Introduction

The original paper related to this dataset with a full description of features are provided in https://www.hindawi.com/journals/bmri/2014/781670. The dataset contains a feature called "readmitted" which indicates if the patient is readmitted within 30 days, more than 30 days, or is not readmitted, and the main aim of this project is to predict this feature. The original paper converts this multi-class problem to a single binary problem by considering if the patient is readmitted within 30 days, or not. However, in this project we build three binary classifications through one-versus-rest approach to separately predict if readmitted<30, readmitted>30, or readmitted=No. We build different classification models, i.e. decision tree, naïve bayes, support vector machine, k-nearest neighbor, ensemble bagging, and ensemble boosting. Moreover, we perform a binary classification on feature "gender" to predict if the patient is male or female using the six mentioned classifiers. Afterward, we make some percentage of data unlabeled, and implement three semi-supervised learning approaches, i.e. label-propagation, label-spreading, and self-training to predict if the patient is readmitted within 30 days or not.

Feature Engineering

Dealing with Null Values

Totally there are 7 features that have null values. Three features, namely "weight", "medical_specialty", "payer_code" that contains very high number of null values. As the number of null values is very high, these columns are removed from the dataset. For the other features with null values, namely race, diag1, diag2, diag3, the null values are replaced with "unknown" to consider them as a new category.

Dimensionality Reduction

For the prescription features, many of them, such as "acetohexamide" and "troglitazone" have only a single value which cannot be informative. So, they are removed from the dataset. Also, some categorical features, such as "acarbose" and "nateglinide" have some categories that are very rare, so we remove those specific categories so that we can convert these features to binary features. For "gender", there are very few invalid values that their rows are removed.

Feature Transformation

The age ranges are converted to ordinal because the order seems meaningful for age. For "admission_type_id", "discharge_disposition_id", "admission_source_id", where there are many different categories, indicated by integer numbers, some values are merged. For example, for "discharge_disposition_id", all the categories for which patient has discharged to home are mapped, or for "admission_source_id", all the categories for which the patient is transferred from a medical center (of any type) are mapped. Full transformation is available in the code. Furthermore, the numeric values of diagnosis columns, i.e. diag1, diag2, diag3 are mapped to categorical based on the defined ranges in the original paper. Finally, all the remaining categorical features containing two categories are converted to binary, and the features with more than two categories are converted to numeric using one-hot encoding.

Normalization

All the features are transferred between 0 and 1 so that no feature will dominate the classification results.

Feature Selection

Feature selection is performed using PCA method in order to reduce the number of features and the size of the dataset. PCA is a very good choice for feature reduction because it can summarize the features by putting the highly correlated features to the same principal component. In fact, PCA does not remove any feature, but it summarizes them.

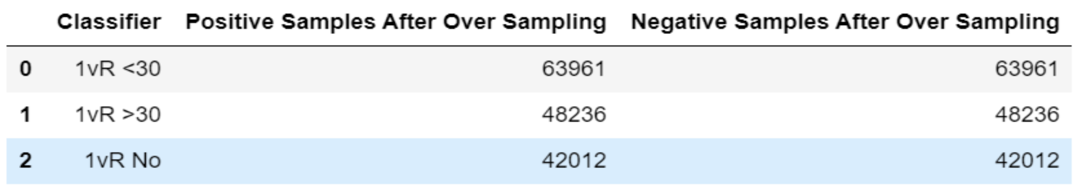
Part1 – Supervised Learning Classification to predict class “readmitted”

Converting multi class problem to three binary classification problems

As the accuracy of multi-class classification is not very good, we convert the problem into three binary ones using one-versus-rest approach so that we can separately predict: if a patient is readmitted within 30 days or not, if a patient is readmitted in more than 30 days or not, and if the patient has no readmission or not. The original paper only considers one binary classification as “<3o” or “other”.

Over Sampling

As the dataset is highly unbalanced, and the number of “<30” class is much less than the other two classes, the data is over sampled using SMOTE function. We also tried under sampling which was not useful, and the data kept unbalanced. Also, based on the results, over sampling is better than balanced sampling. Below is the number of positive and negative samples for each binary classifier after performing SMOTE over sampling.



	Classifier	Positive Samples After Over Sampling	Negative Samples After Over Sampling
0	1vR <30	63961	63961
1	1vR >30	48236	48236
2	1vR No	42012	42012

Building the classifiers

We build 6 classification models for each binary classification problem, i.e. Decision Tree, Naïve Bayes, SVM, KNN, Bagging, and AdaBoost. All the classifiers use 10-fold cross validation strategy, so that we can randomly select training and test set 10 times in order to have accurate model evaluation. The evaluation of the classifiers is done based on Accuracy, Sensitivity, Specificity, F-measure, and Run Time. The results of classification and evaluation of each binary problem are explained below.

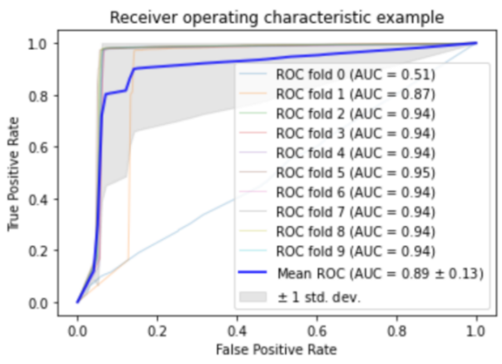
Building classifiers to predict label “<30” versus “rest”

The classifiers are built by using different hyper parameters and the best parameter are selected for the final model. For example, the DTree is built with different depth, or KNN is build based on different number of K, and the best ones are selected. The results show that SVM perform better than the other classifiers for all the evaluation measures. In terms of Mean-AUC, SVM, KNN, Bagging, and AdaBoost perform greatly with AdaBoost having slightly better result. However, in terms of run time, the SVM takes time much more than the other models.

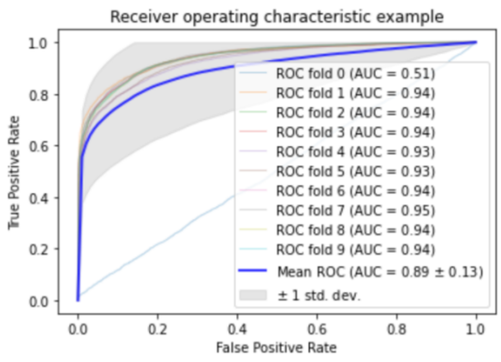
The table containing the evaluation results, and the ROC curves are given as follows.

