How to Handle Class Imbalance in Medical Datasets: A Step-by-Step Guide Using Logistic Regression, Oversampling, and Cross-Validation

CMPT 310 - D200: Introduction to Artificial Intelligence and Machine Learning *(Summer 2025)*

Manjari Prasad

Wan Yu Wendy Wong

Beyzanur Kuyuk

Introduction

In real-world datasets, especially in healthcare, class imbalance is a common challenge. In our project on early Alzheimer's detection using the OASIS dataset, we encountered this firsthand: the number of non-dementia cases vastly outnumbered the dementia cases. While this reflects real-world distributions, it poses problems for machine learning models, which may favour the majority class and fail to identify minority (positive) cases.

This guide walks you through our process of building and improving a binary classifier under class imbalance. Using logistic regression as a baseline, we show how to improve recall and F1 score using class weighting and oversampling. We also compare this with a random forest model. This guide is meant to help CMPT 310 students who are working with imbalanced classification problems and want practical, reproducible solutions.

Dataset Setup & Preprocessing

We used the OASIS Cross-Sectional dataset, available on Kaggle, which includes features such as age, gender, SES, MMSE, eTIV, nWBV, and CDR.

Key preprocessing steps:

1. Removed unnecessary columns (ID, Hand, Delay)
2. Dropped rows where CDR (target) was missing
3. Created a binary label: Dementia = 1 if CDR > 0, else 0
4. Imputed missing values for MMSE, SES, and Educ using median
5. Encoded M/F as binary (M = 1, F = 0)
6. Scaled continuous variables using StandardScaler

Sample code (from preprocessing.py)**:**

```
df = pd.read_csv("oasis_cross-sectional.csv")
df = df.drop(columns=["ID", "Hand", "Delay"])
df = df[df["CDR"].notna()]
df["Dementia"] = df["CDR"].apply(lambda x: 1 if x > 0 else 0)

for col in ["MMSE", "SES", "Educ"]:
    df[col] = df[col].fillna(df[col].median())
df["M/F"] = df["M/F"].map({"M": 1, "F": 0})

features_to_scale = ["Age", "Educ", "SES", "MMSE", "eTIV", "nWBV", "ASF"]
scaler = StandardScaler()
df[features_to_scale] = scaler.fit_transform(df[features_to_scale])
```

Baseline Logistic Regression

We first trained a standard LogisticRegression model using an 80/20 train-test split.

Result: Accuracy ≈ 81%, but recall for dementia cases was poor (0.65).

Although the model reached ~81% accuracy, the recall for dementia cases was only 0.65, meaning it missed 35% of actual dementia cases. This is due to the class imbalance in the dataset, where non-dementia cases outnumber dementia ones. In medical applications, such low recall is problematic, so we needed better balancing techniques to improve detection.

Sample code (from logistic_model.py)**:**

```
X = df[["Age", "Educ", "SES", "MMSE", "eTIV", "nWBV", "ASF", "M/F"]]
y = df["Dementia"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = LogisticRegression()
```

```
model.fit(X_train, y_train)
print(classification_report(y_test, model.predict(X_test)))
```

