Wine Quality Prediction

University of Utah AI Bootcamp

🌍 *Brian Nelson, Eric Caldwell, Jaime Portillo, Nicholas Major*🌎

Instructor- Jason Stewart, Teacher Assistant-Shay Schreurs

July 9th, 2024

Project Overview And Goals

Our project aims to predict the quality of wine based on various chemical properties using machine learning techniques. Data optimization will be key to the projects success.

The goal is to build a robust predictive model that will classify wine quality ratings according to scientific standards.

Our approach was to build multiple models to take the base data and optimize it to predict the best scores using Random Forest Classifier, Gradient Boosting, Adaptive Boosting, Low/High Estimators, Logistic Regression, SVC Poly, and SVC Sigmoid

Once we got the baseline scores Random Forest Classifier was the highest balanced test score. So we decided to use that and built a function that searches through different Random Forest Classifier parameters to get the highest balance test score for the purpose of data optimization.

Executive Summary

Data Model Implementation: Our Jupyter notebook details the data extraction, cleaning, transformation, and exporting of the cleaned data as CSV files. A Python script effectively initializes, trains, and evaluates models, achieving ≥75% classification accuracy or 0.80 R-squared as required.

Data Model Optimization: The project employs iterative changes to model architecture, hyperparameters, and feature engineering, which is documented with clear performance metrics.

Project Outcome: Through structured implementation and optimization phases, the project delivers robust machine learning solutions, ensuring scalability and adaptability in predictive analysis tasks.

Project Challenges

The wine quality ratings are not uniformly distributed in the typical 1-10 class scale, with most wines receiving ratings between 5 and 7. This class imbalance can pose a challenge for the predictive model as the model might be biased towards the more frequently represented classes. To address this challenge, we used bins to categorize our values into “good” or “bad” instead of using just the 1-10 scale.

Determining if we want a model with more accuracy but works less of the time, or one with more accuracy but works more of the time? We decided to think about it from a wine company perspective. Determining that while accuracy is crucial, reliability (consistency of performance) is going to be of greater importance.

Finding more data with the same features to run through our models.

Data Collection Overview

Data was collected from UC Irvine (https://archive.ics.uci.edu/dataset/186/wine+quality)

We combined the red and white wine dataset values to have more data for utilization.

We used bin sampling and feature importance as part of the cleanup and exploration process.

We chose to use the balance test score as part of our data training and testing.

Preprocessing Overview

Importance of pre processing in ML

Steps involved in the pipeline

- Data Cleaning
- Binning
- Sampling
- Feature Importance

Binning

The process of transforming continuous variables into discrete categories or bins. Helping to simplify data and manage outliers.

- Purpose
  - Simplifying complex data
  - Reducing the effects of minor observation errors
  - Improving the performance and accuracy of ML models

In [12]:
```
# Create threshold for bad category
threshold = 6
df['quality'] = df['quality'].where(df['quality'] > threshold, other=0)

# Check quality column for unique value changes
print(df['quality'].unique())
```

```
[0 7 8 9]
```

In [13]:
```
# Create threshold for good category
threshold = 7
df['quality'] = df['quality'].where(df['quality'] < threshold, other=1)

# Check quality column for unique value changes
print(df['quality'].unique())
```

```
[0 1]
```

In [14]:
```
# Check value percentages
df['quality'].value_counts(normalize=True)
```

Out[14]:
```
quality
0    0.803448
1    0.196552
Name: proportion, dtype: float64
```

Sampling

The technique of selecting subsets of data from the larger dataset

- Purpose
  - Balances the dataset
  - Reduces computational load
  - Ensures representative distribution of data
  - Helps in dealing with class imbalance.

Sampling Implementation

```
# Set X and y variables to determine initial baseline score
X0 = df_initial.drop(columns=['quality','wine_type'])
y0 = df_initial['quality']
```

In [8]:

```
# Split data into training and testing data
X_train0,X_test0,y_train0,y_test0 = train_test_split(X0,y0,random_state=13)
```

Feature Importance

The techniques that assign a score to input features based on their importance to predict a target variable

