CSI 5155[F] – Machine Learning

uOttawa

Project Report

Submitted To:

Prof. Herna Viktor

University of Ottawa

Submitted By:

Vasu Mistry (300250384)

vmist089@uottawa.ca

Priya Patel (300152612)

ppate140@uottawa.ca

Master of Computer Science

School of Electrical Engineering and Computer Science

University of Ottawa

Table of Contents

1. Datasets

- The three datasets below are used in this project for implementation:
  1. Online Shoppers Purchasing Intention UCI [1]
  2. Heart Disease UCI [2]
  3. Kaggle – Marketing Campaign [3]

1. Online Shoppers Purchasing Intention Dataset

- The dataset [1] consists of the feature vectors belonging to the online purchasing session of the users in one-year period. 12,330 feature vectors belong to the online session with 17 input features and one output class, "Revenue." The problem is considered as a binary classification problem, whether a user ends up purchasing the product or not. Out of 12,330 sessions, 10,422 belong to class 0 (negative – the user does not buy), and 1908 sessions belong to class 1 (positive–user purchases). Data is highly imbalanced as the distribution of both the classes is not balanced.

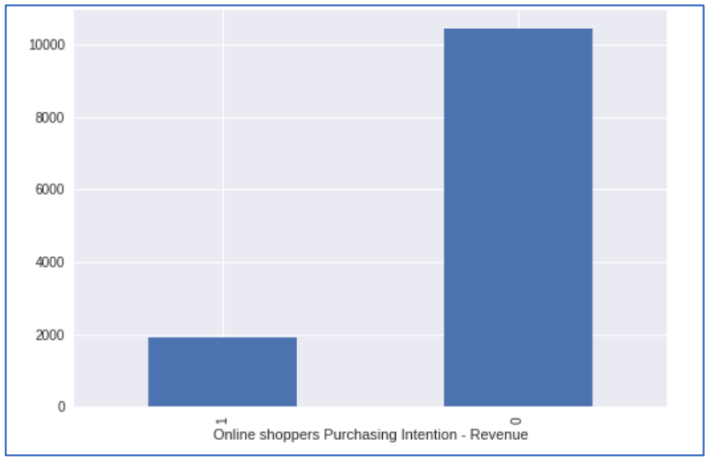


Figure 1: Revenue Count in Online Shoppers Dataset

- The features "Administrative," "Informational," "Product Related" describe the number of various pages visited by the user, and the features "Administrative Duration, "Informational Duration," "Product Related Duration" describe the time spent by the user on these pages. Whereas the features "Bounce Rate," "Exit Rate," and "Page Value" give the metrics given by Google Analytics for each page. The feature "Special Day" represents how near the site's visit to a special day like holidays is.

2. UCI Heart Disease Dataset

- The Heart disease dataset consists of the 303 test results of patients who may or may not have heart disease. The dataset consists of 14 features, including the target feature "target," a binary feature either indicating 0 or 1 viz. The patient has the disease or does not have

the disease. From the 303 target values, 138 belong to class 0, and 165 belong to class 1, indicating that the data is reasonably balanced with more people having heart disease.

- The numeric features "Age" and "Sex" represent patients' age group and gender in the dataset. Categorical feature "cp" means chest pain in 5 categories, 0 being the most severe and five being the least powerful. In contrast, feature "ca" represents several vessels colored during fluoroscopy scans. "thalach" means the maximum heart rate achieved during a stress test. At the same time, "slope, "oldpeak" are features derived from the graph of "thalach."

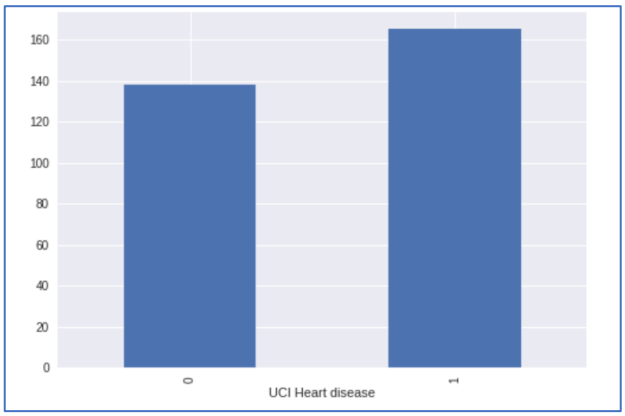


Figure 2: Target Count in Heart Disease Dataset

3. Marketing Campaign – (Teenhome) Dataset

- The Marketing Campaign dataset represents the data of customers collected in a survey to predict whether a customer will respond to a particular offer or not. The dataset has 2240 records with 28 input features and one target variable, "Response." Our study considers feature "Teenhome" as the target feature and performs a binary classification on it. Out of 2240 records, 1158 belong to class 0 (the house does not have any teenage kids), and 1082 belong to class 1 (the house has adolescent kids).

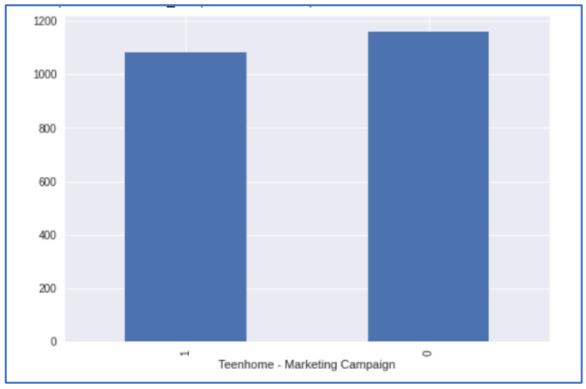


Figure 3: Teenhome count in Marketing Campaign Dataset

- The dataset contains features that indicate monthly expenses occurred on different commodities like Wine, Fish, Meat, Gold, and sweet products. It also captures discount offers utilized by the family through various mediums like in-stores, websites, coupons,

catalogs, etc. Other features like “AcceptedCmp” denote whether a user accepted an offer from three different campaigns, which are one-hot encoded. Feature “Z_Revenue” describes a company’s revenue after a client has accepted a specific campaign.

2.Semi-Supervised Algorithms

- The semi-supervised algorithms allow us to capture the strength of supervised and unsupervised methods by learning from labeled and unlabeled data. We use a semi-supervised learning approach when we have a high amount of unlabeled data and very few labeled data. We train a model on labeled data and use it to make predictions on unlabeled data to generate the pseudo-labels and modify our training set by incorporating highly confident pseudo-labels.

1. **Self-Training Classifier**: The self-training classifier is the basic version of a semi-supervised algorithm that iteratively trains on the labeled data and predicts class labels (pseudo-labels) for the unlabeled data. The pseudo-labels that are predicted with high confidence are added to the labeled training data, and the process continues until the classifier generates no high confident pseudo-labels.

2. **Co-Training Classifier:** The co-training classifier implements two disagreement-based algorithms trained on two different views of the same dataset. In this method, the highly confident pseudo-labels generated by classifier one are merged into labeled data of classifier two and vice versa. This process continues until some stopping criterion is met, such as any one of the two classifiers stopping generating highly confident pseudo-labels. It is vital to have two mutually independent views of the dataset for this method to work well. It is also crucial that each view is self-sufficient to predict the output class. However, finding such independent and autonomous views for the dataset isn't easy [8].

3. **Semi-Supervised Ensemble:** Traditional bagging and boosting-based ensembles implement multiple base learners to produce a model that generalizes well on the testing data. Similarly, in the semi-supervised ensemble, base learners are self-training classifiers that learn from the labeled data and improve on unlabeled data. Bagging-based semi-supervised ensembles consider the majority voting approach to get the final results while boosting attempts to minimize the loss function.

