Applied Machine Learning

CSCI-164: A.I. Programming

04/24/2025

Manuel Arambula, Quoc Dat Cao, Xavier Maravilla, Levi Valencia

Introduction

Our aim for this project was to evaluate classical supervised machine learning models. We did this by utilizing several models for the task of classification on three datasets we found interesting and assessing the performance and effectiveness of these models. Given that for this project we decided to work in a group of 4 we also expanded the scope of the project accordingly. Not only did we evaluate more than the required 2-3 models, but we also decided to tackle an inherently imbalance dataset in order to investigate methods for dealing with such a dataset. We investigated several preprocessing techniques such as standardization, outlier removal, and polynomial feature transformation to name a few along with hyperparameter tuning to improve the performance of our initial models. Logistic Regression, K-Nearest Neighbors, Naïve Bayes, Decision Tree, and Random Forest were the classification models we utilized for comparison.

Related Works

A related study that incorporated the two wine datasets we chose to explore was conducted in 2009 by Paulo Cortez et al. The purpose of that study was to predict human wine taste preferences through the use of various data mining techniques. To accomplish this, they applied three regression techniques with the support vector machine achieving promising result with an overall accuracy of 62.4 for the red wine dataset and 64.6% for the white wine dataset. Their work demonstrated the predictive potential of physicochemical features and inspired many follow-up studies applying different models and preprocessing strategies. Our project will replicate a similar setup but using Logistic Regression, k-NN, and Random Forests as the models being applied as well as incorporating additional preprocessing improvements.

A study published in 1988 by Bohanec Marko and Vladislav Rajkovič shows how they used a decision-making system to evaluate the Car Evaluation dataset. This meant they were using multi-attribute decision-making in which they considered multiple factors like price, safety, number of doors, etc., for the cars. As well as expert systems which were used to mimic the decision-making ability of a human expert but as a computer program. The way this system based its decisions was using tree-structed criteria where choices were made based on different conditions like the price or quality of the car. Their system utilized an expert system shell which is a software tool that helped with the process of building and managing the expert systems. Although their study didn't judge their results by accuracy as we did, for our project the modules that were used were K-NN, Categorical Naive Bayes, and Decision Trees to try to simulate a similar process of evaluating the dataset which resulted in very high accuracy.

Datasets and Preprocessing

From the UC Irvine Machine Learning Repository, we ended up using two datasets that contained data about wine. While they came from the same source we decided to treat them as separate datasets because of the number of instances in each dataset, plus the UCI repository had them as two datasets as well. The data within the two datasets was related to red and white Vinho Verde wine samples with one having 1599 data instances and the other 4898 data instances respectively. With the goal of both datasets being to model wine quality based on physiochemical tests. Both datasets consist of 11 numerical features and a quality label, which ranges from 0 to 10. While part of the reason we chose to use these two datasets was that we found them interesting, another was that they provided a challenge. While both datasets had no missing

values, they were unbalanced with the quality of the majority of the samples falling in between the normal range of 5 to 7, meaning that very few wine samples were poor or excellent quality. This being something we had to deal with since it would affect the performance of our models.

One of the first things we did with these two datasets was merge them and start applying various techniques to deal with the imbalances in the merged dataset. We began by first seeing what the accuracy for the baseline models were with the standardization of all the features being the only preprocessing steps being taken. This obviously gave us low accuracy scores for the models, so we decided to try other techniques like polynomial feature transformation and principal component analysis. Transforming the features using the PolynomialFeatures function with a degree of 3 resulted in an increase in the accuracy of the models, but surprisingly PCA actually deceased the accuracy of the k-NN model. From here looking to further increase the accuracy of the models we chose to use Z-score filtering, with a threshold of 3, to remove the outliners in the dataset to improve data consistency. This technique resulting in a further increase in our models' accuracies. For each of these preprocessing techniques we tested them with an 80/20 data split meaning that we decided to use 80% of the dataset for training and 20% for testing. Overall, these steps helped us deal with the imbalances of the dataset providing us a clean dataset that allowed the models to better capture both linear and non-linear relationships.

The other dataset that we decided to use is the Car Evaluation dataset which is also from the UC Irvine Machine Learning Repository. This dataset is derived from a simple hierarchical decision model which can be useful for testing constructive induction and structure discovery methods. The feature type of the dataset is categorical with 6 features consisting of 1728 instances.

