CSI 5155 Project Report

Name: Alim Manjiyani

Student Number: 300229095

11th December 2021

Contents

1 Introduction

This project deals with the nuances of semi-supervised learning. For this project, I have implemented three different inductive semi-supervised approaches on 3 datasets (Online Shoppers Intention, Marketing Campaign, Heart Disease), the ratio of unlabelled to labelled data varies from 10% to 95% (10%, 20%, 50%, 90%, 95%). The aim is to learn and understand the interplay between class imbalance and semi-supervised learning, the relationship between the runtime and the size of unlabelled data, and to provide an accurate evaluation metric so that these algorithms can be compared for any statistical difference

2 Dataset Generation

One of the challenges faced during generating unlabelled dataset is that one of the datasets (Online Shoppers Intention) is highly imbalanced and a naïve approach for removing the class labels may lead to only 1 type of class for the learner resulting in completely neglecting the presence of a second-class label. To tackle this challenge, I reserve some part of the two class labels for training and testing. For example, If there is a need to create 90% unlabelled data points, then from the remaining 10% datapoints I reserve 5% of the class 1 label and 5% of the class 2 label, this way the learner will have both the class labels at the beginning of the semi-supervised training.

Moreover, I used a new strategy to handle class-imbalance as suggested by Dr. Colin Bellinger during his guest lecture, namely the ReMix (Resampling with data and label mixing). Not to be surprised I achieved very good results with the use of this strategy.

3 Selecting base learners for self-training and semi-supervised ensembles

To decide which algorithms to use as base estimators, I evaluated the scores of Random Forest, Decision Tree, K-NN, SVM, MLP, Gradient Boosting Algorithms with respect to all the three datasets as shown in Figure 1.

Algorithm	Shopper Dataset	Marketing dataset	Heart dataset
Random Forest	0.9051951375559821	0.8830357142857143	0.8533333333333334
Decision Tree	0.852758668305279	0.8553571428571428	0.8066666666666666
KNN	0.8508823651251743	0.5910714285714286	0.8433333333333334
SVM	0.8455610406108894	0.7825892857142858	0.8300000000000001
MLP	0.8801963112811094	0.5977678571428572	0.8566666666666667
Gradient Boosting	0.9543242104972451	0.9035714285714286	0.8066666666666666

Figure 1 Algorithm Scores with respect to datasets

To conclude, I decided to go with Gradient Boosting Algorithm for Shopper and Marketing Dataset and decided to go with Random Forest for the Heart Dataset because of its good performance and easy implementation.

4 Self-Training Algorithm

For implementation of the self-training algorithm, I used SelfTrainingClassifier library from the scikit learn package. This class allows a given supervised classifier to function as a semi-supervised classifier, allowing it to learn from unlabelled data. It does this by iteratively predicting pseudo-labels for the unlabelled data and adding them to the training set.

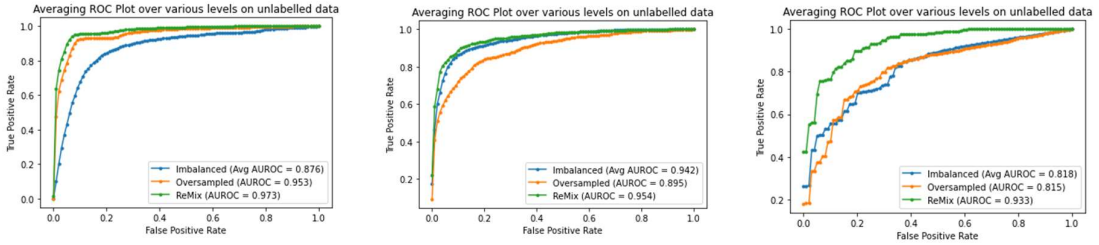
4.1 Results

Table 1 Self-Training Results

Class-Label Distribution	Percent of Unlabelled datapoints	Accuracy	F1-Score	AUROC	Runtime
Dataset: Online Shoppers Intention					
Imbalanced	0%	0.84793	0.59671	0.886	8.485
	10%	0.88104	0.59558	0.878	35.639
	20%	0.86915	0.57394	0.833	64.449
	50%	0.85536	0.63531	0.897	37.096
	90%	0.85158	0.62980	0.895	27.548
	95%	0.86482	0.51361	0.875	28.255
Oversampled	0%	0.95518	0.94738	0.986	13.226
	10%	0.95075	0.95252	0.991	74.015
	20%	0.94403	0.94635	0.989	94.864
	50%	0.92244	0.92521	0.968	88.130
	90%	0.85865	0.86047	0.920	51.570
	95%	0.83786	0.83687	0.897	42.546
ReMix	0%	0.98030	0.97615	0.997	22.557
	10%	0.97412	0.97445	0.996	158.051
	20%	0.97159	0.97196	0.995	167.069
	50%	0.95689	0.95759	0.988	112.831
	90%	0.89990	0.90121	0.948	118.149
	95%	0.87944	0.88129	0.938	95.944
Dataset: Marketing Campaign					
Imbalanced	0%	0.90223	0.88335	0.956	0.709
	10%	0.90029	0.89644	0.954	3.475
	20%	0.89732	0.89269	0.953	5.716
	50%	0.90625	0.90410	0.955	5.468
	90%	0.83630	0.84057	0.927	5.139
	95%	0.83482	0.83936	0.922	5.931
Oversampled	0%	0.90717	0.90014	0.963	2.672
	10%	0.88489	0.88235	0.955	7.805
	20%	0.90503	0.90149	0.962	8.153
	50%	0.87338	0.86249	0.946	7.085
	90%	0.77122	0.75117	0.863	5.346
	95%	0.69352	0.67776	0.750	4.728
ReMix	0%	0.96677	0.95577	0.988	2.374