4. **Unsupervised Preprocessing:** In this approach, we use an unsupervised stage for automatic feature extraction based on the unlabeled data (unsupervised clustering) followed by training a supervised classifier on the labeled data. Features learned from the unsupervised stage are combined with existing features to generate a new version of the training dataset. Then a self-training or co-training classifier can be used to make predictions on the unlabeled dataset.

5. **Intrinsically Semi-Supervised Learning:** In contrast to the above four approaches, intrinsic SSL has inbuilt semi-supervised learning. Instead of following the traditional iterative training on the labeled data, predicting pseudo-labels of unlabeled data, and merging high confident pseudo-labels to labeled data approach, it has a single train and merge step. A hybrid feature set comprising labeled and unlabeled data is fed to the intrinsic algorithm that internally labels the unlabeled data and returns an output vector of class labels. Engelen et. Al [5] classifies such approach as a Graph inference based under Transductive category, however, as covered in class, the algorithm selected by us (Label Spreading) works by generating manifolds covered in Section 6.3

3. Experiments

3.1 Data Pre-processing

3.1.1 Dataset 1 – Online Shoppers Purchasing Intention:

1. **Feature Engineering**: The categorical feature Month was cyclically encoded, and VisitorType was transformed using One-Hot Encoding. Post encoding of VisitorType, two new feature columns were added, resulting in a new 20 column feature set. The 4 Ordinally encoded features like OperatingSystems (#cat=8), Browser(#cat=13), Region(#cat=9), TrafficType(#cat=20) were not encoded because of their large number of categories. The motivation for this decision was that if encoded, the resultant feature set would comprise 75 features. Still, most of the values would be zero (sparse matrix) due to the fundamental property of one-hot encoding. Out of these 75 features, only 11 (10 numerical and one boolean) were guaranteed to contain useful information. Instead, the ten numerical features and the four ordinally encoded features as discussed above were normalized in a range of [0,1].

2. **Feature Selection**: We used Recursive Feature Elimination with 3-fold cross-validation and f1 score as the evaluation parameter due to the class imbalance. As shown in the below figure, we went with selecting 19 features as reported by the RFE function.

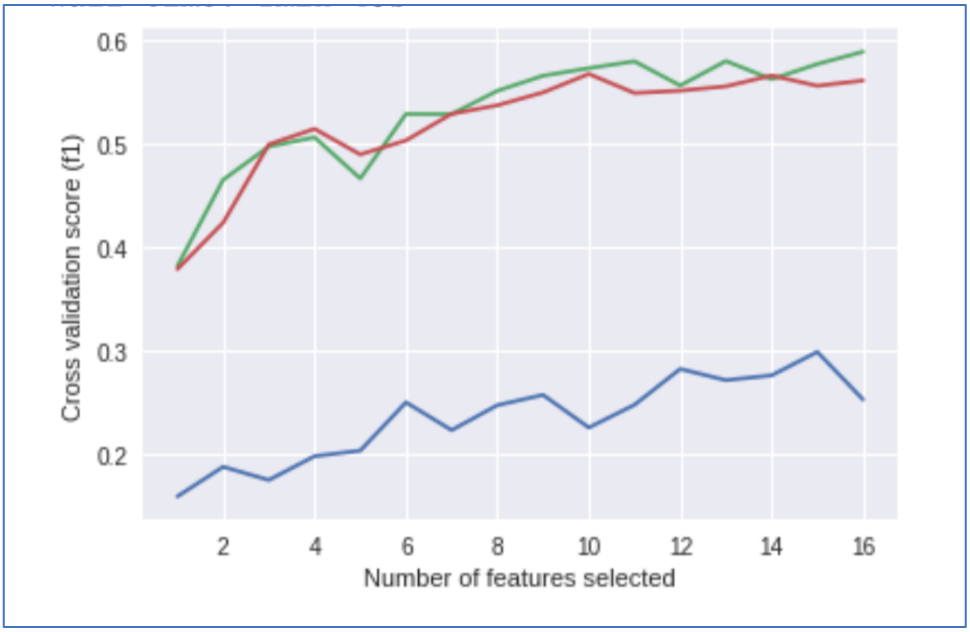


Figure 4: Results of RFE function for Dataset 1

3.1.2 Dataset 2 – UCI Heart Disease

1. **Feature Engineering:** Categorical features ‘cp,’ ‘ca,’ ‘thal,’ ‘thal’ were one-hot encoded. In contrast, numerical features ‘age,’ ‘trestbps,’ ‘chol,’ ‘thalach,’ ‘oldpeak,’ ‘slope’ were standardized using Min-Max scaling algorithm to achieve a normal distribution.

2. **Feature Selection:** Pearson’s correlation-based feature selection technique was used to select features with >0.1 correlation with the target column.

```
thal_2      0.527334
cp_0        0.516015
thal_3      0.486112
ca_0        0.465590
exang       0.436757
oldpeak     0.430696
thalach     0.421741
slope       0.345877
cp_2        0.316742
sex         0.280937
ca_2        0.273998
cp_1        0.245879
ca_1        0.232412
age         0.225439
ca_3        0.210615
trestbps    0.144931
restecg     0.137230
thal_1      0.106589
cp_3        0.086957
chol        0.085239
ca_4        0.066441
fbs         0.028046
thal_0      0.007293
Name: target, dtype: float64
```

Figure 5: Pearson’s Correlation Values for Dataset 2

3.1.3 Dataset 3 – Marketing Campaign

1. **Feature Engineering:** To convert into a binary classification problem, the target column "Teenhome" was binarized to 0 and 1, i.e., all values ≥ 1 were replaced with 1. Feature 'Martial_Status' was also binarized where 0 indicates a single or divorced household and 1 indicates a family. 'Education' was grouped based on three education levels: Basic, medium, and highly educated. A new feature, 'SweetPurchases,' was created by combining 'MntFruits' and 'MntSweetProducts' since a teenage household is likely to spend more on sweets. Other food items were combined to create the 'Spent' feature since a household/family is expected to spend more than an individual. All discount offer features were also combined to a single column with the assumption that a family is likely to avail more discounts than an individual. Income and Age were analyzed for any discrepancies and outliers, and specific observations were removed based on this analysis, like a person earning around 600,000 annually, which was more than the column average. All features were then standardized using the Min-Max scaling algorithm.

2. **Feature Selection:** Pearson's correlation-based feature selection technique selected features with >0.01 correlation with the target column.

```
NumDealsPurchases      0.394803
Age                    0.368572
MntMeatProducts        0.275858
MntFishProducts        0.210854
SweetPurchases         0.195425
Response               0.161871
NumWebPurchases        0.157856
NumWebVisitsMonth      0.142853
Spent                  0.134999
TotalAcceptedOffers    0.124528
NumCatalogPurchases    0.116324
Marital_Status         0.101892
NumStorePurchases      0.048166
Kidhome                0.036687
Income                 0.029775
MntGoldProds           0.023137
Recency                0.003441
Complain               0.003146
MntWines               0.002497
ID                     0.002151
Name: Teenhome, dtype: float64
```

Figure 6: Pearson's Correlation Values for Dataset 3

3.2 Evaluation and Metrics

- Considering the imbalance in dataset #1 F1 score was selected as the evaluation metric and the almost balanced dataset's #2 and #3, accuracy was selected as the evaluation metric. Overall, we report f1 and accuracy scores for all the three datasets. For the evaluation, considering that we had to test and report scores of 5 splits of the each dataset and include balanced & imbalanced versions, we opted for a holdout based evaluation method. We wrote a custom helper function which takes the input features *X* and target column *y* along with desired unlabeled to labelled data split ratio and test size as input. The function first

splits the dataset into $holdout\ H\ and\ unlabelled\ U$ using $unlabelled\_size$ parameter, after which H is further divided into training set T_r and test set T_s. While applying semi supervised learning T_r is modified by appending confident predictions from U made by the SSL algorithm, and after each such modification the resultant model is evaluated on the constant holdout test set T_s. Thus, this approach ensures that all evaluation scores are reported using the same test set.