We first explored the data by checking for missing values and understanding the distribution of the categorical features. Since the dataset contained no missing values, we moved on to preprocessing. Due to all features being categorical, we applied Label Encoding to convert them into numeric values, making them compatible with the machine learning models we intended to use. Once the encoding was complete, we split the data into training and testing sets using an 80/20 split to ensure that the models would be trained and evaluated fairly.

To improve model performance, especially for algorithms like K-Nearest Neighbors that are sensitive to feature scaling, we standardized the feature set using StandardScaler. We then trained baseline models including K-Nearest Neighbors, Decision Trees, and Categorical Naive Bayes using only basic preprocessing. As expected, these baseline models achieved moderate accuracy but left room for improvement. To capture more complex relationships between the features, we applied a Polynomial Feature transformation with a degree of 3, which helped to boost the performance of some models, particularly K-Nearest Neighbors.

Further efforts to improve model accuracy included detecting and removing outliers using Z-score filtering with a threshold of 3. Although the dataset primarily consisted of categorical variables and outliers were not a major issue, this step helped slightly in improving model consistency. Finally, we used GridSearchCV to perform hyperparameter tuning for our models, testing different configurations like the number of neighbors for KNN and the maximum depth for Decision Trees. These preprocessing and tuning steps helped us significantly increase the predictive accuracy of our models and provided cleaner and more consistent data for model training and evaluation.

Methodology

For the wine quality dataset, we decided to implement the following three models using scikit-learn:

- Logistic Regression: We chose this as our first model because it provides a strong baseline for binary and multi-classes classification problems, especially when the features are standardized. Logistic Regression is also fairly easy to interpret, making a good initial model for our wine dataset which contains mostly continuous variables.
- K-Nearest Neighbors: K-NN was selected due to its simplicity and effectiveness in handling non-linear decision boundaries as well as multi-class classification problems, which our task is. It also benefits significantly from feature scaling, which suits our standardized dataset.
- Random Forest Classifier: We decided to use this model mainly to work with a model most of us had no experience working with. And while looking into it we found that it is particularly useful when there are complex relationships among input features which, we thought, fit our dataset. With a bit of research, we found out this model naturally handles high-dimensional data, can capture feature interactions, and is resilient to overfitting due to its ensemble structure.

For the Car Evaluation dataset, we decided to implement the following three models using scikit-learn:

- Shared models with wine classification dataset
    - K-Nearest Neighbors (KNN): We chose KNN as our first model because of its simplicity and strong performance on smaller datasets with categorical features. Since KNN relies heavily on the distances between data points, we standardized

the features to ensure that all attributes contributed equally. KNN is also well-suited for multi-class classification tasks like predicting car acceptability.

- Unique data sets to car evaluation dataset
    - Decision Tree Classifier: We selected the Decision Tree model because it can easily handle categorical data without requiring feature scaling or complex preprocessing. Decision Trees also provide a clear visualization of the decision-making process, which made it easier for us to interpret how different features influenced car acceptability. Additionally, they are quick to train and tend to perform well on structured datasets like ours.
    - Categorical Naive Bayes: We included Categorical Naive Bayes since it is specifically designed to work with categorical features. Given that all features in the Car Evaluation dataset were categorical, this model was a natural fit. Its probabilistic approach provided a different perspective compared to distance-based and tree-based methods, and it allowed us to compare performance across a wider range of machine learning techniques.

As stated, before the input features for all of these models have been through standardization as a baseline preprocessing method. But on top of that they have also gone through polynomial transformation in order to capture non-linear relationships. When it comes to the k-NN model we did preform hyperparameter tuning on it to improve its performance. We did this by utilizing the GridSeatchCV function in the scikit-learn library searching over n_neighbors, weights, and metric to find the best value for each parameter of the k-NN model.