	10%	0.94732	0.94720	0.983	12.437
	20%	0.94892	0.94880	0.982	17.563
	50%	0.93774	0.93769	0.976	16.036
	90%	0.87869	0.87955	0.943	25.089
	95%	0.80766	0.81673	0.884	24.877
Dataset: Heart Disease					
Imbalanced	0%	0.84666	0.80412	0.909	0.109
	10%	0.81111	0.84112	0.879	0.277
	20%	0.83333	0.84848	0.916	0.279
	50%	0.78888	0.82242	0.902	0.829
	90%	0.81111	0.83495	0.894	0.182
	95%	0.54444	0.70503	0.500	0.187
Oversampled	0%	0.85558	0.87999	0.967	0.203
	10%	0.77551	0.78431	0.891	0.750
	20%	0.85714	0.86274	0.923	1.098
	50%	0.81632	0.833333	0.891	1.474
	90%	0.79591	0.81481	0.871	0.373
	95%	0.5000	0.66666	0.500	0.381
ReMix	0%	0.90178	0.90000	0.962	0.124
	10%	0.87050	0.86153	0.957	0.592
	20%	0.89928	0.89705	0.962	0.400
	50%	0.89208	0.89208	0.967	0.574
	90%	0.82733	0.85185	0.902	1.038
	95%	0.76978	0.80487	0.879	1.011

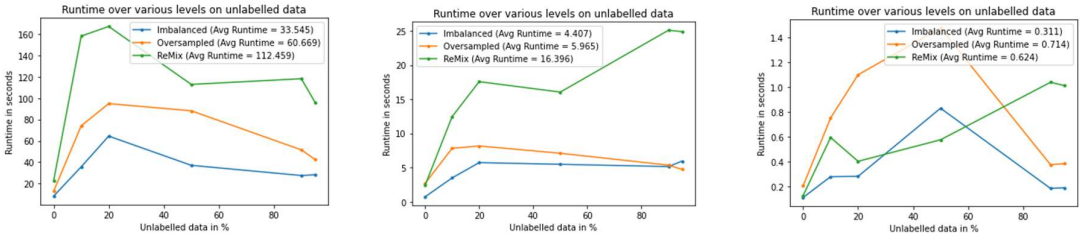
4.1.1 Average ROC Graph



(a) Shopper Dataset (b) Market Dataset (c) Heart Dataset

Figure 2: Average ROC Graphs for Self-Training Algorithm

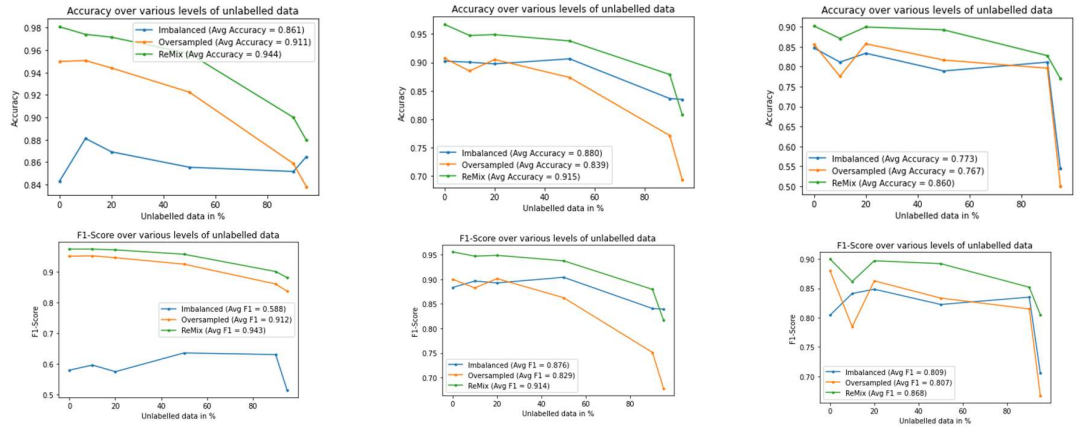
4.1.2 Runtime Graph



(a) Shopper Dataset (b) Market Dataset (c) Heart Dataset

Figure 3: Runtime over unlabelled percent for Self-Training Algorithm

4.1.3 Accuracy and F1-Score Graph



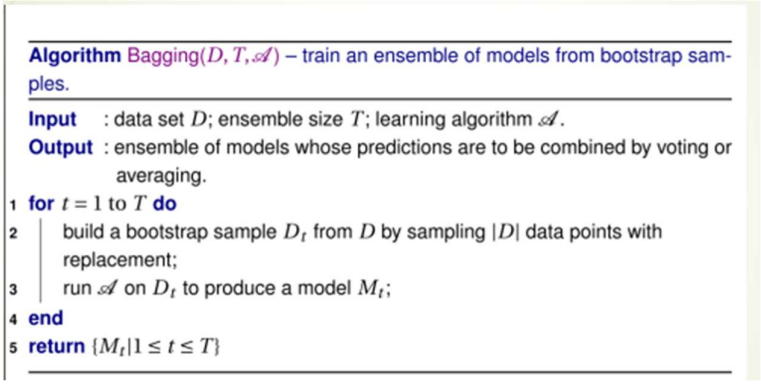
(a) Shopper Dataset (b) Market Dataset (c) Heart Dataset

Figure 4: Accuracy and F1-Score over unlabelled percent for Self-Training Algorithm

5 Semi-Supervised Ensembles

I used Bagging algorithm (Figure 2) to create an ensemble with 3 self-training semi-supervised algorithms. For the predictions, I used max-voting criterion to decide the final prediction.

