Airline Passenger Satisfaction

EE 660 Course Project

Project Type:

(1) Design a system based on real-world data

Number of student authors: 1

Mingjian Shi

mingjian@usc.edu

12/3/2021

1. Abstract

This project explores the dataset from a survey that has almost 130,000 responses towards the satisfaction for a certain airline company with 25 factors of influence, and the goal of this project can be used by the airline company to predict the passenger's satisfaction by different factors.

There are two different schemes in this project:

(1) explore supervised learning models;

(2) explore unsupervised learning models.

In the research of SL, I explored 6 different SL models including Logistic Regression; SVC; K-Neighbors Classifier; Decision Tree Classifier; AdaBoost Classifier; and Random Forest Classifier. After exploring all the 6 different models, I conclude the best model is Random Forest Classifier, which gives the accuracy of 90.01%.

In the research of SSL, I explored 3 different SSL models including Label Propagation, Label Spreading and Self Training. Also, I used the predicted labels from the above three models to fit on a supervised learning model, then verified that the combined performance is better than just use SL to fit on the labeled data alone. The best SSL model from my investigation is Self-Training classifier, which gives the accuracy of 87.82%.

2. Introduction

2.1. Problem Type, Statement and Goals

The project deal with a real-world dataset form a certain airline that made by almost 130,000 people with their final satisfactory of the airline. It intends to help airline to determine the passengers' satisfaction based on different features. I will treat this as a binary classification problem by creating a model that predicts whether a customer was satisfied or unsatisfied with the services(features) that airline provided. This problem is interesting and important to the airline company due to its nature: Customer Satisfaction, and it is not a trivial problem to solve due to the volume and dimensionality of the dataset. One needs to apply many Machine Learning and data science techniques to solve this problem such as feature engineering, data cleaning, and model training and selection. Besides, this project extends the research to Semi-supervised Learning techniques, which will be explained in more details under Section 5.

Example sources of difficulty (nontriviality) include:

(1) **High dimensionality of feature space:**
It is not easy to deal with the real-world data since the output is influenced by many reasons(features). In our dataset, we have originally 25 different features that contribute to a single output. It is always ideal to train the model with all the features, but it is not practical to do it due to the curse of dimensionality. And that is why we need to do some work to deal with the high dimensionality feature space.

(2) **Categorical features**
A constant topic in Machine Learning is to deal with categorical features, and those features are non-numerical and often kept with different levels from different observation stand point. In our dataset, it has 5 different categorical features need to some proper treatment.

(3) **Missing data**
It is always a pain if we encounter missing data when apply machine learning techniques to build the training model. Sometimes we can ignore the missing data and drop them based on situation. However, we cannot always treat missing data without any other consideration. This dataset does have some missing data, and therefore, I need to apply some techniques to deal with them.

(4) **Outliers**

The data points or observation that lies a far distance from other data points will affect the performance of our model. So, it is important to evaluate the way of treating them to benefit our model

(5) Nonlinear behaviors.

There are 25 different features contribute a single output, which means the nonlinear behaviors exist. I will try to explore different model to see which one performs the best.

2.2. Our Prior and Related Work - None

2.3. Overview of Our Approach

In Sections 2.4.1-2.4.2, give an overview of the models and algorithms you used, how they were compared, performance metrics used, and any other key aspects of your project you would like to highlight. Do this in each subsection below, including the main topic and the extension. If your project only has TL/SSL, then one subsection for TL/SSL would be good. (Note that detailed descriptions will be given below in Secs. 3-6.) Overview comments that apply to both the main topic and the extension topic, can be mentioned here.

Baseline systems should also be stated in 2.4.1 and 2.4.2: For type-1 project, there must be two baselines for the main topic, one trivial and one non-trivial. For SSL or TL, there should be at least one baseline; a result from SL that can be compared with the TL/SSL result(s) could serve as the baseline.

2.4.1 Main topic (Supervised Learning)

The baseline systems are

- The trivial baseline: Randomly generate the output label
- The non-trivial baseline: Logistic Regression

2.4.2 Extension (Semi-Supervised Learning)

The baseline system is Logistic Regression fit only on labeled data, which used to compare the results with 3 different semi-supervised learning models to see how well the SSL model is.

