Machine Learning Binary Classification of Glioma Grades.

Aiman Madan, Ken Lopez Martinez, & Cristian Enriquez

Having a Machine Learning Model that can accurately classify the grade of the glioma can have a significant impact on both the clinics and patients. Having an early diagnosis can improve the survivability rate of the patient. It would be faster and cheaper for both the clinic and patient, and would be less invasive for the patient as well.

The **Mean Image Transformation method proposed by Yathirajam and Gutta (2024)** introduced a novel approach to glioma grade classification. By calculating the mean MRI sequences for each patient, they encapsulated tumor characteristics in a single image. This method improved performance metrics such as F1-score, precision, and recall while requiring fewer computational resources. Moreover, features extracted from mean-based CNNs further enhanced the performance of traditional machine learning models like Support Vector Machines and Gradient Boosting, showcasing the potential of combining deep learning and traditional machine learning approaches.

Building upon these prior studies, our project adopts and extends the Mean Radiomic Feature approach, applying it to the radiomic data derived from MRI scans. This enables us to address the challenges of class imbalance, high feature dimensionality, and computational overhead while improving glioma grade classification.

The primary goal of this project is to develop a machine learning model that is capable of distinguishing between high grade and low grade glioma based on radiomic features extracted from MRI scans.

The dataset utilized for this project is the BraTS 2020 Training Dataset.

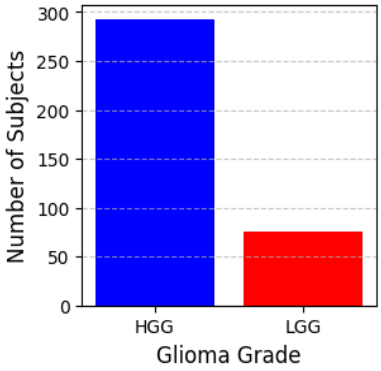
Name Mapping CSV:

This CSV file serves as the primary source for target labels, mapping each patient's Subject ID to their respective glioma classification (High-Grade Glioma - HGG or Low-Grade Glioma - LGG).

This file ensures a clear and consistent relationship between patient identifiers and their diagnostic classes.

Radiomic Features (Provided by Dr. Gutta) :

Radiomic features extracted from the four MRI sequences **(T1, T1CE, T2, and FLAIR)** were the basis for our model development. These features quantify complex characteristics such as texture, intensity, shape, and patterns within the MRI images.
By focusing on these features, we aimed to capture the core characteristics of gliomas while minimizing the dimensionality of raw image data.



To prepare the dataset for machine learning models, we implemented the following pre-processing steps:

- **Removal of unnecessary columns**. (Some columns had repeating values and also the image column was a directory path which is not required for our training)
- **Standardization and normalization** of features across the datasets.
- **Mean Radiomic Feature** (Aiman Madan):
  Retrieved the mean of the radiomic features across all four sequences for each patient.
- **Concatenated Radiomic Features** (Cristian Enriquez):
  A different approach compared to the Mean Radiomic Features. Concatenates the data of the four different sequences of a subject ID into a single row. This is done for every subject ID in the data set.
- **Weighted Radiomic Feature**s (Ken Lopez):
  Instead of taking a simple mean across T1, T2, T1CE, and FLAIR columns, we assigned weights to each modality based on individual accuracy and relevance to glioma grade classification.

To address the class imbalance in our dataset, where High-Grade Gliomas (HGG) significantly outnumber Low-Grade Gliomas (LGG), we employed multiple strategies to balance the dataset and improve the performance of our machine learning model

- **Data Augmentation**: Selected and enhanced relevant information from the radiomic features, adding noise and generalizing the data representation. This approach added variability to the dataset without introducing a lot of redundancy.
- **Oversampling with SMOTE**: Used the Synthetic Minority Oversampling Technique (SMOTE) to generate synthetic samples for the minority class (Low-Grade Gliomas). This balanced the dataset and reduced the bias toward the majority class during training.
- **Calculate Class weights:** Added weights to the minority class to handle class imbalance. This idea assigns higher importance to the minority class during training, by using a weight proportional to the inverse frequency of each class. This helps the model give more focus on the minority class without altering the dataset.