Function signature: $reset\_holdout\_splits(X, y, unlabelled\_size, test\_size)$

For scores reported on 'balanced' dataset, SMOTE oversampler was used with $sampling\_strategy = 1$ which allowed oversampling of minority class such that $\frac{N_{minority}}{N_{majority}} = 1$ where N is number of samples in the training set T_r.

3.3 Classifiers used in various SSL algorithms

- The motivation behind the selection of these classifiers are as follows:

1. **Self-Training:** Random Forest was selected based on the findings reported by Sakar et al. [4] and our experiments in Assignment 1 & 2 on this dataset; Random Forest performed the best.

2. **Co-Training:** Classifiers need to be inherently different; they should complement each other's performance and work well on the specific set of views provided to them. Hence we decided to choose a distance-based classifier and a probabilistic classifier. For our experiment, we tried KNN, LogisticRegression, SVM, decision tree, and RandomForest. Based on our 10% unlabeled data-split experiments, we found the pair of (SVM, Random Forest) to perform the best. KNN inherently had poor performance in our Assignment 1 & 2 experiments, while LogisticRegression and DecisionTree Classifiers could not match SVM and RandomForest's performance.

3. **Semi Supervised Ensemble:** Again we experimented with Random Forests (bagging), Gradient Boosting, and the popular eXtreme Gradient Boosting (XGBoost) classifiers. XGBoost provides a large number of parameters to fine-tune the model. However, we found that it worked the best on the default parameter set, which was quite interesting to see. Later on, we discovered that CATBoost works well without the need for major hyper-parameter tuning. If we had more time, we would have experimented with CATBoost too. Initially, we tried to re-use the earlier trained model and add new labeled instances to reduce computational overhead. However, we noticed that this approach resulted in poor performance. Ideally, this should be because the inherent weak trees in the model remain untouched due to this re-use. The algorithm started from scratch while generating a new model after appending new labeled instances to the training data. It used the boosting principle focused on improving the performance of the weaker trees. Intuitively, we think an OzaBag algorithm that removes vulnerable learners when the ADWIN detector detects

a concept drift can be modified to be used for semi-supervised learning, i.e., when we add new labeled instances to the existing model, we find weaker trees and remove them to improve the performance instead of generating the entire model again from scratch.

4. **Unsupervised Pre-Labeling:** Our first approach was to experiment with semi-supervised pretraining' using Autoencoders. We used a shallow autoencoder to learn feature representations by re-constructing the input and adding a new layer on every iteration. After this training, we added an output layer of size 2 to make predictions based on re-construction loss. However, we could only achieve a maximum F1 score of 0.3 and accuracy 0.62 while training with 90% labeled data. We then decided to implement an unsupervised pre-labeling technique as covered in Section 5.2. We first feed the training data to an unsupervised clustering algorithm and use its labels to create a new feature. We then train a supervised classifier using this new feature set. The idea here is to generate a new feature in an unsupervised setting and train the supervised classifier. If the unsupervised algorithm can correctly identify the clusters, then the resultant feature created will be highly correlated with the prediction class in a supervised setting. We used density-based scanning (DBScan) and the linear algorithm to decide on a supervised algorithm. DBScan identifies the number of clusters automatically in the dataset, while KMeans allows us to specify the number of groups. Even after experimenting with various eps and leaf_size values, DBScan could not outperform KMeans. While we expected KMeans to work well on datasets #2 (heart disease prediction) and #3 (teenhome prediction), since we could see that they were linearly separable, we expected DBScan to perform well on the skewed dataset 1 (online shoppers intention).

5. **Intrinsically semi-supervised:** As covered in class, we used the manifold-based LabelPropagation and LabelSpreading algorithms. We decided LabelSpreading because it works well with noisy data, and we were adding synthetic samples in each data-split which increased the likelihood of adding noise to the dataset. We verified this by comparing their performance on dataset 1 with 10% unlabeled data. Regarding the choice of kernels, for dataset 1, we found that the linear kernel 'knn' worked well on unsampled data while the fully connected kernel 'RBF' worked well on the balanced data. The linear kernel was generally more robust for datasets #2 and #3, except in the 95% unlabeled data split where the fully connected kernel worked well. Another observation was that the features selected as discussed in Sections 3.1 did not work well for this algorithm, with a maximum f1 score of only 0.4. We found that selecting highly correlated features, viz. > 0.5, 0.25, and 0.15 in the three datasets respectively worked well. This makes sense because supplying fewer features with more correlation allows the algorithm to work in lower dimensions containing beneficial information. After running these experiments, we noticed a performance increase of 62% in the f1 score.

- **Confidence level:** We initially started with a confidence score of 0.8 for our experiments on dataset 1. But, we found it too low since the algorithm's performance was relatively low compared to our 100% labeled (fully-supervised) baseline. Thus, considering the

performance v/s computation time trade-off throughout our experiments, we have used confidence scores of 0.9 and 0.95. Specifically, when the ratio of labeled data was less, we used the higher confidence score and vice versa.

- The table below depicts the hyper-parameters used for each algorithm:

Table 1: Hyperparameters used for Algorithms

Sr. No.	**SSL algorithm**	**Classifier**	**Hyperparameters tuned**
1.	Self-Training	Random Forest	n_estimators: {111,1001} max_depth:{3,9} min_samples_leaf: {2,3,4} min_samples_split: {2,3}
2.	Co-Training C_1	SVM	C: {1,3,10} gamma: {1} kernel: {rbf, linear}
3.	Co-Training C_2	Random Forest	n_estimators: {111,1001} max_depth:{3,9} min_samples_leaf: {2,3,4} min_samples_split: {2,3}
4.	Semi-supervised ensemble	XGBoost	learning_rate: {0.1,0.3} max_leaves: {0,10,15} lambda: {1,0.1,5}
5.	Unsupervised preprocessing	KMeans	n_clusters = {2}
6.	Intrinsically semi supervised I_1	Label Spreading with rbf kernel	kernel: rbf alpha: 0.1 max_iter: 100 tol: 0.01
7.	Intrinsically semi supervised I_2	Label Spreading with knn kernel	kernel: knn alpha: 1e-4 max_iter: 1000 n_neighbors: 19 tol: 0.24

4. Results

- Table 3. shows the results for fully supervised model for the three datasets which we consider as baseline. We have used Random Forest for dataset #1 & #2 and Gradient Boosting classifier for dataset #3. We have performed our experiments on imbalanced and balanced data with sampling strategy as discussed in Section 3.2.

Table 3: Baseline Accuracy and F1-score values for Three Datasets

Dataset	Skewness	Classifier	Accuracy	F1
Online Shoppers	Imbalanced	Random Forest	90.59	64.05
Online Shoppers	Balanced	Random Forest	88.19	67.51
UCI Heart Disease	Imbalanced	Random Forest	90.91	90.11
UCI Heart Disease	Balanced	Random Forest	90.72	90.11
Marketing Campaign	Imbalanced	Gradient Boosting	91.8	91.49
Marketing Campaign	Balanced	Gradient Boosting	91.36	90.97

- We consider the results of fully supervised algorithms to be baseline results. Random Forest achieves good accuracy for both the datasets but the F1-score for the first dataset is poor. The results for imbalanced and balanced data are similar for the second and third datasets, but the classifier improves the performance of the F1-score for the first dataset. Gradient Boosting performs equally for imbalanced and balanced data. We got a poor F1-score for the first dataset because the original data is highly imbalanced.