3. Implementation

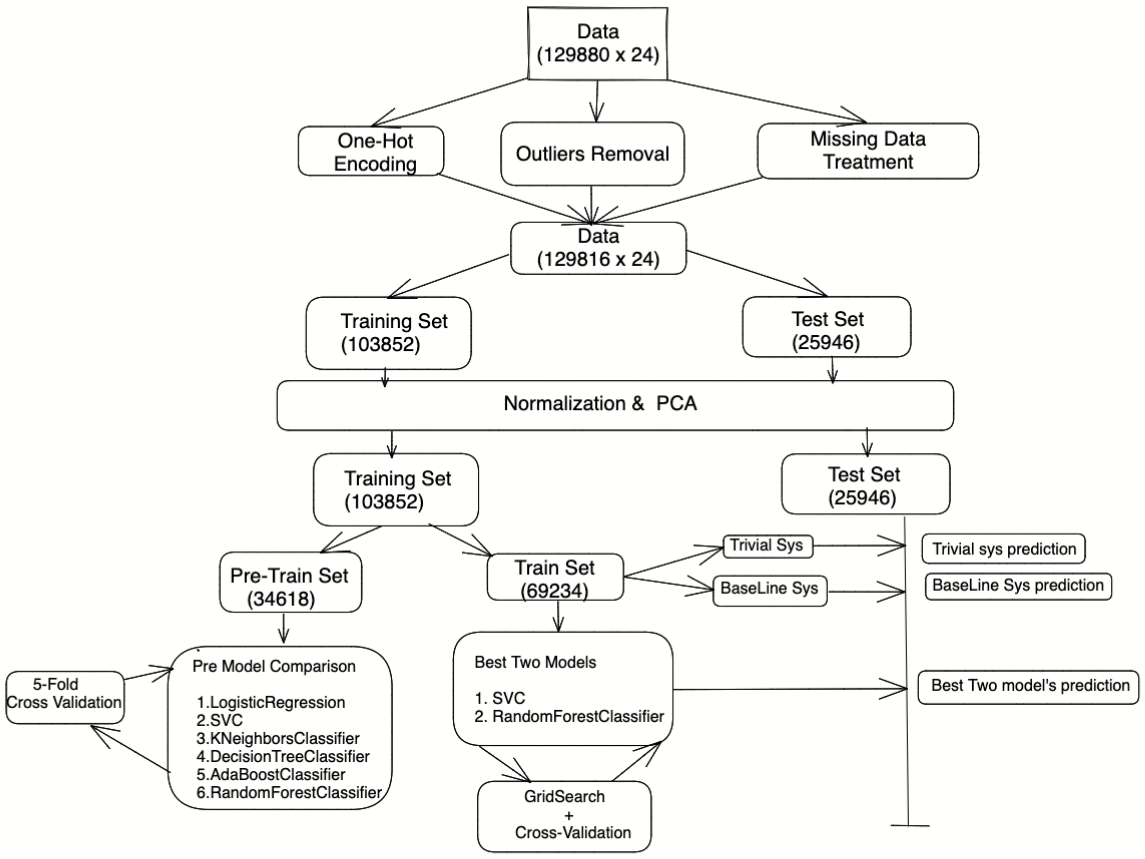
3.1. Data Set

This dataset contains 129880 entries and 24 columns(features). Including 19 numerical features, 5 categorical features and 1 categorical output

Feature name	Type	Description
Unnamed	NUM	Count
Gender	CAT	Male, Female
customer_type	CAT	Loyal, Dis-Loyal
age	NUM	Age
type_of_travel	CAT	Personal, Business
customer_class	CAT	Eco, Eco-plus, Business
flight_distance	NUM	Flight distance
inflight_wifi_service	NUM	Ratings 1-5
departure_arrival_time_convenient	NUM	Ratings 1-5
ease_of_online_booking	NUM	Ratings 1-5
gate_location	NUM	Ratings 1-5
food_and_drink	NUM	Ratings 1-5
online_boarding	NUM	Ratings 1-5
seat_comfort	NUM	Ratings 1-5
inflight_entertainment	NUM	Ratings 1-5
onboard_service	NUM	Ratings 1-5
leg_room_service	NUM	Ratings 1-5
baggage_handling	NUM	Ratings 1-5
checkin_service	NUM	Ratings 1-5
inflight_service	NUM	Ratings 1-5
cleanliness	NUM	Ratings 1-5
departure_delay_in_minutes	NUM	Delays in min
arrival_delay_in_minutes	NUM	Delays in min
satisfaction	CAT	OUTPUT(TARGET)

3.2. Dataset Methodology

The dataset has 129880 entries. And the use of the dataset is shown below:



At first the dataset was divided into training set and test set with a ratio of 8:2, which means training set has 103852 entries and test set has 25964 entries. Then the test set was set aside until the testing phase. The training set then be divided into pre-training set and training set with a ratio of 1:3, which means pre-training set has 34618 entries and training set has 69234 entries. 5-Fold cross validation were used during (1) pre-training when I come up with 6 different models to find the best two out of them; (2) training when tuning the hyperparameters of the best two models by using GridSearchCV method. The test set was only used in the following phase: (1) trivial system prediction; (2) Best two models' prediction; (3) SSL baseline system prediction; (4) SSL best models' prediction

3.3. Preprocessing, Feature Extraction, Dimensionality Adjustment

3.2.1 Categorical Features

The treatment for categorical features is One-hot Encoding.

There are Five different categorical features:

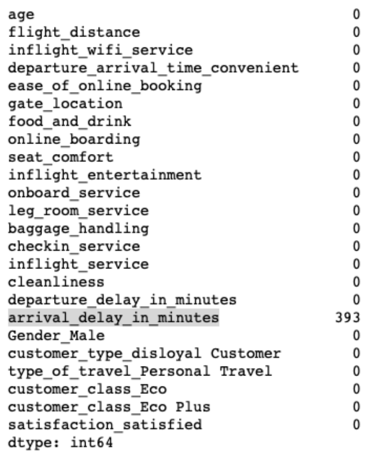
1. "Gender": Male & Female
2. "Customer Type": Loyal Customer & Disloyal Customer
3. "Type of travel": Personal Travel & Business
4. "Customer Class": Eco, Eco-plus, Business
5. "Satisfaction": Satisfied & unsatisfied

The following chart summarizes the changes after one-hot encoding:

Original Categorical Features	After One-hot Encoding
Gender [Male, Female]	gender_Male [0, 1]
Customer Type [Loyal, Disloyal]	customer_type_disloyal Customer[0,1]
Customer Class [Eco, Eco-plus, Business]	customer_class_Eco [0, 1]
	customer_class_Eco Plus [0, 1]
Type of travel[Personal, Business]	type_of_travel_Personal Travel [0, 1]
Satisfaction[satisfied, dissatisfied]	satisfaction_satisfied [0, 1]

3.2.2 Missing Data Treatment

By looking at the following summary, I found there are 393 missing data points in the feature "arrival_delay_in_minutes".



```
age                                  0
flight_distance                      0
inflight_wifi_service                0
departure_arrival_time_convenient    0
ease_of_online_booking               0
gate_location                        0
food_and_drink                       0
online_boarding                      0
seat_comfort                         0
inflight_entertainment               0
onboard_service                      0
leg_room_service                     0
baggage_handling                     0
checkin_service                      0
inflight_service                     0
cleanliness                          0
departure_delay_in_minutes           0
arrival_delay_in_minutes           393
Gender_Male                          0
customer_type_disloyal Customer      0
type_of_travel_Personal Travel       0
customer_class_Eco                   0
customer_class_Eco Plus              0
satisfaction_satisfied               0
dtype: int64
```

Chart 3.2.2 Missing data summary

I tried two different method, and used the second for this project:

(1) Direct removal (since the amount of missing data is not too large)
(2) Data filling (Imputation)

I finally used the second method and found it very interesting and helpful for real-world data.