Result and Evaluation

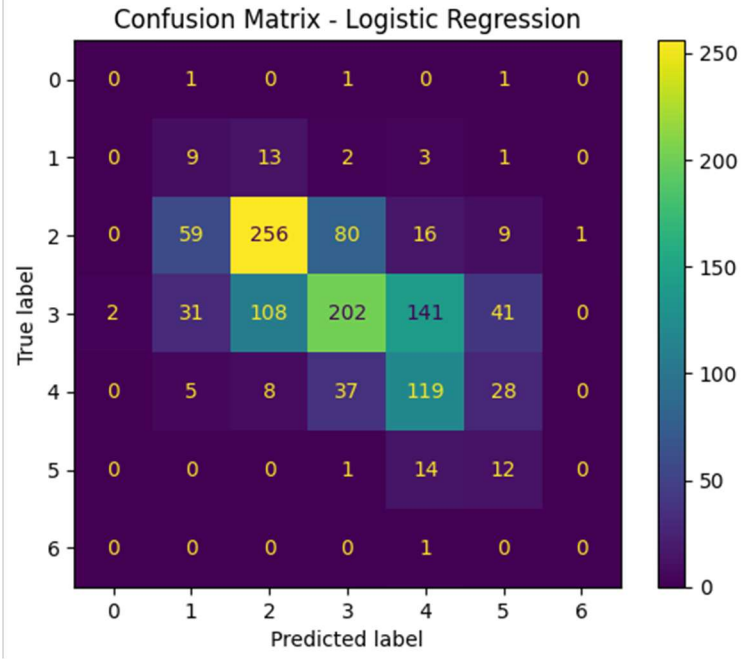
For both the wine and car datasets the metrics that we were looking at to evaluate each model were accuracy, precision, and recall. As well as displaying the confusion matrixes got with each model. We use the function classification_report from the scikit-learn to not only get the precision, recall, and f1-score for each class but also the models overall accuracy. And with the ConfusionMartixDisplay function we were able to output a figure showing the predicted and true labels of each instance.

For the wine dataset we got the following results:

- Logistic Regression:

Logistic Regression Evaluation Metrics:

	precision	recall	f1-score	support
3	0.00	0.00	0.00	3
4	0.09	0.32	0.14	28
5	0.66	0.61	0.64	421
6	0.63	0.38	0.48	525
7	0.40	0.60	0.48	197
8	0.13	0.44	0.20	27
9	0.00	0.00	0.00	1
accuracy			0.50	1202
macro avg	0.27	0.34	0.28	1202
weighted avg	0.58	0.50	0.52	1202

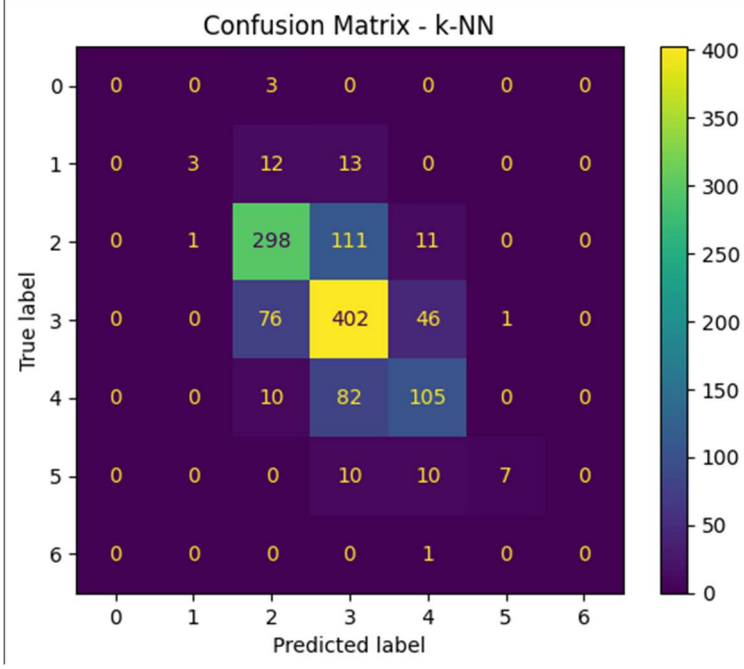


When it came to the logistic regression model, we were able to get a 50% accuracy when employing polynomial feature preprocessing technique. A better performance than the baseline, pure standardization, which got use an accuracy of 32%. Overall, we saw a balanced performance between precision and recall.

- K-Nearest Neighbors:

k-NN Evaluation Metrics:

	precision	recall	f1-score	support
3	0.00	0.00	0.00	3
4	0.75	0.11	0.19	28
5	0.75	0.71	0.73	421
6	0.65	0.77	0.70	525
7	0.61	0.53	0.57	197
8	0.88	0.26	0.40	27
9	0.00	0.00	0.00	1
accuracy			0.68	1202
macro avg	0.52	0.34	0.37	1202
weighted avg	0.68	0.68	0.67	1202



