DL - LAB 9

Analysis of the obtained results

Task: *(Exploring various oversampling / undersampling techniques)*

Original class distribution - Class 0: 797, Class 1: 203, Ratio: 0.255

- Training baseline model (no handling of class imbalance)

Console Output:

Epoch: 1, Loss: 0.6443
Epoch: 2, Loss: 0.5836
Epoch: 3, Loss: 0.5381
Epoch: 4, Loss: 0.5110
Epoch: 5, Loss: 0.4834
Epoch: 6, Loss: 0.4601
Epoch: 7, Loss: 0.4445
Epoch: 8, Loss: 0.4209
Epoch: 9, Loss: 0.4066
Epoch: 10, Loss: 0.3933
Baseline model evaluation:
accuracy: 0.8250
precision: 0.7143
recall: 0.2439
f1: 0.3636
confusion_matrix: {'TP': 10, 'TN': 155, 'FP': 4, 'FN': 31}

- Training with class weighting

Console Output:

Positive class weight: 3.9261
Epoch: 1, Loss: 1.0924
Epoch: 2, Loss: 1.0563
Epoch: 3, Loss: 0.9924
Epoch: 4, Loss: 0.9424
Epoch: 5, Loss: 0.9040
Epoch: 6, Loss: 0.8686
Epoch: 7, Loss: 0.8374
Epoch: 8, Loss: 0.8026
Epoch: 9, Loss: 0.7703
Epoch: 10, Loss: 0.7543
Weighted model evaluation:
accuracy: 0.7700
precision: 0.4627
recall: 0.7561
f1: 0.5741
confusion_matrix: {'TP': 31, 'TN': 123, 'FP': 36, 'FN': 10}

- Random oversampling

Console Output:

After Random Oversampling - Class 0: 638, Class 1: 638
Epoch: 1, Loss: 0.6864
Epoch: 2, Loss: 0.6224
Epoch: 3, Loss: 0.5755
Epoch: 4, Loss: 0.5372
Epoch: 5, Loss: 0.5048
Epoch: 6, Loss: 0.4759
Epoch: 7, Loss: 0.4497
Epoch: 8, Loss: 0.4259
Epoch: 9, Loss: 0.4042
Epoch: 10, Loss: 0.3840
Random Oversampling model evaluation:
accuracy: 0.7750
precision: 0.4688
recall: 0.7317
f1: 0.5714
confusion_matrix: {'TP': 30, 'TN': 125, 'FP': 34, 'FN': 11}

- SMOTE

Console Output:

After SMOTE - Class 0: 638, Class 1: 638
Epoch: 1, Loss: 0.6710
Epoch: 2, Loss: 0.6213
Epoch: 3, Loss: 0.5775
Epoch: 4, Loss: 0.5376
Epoch: 5, Loss: 0.5016
Epoch: 6, Loss: 0.4686
Epoch: 7, Loss: 0.4394
Epoch: 8, Loss: 0.4142
Epoch: 9, Loss: 0.3921
Epoch: 10, Loss: 0.3725
SMOTE model evaluation:
accuracy: 0.7800
precision: 0.4795
recall: 0.8537
f1: 0.6140
confusion_matrix: {'TP': 35, 'TN': 121, 'FP': 38, 'FN': 6}

- ADASYN

Console Output:

After ADASYN - Class 0: 638, Class 1: 647
Epoch: 1, Loss: 0.6627
Epoch: 2, Loss: 0.6195
Epoch: 3, Loss: 0.5885
Epoch: 4, Loss: 0.5434
Epoch: 5, Loss: 0.5069
Epoch: 6, Loss: 0.4914
Epoch: 7, Loss: 0.4568
Epoch: 8, Loss: 0.4399
Epoch: 9, Loss: 0.4243
Epoch: 10, Loss: 0.3939
ADASYN model evaluation:
accuracy: 0.7750
precision: 0.4737
recall: 0.8780
f1: 0.6154
confusion_matrix: {'TP': 36, 'TN': 119, 'FP': 40, 'FN': 5}

- Tomek Links

Console Output:

After Tomek Links - Class 0: 629, Class 1: 162
Epoch: 1, Loss: 0.7340
Epoch: 2, Loss: 0.6555
Epoch: 3, Loss: 0.5895
Epoch: 4, Loss: 0.5457
Epoch: 5, Loss: 0.5117
Epoch: 6, Loss: 0.4828
Epoch: 7, Loss: 0.4545
Epoch: 8, Loss: 0.4379
Epoch: 9, Loss: 0.4217
Epoch: 10, Loss: 0.4053
Tomek Links model evaluation:
accuracy: 0.8050
precision: 0.5833
recall: 0.1707
f1: 0.2642
confusion_matrix: {'TP': 7, 'TN': 154, 'FP': 5, 'FN': 34}