Purpose

- Identify key predictors
- Help in dimensionality reduction

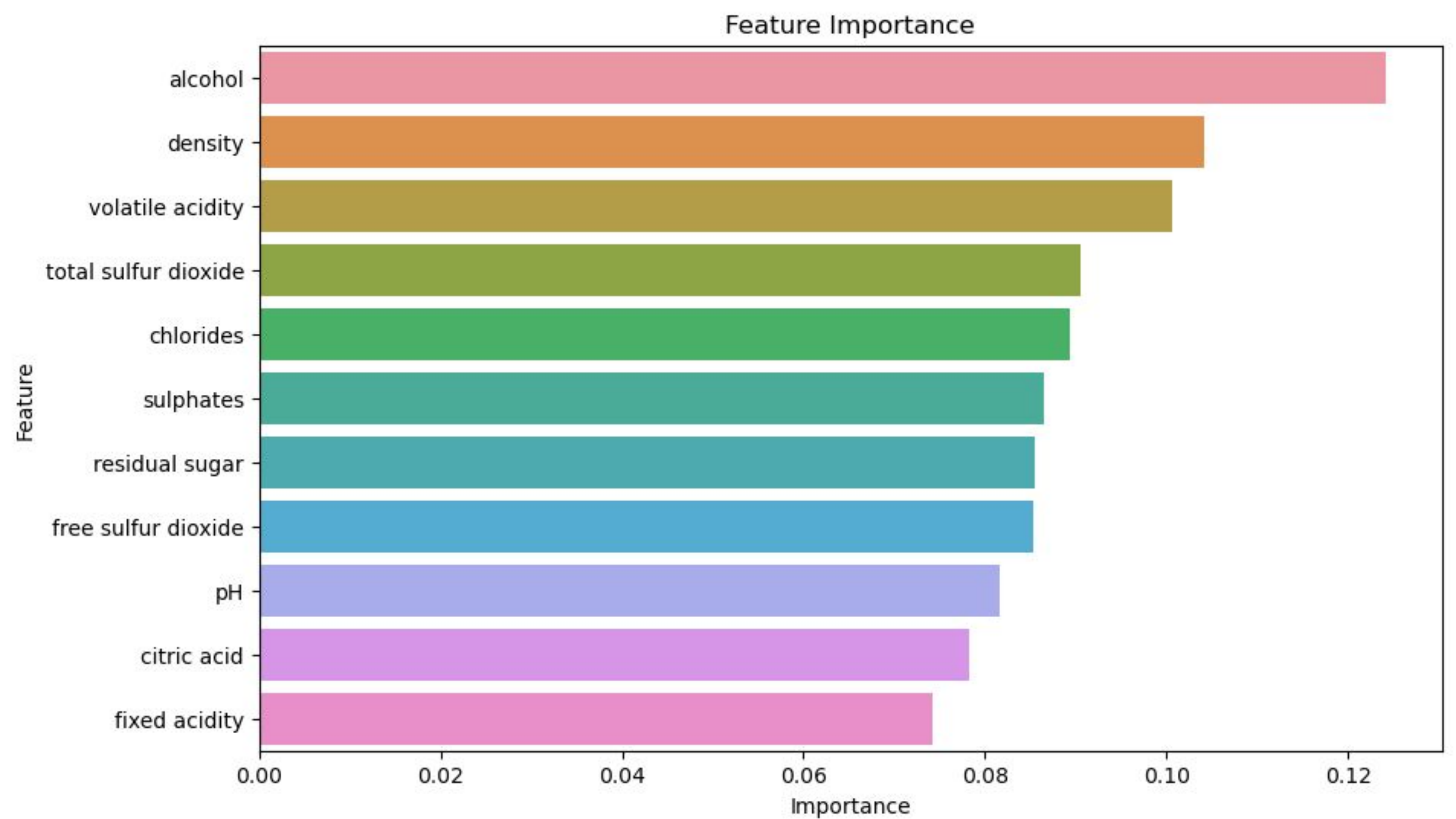
Implementation

- We used random forest to rank feature importance
- Removed features that were not carrying their weight