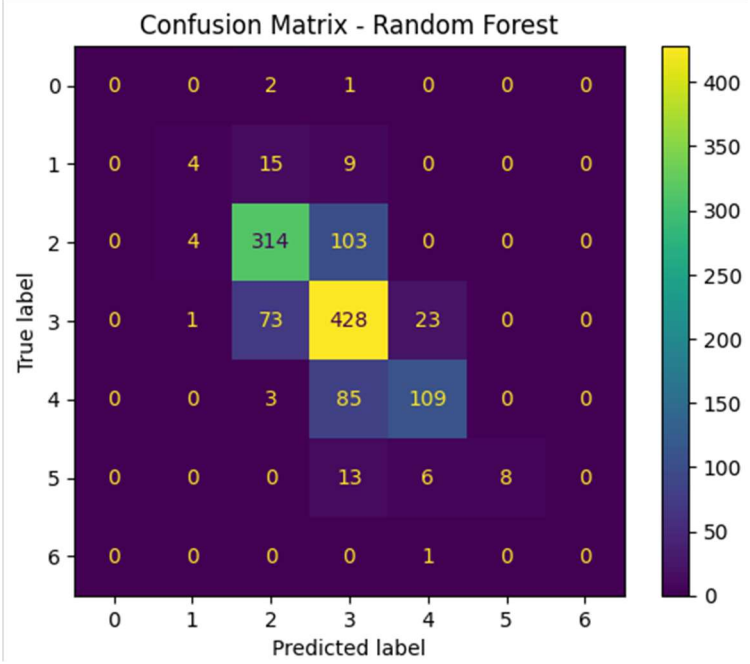
In the case of k-Nearest Neighbors similar to logistic regression we saw a significant improvement through the implementation of polynomial features. With the accuracy of the

model going from 50% to 60% and with the employment of outliner removal, with z-score, we got the accuracy to go up to 68%. After hyperparameter tuning we got the best configuration for this model was n-_neighbors = 5, weights = ‘distance’, and metric = ‘manhattan’. Overall, this model offered a strong performance throughout.

- Random Forest:

Random Forest Classification Report:

	precision	recall	f1-score	support
3	0.00	0.00	0.00	3
4	0.44	0.14	0.22	28
5	0.77	0.75	0.76	421
6	0.67	0.82	0.74	525
7	0.78	0.55	0.65	197
8	1.00	0.30	0.46	27
9	0.00	0.00	0.00	1
accuracy			0.72	1202
macro avg	0.52	0.36	0.40	1202
weighted avg	0.72	0.72	0.71	1202



Overall, the best model implement with the wine dataset was Random Forest with an accuracy of 72%. This model provided strong baseline results even without the use of feature transformation since when implementing polynomial features to the model its performance wasn’t really

improved. These results show that Random Forest is not only naturally able to handle feature interaction well but is also able to deal well with a dataset that contains outliers. While polynomial feature transformation notably improved the Logistic Regression and k-NN models' performance Random Forest remained robust regardless of feature engineering.

Results for the car evaluation dataset:

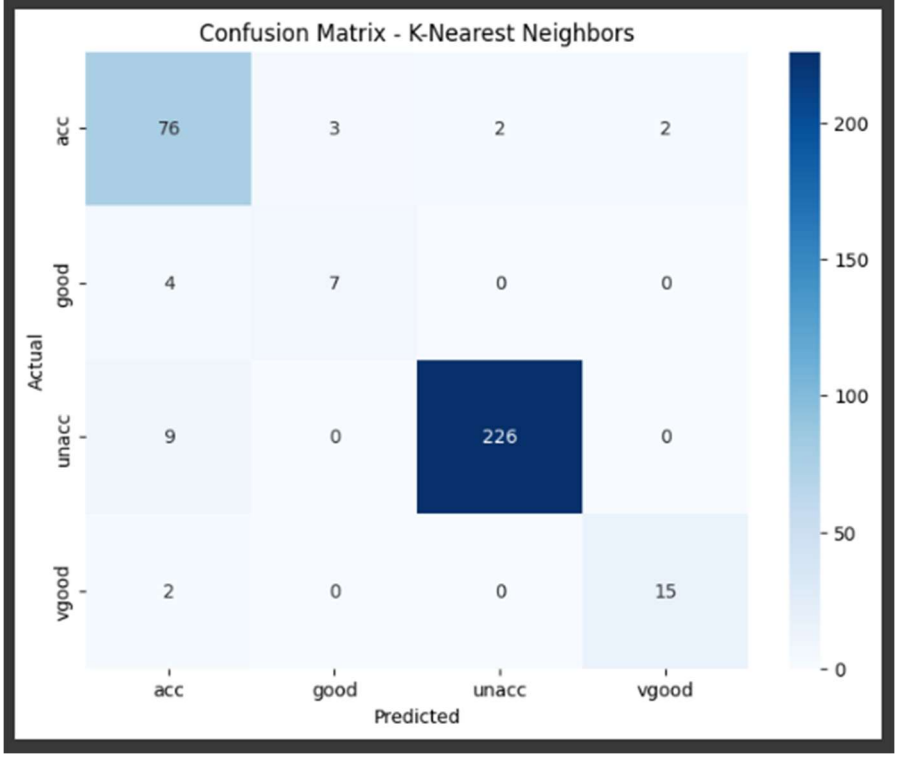
- K-Nearest Neighbor:

```
K-Nearest Neighbors Best Hyperparameters: {'n_neighbors': 3}
K-Nearest Neighbors Best Score: 0.9153037199811646

K-Nearest Neighbors Classification Report (after tuning):
              precision    recall  f1-score   support

           0       0.84      0.92      0.87        83
           1       0.70      0.64      0.67        11
           2       0.99      0.96      0.98       235
           3       0.88      0.88      0.88        17

    accuracy                           0.94       346
   macro avg       0.85      0.85      0.85       346
weighted avg       0.94      0.94      0.94       346
```