The algorithms used were:

Support Vector Classifier
The main reasons for choosing a Support Vector Classifier are that it works well for Binary Classification and pairs well with our dataset since we want to classify between two target classes: HGG and LGG. Additionally it can handle the large amount of features that are present in our data which makes it perfect for preventing overfitting.

Decision Trees
A Decision Tree is great for easily interpreting data thanks to the visual graph that allows us to see how the model is reaching a certain target. Compared to the other two machine learning algorithms that we are using, a Decision Tree is less complex and computationally inexpensive

Gradient Boosting
Gradient boosting is also a good candidate model due to the process of adjusting weights in each iteration of fitting a model. This process makes the model robust against class imbalance which is present in our dataset.

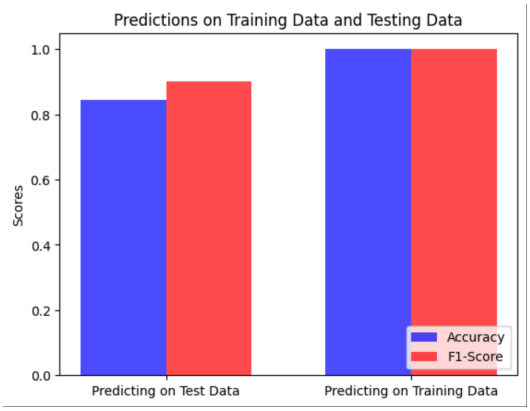
Experiments

Metrics Used

Before explaining our experiments, it is important to discuss the metric utilized to tune our models. In our case, we aimed to improve F1 scores because we want to reduce the amount of both false LGG and false HGG. It is important to always classify Glioma with the correct grade to give a patient the right prescriptions that they need. For instance, if we misclassify a patient as LGG when they are actually HGG, we risk giving a patient that needs intensive treatment very little treatment and leaving their condition to worsen over time; If we misclassify a patient as HGG when they are actually LGG, we risk giving a patient a treatment that is way more intensive and intrusive for their bodies and increasing the possibility harm.

Predicting on Training set vs Testing set

One of the things to note about our current implementation is that, although the results for predicting on the testing set look normal enough, a glaring issue becomes evident when predicting on the training set itself. That is, our models are overfitting and thus always have an accuracy and f1 score of 100% on the training data. The following graph shows the clear gap between results of the training data and testing set for the Decision Tree model:




```
Predicting on Normal Testing Data

Accuracy:  0.8243243243243243  F1:  0.8869565217391304
Confusion Matrix:
             Predicted LGG  Predicted HGG
Actual LGG              40             23
Actual HGG              29            204

Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.63      0.61        63
           1       0.90      0.88      0.89       233

    accuracy                           0.82       296
   macro avg       0.74      0.76      0.75       296
weighted avg       0.83      0.82      0.83       296
```

```
Predicting on Training Data

Accuracy:  1.0  F1:  1.0
Confusion Matrix:
             Predicted LGG  Predicted HGG
Actual LGG             241              0
Actual HGG               0            939

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       241
           1       1.00      1.00      1.00       939

    accuracy                           1.00      1180
   macro avg       1.00      1.00      1.00      1180
weighted avg       1.00      1.00      1.00      1180
```

Although we are only showing the results for a single machine learning model, overfitting occurs on all three of our models and even on experiments. Thus, results of predicting on the training set are not shown in the experiments below since we can expect similar results to the ones shown in the images above.

Hyper Parameter Tuning

To improve the efficiency of our models, we experimented with fine tuning each of the hyperparameters for our models by utilizing RandomizedSearchCV. RandomizedSearchCV was utilized due to its cross validation technique which works well for unbalanced datasets and because it is computationally faster due to the fact that it only looks at a small subset of hyperparameter combinations. Experiments with tuning hyperparameters consisted of taking screenshots of the results that RandomizedSearchCV would give us everytime we ran it and then

analyzing the best models and best hyperparameters along with the confusion matrix and the classification report. After analyzing we would modify our list of hyperparameters so it only included the ones that performed relatively well for our models. This process was repeated until improvements reached a peak.