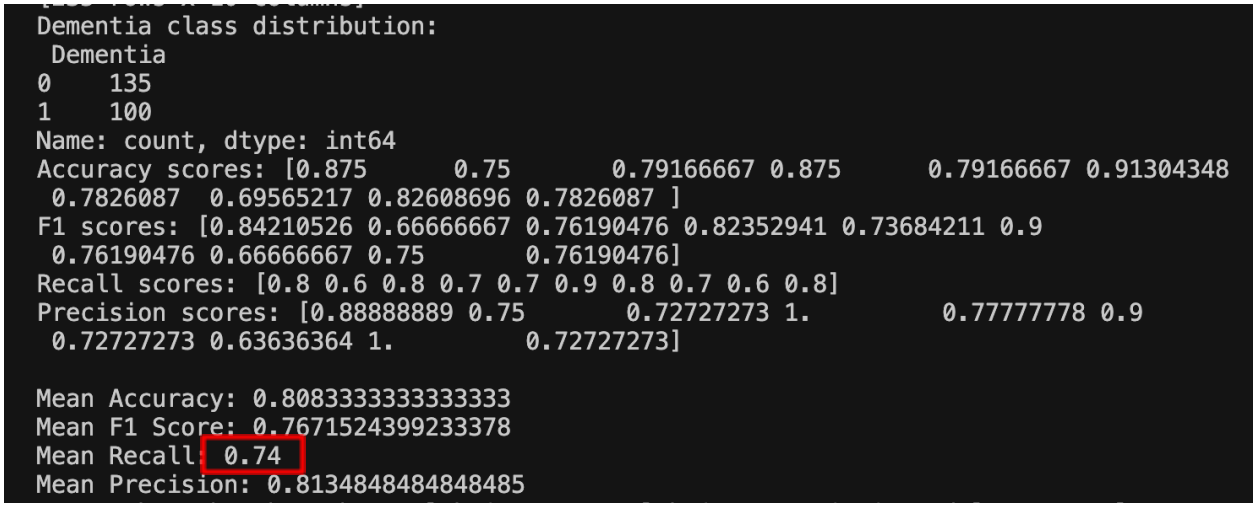
Terminal Output (Without Balancing - Using Train Split)**:**

```
Dementia class distribution:
 Dementia
0    135
1    100
Name: count, dtype: int64
Accuracy:  0.8085106382978723
Confusion Matrix:
 [[27  3]
 [ 6 11]]

Classification Report:              precision    recall  f1-score   support

           0       0.82      0.90      0.86        30
           1       0.79      0.65      0.71        17

    accuracy                           0.81        47
   macro avg       0.80      0.77      0.78        47
weighted avg       0.81      0.81      0.80        47
```

We also trained our LogisticRegression model using 10-fold Cross-Validation. Using 10-fold cross-validation with no balancing, we observed a mean F1 score of 0.76 and a mean recall of only 0.74 for the dementia class. While this is slightly better than the original train-test split (recall 0.65), the model still struggles to consistently identify dementia cases.

The variability across folds (especially in recall scores ranging from 0.6 to 0.9), shows that the model's performance is sensitive to data splits due to the class imbalance. This reinforces the need for balancing techniques to achieve more stable and fair predictions.

Terminal Output (Without Balancing - Using Cross Validation)**:**



Class Weighting with Cross-Validation

To correct the imbalance, we used class_weight={0:1, 1:3} with 10-fold cross-validation.

This tells the logistic regression model to give 3 times more importance to class 1 (dementia) than to class 0 (non-dementia) during training.

Since dementia cases are less frequent in the dataset, this weighting penalizes the model more when it misclassifies a dementia case, encouraging it to focus more on correctly predicting the minority class.

It's a way to manually handle class imbalance and improve recall for underrepresented classes.

Manual class weighting {0:1, 1:3} significantly improved the model's ability to detect dementia cases:

1. Recall increased to 0.90, the highest among all techniques
2. F1 Score rose to 0.80, showing a strong balance between precision and recall

3. Precision remained stable at ~0.73, meaning false positives were still controlled
4. Accuracy stayed consistent at ~80%

Compared to the unbalanced model (recall 0.65–0.74), class weighting made the model much more sensitive to underrepresented dementia cases, which is an essential improvement for early medical detection.

