CS634-Final Term Project Report

Student Name: **Rajeev Kumar Alahari**

Course: **CS634-Data Mining**

Instructor: **Dr. Yasser Abduallah**

Project Title: **Binary Classification of Red vs. White Wine Using Random Forest, KNN, and Conv1D**

Date: **11-16-2025**

1 Introduction

The goal of this project is to build, evaluate, and compare data-mining models for a binary classification task. Specifically, we predict whether a wine sample is red or white based only on its physicochemical properties (acidity, sugar, sulfur dioxide, alcohol, etc.).

Binary classification is a core problem in data mining and machine learning where each instance is assigned to one of two classes (yes or no).

2 Dataset Creation

Name: Wine Type Classification Dataset
Source: Kaggle dataset "Wine Type Classification Dataset" (originally from UCI Wine Quality datasets)
Link: https://www.kaggle.com/datasets/ehsanesmaeili/red-and-white-wine-quality-merged
Description:
Number of rows: The dataset contains 6,497 wine samples.
Features: It includes 11 physicochemical numeric features and 1 numeric quality score used as an additional feature(So total 12 Features).
Target variable: The binary target is wine type (`red` vs `white`).
Preprocessing Steps:
Missing values: Dataset has no missing values, so no imputation was required.
Normalization/Standardization: All numeric features were standardized using StandardScaler to improve model performance.
Label encoding: The categorical target `type` ("red", "white") was label-encoded into 0/1 for machine learning models.

Class balance: Class distribution is moderately imbalanced (nearly 25% red, 75% white), but both classes are sufficiently represented for training.

3. Algorithms Overview

3.1 Random Forest

Random Forest is an ensemble learning method that builds many decision trees on random subsets of the data and features, and then averages their predictions (for classification, using majority vote). It is robust to noise, can model nonlinear decision boundaries, and typically works well out-of-the-box with minimal tuning. In this project, a Random Forest with around 200 trees and default depth settings was used. Random Forest is expected to perform very well here because the classes are well separated and the number of features is modest.

3.2 Conv1D Neural Network

The Conv1D model is a 1-D convolutional neural network that treats the 11 numeric features as a short 1D sequence. Convolutional layers learn local patterns (combinations of nearby features), followed by pooling and fully connected layers to produce a final binary output via a sigmoid activation.
The architecture used:
Input shape: (n_features, 1)
One or two Conv1D layers (e.g., 64 and 32 filters, kernel size 3)
Global max pooling
Dense hidden layer(s) with ReLU
Output layer: 1 neuron with sigmoid activation
Conv1D was chosen as the deep learning model because it is simple, efficient for tabular sequences, and compatible with Keras/TensorFlow. It can capture interactions between neighboring features and potentially achieve performance comparable to or better than classical methods.

3.3 K-Nearest Neighbors (KNN)

KNN is a classic instance-based machine learning method. To classify a new sample, it finds the k most similar training examples (here, using Euclidean distance on standardized features) and predicts the majority class among them.

In this project:
k = 13 neighbors
Distance: Euclidean
Features standardized beforehand
KNN is simple and intuitive. It can perform well on datasets where same-class points are clustered in feature space, which we expect here because red and white wines have distinct ranges of acidity, sulfur, and sugar levels. However, KNN can be sensitive to the choice of k and to class imbalance.

4 Implementation

Programming language: Python

Development environment: Jupyter Notebook and .py scripts.

Required Python packages and installation instructions

pip install numpy pandas scikit-learn tensorflow keras matplotlib seaborn

Screenshots:

Dataset loading/preview

```
# Dataset path
path = "wine_quality.csv"
wine_data = ps.read_csv(path)

print("Raw dataset shape:", wine_data.shape)
print(wine_data.head(3))

#Target value unique count
if "type" in wine_data.columns:
    print("\nType counts:")
    print(wine_data["type"].value_counts())

df = wine_data.copy()
```

```
Raw dataset shape: (6497, 13)
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65

   alcohol  quality type
0      9.4        5  red
1      9.8        5  red
2      9.8        5  red
```

The dataset is successfully loaded and description can be seen.