Classifier1 ("<30" Versus Rest)	Best Depth/Best Number of Neighbors	10-fold Cross Validation Accuracy	Sensitivity	Specificity	F-measure	Time to Train
Support Vector Machine	-	0.946877	0.896781	0.996967	0.944072	4361.52
Naive Bayes	-	0.810519	0.845984	0.775050	0.817008	4.08
K-nearest Neighbor	3	0.914497	0.916433	0.912556	0.914660	3646.27
Decision Tree	16	0.905258	0.887166	0.923344	0.903509	129.99
Boosting (Decision Stumps)	-	0.915131	0.891434	0.938822	0.913068	1081.66
Bagging (Dtree)	16	0.933924	0.895749	0.972092	0.931298	1381.40

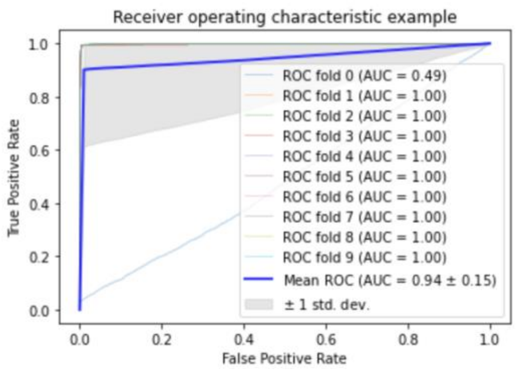
	Classifier	Best Depth	10-fold Cross Validation Accuracy
0	Decision Tree	16	0.905258



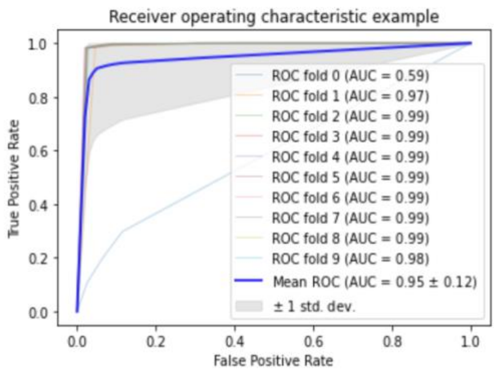
	Classifier	10-fold Cross Validation Accuracy
0	Naive Bayes	0.810519



	Classifier	10-fold Cross Validation Accuracy
0	Support Vector Machine	0.946877

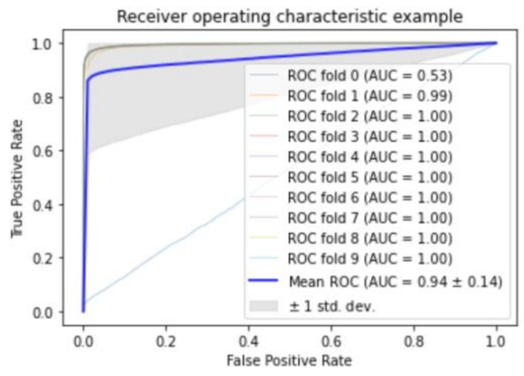


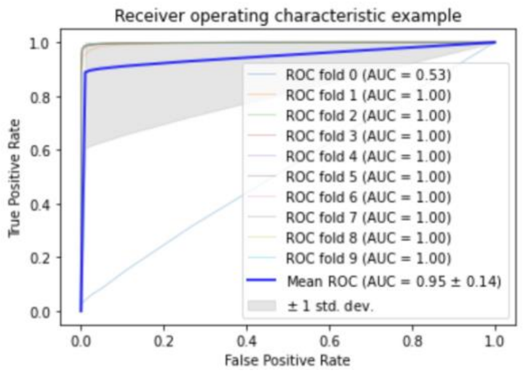
	Classifier	Best number of neighbors	10-fold cross validation accuracy
0	K-nearest neighbor	3	0.914497



	Classifier	10-fold Cross Validation Accuracy
0	Ensemble of Decision Stumps (Boosting)	0.915131

	Classifier	10-fold Cross Validation Accuracy
0	Ensemble of Decision Trees (Bagging)	0.933924





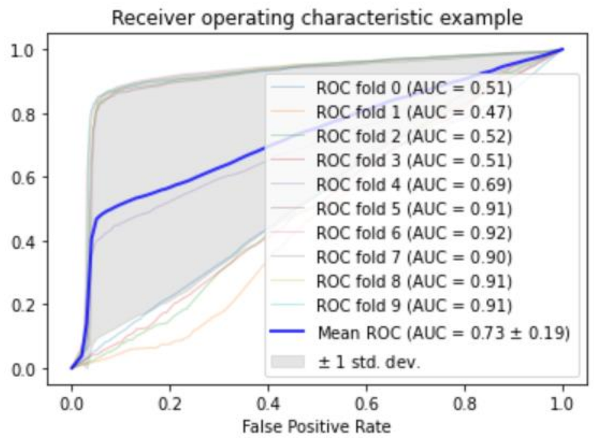
Building classifiers to predict label ">30" versus "rest"

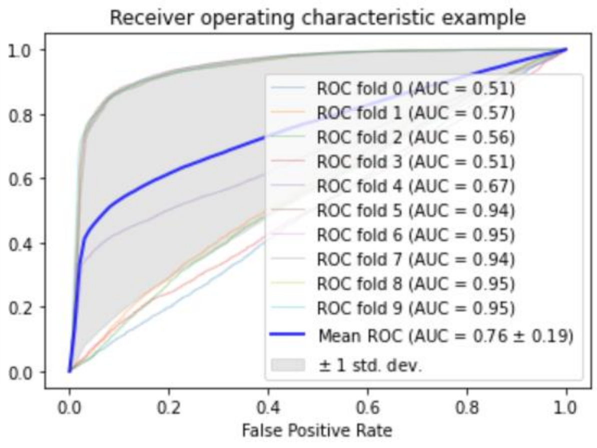
The classifiers and algorithms are exactly the same as the binary classifier "<30". The results show that performance results especially the accuracy and sensitivity is not as good as classifier "<30", but it is still acceptable with again SVM having the best results. However, again the run time of SVM is high. Specificity results are almost good meaning that we can truly predict the negative samples. Also mean-AUC is almost the same for all classifiers.