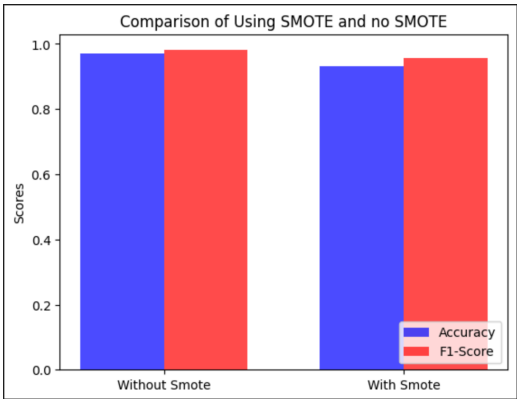
The following screenshots demonstrate how much better GradientBoosting performed after tuning the hyperparameters in which it was able to predict 11 more LGG and 1 more HGG accurately. Similar improvements occurred for all models.

```
Predicting on Normal Testing Data without Tuning

Accuracy:  0.9324324324324325  F1:  0.9585062240663901
Confusion Matrix:
             Predicted LGG  Predicted HGG
Actual LGG              45             18
Actual HGG               2            231

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.71      0.82        63
           1       0.93      0.99      0.96       233

    accuracy                           0.93       296
   macro avg       0.94      0.85      0.89       296
weighted avg       0.93      0.93      0.93       296
```

```
Predicting on Normal Testing Data after Tuning

Accuracy:  0.972972972972973  F1:  0.9830508474576272
Confusion Matrix:
             Predicted LGG  Predicted HGG
Actual LGG              56              7
Actual HGG               1            232

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.89      0.93        63
           1       0.97      1.00      0.98       233

    accuracy                           0.97       296
   macro avg       0.98      0.94      0.96       296
weighted avg       0.97      0.97      0.97       296
```

Over Sampling

As previously mentioned, our data suffers from data imbalance so one of the first experiments we did was to utilize SMOTE to oversample our data and provide a more balanced set. To our surprise, this method actually performed worse when predicting on the testing set as shown by the following graphs where the f1 scores of using SMOTE were slightly worse than the scores of not using it.

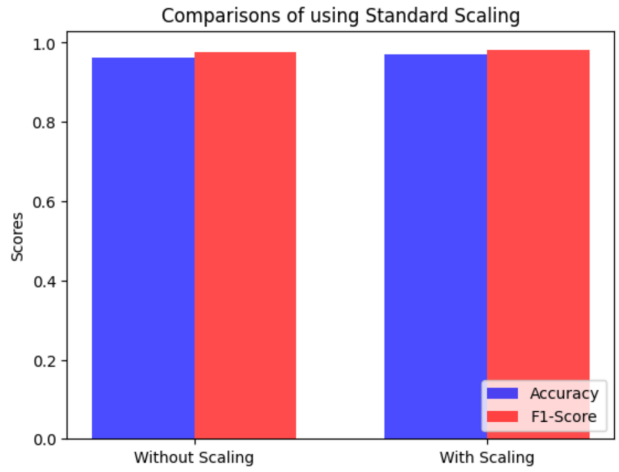


```
Predicting on Normal Testing Data

Accuracy:  0.9695945945945946  F1:  0.9809725158562368
Confusion Matrix:
            Predicted LGG  Predicted HGG
Actual LGG             55              8
Actual HGG              1            232

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.87      0.92        63
           1       0.97      1.00      0.98       233

    accuracy                           0.97       296
   macro avg       0.97      0.93      0.95       296
weighted avg       0.97      0.97      0.97       296
```

```
Predicting on Testing Data with only SMOTE

Accuracy:  0.9324324324324325  F1:  0.9565217391304348
Confusion Matrix:
            Predicted LGG  Predicted HGG
Actual LGG             56              7
Actual HGG             13            220

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.89      0.85        63
           1       0.97      0.94      0.96       233

    accuracy                           0.93       296
   macro avg       0.89      0.92      0.90       296
weighted avg       0.94      0.93      0.93       296
```