- NearMiss

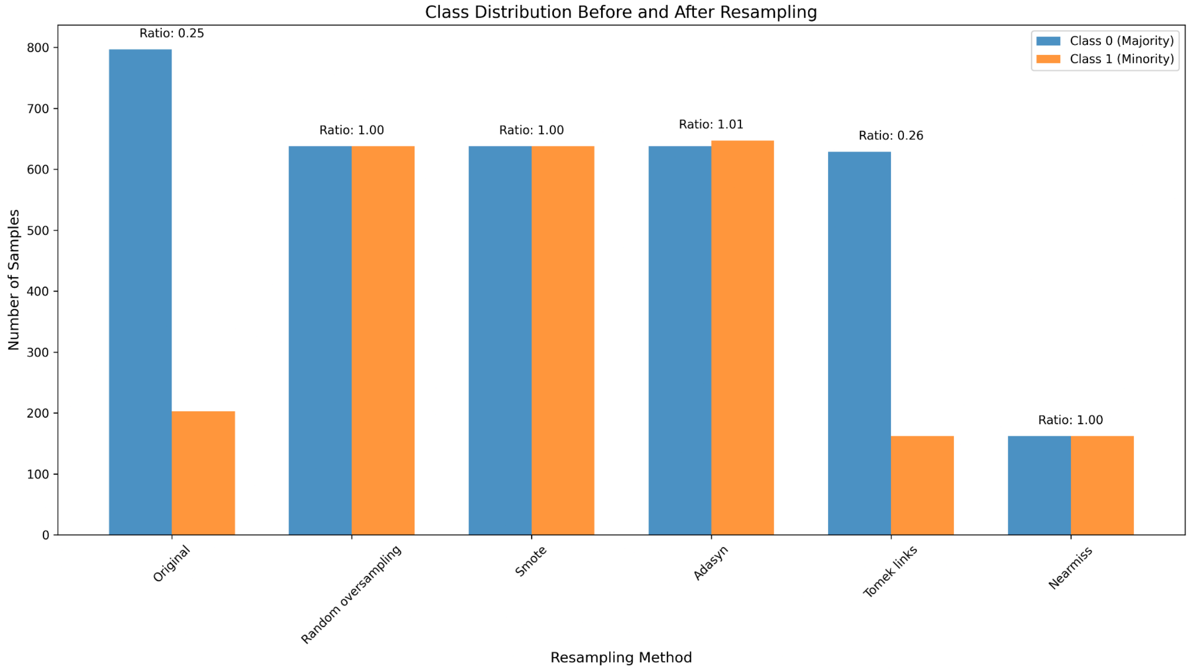
Console Output:

After NearMiss - Class 0: 162, Class 1: 162
Epoch: 1, Loss: 0.7623
Epoch: 2, Loss: 0.7073
Epoch: 3, Loss: 0.6967
Epoch: 4, Loss: 0.7000
Epoch: 5, Loss: 0.6411
Epoch: 6, Loss: 0.6505
Epoch: 7, Loss: 0.6376
Epoch: 8, Loss: 0.6198
Epoch: 9, Loss: 0.6244
Epoch: 10, Loss: 0.5882
NearMiss model evaluation:
accuracy: 0.5850
precision: 0.3056
recall: 0.8049
f1: 0.4430
confusion_matrix: {'TP': 33, 'TN': 84, 'FP': 75, 'FN': 8}

=== Summary of Results ===

Method	*Accuracy*	*Precision*	*Recall*	*F1*
baseline	*0.8250*	*0.7143*	*0.2439*	*0.3636*
weighted	*0.7700*	*0.4627*	*0.7561*	*0.5741*
random_oversampling	*0.7750*	*0.4688*	*0.7317*	*0.5714*
smote	*0.7800*	*0.4795*	*0.8537*	*0.6140*
adasyn	*0.7750*	*0.4737*	*0.8780*	*0.6154*
tomek_links	*0.8050*	*0.5833*	*0.1707*	*0.2642*
nearmiss	*0.5850*	*0.3056*	*0.8049*	*0.4430*