Since I hand-coded the algorithm, one of the challenges was to create the roc-curve. All the estimators had different roc curves with uneven datapoints, so I had to come up with a strategy to get the mean of the 3 curves with uneven datapoints. The solution was to set a specific linspace and interpolate the graphs to find their midpoints.



Algorithm Bagging($D, T, \mathscr{A}$) – train an ensemble of models from bootstrap samples.

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.

Output : ensemble of models whose predictions are to be combined by voting or averaging.

1 **for** $t = 1$ to T **do**
2 build a bootstrap sample D_t from D by sampling $|D|$ data points with replacement;
3 run $\mathscr{A}$ on D_t to produce a model M_t;
4 **end**
5 **return** $\{M_t | 1 \leq t \leq T\}$

Figure 5 Bagging Algorithm

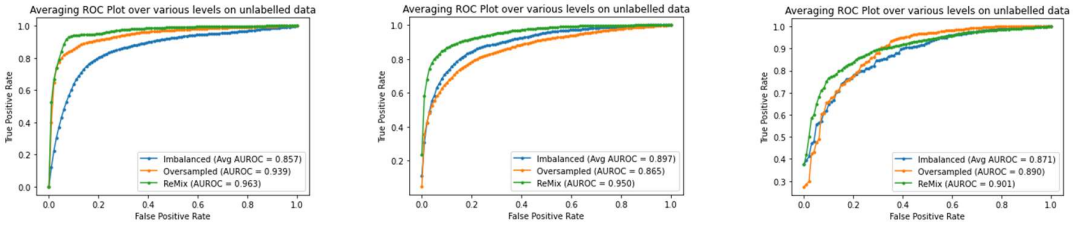
5.1 Results

Table 2 Semi-Supervised Ensemble Results

Class-Label Distribution	Percent of Unlabelled datapoints	Accuracy	F1-Score	AUROC	Runtime
Dataset: Online Shoppers Intention					
Imbalanced	0%	84.793	0.59671	0.886	8.485
	10%	87.861	0.58617	0.853	97.592
	20%	88.294	0.58325	0.842	104.512
	50%	87.023	0.64391	0.892	103.203
	90%	83.536	0.60065	0.870	65.612
	95%	85.915	0.53105	0.827	58.669
Oversampled	0%	95.518	0.94738	0.986	13.226
	10%	93.124	0.92352	0.980	181.793
	20%	92.069	0.94635	0.974	175.212
	50%	89.590	0.89864	0.953	215.110
	90%	84.330	0.83849	0.907	106.788
	95%	78.381	0.73562	0.883	109.652

ReMix	0%	98.030	0.97615	0.997	22.557
	10%	96.251	0.96156	0.993	301.272
	20%	96.080	0.97196	0.990	359.240
	50%	93.958	0.94122	0.978	329.396
	90%	88.923	0.89160	0.936	198.599
	95%	86.820	0.86795	0.919	223.908
Dataset: Marketing Campaign					
Imbalanced	0%	90.223	0.88335	0.956	0.709
	10%	87.053	0.86212	0.942	8.266
	20%	86.309	0.85846	0.933	11.078
	50%	81.250	0.81524	0.904	13.659
	90%	82.440	0.82388	0.893	11.694
	95%	75.000	0.74233	0.814	7.984
Oversampled	0%	90.717	0.90014	0.963	2.672
	10%	89.496	0.89088	0.956	17.614
	20%	89.496	0.88888	0.957	19.130
	50%	82.733	0.80327	0.915	19.415
	90%	72.086	0.68910	0.779	11.537
	95%	71.942	0.71698	0.717	8.775
ReMix	0%	96.677	0.95577	0.988	2.374
	10%	93.948	0.94072	0.987	35.072
	20%	93.328	0.93281	0.979	38.820
	50%	92.397	0.92403	0.977	47.857
	90%	86.268	0.86310	0.935	36.213
	95%	76.183	0.76830	0.871	29.480
Dataset: Heart Disease					
Imbalanced	0%	84.666	0.80412	0.909	0.109
	10%	82.222	0.83673	0.913	1.026
	20%	83.333	0.84848	0.926	1.281
	50%	82.222	0.84000	0.882	1.880
	90%	77.777	0.76190	0.836	2.437
	95%	72.222	0.75247	0.799	2.180
Oversampled	0%	85.558	0.87999	0.967	0.203
	10%	78.571	0.80373	0.894	1.203
	20%	84.693	0.84848	0.943	1.206
	50%	86.734	0.85714	0.935	1.977
	90%	72.448	0.64935	0.874	2.614
	95%	72.448	0.76106	0.802	2.387
ReMix	0%	90.178	0.90000	0.962	0.124
	10%	93.006	0.92753	0.971	1.258
	20%	88.811	0.88732	0.959	1.046
	50%	87.412	0.87142	0.953	1.758
	90%	80.419	0.78124	0.884	2.777
	95%	72.027	0.76190	0.739	2.352