The method of data filling is ideal because I found out there is another feature called “departure_delay_in_minutes” has a 0.97 correlation factors with the missing data feature “arrival_delay_in_minutes”.

The correlation matrix is showing below:

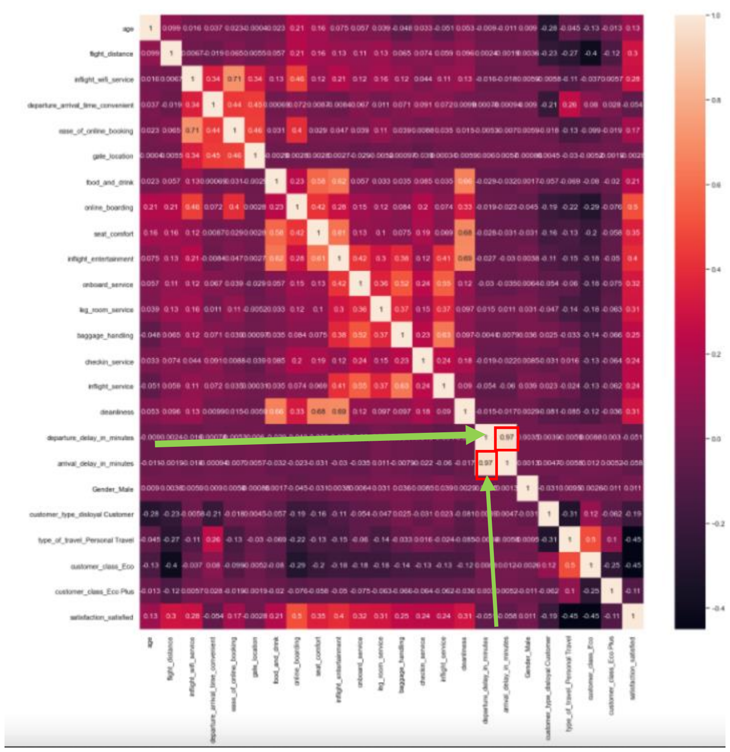


Table 3.2.2 Correlation Matrix

By looking at the feature names, we can also propose the “arrival delay” is related with “departure delay”, which is reasonable. So, I choose to use data filling that utilize the data from “departure delay” to fill in the missing data.

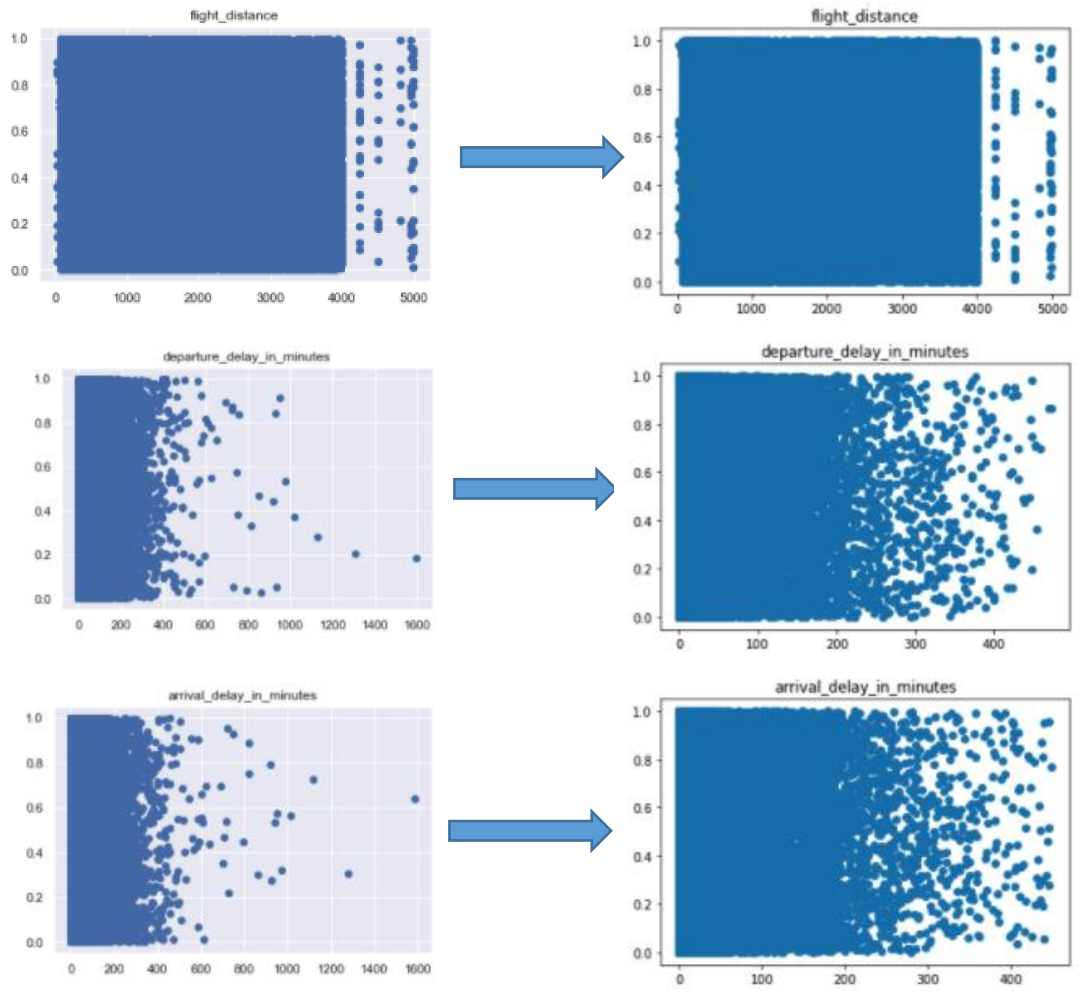
3.3.3 Outliers Removal

Most of the features value ranges from 0 to 5, except 3 features: (1)“flight_distance”, (2)“departure_delay_in_minutes” and (3)“arrival_delay_in_minutes”.