For K-Nearest Neighbor, we found out that the baseline or standard scaling resulted in higher accuracy (94%) compared to when Polynomial Features are used (92%). This is due to the fact that KNN relies on distance measures and adding polynomial features, especially if it's a lot, it can make the accuracy worsen slightly. We can also see that class 2 has the best precision and recall while the other classes are much lower.

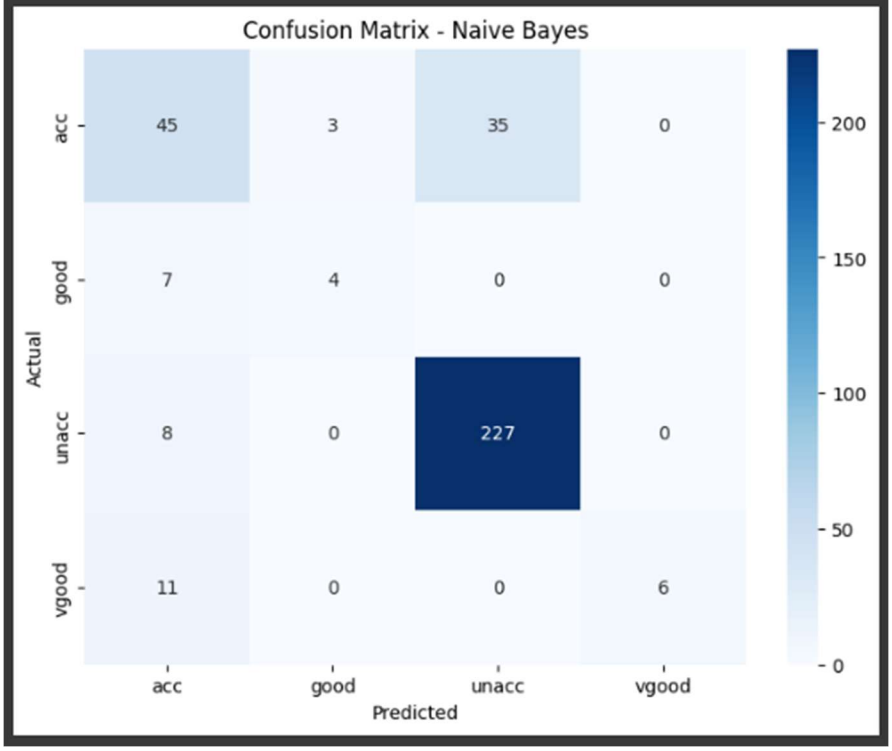
- Categorical Naive Bayes:

```
Naive Bayes Best Hyperparameters: {}
Naive Bayes Best Score: 0.8400826662481033

Naive Bayes Classification Report (after tuning):
              precision    recall  f1-score   support

           0       0.63      0.54      0.58        83
           1       0.57      0.36      0.44        11
           2       0.87      0.97      0.91       235
           3       1.00      0.35      0.52        17

    accuracy                           0.82       346
   macro avg       0.77      0.56      0.62       346
weighted avg       0.81      0.82      0.80       346
```



Naive Bayes seemed to have pretty good accuracy (82%) but saw no positive changes to it. This can possibly be due to the model assumptions that are not compatible with complex feature interactions.