Classifier2 (">30" Versus Rest)	Best Depth/Best Number of Neighbors	10-fold Cross Validation Accuracy	Sensitivity	Specificity	F-measure	Time to Train
Support Vector Machine	-	0.758267	0.559893	0.956630	0.698442	8432.31
Naive Bayes	-	0.671680	0.696865	0.646488	0.679744	3.09
K-nearest Neighbor	5	0.682347	0.674310	0.690376	0.679771	3408.04
Decision Tree	12	0.692402	0.536197	0.848599	0.635456	68.69
Boosting (Decision Stumps)	-	0.702561	0.618459	0.786653	0.675245	899.70
Bagging (Dtree)	12	0.708967	0.587860	0.830065	0.668860	737.34

Classifier	Best Depth	10-fold Cross Validation Accuracy
Decision Tree	12	0.692402

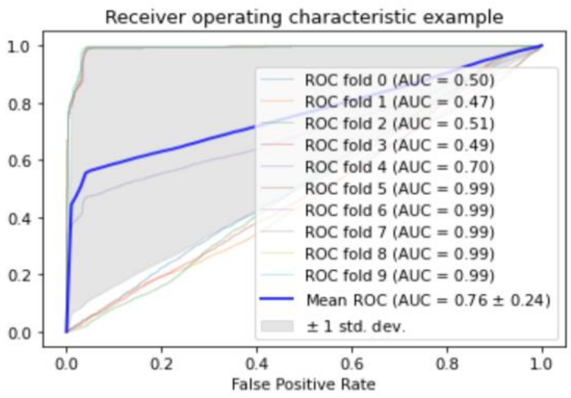
Classifier	10-fold Cross Validation Accuracy
Naive Bayes	0.67168

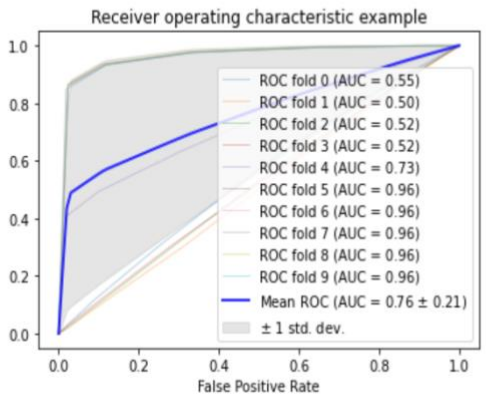




Classifier	10-fold Cross Validation Accuracy
Support Vector Machine	0.758267

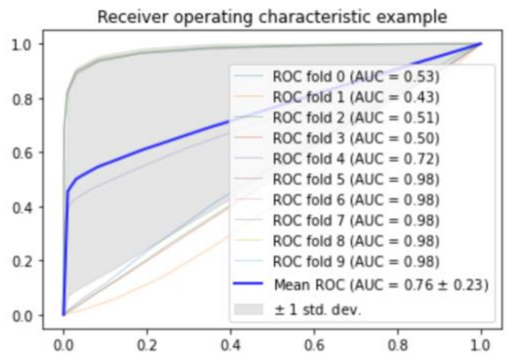
Classifier	Best number of neighbors	10-fold cross validation accuracy
K-nearest neighbor	5	0.682347

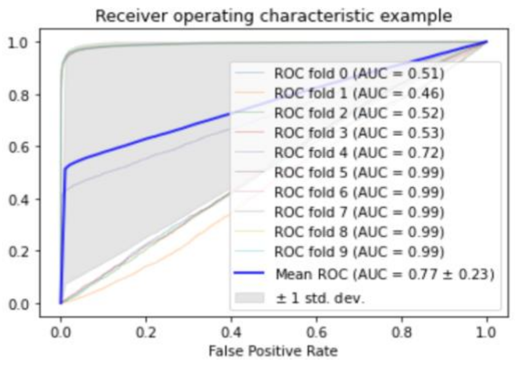




Classifier	10-fold Cross Validation Accuracy
Ensemble of Decision Trees (Bagging)	0.708967

Classifier	10-fold Cross Validation Accuracy
Ensemble of Decision Stumps (Boosting)	0.702561





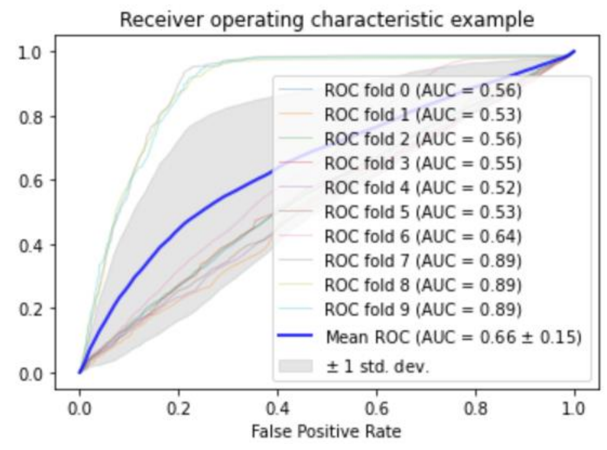
Building classifiers to predict label "No" versus "rest"

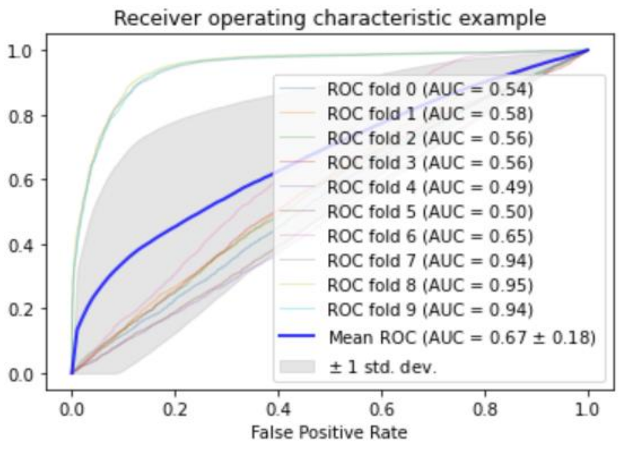
The classifiers and algorithms are exactly the same as the binary classifier "<30". The results show that performance results especially the accuracy is not that good. However, the sensitivity is acceptable meaning that we can almost predict the true samples well. SVM has very bad specificity, so SVM should not be used for predicting class "No" if detecting negative samples has importance for us.