Standard Scaling

By using the StandardScaler method from sci-kit learn we were able to compute the mean and standard deviation of each datapoint in our dataset, effectively normalizing it. This process is perfect for our data with many features as it helps making predicting and computing results slightly more efficient and faster. When testing, we noticed that standard scaling generally improves the efficiency of our models even if only by a slight margin. This can be demonstrated by analyzing the following images in which the scores of f1 are higher by less than 1%:

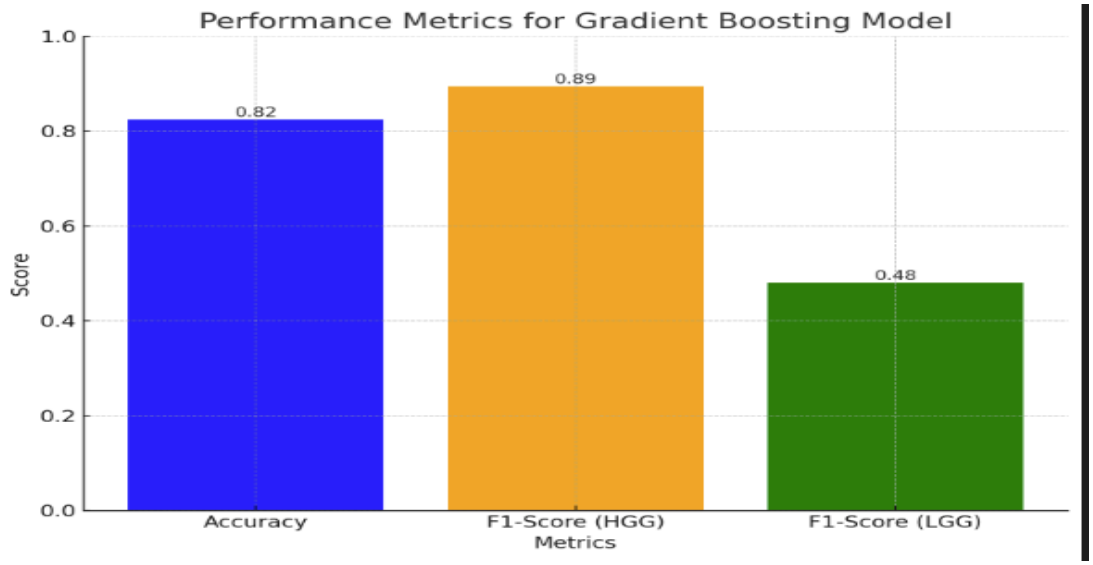



```
Predicting on Normal Testing Data

Accuracy:  0.9628378378378378  F1:  0.9767441860465116
Confusion Matrix:
            Predicted LGG  Predicted HGG
Actual LGG             54              9
Actual HGG              2            231

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.86      0.91        63
           1       0.96      0.99      0.98       233

    accuracy                           0.96       296
   macro avg       0.96      0.92      0.94       296
weighted avg       0.96      0.96      0.96       296
```

```
Predicting on Testing Data with only Standard Scaler

Accuracy:  0.9763513513513513  F1:  0.9851380042462845
Confusion Matrix:
            Predicted LGG  Predicted HGG
Actual LGG             57              6
Actual HGG              1            232

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.90      0.94        63
           1       0.97      1.00      0.99       233

    accuracy                           0.98       296
   macro avg       0.98      0.95      0.96       296
weighted avg       0.98      0.98      0.98       296
```