- Considering our best results from Assignment 2 in a fully supervised setting as a baseline, we compare the performances of different SSL approaches. We also consider the performance when the dataset is imbalanced (No synthetic sampling) and when the dataset is balanced using SMOTE, as discussed in Section 3.2.

Dataset 1 (Online Shoppers Intention):

- Table 4 shows the accuracy scores and table 5 shows the F1-scores for imbalanced and balanced data for dataset 1. The balanced data is achieved through oversampling the minority class in the dataset.

Table 4: Accuracy Scores on Dataset 1 – Online Shoppers
(I = Imbalanced/Un-sampled, B = Balanced data using over-sampling)

Accuracy *scores on Dataset 1 – Online Shoppers*										
unlabelled data	***10%***		***20%***		***50%***		***90%***		***95%***	
SSL Algorithm	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*
Self-Training	90.23	88.24	90.77	87.97	89.94	87.35	87.50	84.21	90.32	87.90
Co-Training	88.37	87.47	86.82	88.19	90.34	87.34	89.48	84.61	92.74	88.70
Ensemble	90.59	88.4	90.48	87.73	90.48	88.79	89.47	85.02	92.74	89.51
Unsupervised Pre-processing	90.9	87.1	88.85	87.43	88.98	86.70	90.32	83.06	91.90	87.10
Intrinsically semi supervised	89.45	88.92	88.85	87.13	88.48	87.59	86.23	88.66	83.87	87.10

Table 5: F1-Scores on Dataset 1 – Online Shoppers
(I = Imbalanced/Un-sampled, B = Balanced data using over-sampling)

F1 *scores on Dataset 1 – Online Shoppers*										
unlabelled data	***10%***		***20%***		***50%***		***90%***		***95%***	
SSL Algorithm	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*
Self-Training	62.26	68.74	65.27	68.10	58.11	67.50	49.18	58.95	57.14	69.39
Co-Training	45.80	61.39	47.47	60.61	60.98	57.14	51.85	56.82	18.18	50.91
Ensemble	66.13	68.70	66.55	67.91	62.69	66.95	60.61	58.70	76.92	70.83
Unsupervised Pre-processing	65.05	65.88	57.03	66.84	54.67	63.72	57.14	58.82	73.68	69.23
Intrinsically semi supervised	60.07	63.06	60.14	60.18	55.90	58.98	48.48	61.11	50.00	55.55

- **Imbalanced:** We can see that the accuracy scores were equal or better for all the 6 data splits, and 70% of the unlabeled data were added to the dataset on average. However, F1 scores were equal to or lower than baseline. This was expected considering the class imbalance and strengthened our decision to use the F1 score as the evaluation metric over accuracy. Co-Training performed the worst while the ensemble was the best.

- **Balanced:** The scores dropped lower than baseline for both accuracy and f1. The best performing approaches were Ensemble-based and Self-Training, while Co-Training was the worst. The f1 scores dropped from 10% to 90% data splits and were higher for the 95% split. This can be explained due to the very few test samples available to evaluate the model. Hence, we consider accuracy in assessing the 95% split where the overall performance was closer to baseline.

- **Balanced v/s Imbalanced:** Performance on imbalance dataset was better than on the synthetically balanced dataset. This can be attributed to the fact that a lot of artificial

samples were added to the dataset due to high imbalance, which introduced noise. If the dataset had been relatively balanced, perhaps a lower difference could have been observed. Also, in both the scenarios, Unsupervised pretraining improved as the labeled data decreased while the ensemble remained stable throughout.

Dataset 2 (Heart Disease UCI):

- Table 6 shows the accuracy scores and table 7 shows the F1-scores for imbalanced and balanced data for dataset 2.

Table 6: Accuracy Scores on Dataset 2 – Heart Disease UCI
(I = Imbalanced/Un-sampled, B = Balanced data using over-sampling)

Accuracy *scores on Dataset 2 – Heart Disease UCI*										
unlabelled data	***10%***		***20%***		***50%***		***90%***		***95%***	
SSL Algorithm	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*
Self-Training	83.64	87.27	89.80	85.45	83.87	87.10	66.67	50.00	100	100
Co-Training	85.45	85.45	85.71	85.71	87.10	87.10	66.67	66.67	33.33	33.33
Ensemble	85.45	85.45	89.80	83.67	87.10	90.32	66.67	66.67	100	100
Unsupervised Pre-processing	85.71	71.43	72.00	92.00	75.00	87.50	100	100	50	100
Intrinsically semi supervised	89.09	87.27	87.76	87.76	87.10	87.10	100	100	66.67	66.67

Table 7: F1-Scores on Dataset 2 – Heart Disease UCI
(I = Imbalanced/Un-sampled, B = Balanced data using over-sampling)

F1 *scores Dataset 2 – Heart Disease UCI*										
unlabelled data	***10%***		***20%***		***50%***		***90%***		***95%***	
SSL Algorithm	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*
Self-Training	89.23	88.52	87.27	88.89	88.24	87.50	66.67	85.71	100	100
Co-Training	88.89	90.63	86.79	86.79	88.89	88.89	75.00	85.71	0	0
Ensemble	87.50	90.63	90.91	89.29	88.24	88.24	66.67	66.67	100	100
Unsupervised Pre-processing	90.32	86.67	82.76	88.89	94.12	94.74	66.67	100	100	100
Intrinsically semi supervised	90.63	88.89	88.00	88.00	88.24	90.32	100	100	66.67	66.67

- **Imbalanced:** Accuracy scores were 5% lower than baseline, acceptable, while 80% of unlabeled data were added to the dataset on average. F1 score was quite like a baseline,

with Unsupervised and Ensemble approaches performing the best while Co-Training was the worst.

- **Balanced:** Co-Training and Ensemble outscored baseline in F1 score and were similar to baseline accuracy. The scores kept dropping from 10% to 90% data splits, and the 50% split had the most comparable performance with baseline accuracy and f1 score.

- **Balance v/s Imbalance:** When unlabeled data was less (10% & 20% splits), approaches performed well on the imbalance dataset, while the opposite was observed in the other three partitions (50%, 90%, and 95%). This can be explained by the less data size; as the available labeled data decreased, very few samples were available to train the model. It was oversampling them generated sufficient samples, which resulted in this performance improvement.

Dataset 3 (Marketing Campaign):

Table 8 shows the accuracy scores and table 9 shows the F1-scores for imbalanced and balanced data for dataset 3.

Table 8: Accuracy Scores on Dataset 3 – Marketing Campaign
(I = Imbalanced/Un-sampled, B = Balanced data using over-sampling)

Accuracy *scores on Dataset 3 – Marketing Campaign*										
unlabelled data	***10%***		***20%***		***50%***		***90%***		***95%***	
SSL Algorithm	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*
Self-Training	89.08	87.84	88.55	88.09	87.50	87.50	80.00	75.56	73.91	78.26
Co-Training	87.84	87.84	89.39	89.39	87.95	87.50	80.00	80.00	78.26	78.26
Ensemble	91.56	90.57	90.50	91.62	88.39	88.39	77.78	80.00	82.61	82.61
Unsupervised Pre-processing	89.11	85.15	89.94	90.50	88.39	81.25	78.26	95.65	66.67	91.67
Intrinsically semi supervised	88.09	86.85	84.36	83.52	82.14	82.14	71.11	71.11	73.91	78.26

Table 9: F1-Scores on Dataset 3 – Marketing Campaign
(I = Imbalanced/Un-sampled, B = Balanced data using over-sampling)

F1 *scores on dataset 3 – Marketing Campaign*										
unlabelled data	***10%***		***20%***		***50%***		***90%***		***95%***	
SSL Algorithm	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*
Self-Training	88.72	87.53	88.18	87.69	87.27	87.16	80.00	74.42	66.67	73.68
Co-Training	87.34	87.40	89.14	89.20	87.56	87.39	79.07	79.07	73.68	73.68
Ensemble	91.10	90.21	90.12	91.48	88.18	88.39	77.27	80.00	80.00	80.00

Unsupervised Pre-processing	89.00	84.69	89.53	89.94	88.29	81.08	70.59	95.65	50.00	90.91
Intrinsically semi supervised	87.88	87.21	84.44	84.53	81.31	81.48	71.11	69.57	70.00	76.19

- **Imbalanced:** Ensemble and Unsupervised pre-labeling approaches managed a similar or better performance over baseline in terms of accuracy while having an average of –1.5% less f1 score than baseline. Self-Training and Co-Training showed stable performances overall data splits compared to the Manifold-based intrinsic approach. All the methods were closer in terms of mean f1 scores than the other two datasets over the 5 data splits, with ensemble being the best and intrinsic being the worst.