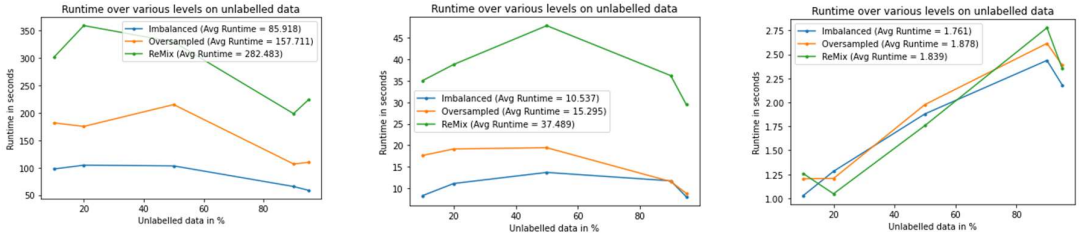
5.1.1 Average ROC Graph



(a) Shopper Dataset (b) Market Dataset (c) Heart Dataset

Figure 6: Average ROC Curve for Semi-Supervised Ensemble

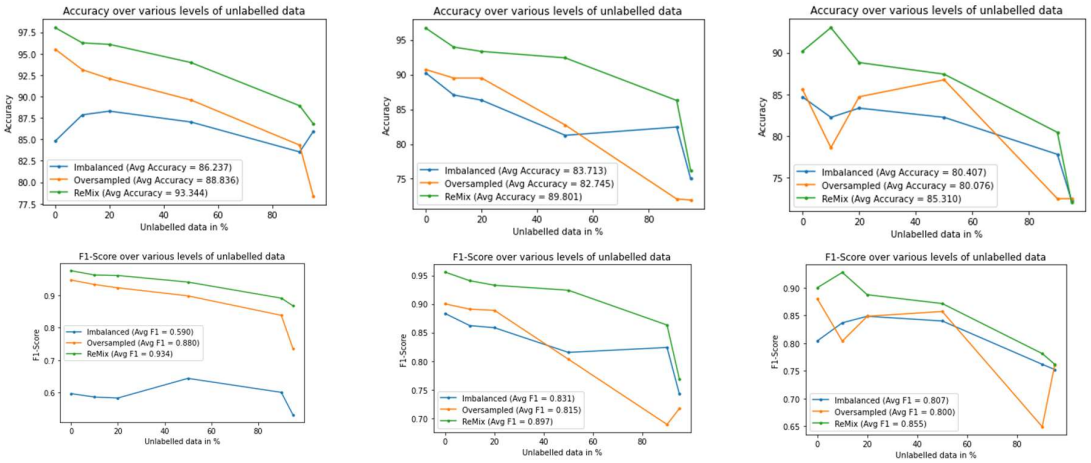
5.1.2 Runtime Graphs



(a) Shopper Dataset (b) Market Dataset (c) Heart Dataset

Figure 7: Runtime over unlabelled percent for Semi-Supervised Ensemble

5.1.3 Accuracy and F1-Score Graphs



(a) Shopper Dataset (b) Market Dataset (c) Heart Dataset

Figure 8: Accuracy and F1-Score over unlabelled percent for Semi-Supervised Ensemble

6 Intrinsic Semi-Supervised Method (S3VM)

S3VM stands for Semi-Supervised Support Vector Machines, objective of which is to maximize the distance between the dense regions where the samples must be. It is also known as a maximum-margin algorithm. I implemented this algorithm using semisupervised library available at pypi.

In lay-man terms, this algorithm follows the self-training approach but instead of choosing k-best labels or setting a threshold for confidence level, it tries to maximize the geometric margins of the SVM.

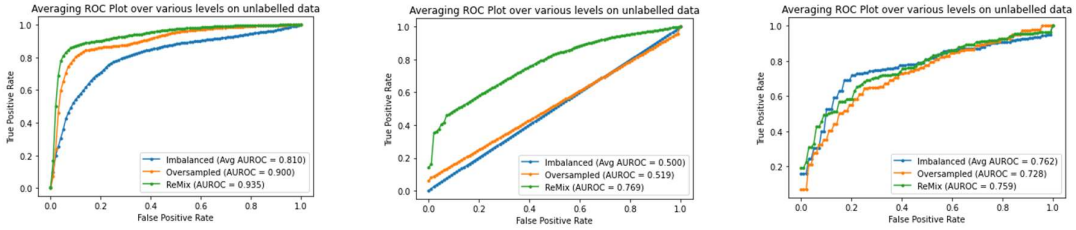
6.1 Results

Table 3 S3VM Results

Class-Label Distribution	Percent of Unlabelled datapoints	Accuracy	F1-Score	AUROC	Runtime
Dataset: Online Shoppers Intention					
Imbalanced	0%	0.84793	0.59671	0.886	8.485
	10%	0.87050	0.43447	0.866	298.250
	20%	0.87807	0.44932	0.862	291.140
	50%	0.86726	0.40772	0.835	289.275
	90%	0.84130	0.22045	0.742	280.181
	95%	0.83563	0.18059	0.748	253.154
Oversampled	0%	0.95518	0.94738	0.986	13.226
	10%	0.86040	0.86365	0.931	520.414
	20%	0.85305	0.85811	0.931	589.522
	50%	0.84873	0.85378	0.921	587.356
	90%	0.78046	0.79325	0.884	751.246
	95%	0.72833	0.75856	0.835	674.959
ReMix	0%	0.98030	0.97615	0.997	22.557
	10%	0.91125	0.91022	0.954	2137.230
	20%	0.90488	0.90624	0.954	2184.052
	50%	0.90784	0.90691	0.950	2141.056
	90%	0.84637	0.85021	0.920	2353.277
	95%	0.79307	0.81033	0.895	2381.770
Dataset: Marketing Campaign					
Imbalanced	0%	0.90223	0.88335	0.956	0.709
	10%	0.51636	0.0	0.505	0.897
	20%	0.51636	0.0	0.512	0.936
	50%	0.51636	0.0	0.498	1.140
	90%	0.51636	0.0	0.489	1.406
	95%	0.51636	0.0	0.497	1.257
Oversampled	0%	0.90717	0.90014	0.963	2.672
	10%	0.54244	0.15425	0.538	1.180