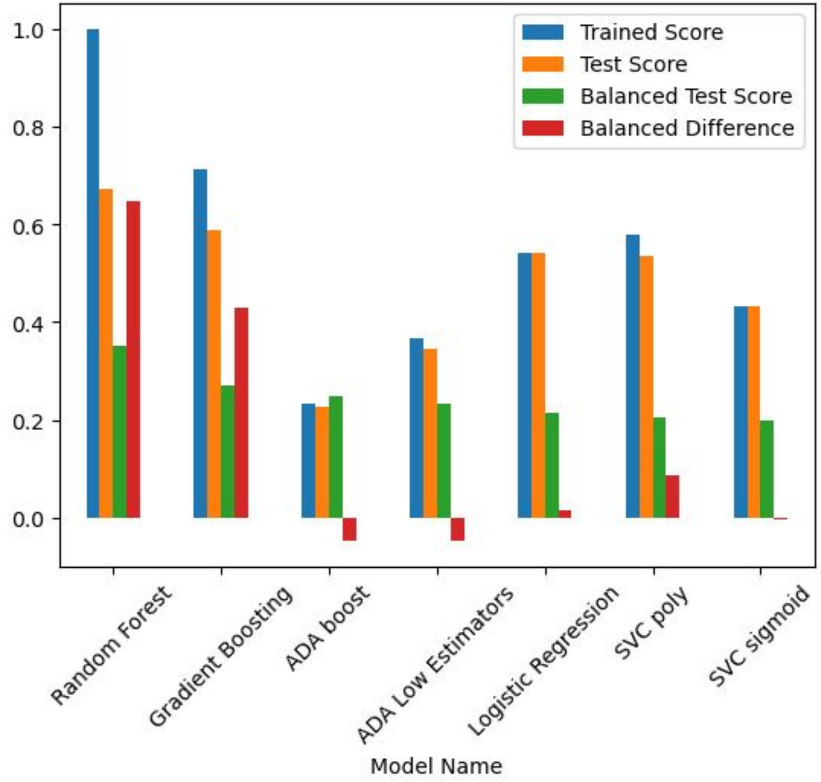
Why?

- Helps in understanding the dataset better
- Gives ideas for feature selection and engineering

```
Most Important Features:
alcohol: 0.1241
density: 0.1041
volatile acidity: 0.1005
total sulfur dioxide: 0.0906
chlorides: 0.0894
sulphates: 0.0865
residual sugar: 0.0854
free sulfur dioxide: 0.0853
pH: 0.0816
citric acid: 0.0782
fixed acidity: 0.0743
```


Feature Importance
alcohol
density
volatile acidity
total sulfur dioxide
chlorides
sulphates
residual sugar
free sulfur dioxide
pH
citric acid
fixed acidity
Feature
0.00
0.02
0.04
0.06
0.08
0.10
0.12
Importance

Baseline Scores



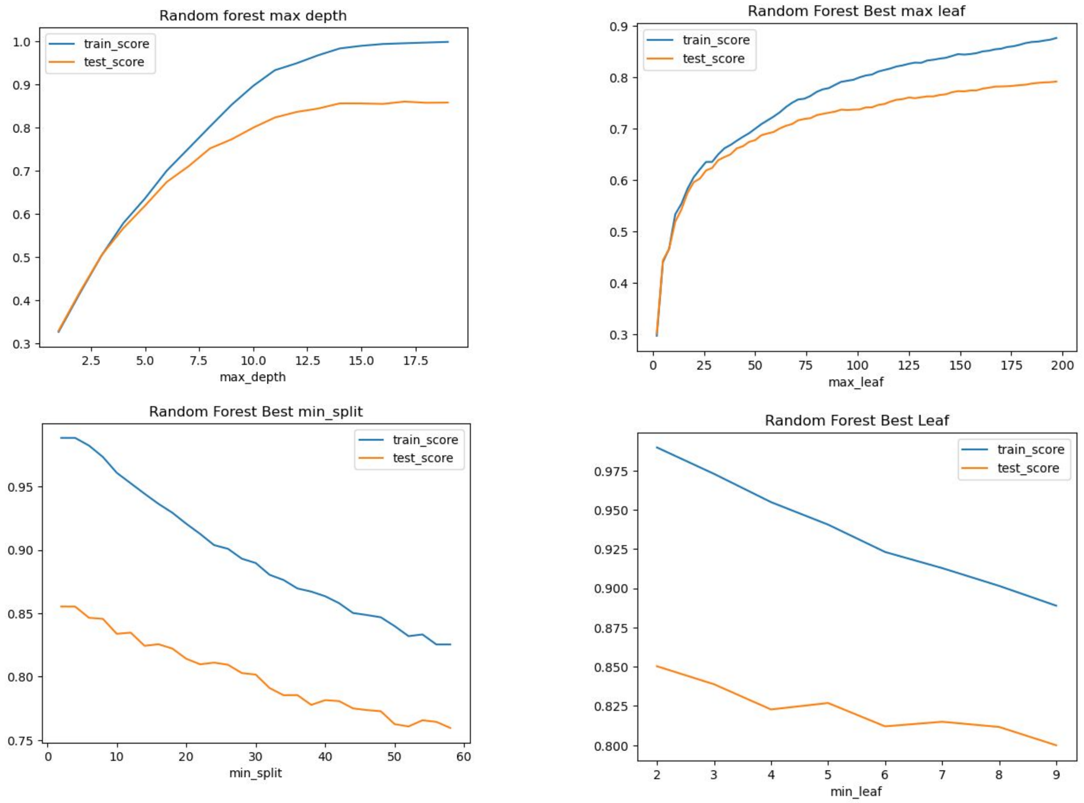
Random Forest
Test Accuracy: 0.6726153846153846
balanced test score: 0.3510074145712444
classification report:

	precision	recall	f1-score	support
3	0.00	0.00	0.00	7
4	0.64	0.15	0.24	48
5	0.70	0.74	0.72	528
6	0.64	0.77	0.70	705
7	0.71	0.46	0.56	282
8	0.95	0.33	0.49	54
9	0.00	0.00	0.00	1
accuracy			0.67	1625
macro avg	0.52	0.35	0.39	1625
weighted avg	0.68	0.67	0.66	1625

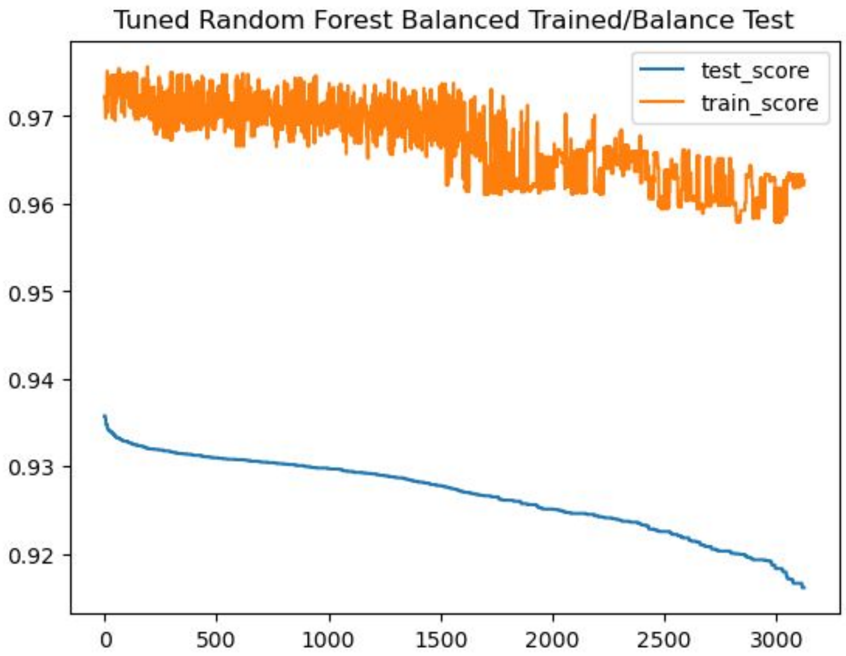
	Trained Score	Test Score	Balanced Test Score
Model Name			
Random Forest	1.000000	0.672615	0.351007
Gradient Boosting	0.713054	0.588923	0.269941
ADA boost	0.234606	0.227077	0.249291
ADA Low Estimators	0.366174	0.344000	0.232777
Logistic Regression	0.542693	0.540923	0.215561
SVC poly	0.580665	0.534769	0.206155
SVC sigmoid	0.433703	0.433846	0.200261

	Balanced Difference
Model Name	
Random Forest	0.648993
Gradient Boosting	0.430798
ADA boost	- 0.048019
ADA Low Estimators	- 0.046457
Logistic Regression	0.015383
SVC poly	0.086844
SVC sigmoid	-0.004813

Best Parameters



Best Results Parameter Search



best results:

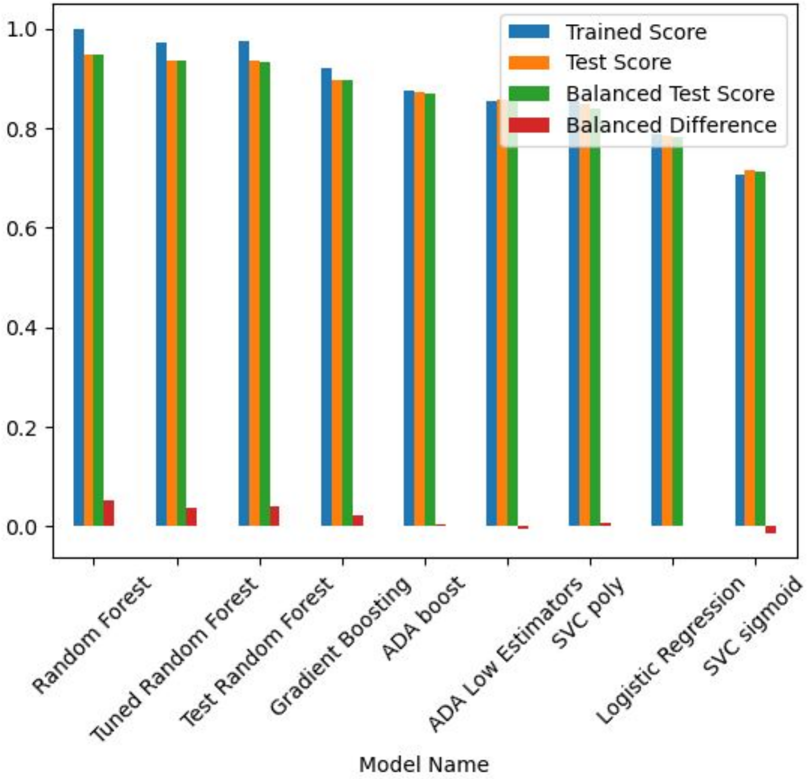
test_score	n_estimators	max_depth	min_split	max_leaf	min_samples_leaf
0.935772	20	13	2	197	2
0.935772	20	13	4	197	2
0.935662	20	13	2	194	2
0.935662	20	13	4	194	2
0.935381	20	13	4	185	2
...	...	...	...	...	...
0.916143	22	17	2	191	5
0.916143	22	17	6	191	5
0.916143	22	17	10	188	5
0.916143	22	17	2	188	5
0.916143	22	17	10	191	5

test_score	train_score
0.935772	0.972253
0.935772	0.972253
0.935662	0.971826
0.935662	0.971826
0.935381	0.969773
...	...
0.916143	0.962649
0.916143	0.962649
0.916143	0.962139
0.916143	0.962139
0.916143	0.962649

[3125 rows x 6 columns]

Top Best Parameters:
best n_estimators: 20
best max_depth: 13
Best min_split: 2
best max_leaf: 197
best min_leaves: 2

Final Scores



Best Random Forest Tuned Parameters Scores
Test Accuracy: 0.9362637362637363
balanced test score: 0.9357724974610844
classification report:

	precision	recall	f1-score	support
0	0.95	0.94	0.94	1278
1	0.92	0.93	0.93	997
accuracy			0.94	2275
macro avg	0.93	0.94	0.94	2275
weighted avg	0.94	0.94	0.94	2275