Mean Radiomic Feature

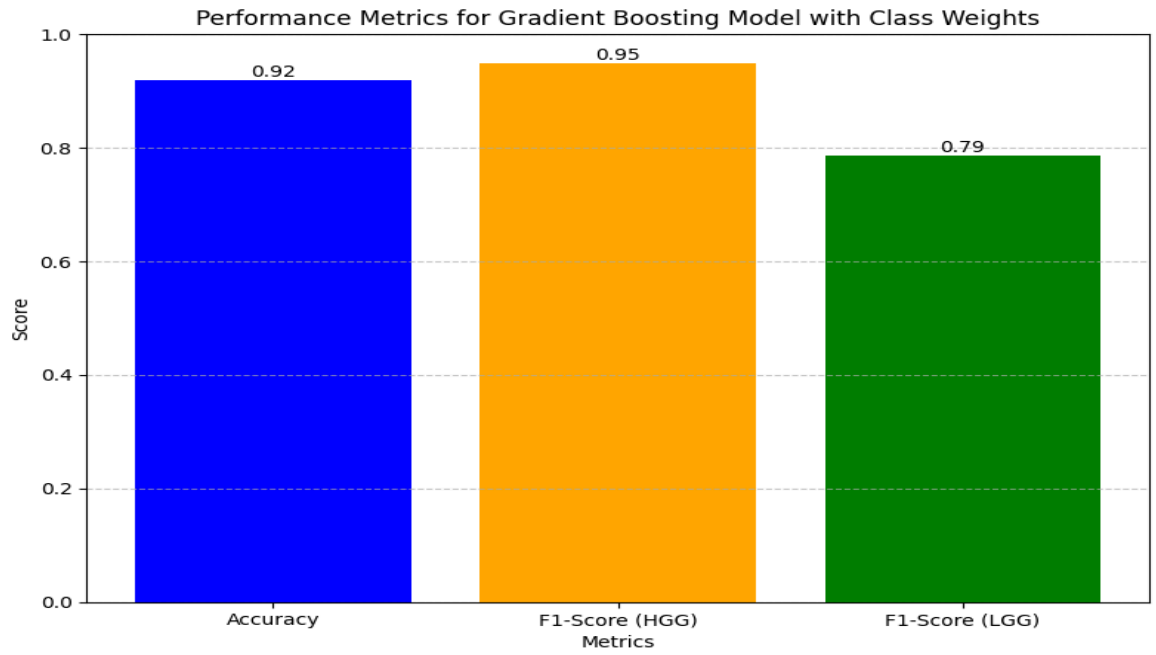
For each patient (subject ID), we had radiomic features derived from four MRI sequences: T1, T1CE, T2, and FLAIR. Instead of treating the features from each sequence separately, we calculated the mean value of the radiomic features across these four sequences for each patient. This method created a single, averaged representation of radiomic features for each subject using the MRI sequences.



This analysis is important because although we have an overall 82% accuracy, we are able to highlight the model's limitations in handling class imbalance, which can lead to biased predictions favoring the majority class. The model has a low F1-score for the minority class (LGG) indicates poor performance in identifying less frequent but equally critical cases. Addressing this imbalance is crucial for ensuring the model's reliability and fairness, especially in applications where minority class predictions are vital medical diagnoses.