	20%	0.53093	0.11891	0.524	1.390
	50%	0.52517	0.09836	0.519	1.496
	90%	0.50647	0.02279	0.502	1.302
	95%	0.50359	0.01709	0.509	1.411
ReMix	0%	0.96677	0.95577	0.988	2.374
	10%	0.78169	0.82080	0.918	4.787
	20%	0.75665	0.80403	0.896	5.142
	50%	0.72769	0.63445	0.833	5.444
	90%	0.57355	0.30749	0.628	4.958
	95%	0.52503	0.65609	0.569	4.865
Dataset: Heart Disease					
Imbalanced	0%	84.666	0.80412	0.909	0.109
	10%	0.54444	0.70503	0.789	0.099
	20%	0.57777	0.72058	0.828	0.078
	50%	0.58888	0.72180	0.773	0.121
	90%	0.55555	0.70588	0.433	0.115
	95%	0.45555	0.0	0.829	0.107
Oversampled	0%	85.558	0.87999	0.967	0.203
	10%	0.60204	0.71532	0.847	0.093
	20%	0.58163	0.70503	0.766	0.136
	50%	0.62244	0.49315	0.788	0.109
	90%	0.51020	0.66666	0.738	0.136
	95%	0.61224	0.38709	0.763	0.107
ReMix	0%	90.178	0.90000	0.962	0.124
	10%	0.77777	0.71428	0.955	0.160
	20%	0.78472	0.82285	0.919	0.137
	50%	0.71527	0.77837	0.889	0.229
	90%	0.60416	0.70466	0.645	0.156
	95%	0.55555	0.67999	0.591	0.169

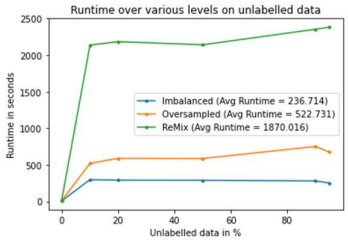
6.1.1 Average ROC Graph



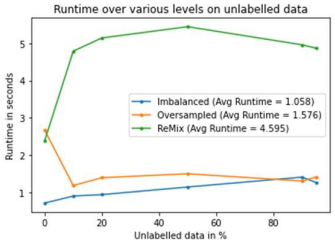
(a) Shopper Dataset (b) Market Dataset (c) Heart Dataset

Figure 9: Average ROC Curve for S3VM

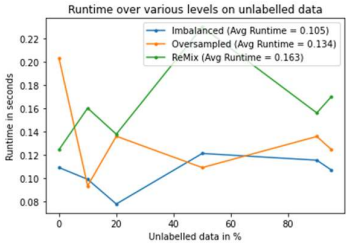
6.1.2 Runtime Graphs



(a) Shopper Dataset



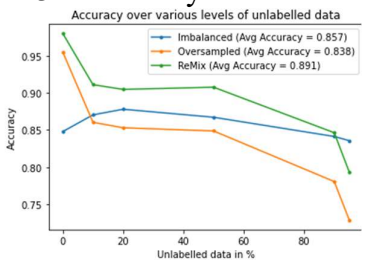
(b) Market Dataset

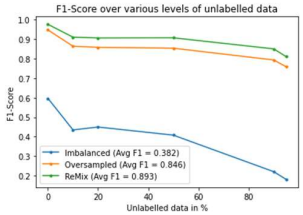


(c) Heart Dataset

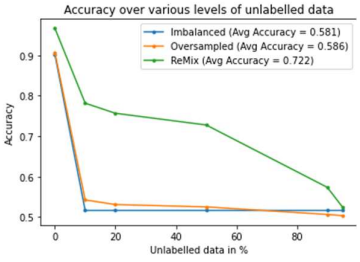
Figure 10: Runtime over unlabelled percent for S3VM