Classifier3 ("No" Versus Rest)	Best Depth/Best Number of Neighbors	10-fold Cross Validation Accuracy	Sensitivity	Specificity	F-measure	Time to Train
Support Vector Machine	-	0.627898	0.860159	0.395625	0.698030	6584.52
Naive Bayes	-	0.603890	0.565743	0.642031	0.588179	2.85
K-nearest Neighbor	9	0.607735	0.610064	0.605398	0.608644	3403.60
Decision Tree	10	0.592132	0.703085	0.481172	0.632864	52.47
Boosting (Decision Stumps)	-	0.620708	0.697753	0.543654	0.647838	696.05
Bagging (Dtree)	10	0.597570	0.567814	0.627321	0.585226	566.72

Classifier	Best Depth	10-fold Cross Validation Accuracy
Decision Tree	10	0.592132

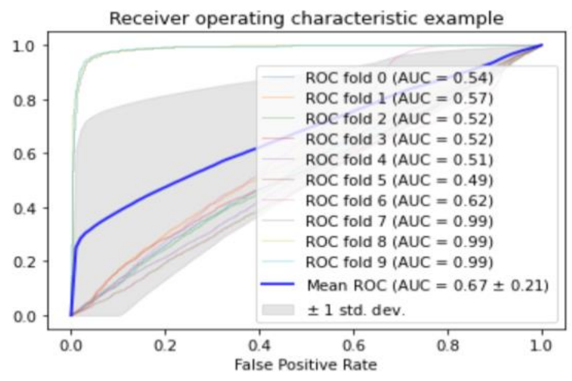
Classifier	10-fold Cross Validation Accuracy
Naive Bayes	0.60389

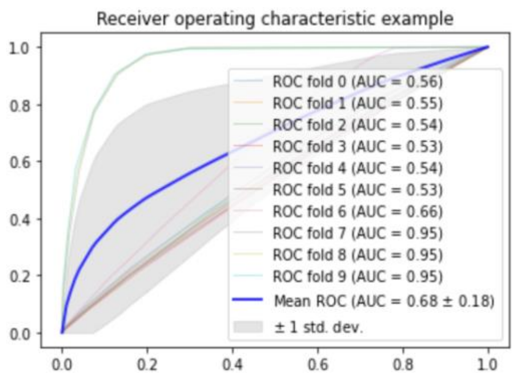




Classifier	10-fold Cross Validation Accuracy
Support Vector Machine	0.627898

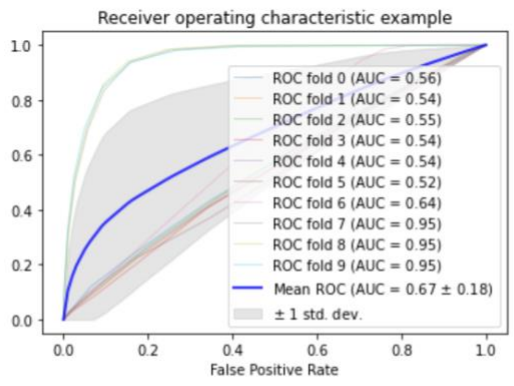
Classifier	Best number of neighbors	10-fold cross validation accuracy
K-nearest neighbor	9	0.607735

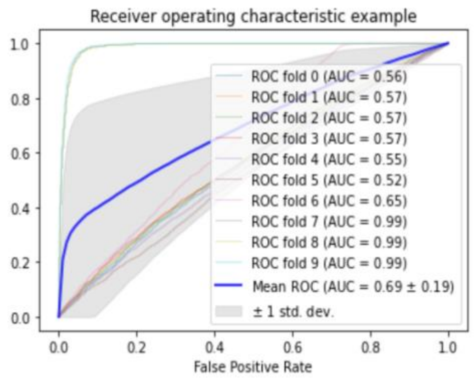




Classifier	10-fold Cross Validation Accuracy
Ensemble of Decision Trees (Bagging)	0.59757

Classifier	10-fold Cross Validation Accuracy
Ensemble of Decision Stumps (Boosting)	0.620708

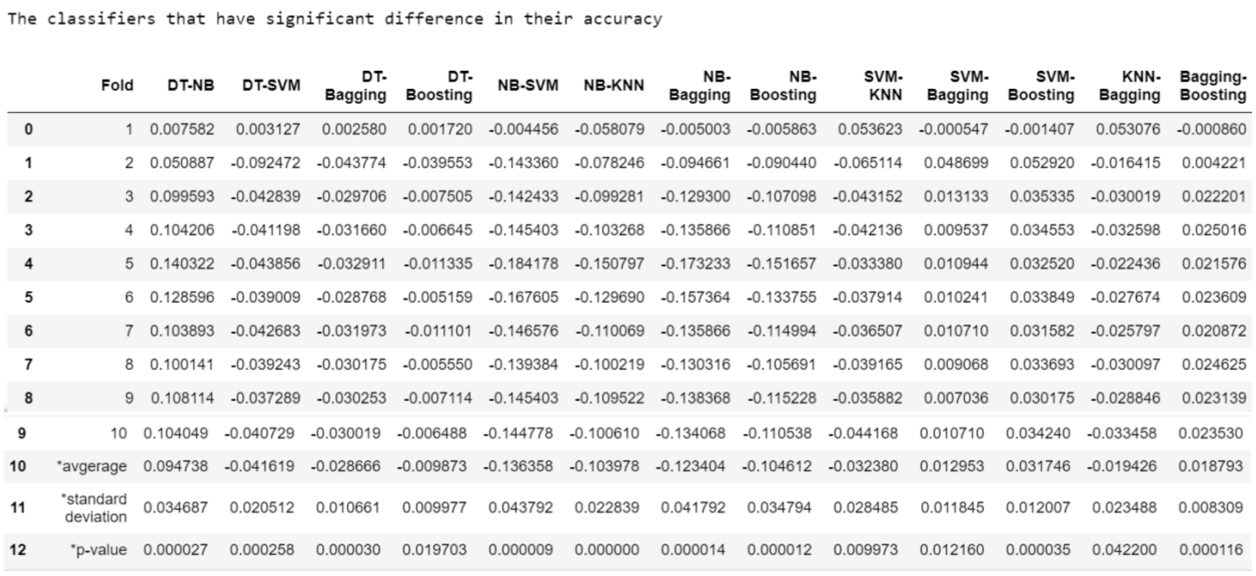




Checking the significant difference between classifiers

For comparing the classifiers with statistic results, paired t-test is done because this test is used when we want to compare different classifiers on a single dataset when we have different subset of observations from original data, which can be the observations of 10 folds. The results show that below classifiers have significant difference. A few of them performs equally such and SVM and KNN so they are not available in the table. Considering that the accuracy of SVM and KNN is also more than the other classifiers, SVM and KNN are selected as the best models.

Note: this is only performed for detecting class "<30" (the exact same setting as the original paper). The same can be applied for "<30" and "NO".