- **Balanced:** Ensemble and Unsupervised approaches were closer to baseline score, while co-training and self-training were lower, with the intrinsic performance the worst in terms of f1 scores and accuracy. The scores remained similar for 10% and 20% splits, dropped 50%, and were back up near the baseline for 90% & 95% splits.

- **Balance v/s Imbalance:** As shown in Fig. 3, the dataset was balanced. Hence no significant performance difference was expected after resampling. As the amount of label data decreased, the performance of the sampled datasets was higher than the un-sampled dataset. Thus, we can infer that generating new synthetic samples worked well on this dataset. We can select less labeled data splits, making the dataset an ideal candidate for semi-supervised learning.

5. Analysis of Results

5.1 ROC and Runtime plots

5.1.1 Runtime Plots

Runtime Values for Fully Supervised algorithm:

Table 10: Runtime Values for Fully supervised algorithm (Baseline)

Dataset	Skew	Classifier	Runtime
Online Shoppers	Imbalanced	Random Forest	1.57s
Online Shoppers	Balanced	Random Forest	2.28s
UCI Heart Disease	Imbalanced	Random Forest	232ms
UCI Heart Disease	Balanced	Random Forest	230ms
Marketing Campaign	Imbalanced	Gradient Boosting	1.14s
Marketing Campaign	Balanced	Gradient Boosting	1.2s

- From the above table 10, the Random Forest and Gradient Boosting classifier runs quickly and do not have a longer runtime.

Runtime Values for Dataset 1 (Online Shoppers):

Table 11: Runtime Values of algorithms for Dataset 1 (Online Shoppers)

Runtime values on Dataset 1 – Online Shoppers										
unlabelled data	***10%***		***20%***		***50%***		***90%***		***95%***	
SSL Algorithm	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*
Self-Training	39.9s	27.2s	35.8s	22.3s	1min 50s	46.8s	1min 49s	4min 31s	28.2s	5min 34s
Co-Training	58.9s	4min 30s	32.1s	4min 53s	5min 25s	8min 18s	43.3s	10min 22s	42.8s	5min 14s
Ensemble	17.9s	11.1s	9.31s	19s	22.7s	44s	30.1s	4min 25s	22.9s	2min 14s
Unsupervised Pre-processing	43.7s	25.1s	31.7s	29.3s	1min 27s	1min 1s	1min 27s	5min 23s	39.2s	5min 42s
Intrinsically semi supervised	531ms	5.48s	523ms	5.53s	580ms	7.62s	625ms	11s	655ms	13.1s

- From above table 11, we can see that for dataset 1, when the percentage of unlabeled data increases, the runtime also increases significantly. The intrinsic semi-supervised algorithm is the fastest, and co-training is relatively slowest.

Runtime Values for Dataset 2 (Heart Disease):

Table 12: Runtime Values of algorithms for Dataset 2 (Heart Disease)

Runtime values on Dataset 2 – Heart Disease										
unlabelled data	***10%***		***20%***		***50%***		***90%***		***95%***	
SSL Algorithm	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*
Self-Training	1.11s	2.25s	3.25 s	1.68 s	4.9 s	6.16 s	548 ms	1.12 s	555 ms	555 ms
Co-Training	601m s	1.38s	643 ms	1.36 s	41.8 ms	47.4 ms	2.9 s	3.1 s	31 ms	33.2 ms
Ensemble	258 ms	155m s	150 ms	322 ms	246 ms	647 ms	465 ms	436 ms	32.4 ms	43 ms
Unsupervised Pre-processing	3.38s	3.41s	1.67 s	2.82 s	4.4 s	5.14 s	8.72 s	5.66 s	574 ms	582 ms

Intrinsically semi supervised	21.3ms	15.9 ms	18.7 ms	21.3 ms	21.4 ms	20.8 ms	19.5 ms	25.7 ms	19.4 ms	21.5 ms

- From above table 12, no algorithm is slow for any split of the data for the second dataset. Even for 95% of the unlabeled data, there are few observations making it fast to execute.

Runtime Values for Dataset 3 (Marketing Campaign):

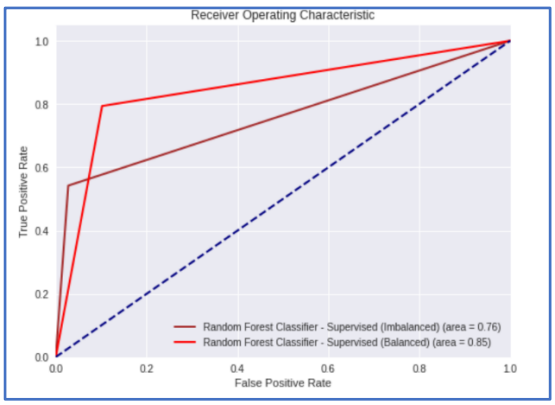
Table 13: Runtime Values of algorithms for Dataset 3 (Marketing Campaign)

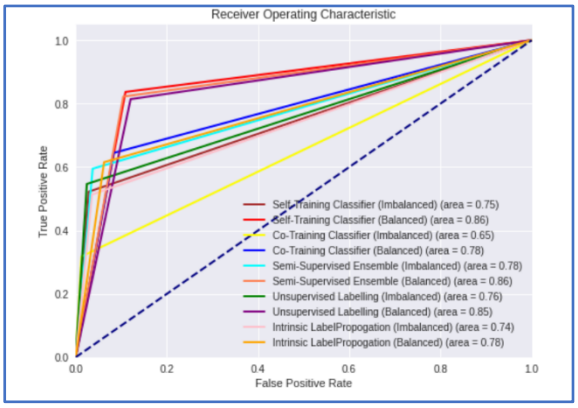
Runtime values on Dataset 3 – Marketing Campaign										
unlabelled data	***10%***		***20%***		***50%***		***90%***		***95%***	
SSL Algorithm	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*	*I*	*B*
Self-Training	6.26s	4.41s	9.35s	4.24s	6.24s	16.8s	29.3s	36.6s	19.5s	31.3s
Co-Training	5.48s	6.15s	8.21s	8.62s	8.84s	10.8s	11.3s	19.5s	9.5s	13s
Ensemble	819ms	1.31s	1.47s	1.3s	2s	3.4s	2.79s	9.92s	1.54s	2.83s
Unsupervised Pre-processing	3.76s	5.84s	9.87s	10.1s	11.2s	16.2s	23.2s	35.1s	16.7s	18.9s
Intrinsically semi supervised	307ms	277ms	97ms	86ms	261ms	314ms	610ms	133ms	374ms	350ms

- From above table 13, no algorithm is slow for any split of the data for the third dataset. Even for 95% of the unlabeled data, there are few observations, so the algorithms run fast. The intrinsic algorithm is again high-speed among all.