For (1) “flight_distance”: values over 5000 miles will be treated as outliers

For (2)“departure_delay_in_minutes” and (3)“arrival_delay_in_minutes”: values over 450min will be treated as outliers

The following chart is a visual help for us to see the before and after removing outliers:



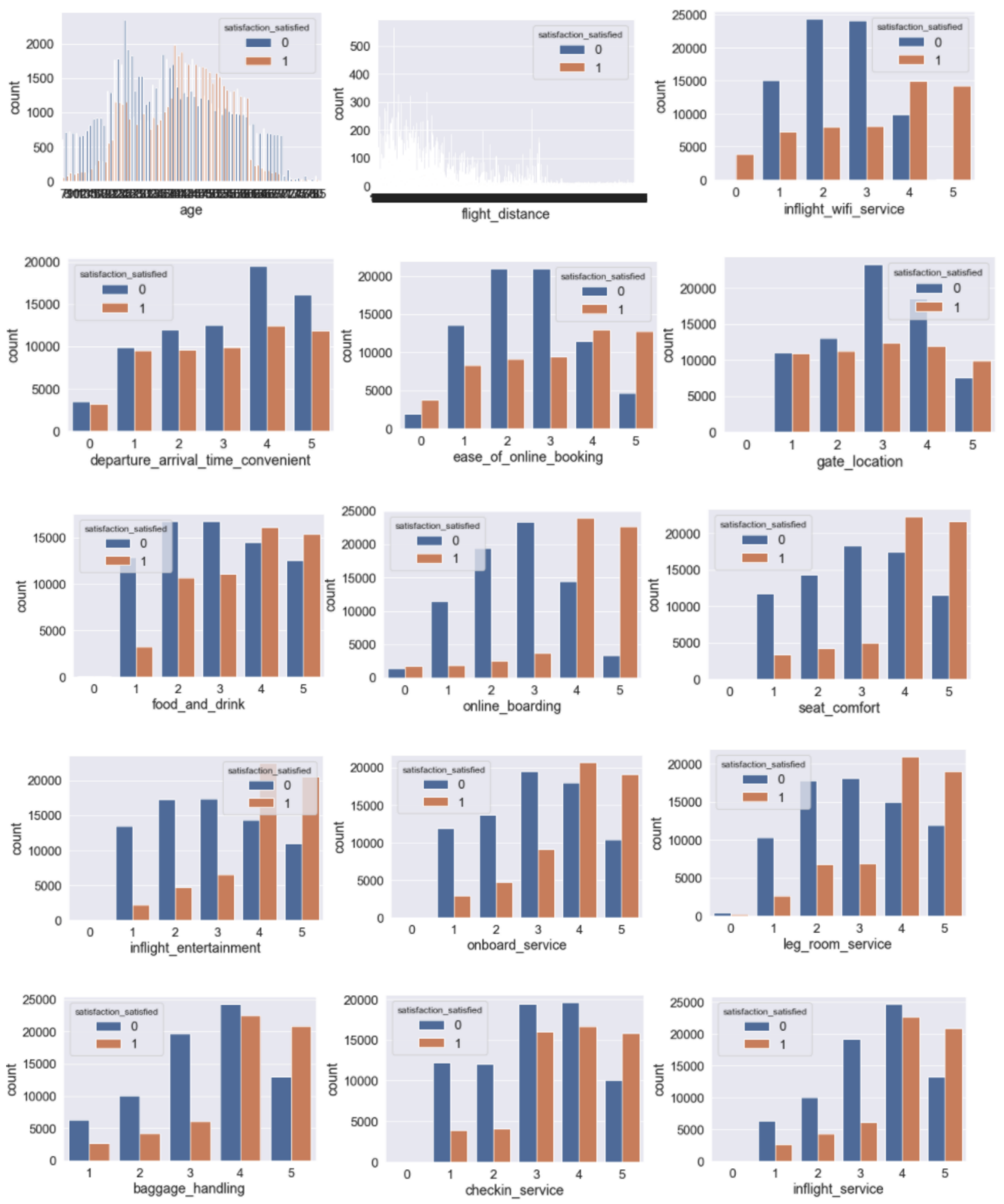
Before Removing Outliers After Removing Outliers