Sample code (from Logistic_model_cross_val.py):

```
balanced_model = LogisticRegression(class_weight={0: 1, 1: 3},
 max_iter=1000)
f1_scores = cross_val_score(balanced_model, X, y, cv=10, scoring='f1')
print("Mean F1 Score (Balanced):", f1_scores.mean())
```

Terminal Output (After Balancing):

```
Dementia class distribution:
 Dementia
0    135
1    100
Name: count, dtype: int64
Accuracy scores: [0.875      0.91666667 0.79166667 0.79166667 0.75       0.86956522
 0.69565217 0.60869565 0.91304348 0.82608696]
F1 scores: [0.85714286 0.90909091 0.76190476 0.7826087  0.72727273 0.86956522
 0.74074074 0.64       0.88888889 0.83333333]
Recall scores: [0.9 1.  0.8 0.9 0.8 1.  1.  0.8 0.8 1. ]
Precision scores: [0.81818182 0.83333333 0.72727273 0.69230769 0.66666667 0.76923077
 0.58823529 0.53333333 1.         0.71428571]

Mean Accuracy: 0.8038043478260869
Mean F1 Score: 0.8010548131417696
Mean Recall: 0.9
Mean Precision: 0.7342847348729702
```

Random Oversampling with imbalanced-learn

By using RandomOverSampler, we balanced the dataset to have equal class counts (135 vs 135). This improved the model's exposure to minority class examples and helped increase performance metrics across the board:

- Mean Accuracy: ~81.4%
- Mean F1 Score: 0.81
- Mean Recall: ~0.79
- Mean Precision: ~0.84

These results show a solid improvement over the unbalanced baseline. However, compared to manual class weighting, which reached a higher recall (0.90) and similar F1 score, oversampling was slightly less effective at maximizing dementia detection.

As a result, we chose to prioritize manual weighting for our final model, since in medical applications, higher recall is critical to avoid missing true dementia cases.

Sample code (from logistic_model_oversampling.py)**:**

```
ros = RandomOverSampler()
X_resampled, y_resampled = ros.fit_resample(X, y)

ros_model = LogisticRegression(class_weight='balanced', max_iter=1000)
f1_scores_ros = cross_val_score(ros_model, X_resampled, y_resampled, cv=10,
scoring='f1')
print("Mean F1 Score (ROS):", f1_scores_ros.mean())
```

Terminal Output (Random Oversampling with Cross Validation)**:**

```
Dementia class distribution:
 Dementia
0    135
1    100
Name: count, dtype: int64
Accuracy scores: [0.88888889 0.77777778 0.81481481 0.85185185 0.88888889 0.85185185
 0.77777778 0.66666667 0.92592593 0.7037037 ]
F1 scores: [0.88       0.75       0.81481481 0.81818182 0.88888889 0.85714286
 0.78571429 0.66666667 0.92307692 0.71428571]
Recall scores: [0.84615385 0.69230769 0.84615385 0.69230769 0.92307692 0.85714286
 0.78571429 0.64285714 0.85714286 0.71428571]
Precision scores: [0.91666667 0.81818182 0.78571429 1.         0.85714286 0.85714286
 0.78571429 0.69230769 1.         0.71428571]

Mean Accuracy: 0.8148148148148147
Mean F1 Score: 0.8098771968771968
Mean Recall: 0.7857142857142857
Mean Precision: 0.8427156177156176

Original class distribution:
 Dementia
0    135
1    100
Name: count, dtype: int64
Resampled class distribution:
 [135 135]
```

Random Forest Comparison

To test a non-linear model, we used RandomForestClassifier with 10-fold CV.Random Forest achieved the highest overall F1 score (≈ 0.805) and maintained a strong accuracy of ~83.4% without requiring any explicit class balancing. Its performance was stable across folds and showed that ensemble methods can naturally mitigate class imbalance due to the diversity of decision trees.

Compared to our best logistic regression model (with manual weighting), Random Forest offered:

- Higher precision (~0.82) → Fewer false positives
- Slightly lower recall → Missed more dementia cases than weighted logistic regression
- Similar AUC (~0.90)

This makes Random Forest a good alternative when precision is prioritized, such as when we want to avoid mislabeling non-dementia cases as dementia (e.g., in clinical decision-support settings).

However, since recall is more important in early screening scenarios (catching as many true cases as possible), we still chose weighted logistic regression as our final model.