> *Class Distribution Comparison Plot*



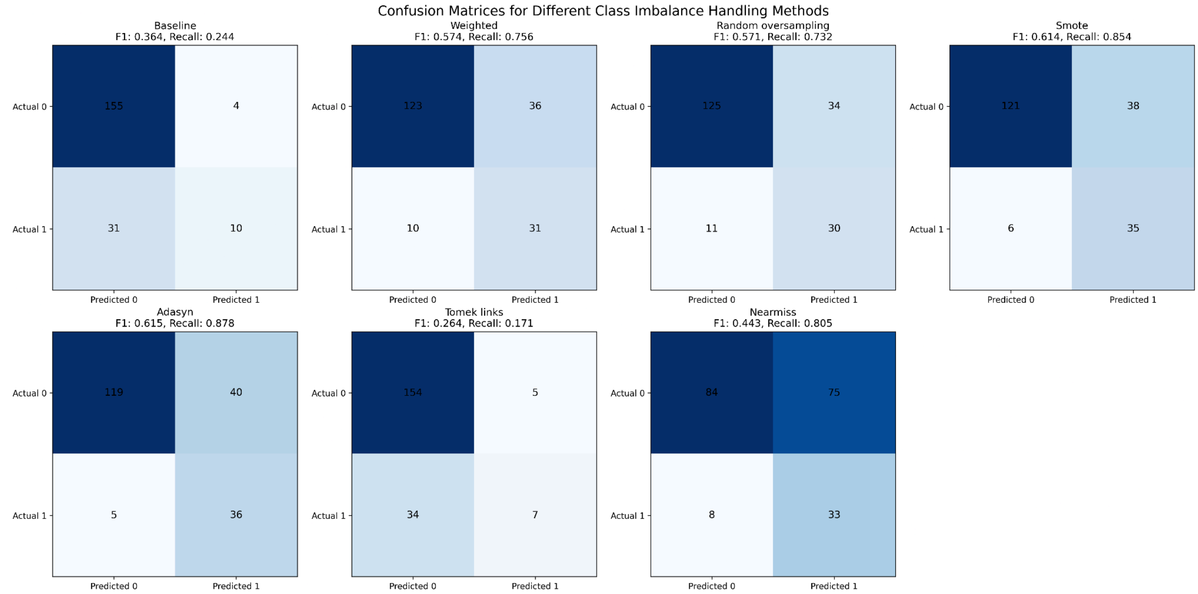
This plot is one's straightforward – it just shows us how many samples of Class 0 (majority) and Class 1 (minority) we fed into the model for training after our resampling tricks.

- Original: We'll see the big gap – lots of Class 0 (e.g., ~638), very few Class 1 (e.g., ~162). That's our starting problem.

- Oversampling (Random, SMOTE, ADASYN): Now Class 1 is beefed up to match Class 0 (both around ~638). The ratio annotation will be close to 1.0 or almost. Here we've balanced the scales by either duplicating or synthetically creating more minority samples. The model can't ignore them now.

- Undersampling (Tomek Links, NearMiss): For Tomek Links, we barely have a scratch on Class 0. It's still very imbalanced. While in NearMiss, Class 0 gets chopped down to match Class 1 (both around ~162). The ratio is 1.0, but the total dataset size is much smaller. Undersampling balances by subtraction. NearMiss is extreme and can lead to losing valuable info from the majority class, which can bite you, as seen in the confusion matrix.

> *Confusion Matrices Plot*



So, these grids are basically showing us where our model got things right and where it face-planted for each method. Remembering that, TP = True Positive (good!), TN = True Negative (good!), FP = False Positive (oops, called a 0 a 1), FN = False Negative (double oops, called a 1 a 0 – usually the one we care most about in imbalance), we could observe:

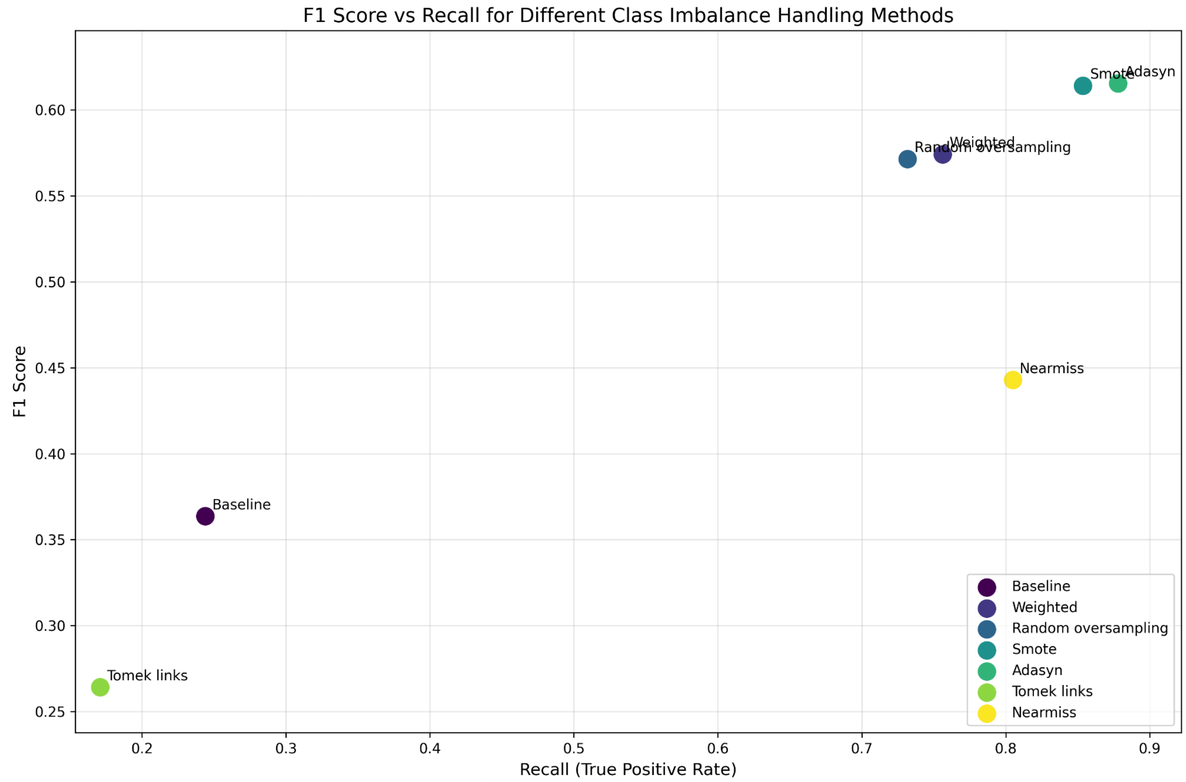
- Baseline (No Frills):
  This one's usually solid on the TNs (e.g., nails 155/160 of the majority class) but totally bombs the TPs for our minority class (like, 10/40 TPs, meaning 30 FNs). The F1 and Recall in the title will be pretty sad. So, we will have a classic case of the model just learning the majority class and pretty much ignoring the little guy. High accuracy, but it's a lie!

- Class Weighting (Giving the Minority Some Love):
  We see more TPs (maybe 28/40) and fewer FNs (down to 12). Sweet! But, the FPs often creep up (say, from 5 to 20). The F1/Recall in the title should look better than the baseline. We told the loss function to care more about the minority class, and it listened. It's trying harder, but sometimes overcorrects and gets a bit trigger-happy on the majority class.

- Oversampling (SMOTE & ADASYN - The Smart Data Generators):
  These are usually our heroes. TPs shoot up (like 33-35/40), FNs drop way down (5-7). FPs are generally managed well. The F1/Recall in their titles should be the highest. By creating smart synthetic samples for the minority class, we've given the model a much richer dataset to learn from. It's not just seeing the same few minority examples over and over.

- Undersampling (NearMiss - The Aggressive Pruner):
  High TPs (maybe 33/40), which looks good at first. But then you see the FPs – they're through the roof (could be 60+!). We threw out so much majority data that the model got hyper-focused on the minority. It's good at spotting them, but now it thinks almost everything might be the minority class. Precision tanks, and the F1 score suffers despite high recall.

> F1 Score vs Recall Plot



This plot is where we see who's really winning. We want to be in that top-right corner – high Recall (catching most of the minority class) and high F1 (good balance of precision and recall).

- Baseline -> Stuck in the mud, bottom-left (Recall ~0.24, F1 ~0.36).
- SMOTE & ADASYN: These are your champions, usually chilling in the top-right (e.g., ADASYN with Recall ~0.88, F1 ~0.62). They're finding the minority class and not making a mess of false positives.
- Class Weighting / Random Oversampling: Decent contenders, somewhere in the middle. They've made progress from the baseline.
- NearMiss: It'll have high Recall, pushing it to the right, but its F1 score will drag it down because of all those FPs we saw earlier.

This plot quickly tells you which methods are actually useful. If it's not heading towards that top-right, it's probably not the best pick for this problem.