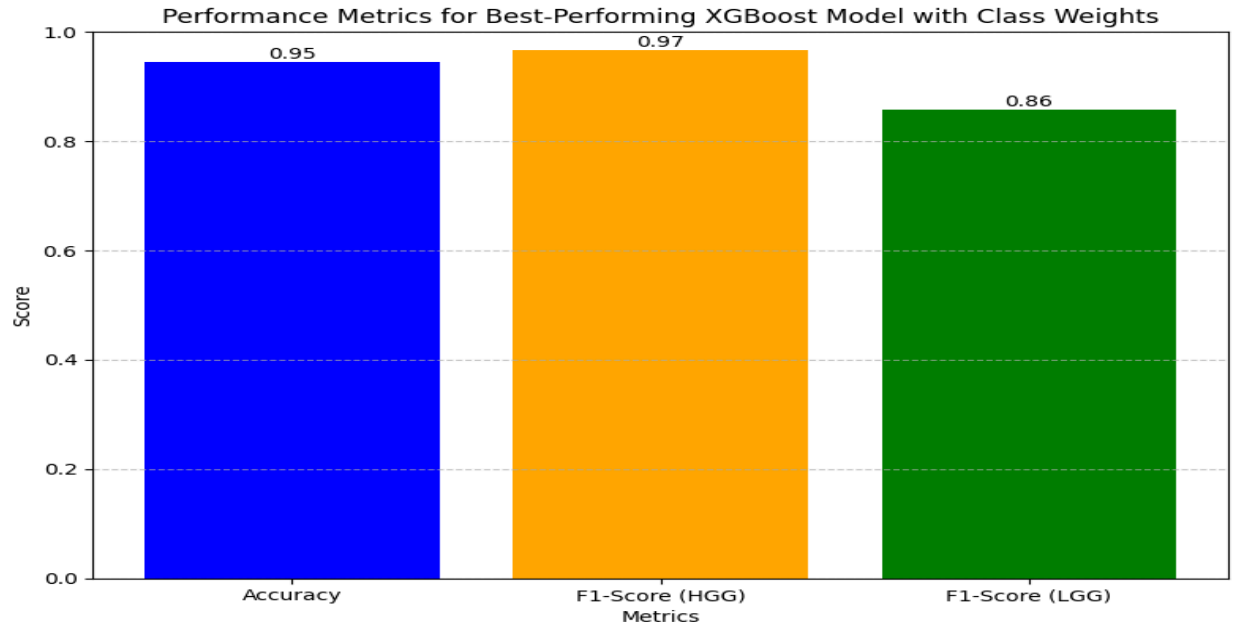
Weighted Classes and Sequences

Adding weights to the minority class, in this case Low Grade Glioma, addresses the class imbalance by having the model pay more attention to the underrepresented samples.



This was simply from adding weights to the minority class. We are able to preserve the dataset from any changes, and we drastically increased the f1 score for LGG

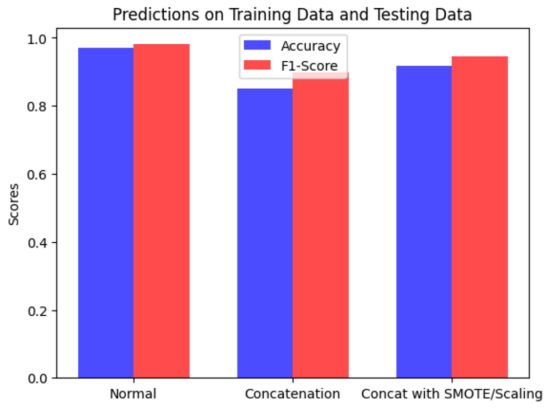
We then test the sequences separately and add weights based on their performance, this allows us to have a deeper understanding of how each imaging sequence contributes to the models overall accuracy, precision and recall. We are able to identify that T1CE and FLAIR are the two best performing sequences. The worst performing are T1 and T2, respectively. We assign weights proportional to their performance, which enables the model to focus more on the sequences that are more predictive.



Together, these strategies create a balanced, performance-optimized model that effectively takes class specific and sequence weighted insights to address the challenges of imbalanced datasets. This approach enhances the model's fairness and accuracy for both classes, meanwhile keeping the integrity of the dataset.

Concatenation

With concatenation, we preprocessed the data so that the four rows of the different sequences of each subject id became concatenated into a single row. Thus, the total number of rows dropped down by 75%. With this approach we will be demonstrating the results for two different experiments: we experimented with concatenation and no other preprocessing (which we will be referring to as concat) and concatenation with both Scaling and SMOTE (we will be referring to as concatBoth). Between both of these experiments, concatBoth generally did better because it had an f1 score of 93% which was higher than the f1 score of 90% from concat. Despite this, concatenation actually performed worse when compared to a normal approach which had a higher f1 score of 98%.




```
Predicting on Normal Testing Data

Accuracy:  0.9695945945945946  F1:  0.9809725158562368
Confusion Matrix:
            Predicted LGG  Predicted HGG
Actual LGG             55              8
Actual HGG              1            232

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.87      0.92        63
           1       0.97      1.00      0.98       233

    accuracy                           0.97       296
   macro avg       0.97      0.93      0.95       296
weighted avg       0.97      0.97      0.97       296
```

```
Predicting on Testing Data after Concatenating

Accuracy:  0.8648648648648649  F1:  0.9090909090909091
Confusion Matrix:
            Predicted LGG  Predicted HGG
Actual LGG             14              5
Actual HGG              5             50

Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.74      0.74        19
           1       0.91      0.91      0.91        55

    accuracy                           0.86        74
   macro avg       0.82      0.82      0.82        74
weighted avg       0.86      0.86      0.86        74
```