Sample code (from [random_forest_crossval.py]):

```
rf = RandomForestClassifier(random_state=42)
rf_scores = cross_val_score(rf, X, y, cv=10, scoring='f1')
print("Random Forest F1 Score:", rf_scores.mean())
```

Terminal Output (Random Forest (Unbalanced) with Cross Validation):

```
Dementia class distribution:
 Dementia
0    135
1    100
Name: count, dtype: int64
Accuracy scores: [0.75       0.79166667 0.79166667 0.875      0.875      0.86956522
 0.86956522 0.73913043 0.86956522 0.91304348]
F1 scores: [0.7        0.73684211 0.76190476 0.85714286 0.85714286 0.85714286
 0.84210526 0.72727273 0.82352941 0.88888889]
Recall scores: [0.7 0.7 0.8 0.9 0.9 0.9 0.8 0.8 0.7 0.8]
Precision scores: [0.7        0.77777778 0.72727273 0.81818182 0.81818182 0.81818182
 0.88888889 0.66666667 1.         1.        ]

Mean Accuracy: 0.8344202898550724
Mean F1 Score: 0.8051971729680707
Mean Recall: 0.8
Mean Precision: 0.8215151515151515
```

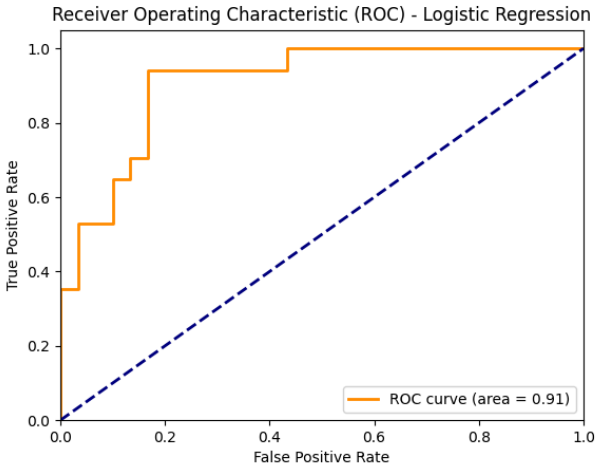
Troubleshooting Tips

Issue	Fix	Why it Happens / How it Helps
Logistic Regression not converging	Increase max_iter=1000	Default iterations may not be enough for convergence, especially with scaled or imbalanced data.
Low recall despite high accuracy	Use class_weight='balanced' or class_weight={0:1, 1:3}	Accuracy alone can be misleading on imbalanced data — these weights make the model focus more on the minority class.
Model overfits after oversampling	Use 10-fold cross-validation instead of a single train-test split	Random oversampling can duplicate data, which risks overfitting. Cross-validation helps ensure generalization.
AUC is high but precision is low	Adjust decision threshold or try a model like Random Forest	A high AUC means good ranking but not necessarily good classification. Adjusting the threshold can better balance precision and recall.
Model performs inconsistently across folds	Apply balancing techniques and ensure standardized features	Inconsistent performance usually means the model is too sensitive to split variance, a sign of imbalance or unscaled features.
Manual class weighting doesn't help	Experiment with different ratios, e.g. {0:1, 1:2} or {0:1, 1:5}	The optimal weight ratio depends on how imbalanced your data is and how aggressive the correction needs to be.
Recall is improving but precision drops	Consider trade-offs based on your goal (e.g., prioritize recall in screening)	In healthcare, higher recall is often worth slightly lower precision — but it should be an informed trade-off.