Per-fold evaluation outputs/results

-- Fold 1 results --

	TP	TN	FP	FN	P	N	TPR	TNR	FPR	FNR
RF	477.0	170.0	2.0	1.0	478.0	172.0	0.998	0.988	0.012	0.002
KNN	472.0	169.0	3.0	6.0	478.0	172.0	0.987	0.983	0.017	0.013
Conv1D	475.0	162.0	10.0	3.0	478.0	172.0	0.994	0.942	0.058	0.006

	Precision	F1	Accuracy	Error_rate	BACC	TSS	HSS	AUC
RF	0.996	0.997	0.995	0.005	0.993	0.986	0.988	1.000
KNN	0.994	0.991	0.986	0.014	0.985	0.970	0.965	0.999
Conv1D	0.979	0.987	0.980	0.020	0.968	0.936	0.948	0.997

	BS	BSS
RF	0.005	0.974
KNN	0.009	0.953
Conv1D	0.015	0.926

-- Fold 2 results --

	TP	TN	FP	FN	P	N	TPR	TNR	FPR	FNR
RF	481.0	165.0	4.0	0.0	481.0	169.0	1.000	0.976	0.024	0.000
KNN	480.0	164.0	5.0	1.0	481.0	169.0	0.998	0.970	0.030	0.002
Conv1D	479.0	144.0	25.0	2.0	481.0	169.0	0.996	0.852	0.148	0.004

	Precision	F1	Accuracy	Error_rate	BACC	TSS	HSS	AUC
RF	0.992	0.996	0.994	0.006	0.988	0.976	0.984	0.999
KNN	0.990	0.994	0.991	0.009	0.984	0.968	0.976	0.991
Conv1D	0.950	0.973	0.958	0.042	0.924	0.848	0.887	0.989

	BS	BSS
RF	0.007	0.962
KNN	0.009	0.955
Conv1D	0.033	0.828

-- Fold 3 results --

	TP	TN	FP	FN	P	N	TPR	TNR	FPR	FNR
RF	489.0	159.0	2.0	0.0	489.0	161.0	1.000	0.988	0.012	0.000
KNN	485.0	159.0	2.0	4.0	489.0	161.0	0.992	0.988	0.012	0.008
Conv1D	489.0	148.0	13.0	0.0	489.0	161.0	1.000	0.919	0.081	0.000

	Precision	F1	Accuracy	Error_rate	BACC	TSS	HSS	AUC
RF	0.996	0.998	0.997	0.003	0.994	0.988	0.992	1.000
KNN	0.996	0.994	0.991	0.009	0.990	0.979	0.975	0.996
Conv1D	0.974	0.987	0.980	0.020	0.960	0.919	0.945	0.991