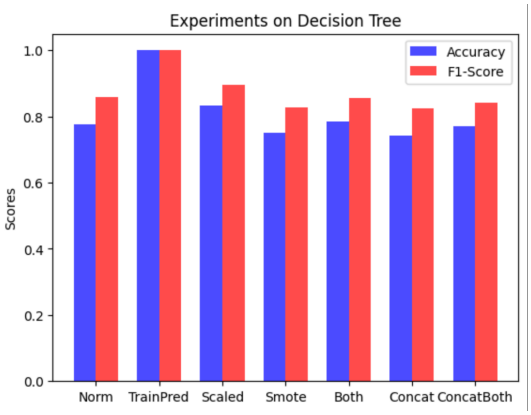
```
Predicting on Testing Data after Concatenating with SMOTE and Scaling

Accuracy:  0.9054054054054054  F1:  0.9369369369369369
Confusion Matrix:
            Predicted LGG  Predicted HGG
Actual LGG             15              4
Actual HGG              3             52

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.79      0.81        19
           1       0.93      0.95      0.94        55

    accuracy                           0.91        74
   macro avg       0.88      0.87      0.87        74
weighted avg       0.90      0.91      0.90        74
```

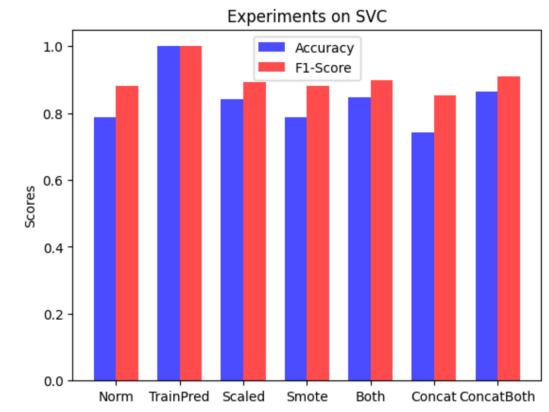
Model Comparisons

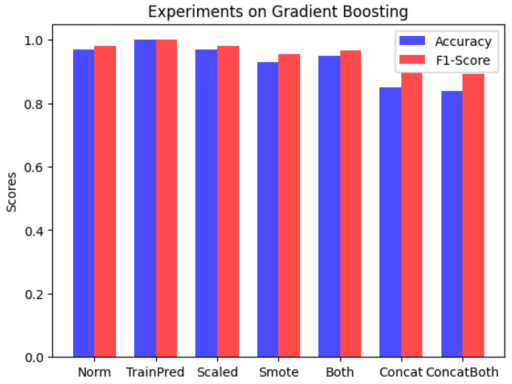
Having discussed the experiments taken, we can now compare how each of the machine learning models compare to each other. For each of the models we used the following hyperparameters to fit the model that were obtained after tuning with RandomSearchCV:

Decision Tree: max_depth=30, splitter= 'random', random_state=42
Support Vector Classifier: C=1000, class_weight = 'balanced', gamma = 0.001
GradientBoostingClassifier: learning_rate=0.2, min_samples_leaf=2, n_estimators=200, subsample=0.8

Before discussing the graphs below, it is important to discuss each of the labels on the x-axis:
Norm: predicting the test set on normal data with no preprocessing.
TrainPred: predicting on the training data set with no preprocessing.
Scaled: predicting on the test set after applying only the Standard Scaler.
Smote: predicting on the test set after applying only SMOTE.
Both: predicting on the test set after applying both Standard Scaler and Smote.
Concat: predicting on the test set after applying the concatenation strategy.
ConcatBoth: predicting on the test set after applying concatenation, Standard Scaler and Smote.