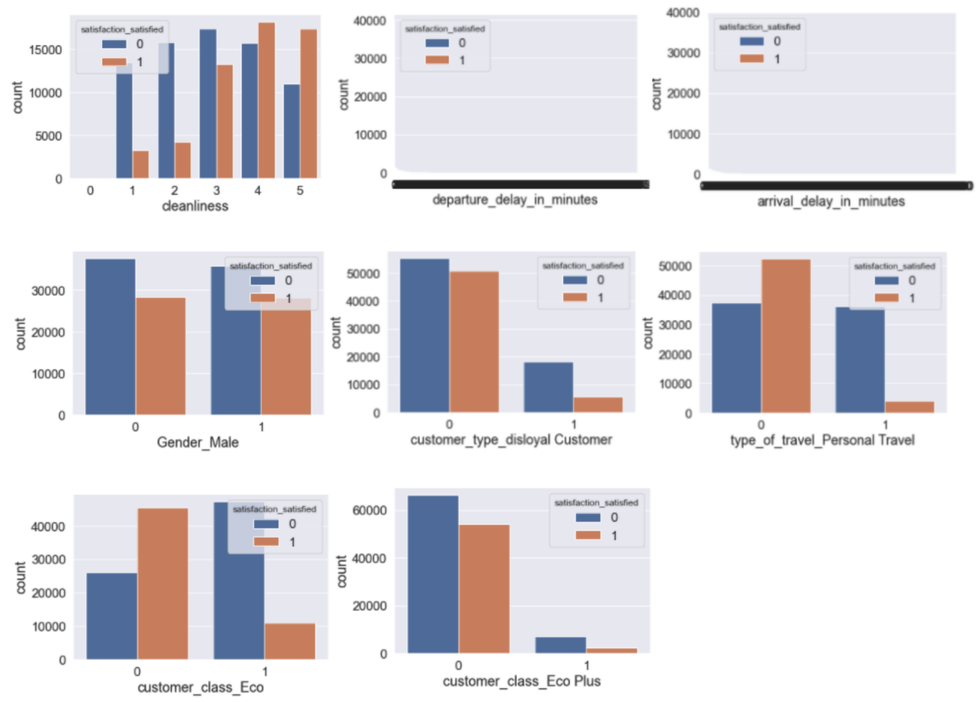
3.3.4 Normalize the Data

It is important to normalize our data before the training process. In order to normalize the data, at first, I split the data into training and testing. Then I use StandardScaler() to fit transform the training data for normalizing, and use the same scaler to fit transform the test data.

3.3.4 Feature dimensionality adjustment PCA

First, I created the correlation matrix for all the features that shown in Table 3.3.2. Then I analyzed all the features individually with respect to the label to see if there are some patterns. Besides, I want to find out if there are some features have more influence on the prediction. The table below is showing the histogram with all the individual features:





From the observation, I found a few interesting patterns:

(1). For almost all the features who have the values rating from 0-5 will tend to have more satisfied label with “4” or “5” values. The features are Inflight wifi service, Departure/Arrival time convenient, Ease of Online booking, Gate location, Food and drink, Online boarding, Seat comfort, Inflight entertainment, On-board service, Leg room service, Baggage handling, Checkin service, Inflight service, CleanlinessTraining Process.

(2) The features such as Flight Distance, Class, Gender, Age have similar amount of satisfied vs unsatisfied label. But we can’t conclude these features is not important for the label prediction.

(3) For features such as Loyal customer and travel type tends to have more satisfaction among those who paid for higher price services.

In my approach after the observation, I choose to us Principal Component Analysis to downgrade the feature space. I used n_components = 0.85 for PCA and fit and transform on training data set, then apply it to test dataset. The feature importance list after PCA is showing some very informative information: [0.37222178, 0.10889023, 0.03808818, 0.0468296, 0.13501453, 0.08331977, 0.03396086, 0.02336743, 0.03726279, 0.0288797 , 0.0361474 , 0.02597438, 0.03004337]. All the other features that been downgraded are not considered much in the training process.

3.4. Model Selection and Comparison of Results

<u>**3.4.1 Trivial System**</u>

The Trivial System is used and it is just for simple comparison with the actual machine learning models. The Trivial System is simply not learning from the training data, but direct generate the output labels based on random choice between 0 and 1 without taking consideration about the input.

The function of this Trivial System serves as an indicator if our non-trivial model is learning. If the non-trivial model performs even worse than this Trivial System, it means something is wrong.

<u>The test accuracy of the trivial System is 50.12%.</u>

<u>**3.4.2. Note for all the non-Trivial Systems**</u>

In my approach, I first picked 6 different models, including:

<u>1.Logistic Regression;</u>

<u>2.SVC;</u>

<u>3.KNeighbors Classifier;</u>

<u>4.Decision Tree Classifier;</u>

<u>5.AdaBoost Classifier;</u>

<u>6.Random Forest Classifier</u>

I use all the above models to train on Pre-train dataset with 5-fold cross validation with their default hyperparameters to get the validation accuracy. Then I compare all the models based on the accuracy in order to pick the best two models for fine tuning their hyperparameter for achieving better accuracy. The short answer of why I choose these six models is because all of them are suitable for a 2-class classification problem. Besides, Logistic Regression model also server as the base line model due to its simplicity and ease of modeling.

At the end of section 3.4, I will show that the best perform models are SVC and RandomForestClassifier. Pre-training set severs as the data set for the initial model comparison, it intended to find the best two models, default hyperparameters were used for all the 6 models trained with pre-train data.

All the fine tuning of hyperparameters for the best two models will be reported in section 3.5.

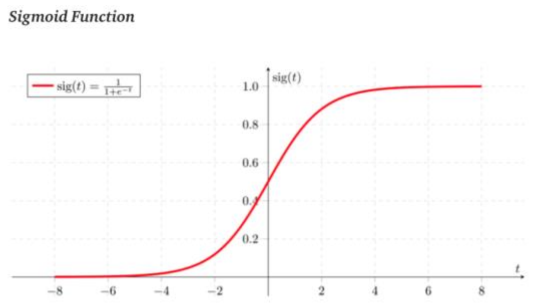
The performance measure is accuracy score, which gives us the most direct guidance how well the model is:

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_\text{samples}} \sum_{i=0}^{n_\text{samples}-1} 1(\hat{y}_i = y_i)$$

3.4.3 Logistic Regression (Baseline Model)

Logistic Regression is a classification algorithm that is used to predict the label of the categorical dependent variable, which is suitable in this project. It requires the dependent variable to be binary, which is a match in our situation. For the best result, it acquires large amount of data, and our Pre-training data set has 103852 entries with 13 features after PCA, overfitting will not show up due to the shape of training data.

The output of our model is 0 or 1, our hypothesis: Z = WX + B, with H = sigmoid(Z). When Z goes to infinity, the prediction will become 1 and if Z goes to negative infinity, the prediction will become 0.



The output / prediction we get from the hypothesis is the estimated value based on probability, which means the confidence for a predicted value when given an input. The model algorithm uses the one-vs-rest. From the nature of this model, we can use it to solve classification problem.

Our Pre-training data set has 103852 entries with 13 features after PCA, overfitting will not show up due to the shape of training data. 5-fold cross validation is used for this classifier.