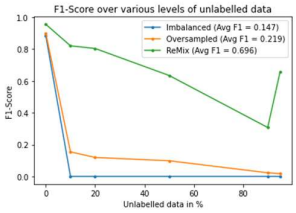
6.1.3 Accuracy and F1-Score Graphs



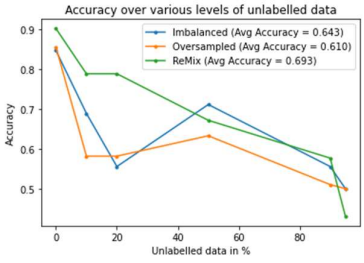


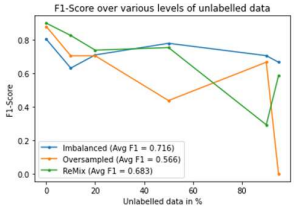
(a) Shopper Dataset





(b) Market Dataset





(c) Heart Dataset

Figure 11: Accuracy and F1-Score over unlabelled percent for S3VM

7 Statistically Significant Test

7.1 Nemenyi Test and Friedman's Test

7.1.1 Shopper Imbalanced

Dataset Name	Unlabelled Percent	Self-Training	Semi-Supervised Ensembles	Semi-Supervised Support Vector Machine (S3VM)
Online Shopper Intention (Imbalanced)	10	0.88104 (1)	0.87861(2)	0.87050 (3)
	20	0.86915 (3)	0.88294 (1)	0.87807 (2)
	50	0.85536 (3)	0.87023 (1)	0.86726 (2)
	90	0.85158 (1)	0.83536 (3)	0.84130 (2)
	95	0.86482 (1)	0.85915 (2)	0.83563 (3)
Average Rank		1.8	1.8	2.4

```
self_train = [0.88104, 0.86915, 0.85536, 0.85158, 0.86482]
ss_ensemble = [0.87861, 0.88294,0.87023, 0.83536, 0.85915]
s3vm = [0.87050,0.87807,0.86726,0.84130,0.83563]

stats.friedmanchisquare(self_train,ss_ensemble,s3vm)
```

```
FriedmanchisquareResult(statistic=1.2000000000000028, pvalue=0.5488116360940257)
```

```
print("alpha:0.05 has critical value: 6.4")
print("since statistic < critical value, we say All algorithms perform equally")
```

```
alpha:0.05 has critical value: 6.4
since statistic < critical value, we say All algorithms perform equally
```

```
cd= 1.4822866628828582
```

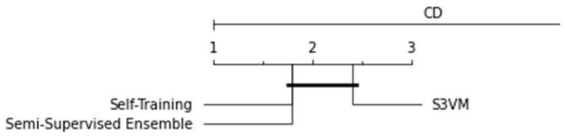


Figure 12 Nemenyi and Friedman Graph for Shopper Imbalanced Dataset

Result: No Significant Difference

7.1.2 Oversampled Shopper

Dataset Name	Unlabelled Percent	Self-Training	Semi-Supervised Ensembles	Semi-Supervised Support Vector Machine (S3VM)
Online Shopper Intention (Oversampled)	10	0.95075 (1)	93.124 (2)	0.86040 (3)
	20	0.94403 (1)	92.069 (2)	0.85305 (3)
	50	0.92244 (1)	89.590 (2)	0.84873 (3)
	90	0.85865 (1)	84.330 (2)	0.78046 (3)
	95	0.83786 (1)	78.381 (2)	0.72833 (3)
Average Rank		1	2	3

```
FriedmanchisquareResult(statistic=10.0, pvalue=0.006737946999085468)
```

```
print("alpha:0.05 has critical value: 6.4")
print("since statistic > critical value, we say there is statistically significant difference")
```

```
alpha:0.05 has critical value: 6.4
since statistic > critical value, we say there is statistically significant difference
```

```
cd= 1.4822866628828582
```

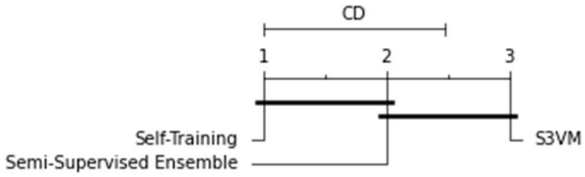


Figure 13 Nemenyi and Friedman's graph for shopper Oversampled Dataset

Result: Statistically Significant difference between Self-Training and S3VM

7.1.3 ReMix Shopper

Dataset Name	Unlabelled Percent	Self-Training	Semi-Supervised Ensembles	Semi-Supervised Support Vector Machine (S3VM)
Online Shopper Intention (ReMix)	10	0.97412 (1)	96.251 (2)	0.91125 (3)
	20	0.97159 (3)	96.080 (2)	0.90488 (3)
	50	0.95689 (1)	93.958 (2)	0.90784 (3)
	90	0.89990 (1)	88.923 (2)	0.84637 (3)
	95	0.87944 (1)	86.820 (2)	0.79307 (3)
Average Rank		1	2	3

```
FriedmanchisquareResult(statistic=10.0, pvalue=0.006737946999085468)
```

```
print("alpha:0.05 has critical value: 6.4")
print("since statistic > critical value, we say there is statistically significant difference")
```

```
alpha:0.05 has critical value: 6.4
since statistic > critical value, we say there is statistically significant difference
```

```
cd= 1.4822866628828582
```

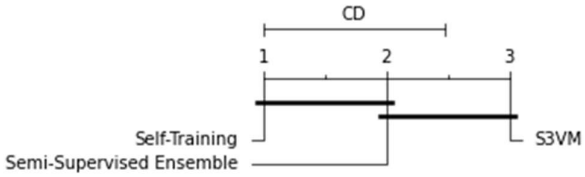


Figure 14 Nemenyi and Friedman's Test on Shopper ReMix

Result: Statistically Significant difference between Self-Training and S3VM

7.1.4 Imbalanced Market

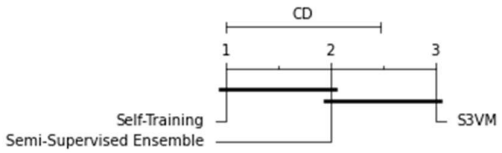
Dataset Name	Unlabelled Percent	Self-Training	Semi-Supervised Ensembles	Semi-Supervised Support Vector Machine (S3VM)
Marketing Dataset (Imbalanced)	10	0.90029 (1)	87.053 (2)	0.51636 (3)
	20	0.89732 (3)	86.309 (2)	0.51636 (3)
	50	0.90625 (1)	81.250 (2)	0.51636 (3)
	90	0.83630 (1)	82.440 (2)	0.51636 (3)
	95	0.83482 (1)	75.000 (2)	0.51636 (3)
Average Rank		1	2	3

```
FriedmanchisquareResult(statistic=10.0, pvalue=0.006737946999085468)
```

```
print("alpha:0.05 has critical value: 6.4")
print("since statistic > critical value, we say there is statistically significant difference")
```

```
alpha:0.05 has critical value: 6.4
since statistic > critical value, we say there is statistically significant difference
```

```
cd= 1.4822866628828582
```



