifying complex relationships between variables and milk grade outcomes. Future studies would benefit substantially from data that captures a broader spectrum of values for predictors, providing richer, more detailed insights into the factors influencing milk quality.

7. References

1. Scikit-learn developers. (n.d.). Scikit-learn: Machine learning in Python. Retrieved February 2, 2024, from https://scikit-learn.org
2. Scikit-learn developers. (n.d.). Plot confusion matrix. Retrieved February 2, 2024, from https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html