	BS	BSS
RF	0.004	0.976
KNN	0.007	0.960
Conv1D	0.015	0.920

```
-- Fold 4 results --
           TP     TN    FP   FN      P      N    TPR    TNR    FPR    FNR  \
RF      471.0  170.0   6.0  3.0  474.0  176.0  0.994  0.966  0.034  0.006
KNN     472.0  172.0   4.0  2.0  474.0  176.0  0.996  0.977  0.023  0.004
Conv1D  468.0  162.0  14.0  6.0  474.0  176.0  0.987  0.920  0.080  0.013

        Precision     F1  Accuracy  Error_rate   BACC    TSS    HSS    AUC  \
RF          0.987  0.991     0.986       0.014  0.980  0.960  0.965  0.999
KNN         0.992  0.994     0.991       0.009  0.987  0.973  0.977  0.997
Conv1D      0.971  0.979     0.969       0.031  0.954  0.908  0.921  0.996

           BS    BSS
RF      0.009  0.953
KNN     0.007  0.963
Conv1D  0.020  0.898
```

```
-- Fold 5 results --
           TP     TN   FP   FN      P      N    TPR    TNR    FPR    FNR  \
RF      503.0  146.0  0.0  1.0  504.0  146.0  0.998  1.000  0.000  0.002
KNN     502.0  145.0  1.0  2.0  504.0  146.0  0.996  0.993  0.007  0.004
Conv1D  499.0  139.0  7.0  5.0  504.0  146.0  0.990  0.952  0.048  0.010

        Precision     F1  Accuracy  Error_rate   BACC    TSS    HSS    AUC  \
RF          1.000  0.999     0.998       0.002  0.999  0.998  0.996  1.000
KNN         0.998  0.997     0.995       0.005  0.995  0.989  0.987  1.000
Conv1D      0.986  0.988     0.982       0.018  0.971  0.942  0.947  0.994

           BS    BSS
RF      0.004  0.979
KNN     0.004  0.977
Conv1D  0.015  0.915
```

```
-- Fold 6 results --
           TP     TN    FP   FN      P      N    TPR    TNR    FPR    FNR  \
RF      495.0  154.0   1.0  0.0  495.0  155.0  1.000  0.994  0.006  0.000
KNN     494.0  154.0   1.0  1.0  495.0  155.0  0.998  0.994  0.006  0.002
Conv1D  492.0  130.0  25.0  3.0  495.0  155.0  0.994  0.839  0.161  0.006

        Precision     F1  Accuracy  Error_rate   BACC    TSS    HSS    AUC  \
RF          0.998  0.999     0.998       0.002  0.997  0.994  0.996  1.000
KNN         0.998  0.998     0.997       0.003  0.996  0.992  0.992  1.000
Conv1D      0.952  0.972     0.957       0.043  0.916  0.833  0.875  0.992

           BS    BSS
RF      0.003  0.983
KNN     0.003  0.984
Conv1D  0.030  0.832
```

-- Fold 7 results --

	TP	TN	FP	FN	P	N	TPR	TNR	FPR	FNR \
RF	488.0	159.0	2.0	1.0	489.0	161.0	0.998	0.988	0.012	0.002
KNN	488.0	159.0	2.0	1.0	489.0	161.0	0.998	0.988	0.012	0.002
Conv1D	485.0	147.0	14.0	4.0	489.0	161.0	0.992	0.913	0.087	0.008

	Precision	F1	Accuracy	Error_rate	BACC	TSS	HSS	AUC \
RF	0.996	0.997	0.995	0.005	0.993	0.986	0.988	1.000
KNN	0.996	0.997	0.995	0.005	0.993	0.986	0.988	0.997
Conv1D	0.972	0.982	0.972	0.028	0.952	0.905	0.924	0.991

	BS	BSS
RF	0.004	0.978
KNN	0.005	0.975
Conv1D	0.023	0.877

-- Fold 8 results --

	TP	TN	FP	FN	P	N	TPR	TNR	FPR	FNR \
RF	472.0	173.0	4.0	0.0	472.0	177.0	1.000	0.977	0.023	0.000
KNN	470.0	174.0	3.0	2.0	472.0	177.0	0.996	0.983	0.017	0.004
Conv1D	467.0	150.0	27.0	5.0	472.0	177.0	0.989	0.847	0.153	0.011

	Precision	F1	Accuracy	Error_rate	BACC	TSS	HSS	AUC \
RF	0.992	0.996	0.994	0.006	0.989	0.977	0.984	0.996
KNN	0.994	0.995	0.992	0.008	0.989	0.979	0.981	0.994
Conv1D	0.945	0.967	0.951	0.049	0.918	0.837	0.871	0.986

	BS	BSS
RF	0.007	0.964
KNN	0.007	0.964
Conv1D	0.037	0.812

-- Fold 9 results --

	TP	TN	FP	FN	P	N	TPR	TNR	FPR	FNR \
RF	510.0	135.0	4.0	0.0	510.0	139.0	1.000	0.971	0.029	0.000
KNN	505.0	138.0	1.0	5.0	510.0	139.0	0.990	0.993	0.007	0.010
Conv1D	506.0	123.0	16.0	4.0	510.0	139.0	0.992	0.885	0.115	0.008