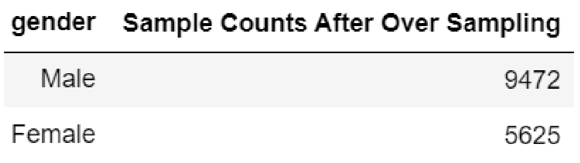
```
The classifiers that have significant difference in their accuracy
```

	Fold	DT-NB	DT-SVM	DT-Bagging	DT-Boosting	NB-SVM	NB-KNN	NB-Bagging	NB-Boosting	SVM-KNN	SVM-Bagging	SVM-Boosting	KNN-Bagging	Bagging-Boosting
0	1	0.007582	0.003127	0.002580	0.001720	-0.004456	-0.058079	-0.005003	-0.005863	0.053623	-0.000547	-0.001407	0.053076	-0.000860
1	2	0.050887	-0.092472	-0.043774	-0.039553	-0.143360	-0.078246	-0.094661	-0.090440	-0.065114	0.048699	0.052920	-0.016415	0.004221
2	3	0.099593	-0.042839	-0.029706	-0.007505	-0.142433	-0.099281	-0.129300	-0.107098	-0.043152	0.013133	0.035335	-0.030019	0.022201
3	4	0.104206	-0.041198	-0.031660	-0.006645	-0.145403	-0.103268	-0.135866	-0.110851	-0.042136	0.009537	0.034553	-0.032598	0.025016
4	5	0.140322	-0.043856	-0.032911	-0.011335	-0.184178	-0.150797	-0.173233	-0.151657	-0.033380	0.010944	0.032520	-0.022436	0.021576
5	6	0.128596	-0.039009	-0.028768	-0.005159	-0.167605	-0.129690	-0.157364	-0.133755	-0.037914	0.010241	0.033849	-0.027674	0.023609
6	7	0.103893	-0.042683	-0.031973	-0.011101	-0.146576	-0.110069	-0.135866	-0.114994	-0.036507	0.010710	0.031582	-0.025797	0.020872
7	8	0.100141	-0.039243	-0.030175	-0.005550	-0.139384	-0.100219	-0.130316	-0.105691	-0.039165	0.009068	0.033693	-0.030097	0.024625
8	9	0.108114	-0.037289	-0.030253	-0.007114	-0.145403	-0.109522	-0.138368	-0.115228	-0.035882	0.007036	0.030175	-0.028846	0.023139
9	10	0.104049	-0.040729	-0.030019	-0.006488	-0.144778	-0.100610	-0.134068	-0.110538	-0.044168	0.010710	0.034240	-0.033458	0.023530
10	*avgerage	0.094738	-0.041619	-0.028666	-0.009873	-0.136358	-0.103978	-0.123404	-0.104612	-0.032380	0.012953	0.031746	-0.019426	0.018793
11	*standard deviation	0.034687	0.020512	0.010661	0.009977	0.043792	0.022839	0.041792	0.034794	0.028485	0.011845	0.012007	0.023488	0.008309
12	*p-value	0.000027	0.000258	0.000030	0.019703	0.000009	0.000000	0.000014	0.000012	0.009973	0.012160	0.000035	0.042200	0.000116

Part2 – Supervised Learning Classification to predict class "gender"

Building Classifiers

Here we have a binary classification to predict the gender, and we build the exact same classifiers as task 1, i.e. Decision Tree, Naïve Bayes, SVM, KNN, Bagging, and AdaBoost. As we again have an unbalanced dataset with more males than females, we used over sampling and balanced sampling, where the results of balanced sampling with SMOTEEN function was better. Below is the number of males and females after performing balanced sampling.



gender	Sample Counts After Over Sampling
Male	9472
Female	5625

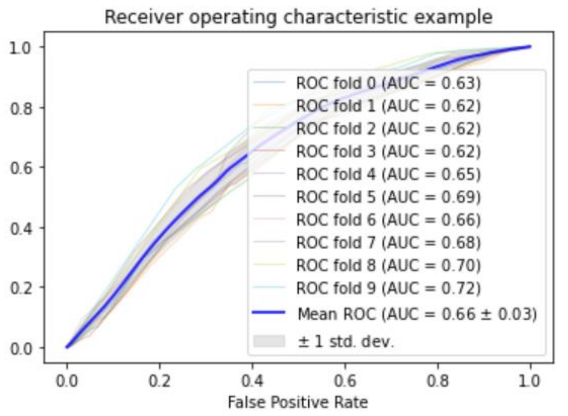
Performance Results

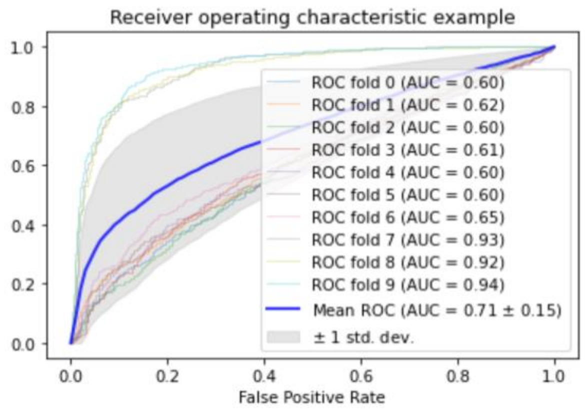
Below are the performance results of all classifiers in terms of accuracy, sensitivity, specificity, F-measure, Run Time. We also demonstrate the ROC Curves of all folds as well as their average AUC. As depicted in the below table, the classifiers perform not very good and their accuracy is below 70%. However, in terms of sensitivity, the results are good and SVM has a very good sensitivity more that 83%. Specificity is almost low for all classifiers, and F-measure is almost good. Also, as we used balanced sampling and the number of instances reduced, the Run Time is small for all the classifiers. Overall, SVM has the best performance among all. Mean AUC is in the same range for all classifiers, but SVM again has better results.