5.1.2 ROC Curve Plots

ROC Plots for Dataset 1 – Online shoppers





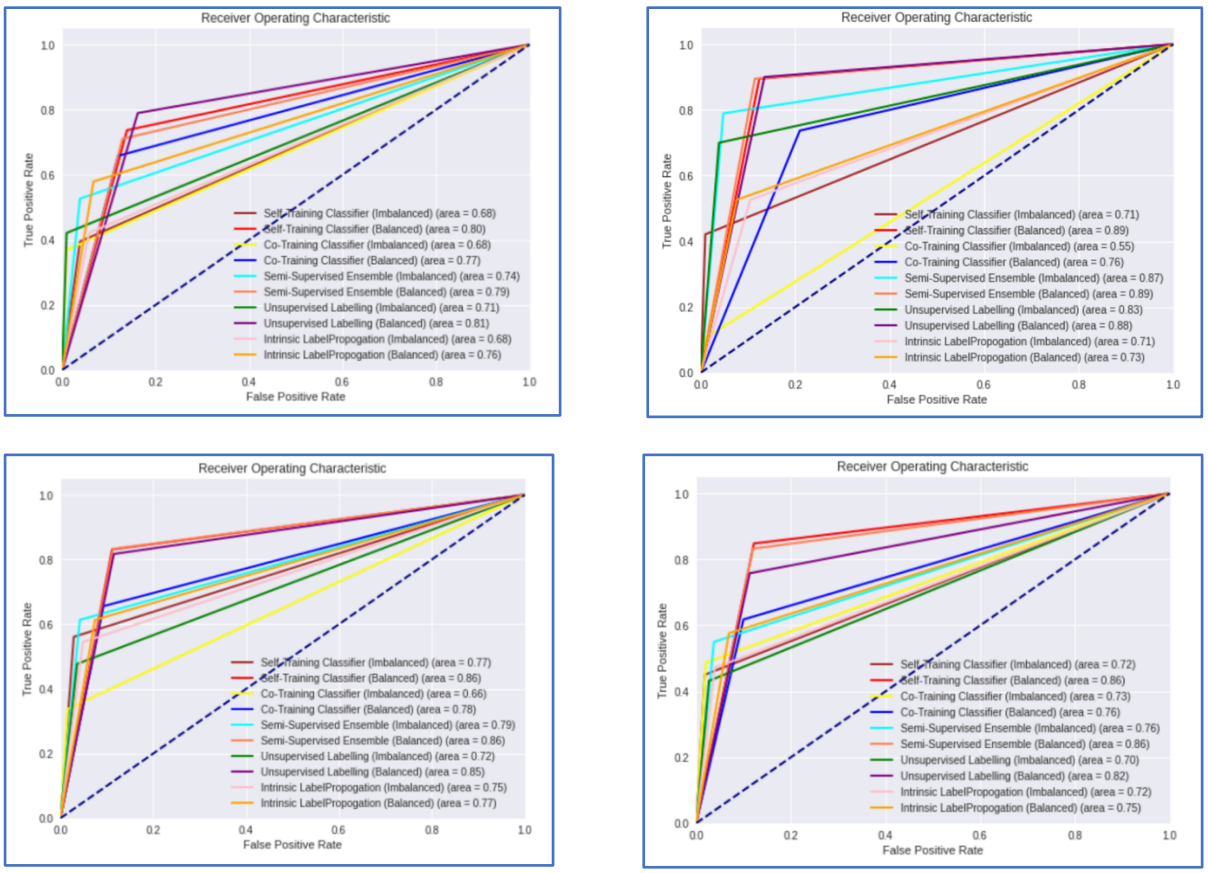
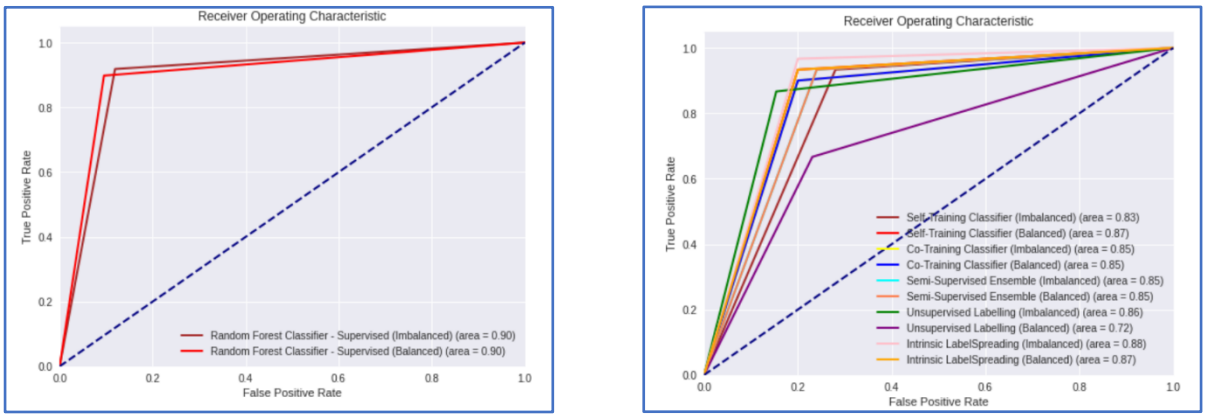


Figure 7: ROC Plots for Dataset 1 (0%, 10%, 20%, 50%, 90%, 95% Unlabeled Data)

- The above Figure 7 shows the ROC plots for the various splits of dataset 1. For fully supervised Random Forest, the roc for balanced data has more area under the curve as it performs better than the imbalanced data. From the figure, the self-training classifier for the balanced data, the semi-supervised ensemble for the balanced data, and Unsupervised labeling for balanced data have the higher area under the curve. In comparison, the co-training classifier for the unbalanced data and unsupervised labeling for the imbalanced data have the lower area under the curve.