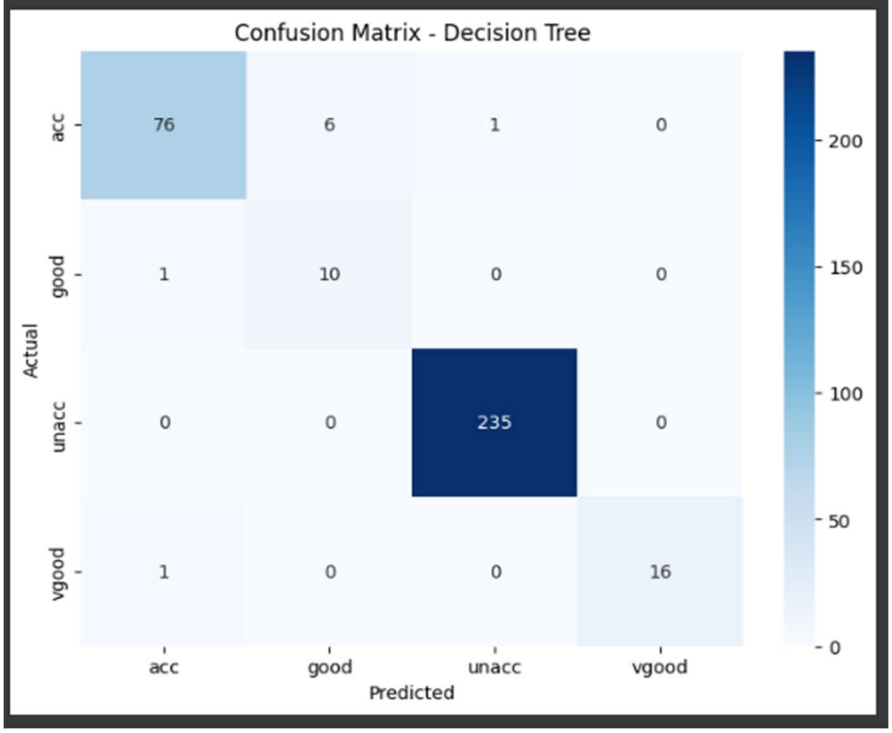
- Decision Tree:

```
Decision Tree Best Hyperparameters: {'max_depth': None, 'min_samples_split': 2}
Decision Tree Best Score: 0.976845602469523

Decision Tree Classification Report (after tuning):
              precision    recall  f1-score   support

           0       0.97      0.92      0.94        83
           1       0.62      0.91      0.74        11
           2       1.00      1.00      1.00       235
           3       1.00      0.94      0.97        17

    accuracy                           0.97       346
   macro avg       0.90      0.94      0.91       346
weighted avg       0.98      0.97      0.98       346
```



Using the Decision Tree module, we see that it performs almost perfectly (98%). Having only minor imperfections in some of the classes but overall performing very well. Decision Trees don't need polynomial features since they already capture nonlinear feature interactions natively by splitting on different features.

Conclusion

In this project, we evaluated several classical machine learning models across two datasets using extensive preprocessing and tuning techniques. Through standardization, polynomial feature transformation, outlier removal, and hyperparameter optimization, we significantly improved model performance. Logistic Regression and k-NN benefited notably from feature transformations on the Wine Quality dataset, while Random Forest showed strong performance even without heavy preprocessing. On the Car Evaluation dataset, the Decision Tree model achieved the highest accuracy, leveraging its ability to handle categorical and non-linear relationships naturally.

Our results highlight the importance of choosing the right model and preprocessing strategy based on dataset characteristics. This project also demonstrated how critical it is to tailor machine learning pipelines to the specific challenges of each dataset to maximize predictive accuracy. Future improvements could include exploring ensemble methods or additional techniques for handling data imbalance.

References

GitHub Repository: https://github.com/Liv-1804/164-Project

Bohanec, Marko and Vladislav Rajkovič. “KNOWLEDGE ACQUISITION AND EXPLANATION FOR MULTI-ATTRIBUTE DECISION MAKING ∗.” (1988).

Bohanec, Marko. "Car Evaluation." UCI Machine Learning Repository, 1988, https://doi.org/10.24432/C5JP48.

Cortez, P. et al. “Modeling wine preferences by data mining from physicochemical properties.” *Decis. Support Syst.* 47 (2009): 547-553.

Cortez, Paulo, et al. "Wine Quality." UCI Machine Learning Repository, 2009, https://doi.org/10.24432/C56S3T.