Best results for Decision Tree

Accuracy: 0.8344594594594594 F1: 0.8946236559139785
Confusion Matrix:

	Predicted LGG	Predicted HGG
Actual LGG	39	24
Actual HGG	25	208

Classification Report:

	precision	recall	f1-score	support
0	0.61	0.62	0.61	63
1	0.90	0.89	0.89	233
accuracy			0.83	296
macro avg	0.75	0.76	0.75	296
weighted avg	0.84	0.83	0.83	296

Best results for SVC

Accuracy: 0.8918918918918919 F1: 0.9259259259259259
Confusion Matrix:

	Predicted LGG	Predicted HGG
Actual LGG	16	3
Actual HGG	5	50

Classification Report:

	precision	recall	f1-score	support
0	0.76	0.84	0.80	19
1	0.94	0.91	0.93	55
accuracy			0.89	74
macro avg	0.85	0.88	0.86	74
weighted avg	0.90	0.89	0.89	74

Best results for Gradient Boosting

Accuracy: 0.9797297297297297 F1: 0.9872340425531915
Confusion Matrix:

	Predicted LGG	Predicted HGG
Actual LGG	58	5
Actual HGG	1	232

Classification Report:

	precision	recall	f1-score	support
0	0.98	0.92	0.95	63
1	0.98	1.00	0.99	233
accuracy			0.98	296
macro avg	0.98	0.96	0.97	296
weighted avg	0.98	0.98	0.98	296

By analyzing the graphs, the confusion matrix and the classification reports, we can give a ranking for each of the models used:
The model with the worst performance is the Decision Tree after applying only Standard Scaling. It has the worst f1 score of 89% and for each of the experiments performed it generally had the worse results compared to the other models with a few exceptions
Meanwhile, the model with the best performance was Gradient Boosting without applying any preprocessing (no SMOTE and no Standard Scaling). It had the best f1 score of 98%
The Support Vector Classifier lies in the sweet spot in between both of these models where it performed best after applying the concatenation strategy as well as both SMOTE and Standard Scaling. It had an f1 score of 92%.

Result Analysis

There are a few different things that stand out from analyzing our results and that we believe deserve doing more research on.
First, it is interesting to see that utilizing SMOTE does not improve our scores and instead brings them down despite its main purpose being to solve the issue of unbalanced datasets. It is possible that the synthetic data created is not truly representative nor accurate of the full length of features utilized to determine whether a glioma is high grade or low grade. Oddly enough, when it comes to the support vector machine model there is a weird occurrence as using the *ConcatBoth* settings actually resulted in the highest scores despite these generally being one of the worst performing strategies discussed in previous sections. It could be an issue with overfitting in our model or with the way these preprocessing steps interact with the hyperparameter models, it is something to investigate further.

When comparing our results to **Mean Image Transformation method proposed by Yathirajam and Gutta (2024)**, the order of the best models are also very similar to those described in the article as the model with the best performance is Gradient Boosting (in their paper they used XG Boost), the one with the worst performance is Decision Tree, and Support Vector Machine falls somewhere in the middle for both studies. One other interesting detail is that when comparing actual scores, our scores for SVC and Decision Trees fall slightly under the scores presented in their paper but our scores for Gradient Boosting are equal with both standing with a 98% f1 score. However this brings up our next point.

As previously mentioned when predicting our models on the training set, our current implementation is prone to overfitting as it is correctly identifying every single target despite the imbalance in our data. This makes us wonder if it is possible that this overfitting is also affecting how accurate the results when predicting on the test data set really are. It is possible that they are worse or even better for some models. Thus, a future improvement that we can try to solve is this very issue of overfitting.

Aiman Madan:

- Initialized the set up for environments
- Download the data
- Initialized project overview
- Preprocessed data
- Applied the mean method
- Tested using Augmentation & Noise
- Tested using SMOT

Ken Lopez:

- Contributed to project overview and final report
- Initialized progress report
- Preprocessed data, sequence mean, sequence weights
- Hypertuning, minority class weights

Cristian Enriquez:

- Initialized testing for the three algorithms (svm, decision tree, gradient boosting)
- Initialized final project report
- Contributed to project overview and progress report
- Preprocessed data, concatenation