Result: Statistically Significant difference between Self-Training and S3VM

7.1.5 Oversampled Market

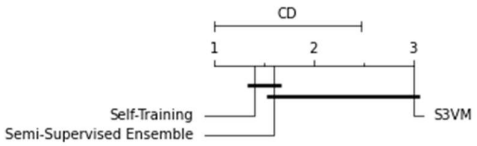
Dataset Name	Unlabelled Percent	Self-Training	Semi-Supervised Ensembles	Semi-Supervised Support Vector Machine (S3VM)
Marketing Dataset (Oversampled)	10	0.88489 (2)	89.496 (1)	0.54244 (3)
	20	0.90503 (1)	89.496 (2)	0.53093 (3)
	50	0.87338 (1)	82.733 (2)	0.52517 (3)
	90	0.77122 (1)	72.086 (2)	0.50647 (3)
	95	0.69352 (2)	71.942 (1)	0.50359 (3)
Average Rank		1.4	1.6	3

```
FriedmanchisquareResult(statistic=10.0, pvalue=0.006737946999085468)
```

```
print("alpha:0.05 has critical value: 6.4")
print("since statistic > critical value, we say there is statistically significant difference")
```

```
alpha:0.05 has critical value: 6.4
since statistic > critical value, we say there is statistically significant difference
```

```
cd= 1.4822866628828582
```



Result: Statistically Significant difference between Self-Training and S3VM

7.1.6 ReMix Market

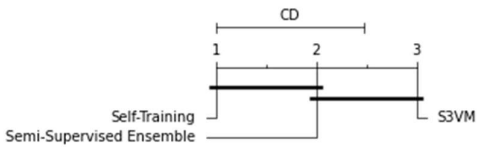
Dataset Name	Unlabelled Percent	Self-Training	Semi-Supervised Ensembles	Semi-Supervised Support Vector Machine (S3VM)
Marketing Dataset (ReMix)	10	0.94732 (1)	93.948 (2)	0.78169 (3)
	20	0.94892 (1)	93.328 (2)	0.75665 (3)
	50	0.93774 (1)	92.397 (2)	0.72769 (3)
	90	0.87869 (1)	86.268 (2)	0.57355 (3)
	95	0.80766 (1)	76.183 (2)	0.52503 (3)
Average Rank		1	2	3

```
FriedmanchisquareResult(statistic=10.0, pvalue=0.006737946999085468)
```

```
print("alpha:0.05 has critical value: 6.4")
print("since statistic > critical value, we say there is statistically significant difference")
```

```
alpha:0.05 has critical value: 6.4
since statistic > critical value, we say there is statistically significant difference
```

```
cd= 1.4822866628828582
```



Result: Statistically Significant difference between Self-Training and S3VM

7.1.7 Imbalanced Heart

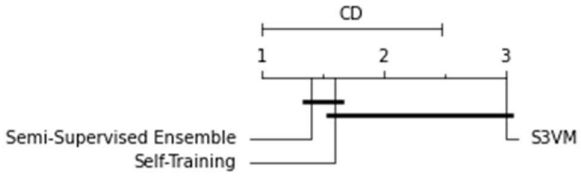
Dataset Name	Unlabelled Percent	Self-Training	Semi-Supervised Ensembles	Semi-Supervised Support Vector Machine (S3VM)
Heart Dataset (Imbalanced)	10	0.81111 (2)	82.222 (1)	0.68888 (3)
	20	0.83333 (1)	83.333 (2)	0.55555 (3)
	50	0.78888 (2)	82.222 (1)	0.71111 (3)
	90	0.81111 (1)	77.777 (2)	0.55555 (3)
	95	0.54444 (2)	72.222 (1)	0.5 (3)
Average Rank		1.6	1.4	3

```
FriedmanchisquareResult(statistic=10.0, pvalue=0.006737946999085468)
```

```
print("alpha:0.05 has critical value: 6.4")
print("since statistic > critical value, we say there is statistically significant diffe
```

```
alpha:0.05 has critical value: 6.4
since statistic > critical value, we say there is statistically significant difference
```

```
cd= 1.4822866628828582
```



Result: Statistically Significant difference between Self-Training and S3VM

7.1.8 Oversampled Heart

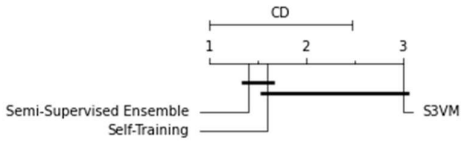
Dataset Name	Unlabelled Percent	Self-Training	Semi-Supervised Ensembles	Semi-Supervised Support Vector Machine (S3VM)
Heart Dataset (Oversampled)	10	0.77551 (2)	78.571 (1)	0.58163 (3)
	20	0.85714 (1)	84.693 (2)	0.58163 (3)
	50	0.81632 (2)	86.734 (1)	0.63265 (3)
	90	0.79591 (1)	72.448 (2)	0.51020 (3)
	95	0.5 (2)	72.448 (1)	0.5 (3)
Average Rank		1.6	1.4	3

```
FriedmanchisquareResult(statistic=9.578947368421062, pvalue=0.00831683351100441)
```

```
print("alpha:0.05 has critical value: 6.4")
print("since statistic > critical value, we say there is statistically significant difference")
```

```
alpha:0.05 has critical value: 6.4
since statistic > critical value, we say there is statistically significant difference
```

```
cd= 1.4822866628828582
```