Classifier1 ("<30" Versus Rest)	Best Depth/Best Number of Neighbors	10-fold Cross Validation Accuracy	Sensitivity	Specificity	F-measure	Time to Train
Support Vector Machine	-	0.688429	0.832137	0.446400	0.770178	169.07
Naive Bayes	-	0.646849	0.712627	0.536000	0.716865	0.94
K-nearest Neighbor	9	0.685979	0.851774	0.406756	0.772908	124.36
Decision Tree	4	0.667157	0.842694	0.371556	0.760589	4.80
Boosting (Decision Stumps)	-	0.679892	0.780405	0.510578	0.753632	126.70
Bagging (Dtree)	4	0.669100	0.715372	0.591111	0.730645	82.43

Classifier	Best Depth	10-fold Cross Validation Accuracy
Decision Tree	4	0.667157

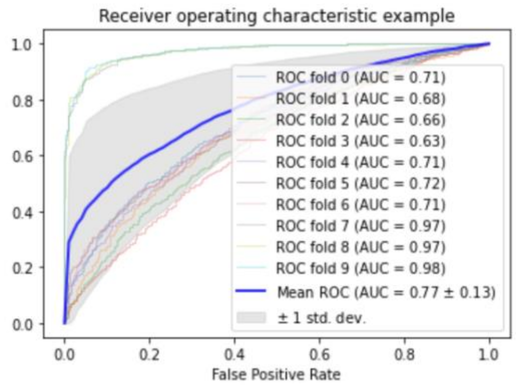
Classifier	10-fold Cross Validation Accuracy
Naive Bayes	0.646849

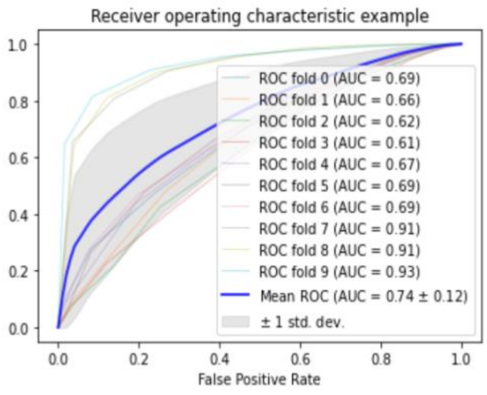




Classifier	10-fold Cross Validation Accuracy
Support Vector Machine	0.688429

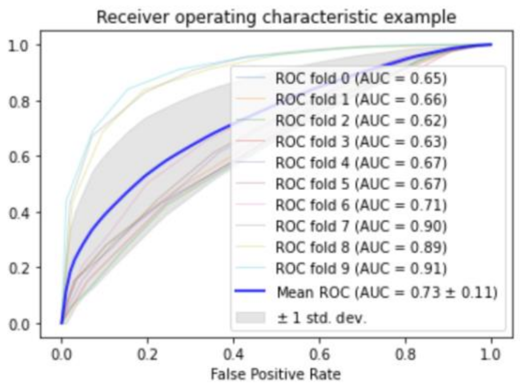
Classifier	Best number of neighbors	10-fold cross validation accuracy
K-nearest neighbor	9	0.685979

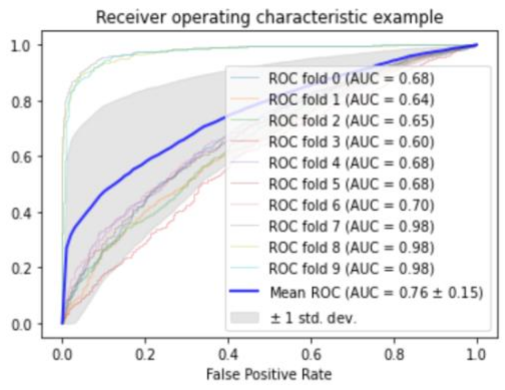




	Classifier	10-fold Cross Validation Accuracy
	Ensemble of Decision Trees (Bagging)	0.6691

	Classifier	10-fold Cross Validation Accuracy
	Ensemble of Decision Stumps (Boosting)	0.679892





Checking the significant difference between classifiers

Again paired t-test is permored for comparing all classifiers statistically. The results of all comparisions and the P values are available in the code. Below table shows the classifiers that have significant difference wih each other considering the accuracy results of 10 folds. As the table shows, for predicting class "gender", Naïve Bayes has significant difference with both Bagging and Adaboost. However, the other classifiers perform equally and the difference between their accuacies are by chance.

```
The classifiers that have significant difference in their accuracy
```

	Fold	NB-Bagging	NB-Boosting
0	1	-0.004636	-0.074834
1	2	-0.026490	-0.032450
2	3	-0.002649	-0.046358
3	4	-0.006623	-0.007285
4	5	-0.060265	-0.080132
5	6	-0.066225	-0.082119
6	7	-0.053642	-0.045033
7	8	-0.007952	0.013917
8	9	0.009278	0.009940
9	10	-0.003313	0.013917
10	*avgerage	-0.022252	-0.033044
11	*standard deviation	0.025006	0.035182
12	*p-value	0.031432	0.024936

Part3 – Semi-Supervised Learning Classification

Building Classifiers

In this part, we use semi-supervised learning in order to predict class "readmitted" as we did in the first part. As we had three binary classification problems in task1, we choose one of the binary classifiers, i.e. "<30" versus the rest, the exact same setting that is used in original paper, in order to continue this part and build semi-supervised models.

Three different semi-supervised approaches are used in this section, namely label-propagation, label-spreading, and self-training. Label-propagation and label-spreading have available libraries in python that we use. However, self-training approach is implemented manually. Label-propagation and label-spreading work by manifold detection in data, both labeled and unlabeled instances, with the use of similarity between the samples. In other words, they try to find the low-dimensional regions in data that belong to the same class. Moreover, label-propagation and label-spreading need a kernel that should be selected. We select KNN kernel, as it works with sparse graph and the run time is much less than RBF kernel. As the number of instances is high and running these algorithms takes time, we perform a random sampling on data and see the results. Surprisingly, when selecting 20000 samples and perform SMOTE over sampling, the classifiers perform almost the same as the original models. So, below number of positive and negative samples are used for this part.

Positive Samples After Over Sampling	Negative Samples After Over Sampling
18173	18173

As the third SSL approach, we use self-training method which works based on pseudo labels. This algorithm starts with building a model with a portion of labeled data. Then, the built model is used to predict the pseudo labels of unlabeled data. Afterwards, the pseudo labeled data that have the confidence more than a threshold are selected and fed into the model in the next round. So, at each round a combination of labeled data, and newly pseudo labeled data are combined and fed into the model. In this task we naïve bayes classifier as the base algorithm of self-training, as its accuracy was less than the other models in task1 and we want to see if semi-supervised learning can improve its accuracy. we perform related steps, namely building a classifier, using it to label data, and fed unlabeled data into model, iteratively for 10 times. We can also repeat until we reach a specific accuracy.

Moreover, semi-supervised learning is performed with different percentage of unlabeled data, i.e. 0% (fully labeled), 10%, 20%, 50%, 90%, and 95%.