Since this serves as a baseline model, the hyperparameter of the model is using default value:

penalty	L2
tol	1e-4
C	1
fit_intercept	True

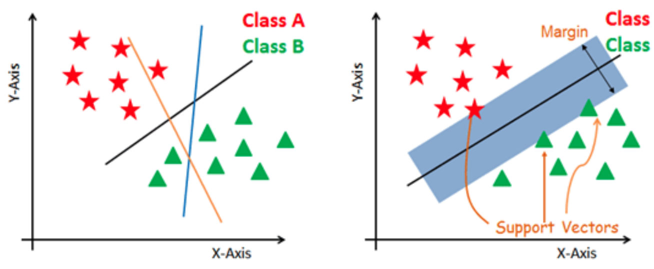
The performance will be evaluated by the accuracy scores:

Performance measure	Value
Training/ cross-validation accuracy	86.05%
Test Accuracy	84.21%

3.4.4 SVC

The first model I choose to explore after Logistic Regression is SVC, which is considered to a good classification approach. It often gives higher accuracy compared to our baseline model. The classifier draws the decision boundary using a hyperplane with the largest amount of margin and it finds the optimal boundary for classification. The hyperplane is generated iteratively to reduce the error.

Basically, the classifier generates hyperplanes to separate the classes, and the right hyperplane with maximum segregation from the nearest data points from both sides of the classes. Also, due to the shape of the dataset, it will not have overfitting problems. 5-fold cross validation is used for this classifier.



The optimization technique is to minimize ||w|| subject to y_i ($w^T x_i$ - b) >= 1 for i = 1….n. And it usually extends to a soft margin when dealing with data that are not linearly separable: $\lambda\|\mathbf{w}\|^2 + \left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - b)\right)\right]$

Since for this is just for selecting the best two model with pre-training set, the parameter uses the default value, the fine tuning of this model will be covered in section 3.5

C	1.0
Kernel	Rbf
Degree	3
Gamma	Scale
tol	1e – 3
Verbose	False
Max_iter	-1
Decision_function_shape	ovr

Performance measure	Value
Training/ cross-validation accuracy	92.80%

3.4.5 K-Neighbors Classifier

K-Neighbors Classifier is a common technique that solves the classification problems, which is very suitable for this project. KNN implements the idea of similarity, and for each data it calculates the distance between the current data and the next data available in the query, and it picks the first K entries from the sorted distance list to get the labels of the selected K entries.

Our Pre-training data set has 103852 entries with 13 features after PCA, overfitting will not show up due to the shape of training data. 5-fold cross validation is used for this classifier.

This stage is not tuning the hyperparameters, but to have a comparison with all other 5 classifier models' performance. The hypermeters used for on pre-training data set is listed below:

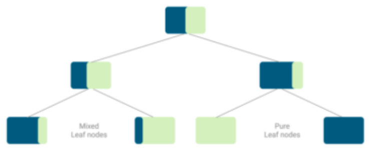
N_neighbors	5
Weights	Uniform
Algorithm	Auto
Leaf_size	30

Performance measure	Value
Training/ cross-validation accuracy	91.15%

3.4.6 Decision Tree Classifier

Decision Tree is a great tool for solving classification problems in supervised learning scheme. It uses a set of decision rules to draw the decision boundary between two classes. It uses the features of the data to form the tree and continually split the dataset until classified all the data points to make them into their classes.

The algorithm will work the best to separate the dataset until all leaf nodes that will not be split further. And the algorithm will assign one class to the data points in each leaf node.



The algorithm separates data points based on the loss function that gives the evaluation of purity of the resulting nodes. It often uses Gini Impurity and Entropy as the measure of how different the classes are for different classes.

$$\mathrm{G(node)} = \sum_{k=1}^{c} \underbrace{p_k}_{\text{Probability of picking a data point from class } k} \overbrace{(1-p_k)}^{\text{Probability of not picking a data point from class } k}$$

$$p_k = \frac{\text{number of observations with class k}}{\text{all observations in node}}$$

Gini Impurity of a node.

$$\mathrm{Entropy(node)} = -\sum_{i=1}^{c} \underbrace{p_k}_{\text{Probability of picking a data point from class } k} \log(p_k)$$

$$p_k = \frac{\text{number of observations with class k}}{\text{all observations in node}}$$

Entropy of a node.

Our Pre-training data set has 103852 entries with 13 features after PCA, overfitting will not show up due to the shape of training data. 5-fold cross validation is used for this classifier.

This stage is not tuning the hyperparameters, but to have a comparison with all other 5 classifier models' performance. The hypermeters used for on pre-training data set is listed below:

Criterion	Gini
Splitter	Best
Max_depth	None
Min_sample_split	2
Min_sample leaf	1
Max_leaf_nodes	None

Performance measure	Value
Training/ cross-validation accuracy	86.97%

3.4.7 AdaBoost Classifier

Boosting is a very strong method to solve classification problem, it usually uses a number of weaker classifiers to form a strong classifier. In our case, we use AdaBoost to solve our classification problem.

In this algorithm, the classifier assign weight to each instance and the initial weight is set to weight(x_i) = 1/n. Then the classifier uses a weak learner to train classifier (1-node CART) on the weighted dataset and make one decision with output 1 or -1. By calculating the misclassification by error = (correct - N) /N to calculated the stage value = ln((1- error) / error). Then the training weights are updated by a certain rule: more weight will be applied to incorrectly predicted data points and less weight to correctly predicted data points. Which will have an effect of putting more penalty to the misclassified data points.

Our Pre-training data set has 103852 entries with 13 features after PCA, overfitting will not show up due to the shape of training data. 5-fold cross validation is used for this classifier.