Result: Statistically Significant difference between Self-Training and S3VM

7.1.9 ReMix Heart

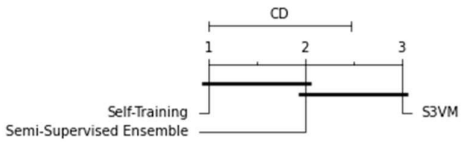
Dataset Name	Unlabelled Percent	Self-Training	Semi-Supervised Ensembles	Semi-Supervised Support Vector Machine (S3VM)
Marketing Dataset (ReMix)	10	0.87050 (2)	93.006 (1)	0.78832 (3)
	20	0.89928 (1)	88.811 (2)	0.78832 (3)
	50	0.89208 (1)	87.412 (2)	0.67153 (3)
	90	0.82733 (1)	80.419 (2)	0.57664 (3)
	95	0.76978 (1)	72.027 (2)	0.43065 (3)
Average Rank		1	2	3

```
FriedmanchisquareResult(statistic=10.0, pvalue=0.006737946999085468)
```

```
print("alpha:0.05 has critical value: 6.4")
print("since statistic > critical value, we say there is statistically significant difference")
```

```
alpha:0.05 has critical value: 6.4
since statistic > critical value, we say there is statistically significant difference
```

```
cd= 1.4822866628828582
```



Result: Statistically Significant difference between Self-Training and S3VM

7.2 Result

Through the two tests we find out that there is statistically significant difference between the Self-Training and S3VM algorithms for Oversampled and ReMix version of Shopper Dataset and all variations of the market dataset

8 Observations and Comments

Since each dataset had 3 variation of class ratios (Imbalanced, Oversampled and ReMix) which in turn had 6 variations of unlabelled class percentage (0%, 10%, 20%, 50%, 90%, 95%), It was a challenge to visually represent the results. For, example for each dataset generated 6 ROC graphs.

8.1 ROC and AUROC (Class-Imbalance vs Unlabelled percentage)

To summarize the ROC Graphs I used interpolation to get the mean graph, to be able to visualize the results properly.

In almost all the ROC curves, I have noticed that the ReMix version of the dataset always had the best result. The Imbalanced versions outperformed the Oversampled versions when the level of imbalance was low (Market and Heart Dataset), but Oversampled was always better when it came to highly imbalanced/skewed dataset (Shopper Dataset)

Moreover, there were some configurations where Oversampled and Imbalanced datasets proved to be beneficial than the ReMix dataset (Figure 16)

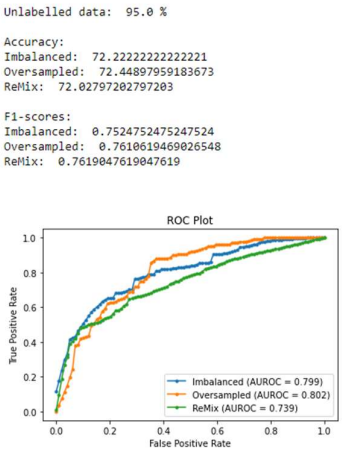


Figure 16 Heart Dataset used by Semi-Supervised Ensemble

8.2 Runtime

It can be noticed from the results above (Figure 10,7,3) that ReMix dataset almost always had the highest runtime, followed by oversampled. This result can be attributed to the size of the dataset, since ReMix dataset is huge, it requires more time to train.

Moreover the relation between the runtime and unlabelled percentage was not clear, meaning that sometimes runtime would increase with increase in the percent of unlabelled data, but sometimes it would decrease as well.

8.3 Accuracy and F1-Score

F1-Score can sometimes prove to be a better metric of evaluation, example for highly imbalanced dataset, because it leverages precision and recall.

Like ROC, ReMix dataset's accuracy was better than the others most of the time, but during the testing for the market dataset by self-training algorithm at 95% of unlabelled classes, the Imbalanced dataset outperformed the ReMix and Oversampled (Figure 4.b), complying to the "No Lunch Theorem" for class-balancing strategies.

8.4 Tests to determine statistically significant difference

I performed Friedman's test followed by Nemenyi's test and found out that there was statistically significant different between the Self-Training and ReMix algorithms for Oversampled and ReMix version of Shopper dataset and all versions of Market dataset at alpha=0.05.

9 Conclusion

I herby conclude the project after stating my observations, comments and techniques used to overcome the challenges I faced. To Recapitulate, Class-Balancing plays a huge role in the performance of a machine learning model and is a very crucial aspect. Also, it is to be noted there exists a possibility of trade-off between the runtime and accuracy, as found out in the experiment, that although ReMix datasets had high runtimes, they were also the one with the best accuracies. Moreover, it is not plausible to admit that an algorithm is significantly better than the other by just looking at the accuracies and the f1-scores, one must perform tests such as Friedman's test or Nemenyi's test to reach to a conclusion.

Finally, I would like to thank prof. Herna Viktor for being such an amazing teacher and guide, and to the TA Farnaz Sadeghi for always being cooperative and supportive.