	Precision	F1	Accuracy	Error_rate	BACC	TSS	HSS	AUC \
RF	0.992	0.996	0.994	0.006	0.986	0.971	0.981	1.000
KNN	0.998	0.994	0.991	0.009	0.992	0.983	0.973	0.996
Conv1D	0.969	0.981	0.969	0.031	0.939	0.877	0.905	0.988

	BS	BSS
RF	0.006	0.963
KNN	0.007	0.958
Conv1D	0.028	0.836

-- Fold 10 results --

	TP	TN	FP	FN	P	N	TPR	TNR	FPR	FNR
RF	505.0	143.0	0.0	1.0	506.0	143.0	0.998	1.000	0.000	0.002
KNN	506.0	142.0	1.0	0.0	506.0	143.0	1.000	0.993	0.007	0.000
Conv1D	501.0	130.0	13.0	5.0	506.0	143.0	0.990	0.909	0.091	0.010

	Precision	F1	Accuracy	Error_rate	BACC	TSS	HSS	AUC
RF	1.000	0.999	0.998	0.002	0.999	0.998	0.996	1.000
KNN	0.998	0.999	0.998	0.002	0.997	0.993	0.996	1.000
Conv1D	0.975	0.982	0.972	0.028	0.950	0.899	0.918	0.996

	BS	BSS
RF	0.003	0.982
KNN	0.002	0.986
Conv1D	0.018	0.895

5 Evaluation Setup

Method: 10-fold cross-validation.

Metrics to report for each fold and overall averages: TP, TN, FP, FN, Accuracy, Precision, Recall, F1, FPR, FNR, Specificity, Balanced Accuracy, TSS, HSS, ROC, AUC, BS, BSS.

6 Results

Per Fold Results for all three Algorithms

All KNN folds metrics:

	fold1	fold2	fold3	fold4	fold5	fold6	fold7
TP	472.000	480.000	485.000	472.000	502.000	494.000	488.000
TN	169.000	164.000	159.000	172.000	145.000	154.000	159.000
FP	3.000	5.000	2.000	4.000	1.000	1.000	2.000
FN	6.000	1.000	4.000	2.000	2.000	1.000	1.000
P	478.000	481.000	489.000	474.000	504.000	495.000	489.000
N	172.000	169.000	161.000	176.000	146.000	155.000	161.000
TPR	0.987	0.998	0.992	0.996	0.996	0.998	0.998
TNR	0.983	0.970	0.988	0.977	0.993	0.994	0.988
FPR	0.017	0.030	0.012	0.023	0.007	0.006	0.012
FNR	0.013	0.002	0.008	0.004	0.004	0.002	0.002
Precision	0.994	0.990	0.996	0.992	0.998	0.998	0.996
F1	0.991	0.994	0.994	0.994	0.997	0.998	0.997
Accuracy	0.986	0.991	0.991	0.991	0.995	0.997	0.995
Error_rate	0.014	0.009	0.009	0.009	0.005	0.003	0.005
BACC	0.985	0.984	0.990	0.987	0.995	0.996	0.993
TSS	0.970	0.968	0.979	0.973	0.989	0.992	0.986
HSS	0.965	0.976	0.975	0.977	0.987	0.992	0.988
AUC	0.999	0.991	0.996	0.997	1.000	1.000	0.997
BS	0.009	0.009	0.007	0.007	0.004	0.003	0.005
BSS	0.953	0.955	0.960	0.963	0.977	0.984	0.975

	fold8	fold9	fold10
TP	470.000	505.000	506.000
TN	174.000	138.000	142.000
FP	3.000	1.000	1.000
FN	2.000	5.000	0.000
P	472.000	510.000	506.000
N	177.000	139.000	143.000
TPR	0.996	0.990	1.000
TNR	0.983	0.993	0.993
FPR	0.017	0.007	0.007
FNR	0.004	0.010	0.000
Precision	0.994	0.998	0.998
F1	0.995	0.994	0.999
Accuracy	0.992	0.991	0.998
Error_rate	0.008	0.009	0.002
BACC	0.989	0.992	0.997
TSS	0.979	0.983	0.993
HSS	0.981	0.973	0.996
AUC	0.994	0.996	1.000
BS	0.007	0.007	0.002
BSS	0.964	0.958	0.986