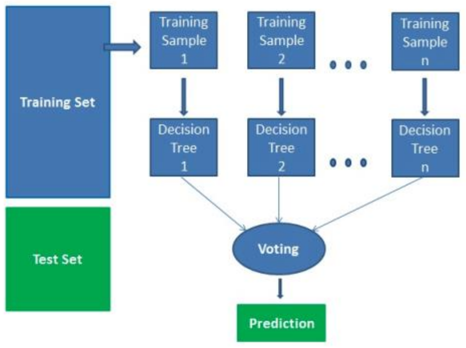
This stage is not tuning the hyperparameters, but to have a comparison with all other 5 classifier models' performance. The hypermeters used for on pre-training data set is listed below:

Base_estimator	DecisionTreeClassifier
N_estimators	50
Learning_rate	1.0
Algorithm	Default
Random_state	None

Performance measure	Value
Training/ cross-validation accuracy	87.15%

3.4.8 Random Forest Classifier

Random Forest is a powerful supervised learning algorithm to solve classification problems. Random forest gets prediction from each tree it creates randomly, and selects the best solution by means of voting.



Random forest will first choose random samples from the dataset, then construct a decision tree for each sample. It gets a prediction from each tree to perform a voting mechanism for the prediction.

Just like Decision Tree Classifier, Random Forest uses gini importance to calculate the importance of each feature.

Our Pre-training data set has 103852 entries with 13 features after PCA, overfitting will not show up due to the shape of training data. 5-fold cross validation is used for this classifier.

This stage is not tuning the hyperparameters, but to have a comparison with all other 5 classifier models' performance. The hypermeters used for on pre-training data set is listed below:

N_estimators	100
Criterion	Gini
Max_depth	None
Min_samples_split	2
Min_sample_leaf	1
Max_feature	Auto
Bootstrap	True

Performance measure	Value
Training/ cross-validation accuracy	91.78%

3.4.9 Quick Summary

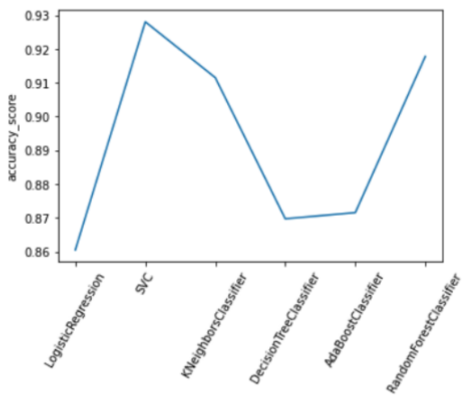
In this section, I presented 6 different classifier models that I trained on preTrain dataset and used 5-fold cross validation to find out their performance. The goal of this is to find the best two models so I can fine tune them to boost their performance. Based on the results presented in section 3.4, the best two models are (1) Random Forest with accuracy 91.87%; (2) SVC with accuracy 92.80%. The next step is to fine tune the hyperparameters for these two best models, which will be presented in the next section.

3.5. Model Selection and Comparison of Results

3.5.1 Models' performance comparison

6 models were trained and had cross validation on pre-train model, the results are shown below:

Model	Accuracy
Trivial Classifier	50.12%
Logistic Regression (Baseline Sys)	86.05%
SVC	92.80%
K-Neighbors Classifier	91.15%
Decision Tree Classifier	86.97%
AdaBoost Classifier	87.15%
Random Forest Classifier	91.78%



Based on the results, we proved that all the models have much higher performance than the Trivial System, which means our models are learning from the data(Pre-Train dataset). From the 6 models, I choose the best two models 1) Random Forest with accuracy 91.87%; (2) SVC with accuracy 92.80% for a more in depth hyperparameter tuning and training on Training dataset.

For hyperparameters tuning, I choose to use the Training dataset and apply the method called GridSearchCV from scikit-learns. It runs all the different parameter that fed into the parameter grid and returns the best combination of parameters based on the metric I choose(accuracy). Please see the results in below:

3.5.2 SVC Hyperparameters Tuning

I fed in the following parameters to GridSearchCV:

Parameters Name	Values
C	[0.05, 0.1, 0.5, 1 ,10]
gamma	[1, 0.5 ,0.1 ,0.01, 0.001]
kernel	[‘linear’, ‘rbf’]

The best combination of parameter returned from GridSearchCV is:

Parameters Name	Values
C	1
gamma	0.1
kernel	rbf

Then I use the above parameters to form my best SVC model. I trained my model on Train dataset (cross-validation included in GridSearchCV) to give the best model performance:

Training / Validation State	Accuracy
Before Tuning	92.80%
After Tuning	94.45%
Improvements	1.78%

We can see we had a 1.78% improvement after the tuning by GridSearchCV, which means the tuning works. Finally, I use this model to predict on the Test dataset, the results are shown below:

State	Accuracy
SVC Training	94.45%
SVC Testing	87.96%

3.5.3 Random Forest Classifier Hyperparameters Tuning

I fed in the following parameters to GridSearchCV:

Parameters Name	Values
bootstrap	True
Max_depth	[5, 10, 50, 100]
Max_features	[2, 4, 6]

N_estimators	[100, 200, 300]

The best combination of parameter returned from GridSearchCV is:

Parameters Name	Values
bootstrap	True
Max_depth	50
Max_features	6
N_estimators	100

Then I use the above parameters to form my best Random Forest Classifier model. I trained my model on Train dataset (cross-validation included in GridSearchCV) to give the best model performance:

Training / Validation State	Accuracy
Before Tuning	91.78%
After Tuning	99.99%
Improvements	8.96%

We can see we had an 8.96% improvement after the tuning by GridSearchCV, which means the tuning works. Finally, I use this model to predict on the Test dataset, the results are shown below:

State	Accuracy
Random Forest Training	99.99%
Random Forest Testing	90.01%

4. Final Results and Interpretation

By comparing all the models' performance, the best model is Random Forest, the parameters and the performance measure (accuracy) are listed below:

Parameters Name	Values
bootstrap	True
Max_depth	50
Max_features	6
N_estimators	100

State	Accuracy
Random Forest Training	99.99%
Random Forest Testing	90.01%

The baseline system (Logistic Regression) gives the training accuracy 86.05%, and test accuracy 84.21%. The best model (Random Forest) outperforms the baseline system. The improvements for training accuracy is 16.20%, and the improvements for testing accuracy is 6.89%.