updated comparison DataFrame

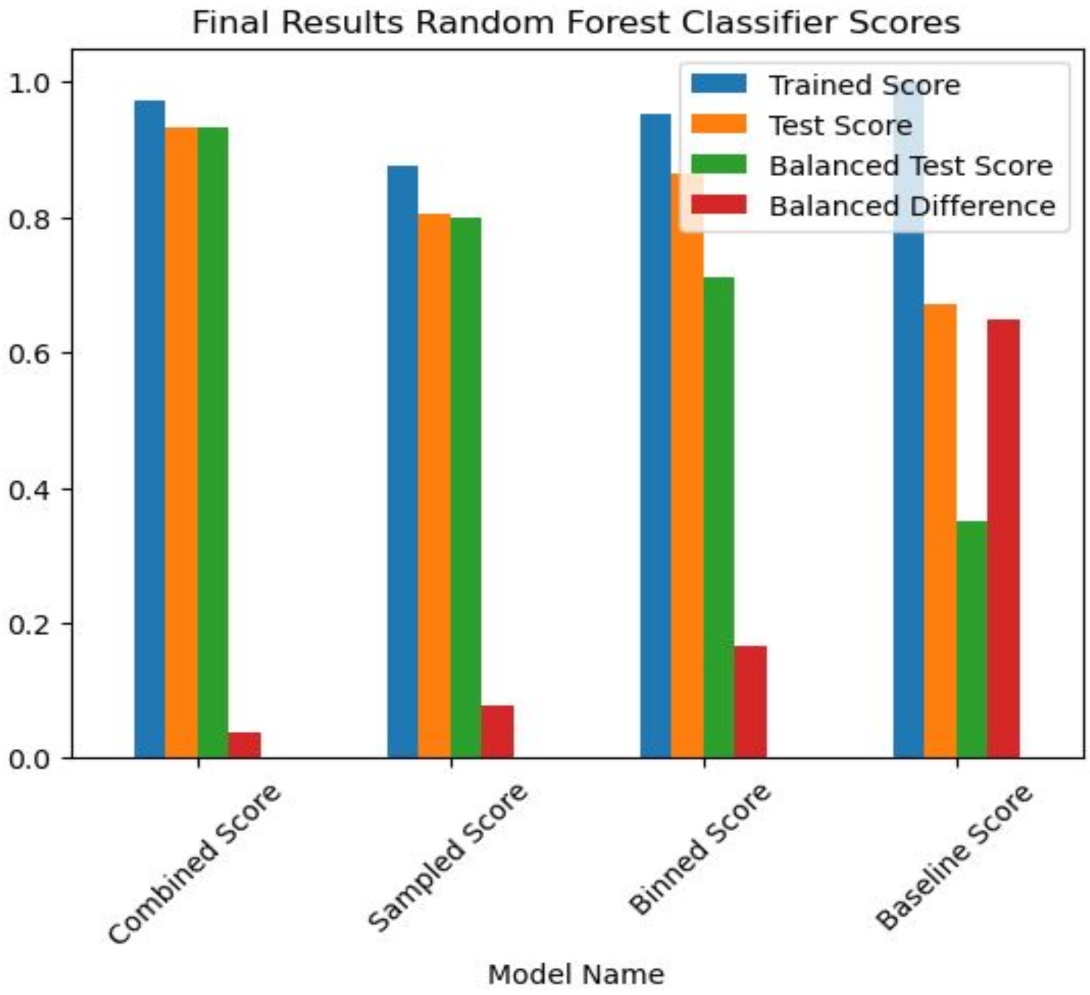
Model Name	Trained Score	Test Score	Balanced Test Score
Random Forest	1.000000	0.947253	0.946656
Tuned Random Forest	0.972894	0.936264	0.935772
Test Random Forest	0.975238	0.934505	0.933546
Gradient Boosting	0.919414	0.897143	0.896873
ADA boost	0.876337	0.871648	0.868447
ADA Low Estimators	0.854799	0.858022	0.854113
SVC poly	0.853919	0.846593	0.837656
Logistic Regression	0.788278	0.785495	0.782723

Model Name	Balanced Difference
Random Forest	0.053344
Tuned Random Forest	0.036481
Test Random Forest	0.040881
Gradient Boosting	0.021510
ADA boost	0.003236
ADA Low Estimators	-0.006586
SVC poly	0.005751
Logistic Regression	-0.000368

Final Data Set Scores

UCI Wine Quality Dataset(red/white combined)

	Trained Score	Test Score	Balanced Test Score	Balanced Difference
Initial	1.000000	0.672615	0.351007	0.648993
Binned	0.951149	0.865231	0.711225	0.167036
Sampled	0.883516	0.813626	0.808155	0.076632
Sampled and Binned	0.972601	0.933626	0.932433	0.039073


Final Results Random Forest Classifier Scores
Trained Score
Test Score
Balanced Test Score
Balanced Difference
1.0
0.8
0.6
0.4
0.2
0.0
Combined Score
Sampled Score
Binned Score
Baseline Score
Model Name

Analysis and Conclusion

- The Initial model shows overfitting and poor handling of class imbalance.
- The Binned model shows significant improvement in generalization and handling of class imbalance.
- The Sampled model shows good performance and handles class imbalance well, with a very small balanced difference.
- The Sampled and Binned model shows the best overall performance, generalizing well to the test data and handling class imbalance effectively, with the smallest balanced difference.
- The Sampled and Binned model is the most robust and well-performing approach based on these metrics

Questions & Challenges

Additional Questions

- How will climate change affect wine quality?
  - No linking data available to add temperature and weather
- What regions have the best wine quality?
  - Data not available to create correlation between wine and region