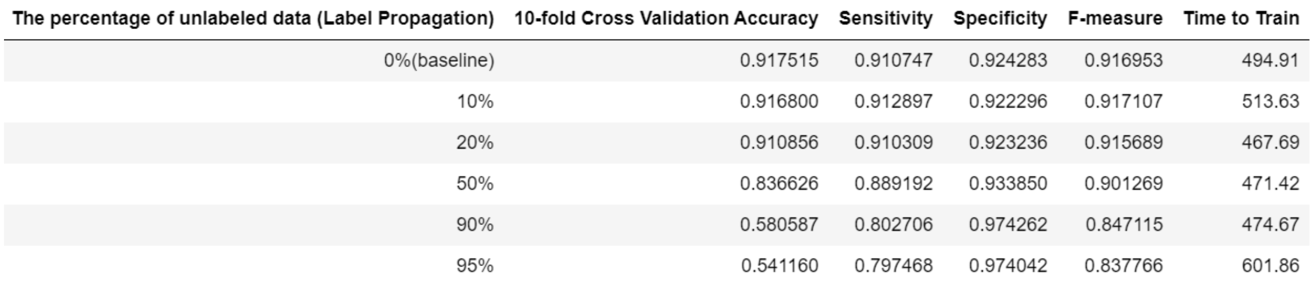
A random generator that creates random numbers based on normal distribution is used in all three models to randomly generate the unlabeled data from the fully labeled dataset.

Analyzing the Results

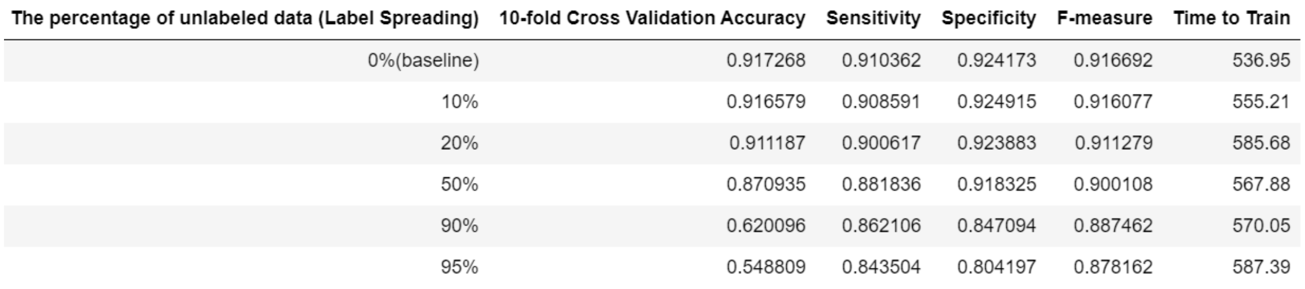
Below tables show the results of label-propagation, label-spreading, and self-training for different percentage of unlabeled data. As the results show, in label-propagation and label-spreading, the accuracy results of 10%, 20%, and 50% unlabeled data is very close to the baseline. However, as the number of unlabeled data increase to 90% and 95%, the accuracy results decrease significantly. Also, label-spreading perform slightly better than label-propagation when the number of unlabeled data increases.

For self-training, as the third table shows, as the number of unlabeled data increases, the accuracy and all other measures increase significantly, where we reach around 95% accuracy by using 95% unlabeled data. Also, as we use only a portion of data, the run time decrease significantly compared to task 1.

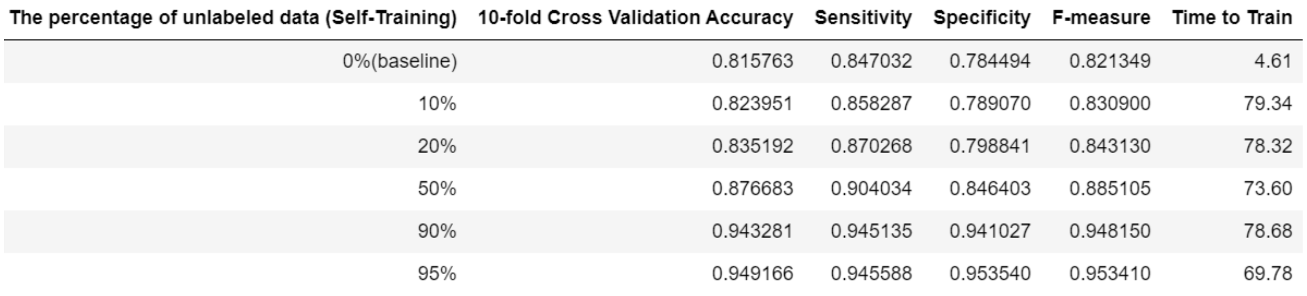
So, self-training is selected as the best SSL approach for this task.



The percentage of unlabeled data (Label Propagation)	10-fold Cross Validation Accuracy	Sensitivity	Specificity	F-measure	Time to Train
0%(baseline)	0.917515	0.910747	0.924283	0.916953	494.91
10%	0.916800	0.912897	0.922296	0.917107	513.63
20%	0.910856	0.910309	0.923236	0.915689	467.69
50%	0.836626	0.889192	0.933850	0.901269	471.42
90%	0.580587	0.802706	0.974262	0.847115	474.67
95%	0.541160	0.797468	0.974042	0.837766	601.86