All RF folds metrics:

	fold1	fold2	fold3	fold4	fold5	fold6	fold7 \
TP	477.000	481.000	489.000	471.000	503.000	495.000	488.000
TN	170.000	165.000	159.000	170.000	146.000	154.000	159.000
FP	2.000	4.000	2.000	6.000	0.000	1.000	2.000
FN	1.000	0.000	0.000	3.000	1.000	0.000	1.000
P	478.000	481.000	489.000	474.000	504.000	495.000	489.000
N	172.000	169.000	161.000	176.000	146.000	155.000	161.000
TPR	0.998	1.000	1.000	0.994	0.998	1.000	0.998
TNR	0.988	0.976	0.988	0.966	1.000	0.994	0.988
FPR	0.012	0.024	0.012	0.034	0.000	0.006	0.012
FNR	0.002	0.000	0.000	0.006	0.002	0.000	0.002
Precision	0.996	0.992	0.996	0.987	1.000	0.998	0.996
F1	0.997	0.996	0.998	0.991	0.999	0.999	0.997
Accuracy	0.995	0.994	0.997	0.986	0.998	0.998	0.995
Error_rate	0.005	0.006	0.003	0.014	0.002	0.002	0.005
BACC	0.993	0.988	0.994	0.980	0.999	0.997	0.993
TSS	0.986	0.976	0.988	0.960	0.998	0.994	0.986
HSS	0.988	0.984	0.992	0.965	0.996	0.996	0.988
AUC	1.000	0.999	1.000	0.999	1.000	1.000	1.000
BS	0.005	0.007	0.004	0.009	0.004	0.003	0.004
BSS	0.974	0.962	0.976	0.953	0.979	0.983	0.978

	fold8	fold9	fold10
TP	472.000	510.000	505.000
TN	173.000	135.000	143.000
FP	4.000	4.000	0.000
FN	0.000	0.000	1.000
P	472.000	510.000	506.000
N	177.000	139.000	143.000
TPR	1.000	1.000	0.998
TNR	0.977	0.971	1.000
FPR	0.023	0.029	0.000
FNR	0.000	0.000	0.002
Precision	0.992	0.992	1.000
F1	0.996	0.996	0.999
Accuracy	0.994	0.994	0.998
Error_rate	0.006	0.006	0.002
BACC	0.989	0.986	0.999
TSS	0.977	0.971	0.998
HSS	0.984	0.981	0.996
AUC	0.996	1.000	1.000
BS	0.007	0.006	0.003
BSS	0.964	0.963	0.982

All Conv1D folds metrics:

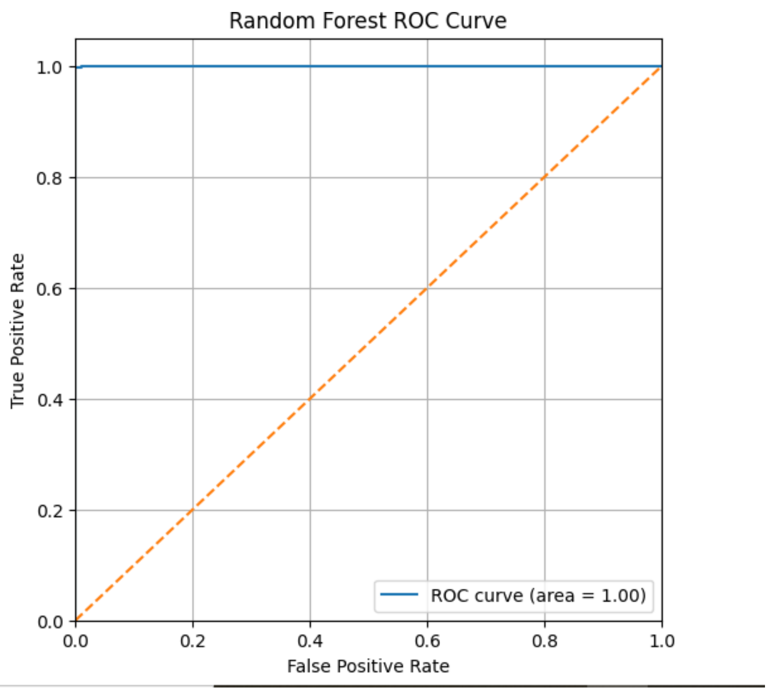
	fold1	fold2	fold3	fold4	fold5	fold6	fold7	\
TP	475.000	479.000	489.000	468.000	499.000	492.000	485.000	
TN	162.000	144.000	148.000	162.000	139.000	130.000	147.000	
FP	10.000	25.000	13.000	14.000	7.000	25.000	14.000	
FN	3.000	2.000	0.000	6.000	5.000	3.000	4.000	
P	478.000	481.000	489.000	474.000	504.000	495.000	489.000	
N	172.000	169.000	161.000	176.000	146.000	155.000	161.000	
TPR	0.994	0.996	1.000	0.987	0.990	0.994	0.992	
TNR	0.942	0.852	0.919	0.920	0.952	0.839	0.913	
FPR	0.058	0.148	0.081	0.080	0.048	0.161	0.087	
FNR	0.006	0.004	0.000	0.013	0.010	0.006	0.008	
Precision	0.979	0.950	0.974	0.971	0.986	0.952	0.972	
F1	0.987	0.973	0.987	0.979	0.988	0.972	0.982	
Accuracy	0.980	0.958	0.980	0.969	0.982	0.957	0.972	
Error_rate	0.020	0.042	0.020	0.031	0.018	0.043	0.028	
BACC	0.968	0.924	0.960	0.954	0.971	0.916	0.952	
TSS	0.936	0.848	0.919	0.908	0.942	0.833	0.905	
HSS	0.948	0.887	0.945	0.921	0.947	0.875	0.924	
AUC	0.997	0.989	0.991	0.996	0.994	0.992	0.991	
BS	0.015	0.033	0.015	0.020	0.015	0.030	0.023	
BSS	0.926	0.828	0.920	0.898	0.915	0.832	0.877	

Mean Average Results of all three algorithms of all 10 folds

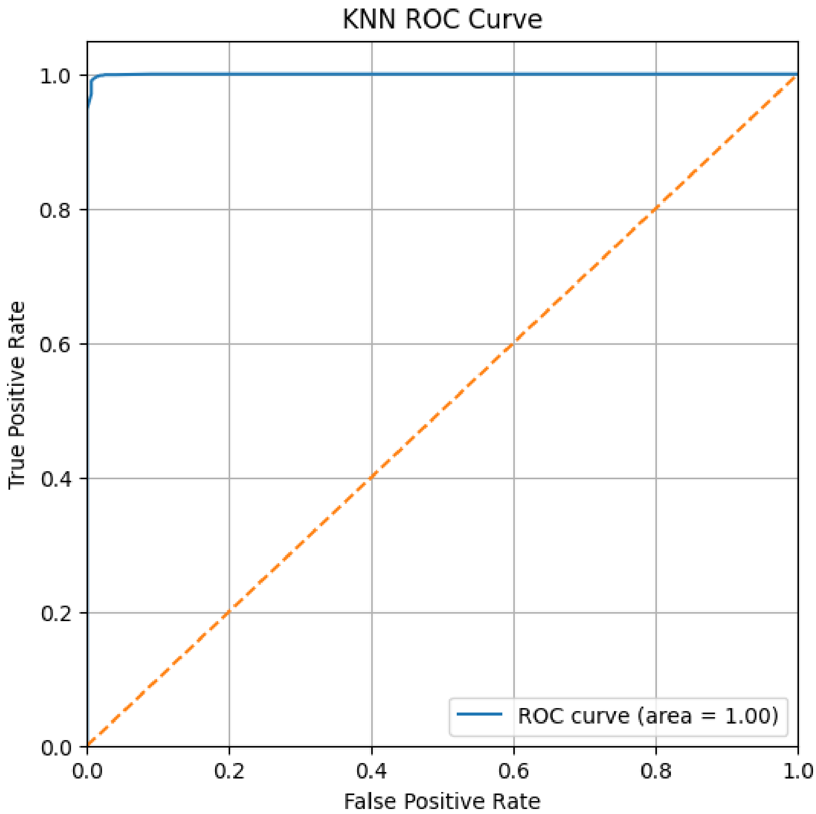
Mean metric avg values for 10 folds:

	RF	KNN	Conv1D
TP	489.100	487.400	486.100
TN	157.400	157.600	143.500
FP	2.500	2.300	16.400
FN	0.700	2.400	3.700
P	489.800	489.800	489.800
N	159.900	159.900	159.900
TPR	0.999	0.995	0.992
TNR	0.985	0.986	0.898
FPR	0.015	0.014	0.102
FNR	0.001	0.005	0.008
Precision	0.995	0.995	0.967
F1	0.997	0.995	0.980
Accuracy	0.995	0.993	0.969
Error_rate	0.005	0.007	0.031
BACC	0.992	0.991	0.945
TSS	0.983	0.981	0.890
HSS	0.987	0.981	0.914
AUC	0.999	0.997	0.992
BS	0.005	0.006	0.023
BSS	0.971	0.967	0.874

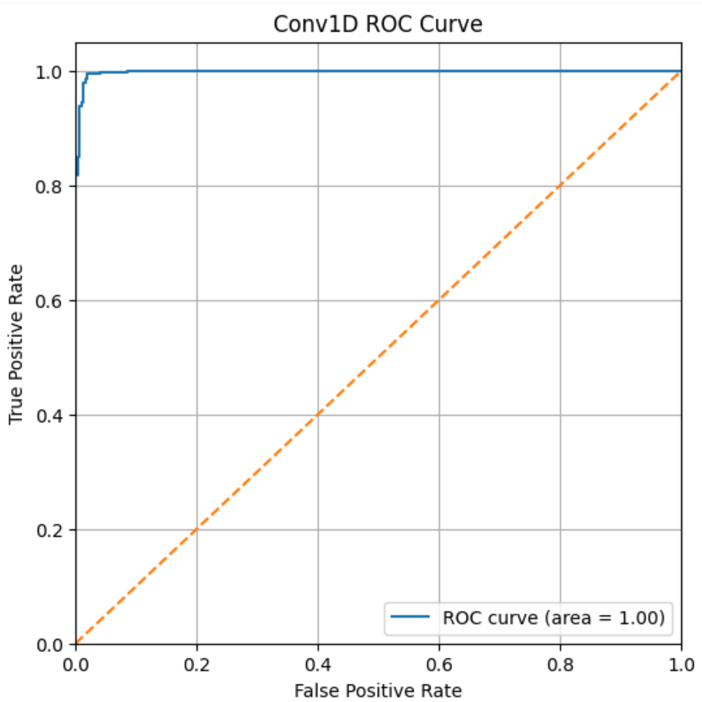
	fold8	fold9	fold10
TP	467.000	506.000	501.000
TN	150.000	123.000	130.000
FP	27.000	16.000	13.000
FN	5.000	4.000	5.000
P	472.000	510.000	506.000
N	177.000	139.000	143.000
TPR	0.989	0.992	0.990
TNR	0.847	0.885	0.909
FPR	0.153	0.115	0.091
FNR	0.011	0.008	0.010
Precision	0.945	0.969	0.975
F1	0.967	0.981	0.982
Accuracy	0.951	0.969	0.972
Error_rate	0.049	0.031	0.028
BACC	0.918	0.939	0.950
TSS	0.837	0.877	0.899
HSS	0.871	0.905	0.918
AUC	0.986	0.988	0.996
BS	0.037	0.028	0.018
BSS	0.812	0.836	0.895