ROC Plots for Dataset 2 – Heart Disease



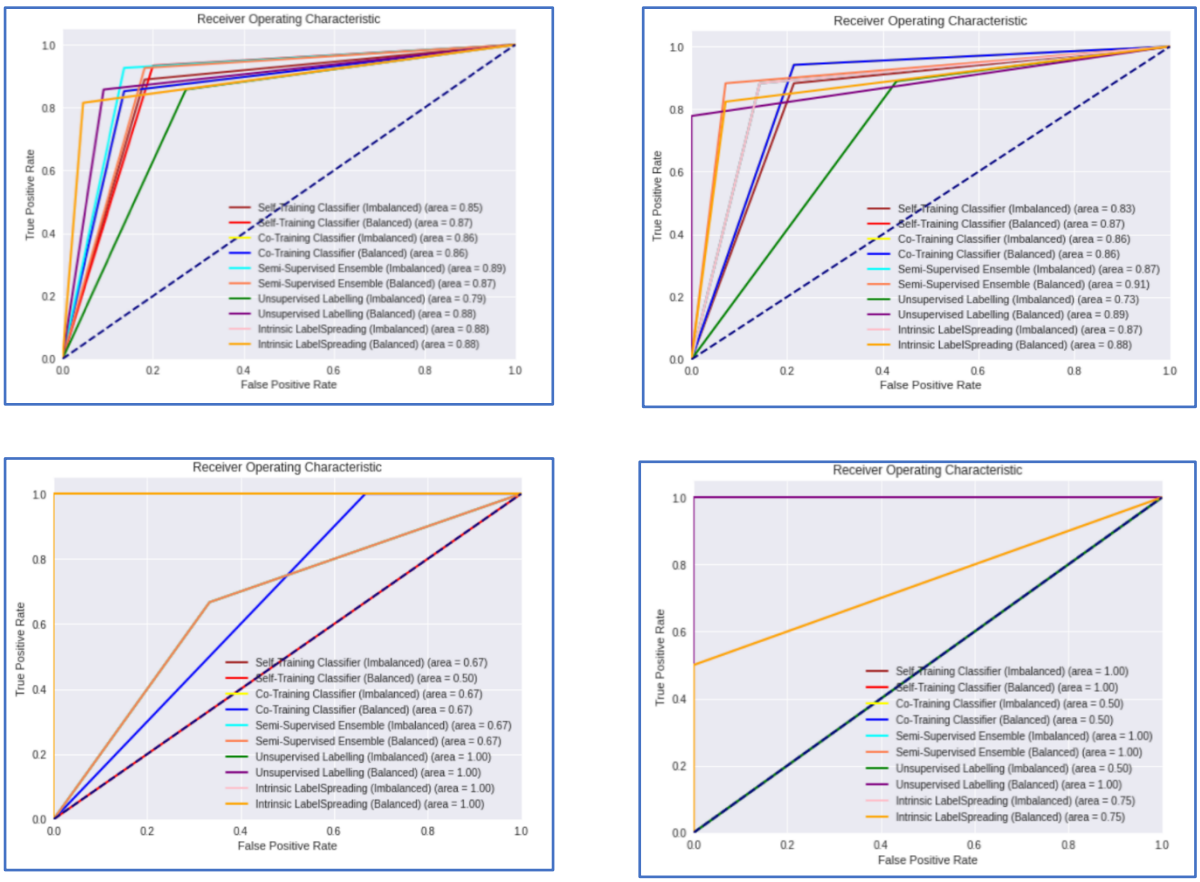
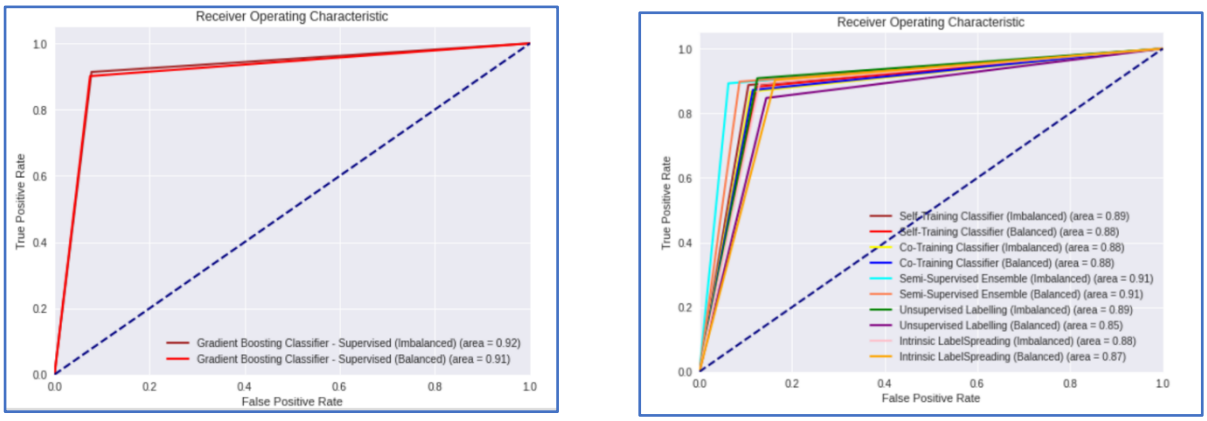


Figure 8: ROC Plots for Dataset 2 (0%, 10%, 20%, 50%, 90%, 95% Unlabeled Data)

- From Figure 8, the area under the curve for the heart dataset becomes more skewed as the percentage of unlabeled data increases. For 0% unlabeled data (fully supervised), the area under the curve is almost similar for the imbalanced and balanced datasets. For 10%, 20%, and 50% unlabeled data, the skewness is moderate, and for 90% and 95% unlabeled data, the area under the curve is random. The 90% and 95% splits are either random guessing or overfitting the training data.

ROC Plots for Dataset 3 – Marketing Campaign



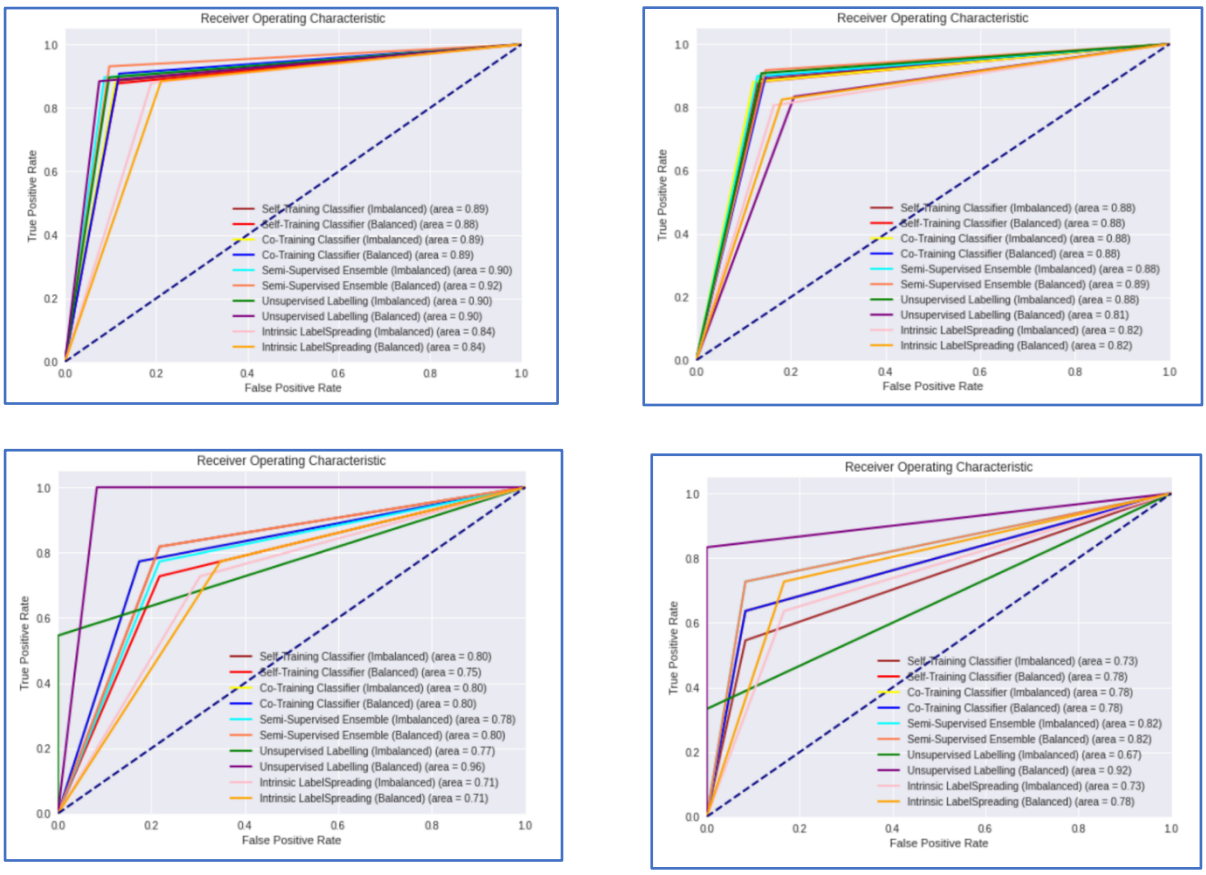


Figure 9: ROC Plots for Dataset 3 (0%, 10%, 20%, 50%, 90%, 95% Unlabeled Data)

- From Figure 9, the area under the curve for dataset 3 becomes more skewed as the percentage of unlabeled data increases, as we discussed for dataset 2. For 0% unlabeled data (fully supervised), the area under the curve is almost similar for the imbalanced and balanced datasets. For 10%, 20%, and 50% unlabeled data, the skewness is moderate, and for 90% and 95% unlabeled data, the area under the curve is random. The 90% and 95% splits are random guessing or overfitting the training data. The skewness for 90% and 95% is not extreme compared to the heart dataset, but we can not trust the model.