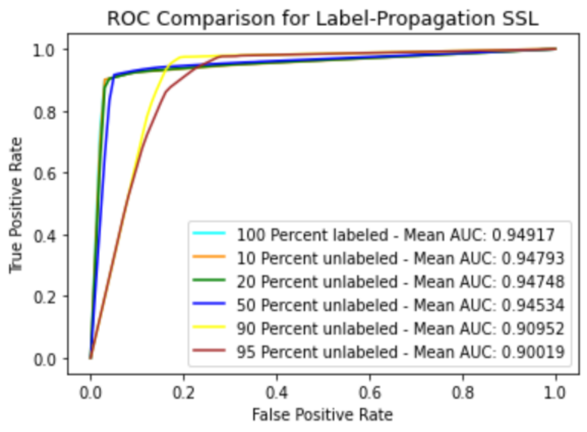


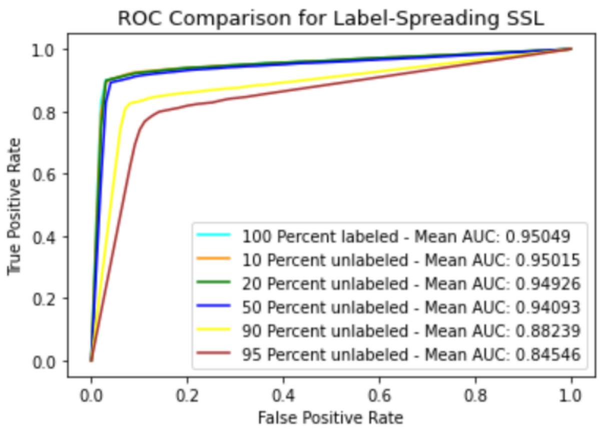
The percentage of unlabeled data (Label Spreading)	10-fold Cross Validation Accuracy	Sensitivity	Specificity	F-measure	Time to Train
0%(baseline)	0.917268	0.910362	0.924173	0.916692	536.95
10%	0.916579	0.908591	0.924915	0.916077	555.21
20%	0.911187	0.900617	0.923883	0.911279	585.68
50%	0.870935	0.881836	0.918325	0.900108	567.88
90%	0.620096	0.862106	0.847094	0.887462	570.05
95%	0.548809	0.843504	0.804197	0.878162	587.39

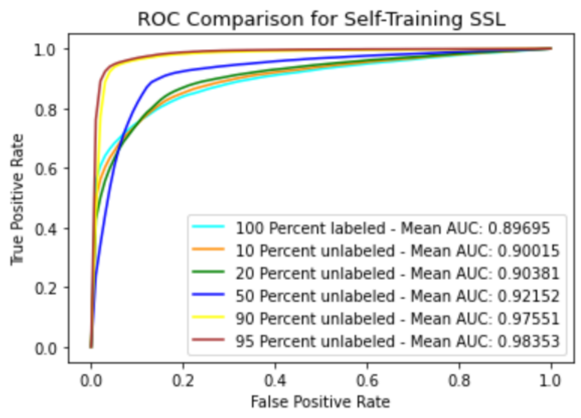


The percentage of unlabeled data (Self-Training)	10-fold Cross Validation Accuracy	Sensitivity	Specificity	F-measure	Time to Train
0%(baseline)	0.815763	0.847032	0.784494	0.821349	4.61
10%	0.823951	0.858287	0.789070	0.830900	79.34
20%	0.835192	0.870268	0.798841	0.843130	78.32
50%	0.876683	0.904034	0.846403	0.885105	73.60
90%	0.943281	0.945135	0.941027	0.948150	78.68
95%	0.949166	0.945588	0.953540	0.953410	69.78

Below graphs depict the ROC curves of different SSL methods when using different percentage of unlabeled data. As the graphs show, the best ROC (the highest mean AUC) is for self-training, in the third figure, when we use 95% unlabeled data.







Checking the significant difference between classifiers

For comparing the performance of SSL classifiers statistically, when we have one dataset and 10 folds, we again use paired t-test. In order to have reasonable comparison between the models, we first compare the three models when they use fully labeled data, then compare the models when they use 10% unlabeled data, etc. The comparison tables of differences with corresponding P values are all available in code. In order to keep this report short, we only provide two comparison tables which correspond to 10% and 95% unlabeled data. As the results show, there is a significant difference between self-training and labeled-propagation, and also between self-training and label-spreading, so self-training method performs really the best among the three. However, there is no significant difference between label-propagation and label-spreading.

Fold	LabelPropagation(10% unlabeled)-SelfTraining(10% unlabeled)	LabelSpreading(10% unlabeled)-SelfTraining(10% unlabeled)
1	0.0940	0.0984
2	0.0920	0.0931
3	0.0974	0.0991
4	0.0952	0.0996
5	0.0867	0.0850
6	0.0872	0.0894
7	0.0932	0.0885
8	0.0958	0.0878
9	0.0944	0.0930
10	0.0927	0.0924
*avgerage	0.0928	0.0926
*standard deviation	0.0032	0.0046
*p-value	0.0000	0.0000

Fold	LabelPropagation(95% unlabeled)-SelfTraining(95% unlabeled)	LabelSpreading(95% unlabeled)-SelfTraining(95% unlabeled)
1	-0.4092	-0.4059
2	-0.4088	-0.3959
3	-0.4054	-0.4013
4	-0.4199	-0.4144
5	-0.4079	-0.3955
6	-0.4089	-0.3913
7	-0.4043	-0.3908
8	-0.4088	-0.3868
9	-0.4032	-0.3969
20	-0.4035	-0.4247
*avgerage	-0.4080	-0.4004
*standard deviation	0.0044	0.0106
*p-value	0.0000	0.0000

Conclusion and Future Works

In this project we used several classification tasks, supervised and semi-supervised on a dataset containing the patient information of more than 100,000 patients. For the supervised tasks, we built six classifiers to predict class "readmitted" and class "gender". Regarding "readmitted" class, the classifiers can perfectly predict the class "<30", especially SVM reach the accuracy more than 94%. Also, as in this project we have medical domain, the sensitivity and specificity are very important, and the built classifiers have very good results in terms of both sensitivity and specificity. Regarding the specificity, SVM has the performance of more than 96% which shows very high rate of true negative samples. Regarding specificity, the bagging method has performed very well with more than 97%. So, depending on whether we need higher accuracy, higher sensitivity, or higher specificity, one of these models can be selected.

Regarding the semi-supervised learning approach, what we found very interesting is that using self-training method can potentially increase the performance of the classifiers, as shown in the results.

Increasing the number of unlabeled data from 10% to 95% gradually increased the accuracy and using 95% unlabeled data gave us the high accuracy of more than 95%. Similarly, increasing the percentage of unlabeled data increased the sensitivity, specificity, and F-measure which was the most interesting part of this project.

Another important thing we realized is that this dataset seems having a normal distribution, because in semi-supervised part, when we used a small subset of samples (20,000 before over sampling), the accuracy results was very close to the original dataset.

For the future work, one direction is to try other semi-supervised algorithms such as co-training to see how the combination of two classifiers to feed pseudo labels o each other can improve the accuracy of the classification. Another direction could be to improve the results of class "<30" and class "NO", since they don't have as good results as class ">30". Finally, as one of the most important parts of any machine learning algorithm, feature engineering is another possible direction for future studies about this dataset. As saw in this project, this dataset is a huge dataset with so many features and rows. So, feature engineering is of great important to increase the performance, specifically reducing the training time, as its high for SVM and KNN. Feature engineering might also result in better performance to predict class ">30" and class "No".