The ROC curve for Random Forest almost hugs the top-left corner with an AUC of 1.00. which means it separates red and white wines almost perfectly. It keeps the true positive rate very close to 1.0 even when the false positive rate is near zero.



The KNN ROC curve is also very close to the top-left corner with AUC of 1.00, showing that it is nearly as strong as Random Forest. There is a tiny bit more curvature near the start, but it still maintains a very high TPR for low FPR values.



The Conv1D ROC curve rises quickly and stays high, with an AUC close to 1.00 as well, indicating excellent discriminative power. However, compared to RF and KNN, it dips slightly lower near the beginning, which matches its slightly higher false positive rate.

7 Discussion

Overall, all three models – Random Forest, KNN, and Conv1D – performed extremely well on the wine type classification task. The average accuracies over 10 folds are 0.995 for RF, 0.993 for KNN, and 0.971 for Conv1D, with RF having the best scores on almost every metric. Random Forest also reaches the highest recall (TPR = 0.999) and specificity (TNR = 0.985), meaning it almost never misses red wines and rarely mislabels white wines. Its F1 score (0.997), Balanced Accuracy (0.992), and skill scores (TSS = 0.983, HSS = 0.987) show that it is very reliable even with the moderate class imbalance. KNN is only slightly behind RF with very similar precision, F1, and BACC values, while Conv1D has noticeably lower TNR (0.904) and higher FPR (0.096), and its Brier Score (0.023) indicates less accurate probability estimates compared to RF and KNN (BS ≈ 0.005–0.006).

The ROC and AUC results support these observations. All three models have AUC values above 0.99 (RF = 0.999, KNN = 0.997, Conv1D = 0.991), which means they separate red and white wines almost perfectly. In the ROC plots, the curves for RF and KNN stay very close to the top-left corner, showing a high true positive rate even when the false positive rate is low. The Conv1D curve is still strong but slightly lower, matching its weaker TNR and higher FPR. These results suggest that the dataset is highly separable and that classical machine-learning methods, especially Random Forest and KNN, are able to capture the patterns in the physicochemical features more efficiently than the deeper Conv1D model in this case.

8 Conclusion

In this project I used the merged red and white wine dataset from Kaggle to build a binary classifier that predicts whether a wine is red or white based on its physicochemical properties. After encoding the wine type, standardizing the numerical features, and applying 10-fold cross-validation, I compared three algorithms: Random Forest, KNN, and a Conv1D neural network. The experiments showed that all three models achieved very high accuracy and AUC, but Random Forest consistently delivered the best overall performance, with the highest accuracy, F1 score, balanced accuracy, and the lowest error and Brier scores. KNN was a close second, while Conv1D, although still strong, did not outperform the simpler models.

These findings suggest that for structured tabular data like this wine dataset, traditional machine learning models are not only simpler but also highly effective. A well-tuned Random Forest could realistically be used in a real lab or production setting to automatically verify wine type from chemical measurements. In the future, this work could be extended by predicting wine quality levels instead of just type, performing more systematic hyperparameter tuning, and trying additional ensemble methods such as Gradient Boosting or XGBoost. This would help to see whether any further gains are possible beyond the already very strong results obtained in this study.

9 References

Link: https://www.kaggle.com/datasets/ehsanesmaeili/red-and-white-wine-quality-merged

10 GitHub Repository

Link: https://github.com/rajeevalahari/alahari_rajeevkumar_finalproject

[OPTIONAL]

For Supporting notes or clarifications and instructions to run the code visit github readme.