5.2 Statistical analysis – Friedman and Nemenyi post hoc tests

Table 14: Average Rank of the algorithms for Imbalanced and Balanced Datasets

Average Ranks	self_training	co_training	ss_ensemble	unsupervised_ labelling	intrinsic
Imbalanced dataset	8.8	10.6	5.2	8.4	10.6
Balanced dataset	7.6	11	6	8	10.8

The above table shows the average ranks of the five algorithms obtained for the analysis from the results for the three datasets.

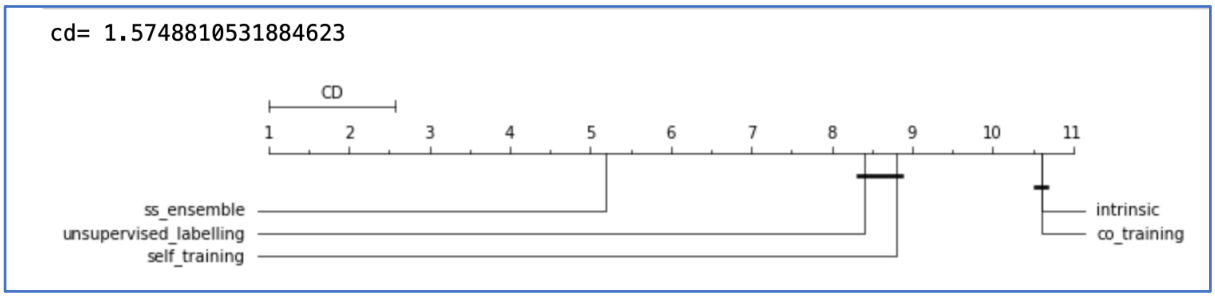


Figure 10: Nemenyi Post-Hoc for Imbalanced Dataset

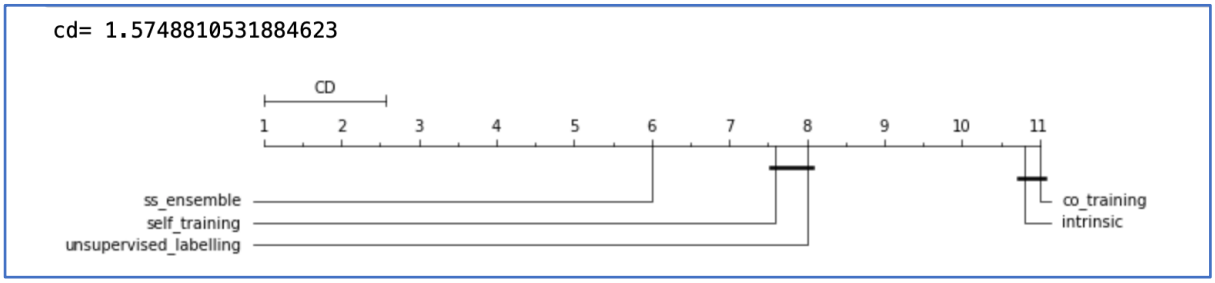


Figure 11: Nemenyi Post-Hoc for Balanced Dataset

- Fig. 10 and Fig. 11 represent the Nemenyi post-hoc analysis output on our experiments with significance level on the imbalanced/un-sampled and sampled dataset.

1. **Imbalanced/Un-sampled Datasets:** Semi-supervised algorithms performed the best. At the same time, intrinsic and co-training approaches were precisely similar in performance (this can also be noticed in Table 14. as they have the same average ranks). Unsupervised pre-labeling and Self Training performed similarly, with a slight performance improvement of the unsupervised approach over Self-Training; we discuss this further in Section 6.

2. **Balanced Datasets:** Again, the semi-supervised ensemble performed the best while self-training and unsupervised pre-labeling had similar performances. However, self-training was slightly better than the unsupervised approach. Similarly, the intrinsic approach had marginally better performance over co-training; however, they were statistically similar.

- While these observations are based on statistical analysis, we discuss the performance of different approaches over different data splits in Section 6, followed by our recommendation of the best strategy and best data split for each of the three datasets.

6. Insights & Learnings

- Apart from the statistical post-hoc analysis results, we can observe that semi-supervised ensemble was the most stable approach on the three datasets, which class imbalance and on different labeled & unlabeled data ratios. Unsupervised pre-labeling had the best performance when there were high unlabeled data samples. In other scenarios, it had similar performance to the ensemble but performed weak in certain 90% splits, which can be observed from Tables 5. and 6. Thus, we can conclude that when labeled is less unsupervised approach would be the best option – obviously relevant explanatory data analysis has to be performed to visualize possible clusters as a pre-requisite. A semi-supervised ensemble should be tried if sufficient information cannot be obtained after EDA. Also, we observe that while autoencoders work well in theory by learning input features from data and using it to make predictions, their performance is limited unless one of the two criteria is met - the first being a large number of training data so that the loss function can efficiently converge and the second being supplying information reach inputs like images. They are powerful enough to identify minute patterns/signatures in these information-rich features and enhance their learning, which is impossible with standard classifiers. Co-Training performed did not perform well in our experiments; this can be explained by the underlying two assumptions which are required to be satisfied for co-training to work well on actual data as indicated by Jun Du et Al. [8], viz. Two views should be self-sufficient to predict the output class labels, and they should be independent, i.e., they should be non-correlated. Experimenting with different combinations of feature view subsets may have resulted in an improved model. Another important observation is the performance of the manifold-based intrinsic approach, which like the unsupervised approach, worked well when available label data was low. The performance almost remained constant even when a high amount of label data was available, which showcases that only a few good samples that can help generate the manifold are sufficient for this algorithm to work. Thus, if one can identify possible clusters through EDA, this manifold-based algorithm is one of the best approaches to performing semi-supervised learning.

To conclude, the following table shows the semi-supervised approach we would use and the ideal labeled-unlabeled data ratio preferred on the three datasets:

Dataset	SSL Approach	Dataset distribution	Ideal unlabel/label split ratio	Difference with baseline (f1 score)
Online shoppers Intention	Semi supervised ensemble	Sampled	20%	+ 0.5
UCI Heart disease	Unsupervised Pre-labeling	Sampled	50%*	+ 5.16
Teenhome – Marketing campaign	Semi supervised ensemble	Sampled	50%	- 0.61

* 90% could have been selected, however considering that this dataset belongs to medical domain, we choose the stable option with 50% ratio.

7. References

1. https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset
2. https://archive.ics.uci.edu/ml/datasets/Heart+Disease
3. https://www.kaggle.com/rodsaldanha/arketing-campaign
4. Sakar, C. Okan & Polat, S. & Katircioglu, Mete & Kastro, Yomi. (2019). Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks. Neural Computing and Applications. 31. 10.1007/s00521-018-3523-0.
5. Engelen, Jesper & Hoos, Holger. (2020). A survey on semi-supervised learning. Machine Learning. 109. 10.1007/s10994-019-05855-6.
6. Oza, Nikunj. (2005). Online Bagging and Boosting. Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics. 3. 2340 - 2345 Vol. 3. 10.1109/ICSMC.2005.1571498.
7. L. Berton, A. de Andrade Lopes and D. A. Vega-Oliveros, "A Comparison of Graph Construction Methods for Semi-Supervised Learning," 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1-8, doi: 10.1109/IJCNN.2018.8489524.
8. Ling, Charles & Du, Jun & Zhou, Zhi-Hua. (1970). When does Co-training Work in Real Data?. 5476. 596-603. 10.1007/978-3-642-01307-2_58.