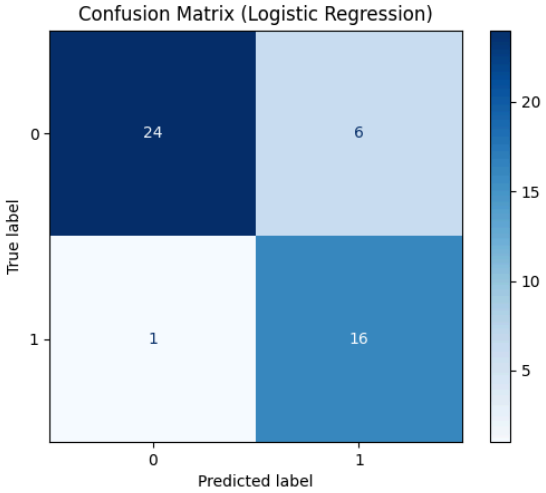
Takeaways

1. **Class imbalance** had a significant impact on our model's ability to detect dementia cases, especially when using standard logistic regression.
2. The **baseline model** achieved good accuracy (~81%) but had poor recall (0.65), missing many true dementia cases.
3. Applying **manual class weighting {0:1, 1:3}** resulted in the **best recall (0.90)** and a strong F1 score (0.80), making it our final choice.
4. **Random oversampling** also improved results, achieving F1 ≈ 0.81 and recall ≈ 0.79, but didn't outperform class weighting.
5. **Random Forest** performed the best overall in terms of F1 score (~0.805) and precision (~0.82) without requiring any balancing. However, its slightly lower recall made it less ideal for early screening.
6. Ultimately, we prioritized **recall** in our model selection, as missing dementia cases would be more harmful than predicting a few false positives.

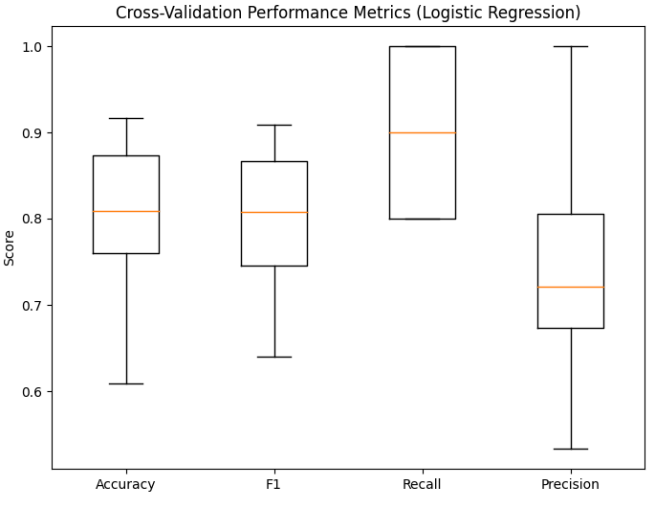
Plots and Visuals



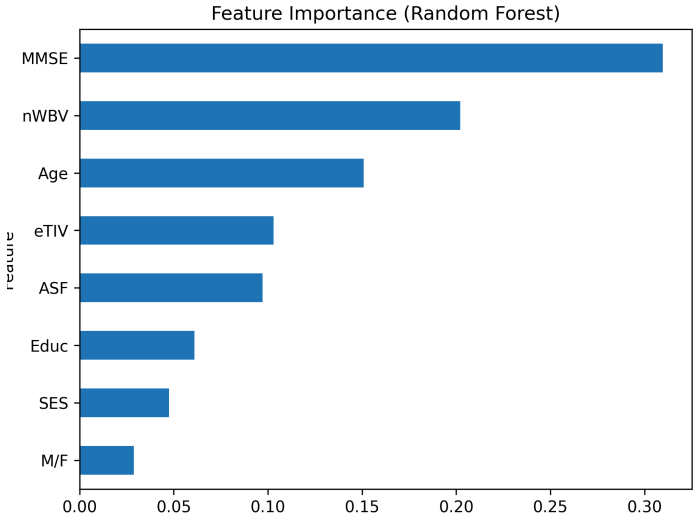
Receiver Operating Characteristic (ROC) curve for logistic regression, with AUC indicating the model's discrimination power between dementia and non-dementia classes.



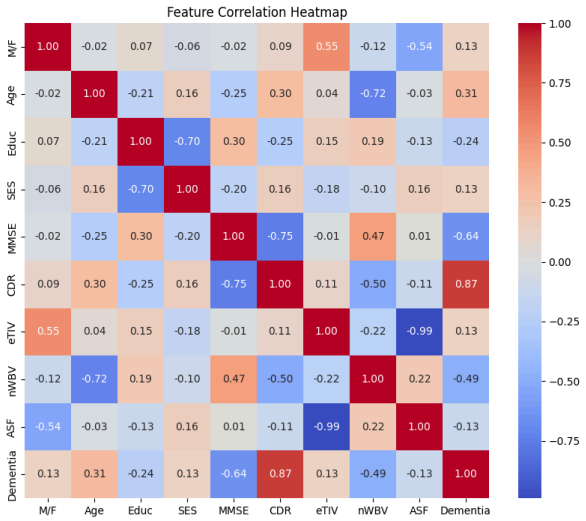
Confusion matrix for logistic regression model, showing true positives, false positives, true negatives, and false negatives for dementia classification.



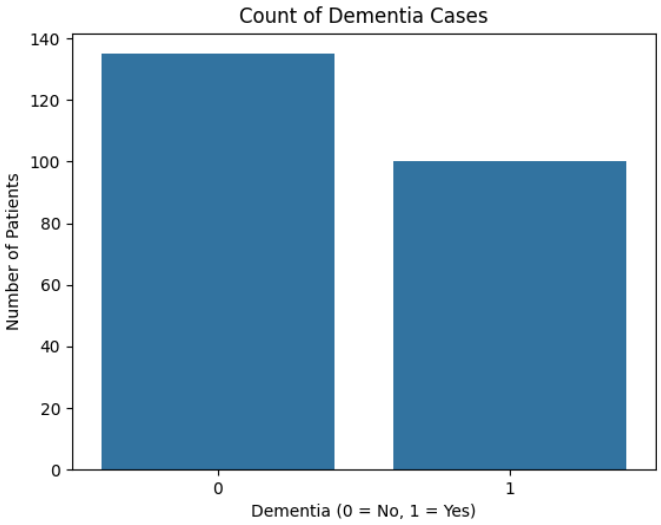
Boxplot showing how accuracy, F1 score, recall, and precision varied across the 10 folds of cross-validation for logistic regression



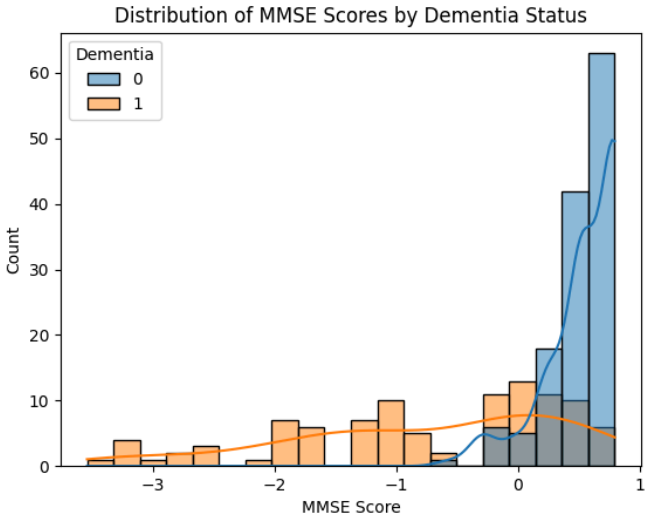
Random Forest feature importance showing MMSE, nWBV, and age as top dementia predictors.



This heatmap visualizes the pairwise correlations between numeric features, helping identify which variables are strongly related.



This plot shows the number of patients with and without dementia highlighting a class imbalance in the dataset.



This histogram compares the distribution of MMSE scores between dementia and non-dementia patients, revealing lower MMSE scores among those with dementia.