	Baseline (Logistic Regression)	Best Model (Random Forest)	Improvements
Training Accuracy	86.05%	99.99%	16.20%
Testing Accuracy	84.21%.	90.01%	6.89%

From above results, we can calculate that Eout (hg)<= 0.1429, and our results satisfy this constraint.

It is not surprising that the best model wins the baseline, and this is because the random forest classifier is a much robust model when solving classification problems. I believe the final result is very satisfying and there is no overfitting occurred. There are a few reasons that the final best model performs well: (1) Appropriate pre-processing techniques eliminated some unrelated data points and features for our model. (2) Good hyperparameters tuning gives the best performance.

What we can do to make this model better:

(1) Better feature engineering

(2) Wider range of the selection of hyperparameters.

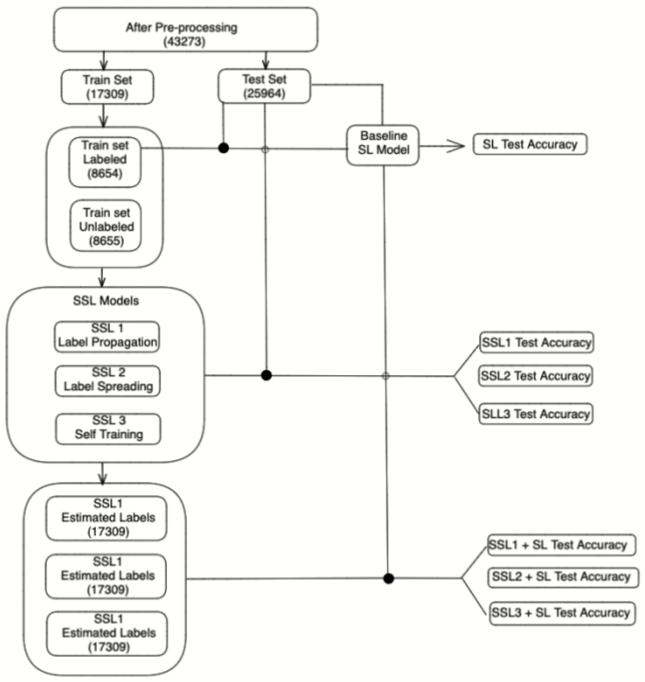
For example, I choose one of the hyperparameters n_estimators from the range [100, 300] and the GridSearchCV gives the final pick of the lowest in the range (n_estimators = 100) for this parameter. Which means we can still give a lower range of this parameter, and it may boost our testing accuracy as well.

5. Implementation for the extension - SSL

For Semi-supervised learning, I choose the training data set size of (17309) and test data set size of (25964). Before deleting the labels from training set, I asset a baseline model to be Logistic Regression and train it on labeled training set and test on the test set. Then I divide the training set to be a labeled set and unlabeled set with a ratio of 1:1 for the preparation of SSL part.

I tried 3 different SSL method: Label Propagation, Label Spreading, and Self Training, and compared their performance based on the accuracy.

After that, I also take another approach and research that how much improvement will we get if we combine SSL and SL together. I do this by use the estimated labels from SSL and fit with SL, and the expectation is that the SL model fit on the entire training data (with estimated labels from SSL) will have better performance than SSL model alone.



5.1 Baseline System (Logistic Regression)

I build a baseline system on SSL dataset using a SL (Logistic Regression) fit only on the labeled data.

The purpose: our expectation is to see an SSL outperforms a SL that fit on the labeled data alone. This is important since if that doesn't match our expectation, then SSL is not doing a good job.

Model name	Test Accuracy
Baseline SL (Logistic Regression)	84%

5.2 SSL Model 1 (Label Propagation)

Label Propagation is a good approach to solve SSL problems. It basically creates a graph to connect all the data based on their distance, then create the soft labels based on the label distribution that nearby. It achieved iteratively to assign labels to unlabeled portion.

The Label Propagation method is available in the scikit-learn Python machine learning library under semi_supervised.LabelPropagation class.

Pre-requisite: This algorithm requires the training data to include labeled examples and unlabeled examples. The unlabeled examples need to marked with a label of -1. I marked all the unlabeled training data to be -1 by “u_label = [-1 for _ in range(len(Y_train_U))]”.

The classification result is shown below:

Model name	Test Accuracy
SSL1 (Label Propagation)	84.01%

We can see that SSL1 model gives a slightly better Test accuracy than the baseline model.

Extensive approach:

We can collect all the estimated labels from SSL model and use them to fir a supervised learning model, in our case I fit the estimated labels to our baseline model. The hope is that the SL model fit on the entire training data with estimated labels from SSL will achieve even better performance than the SL model alone.

The results and comparison results are show below:

Model name	Test Accuracy
Baseline model (Logistic Regression)	84%
SSL1 (Label Propagation)	84.01%
Label Propagation + Logistic Regression	84.54%

We can see the combined SSL + SL performs slightly better than SL models alone, which meets our expectation.

5.2 SSL Model 2 (Label Spreading)

Label Propagation is a good approach to solve SSL problems. It basically assume that the points that are nearby each other should have the same label, and the structure in the input space with similar prosperities should have the same labels.

It achieved by using the weight matrix of a graph and use information passing to capture the structure.

The Label Spreading method is available in the scikit-learn Python machine learning library under semi_supervised.LabelSpreading class.

Pre-requisite: This algorithm requires the training data to include labeled examples and unlabeled examples. The unlabeled examples need to marked with a label of -1. I marked all the unlabeled training data to be -1 by "u_label = [-1 for _ in range(len(Y_train_U))]".

The classification result is shown below:

Model name	Test Accuracy
SSL2 (Label Spreading)	84.01%

We can see that SSL2 model gives a slightly better Test accuracy than the baseline model.

Extensive approach:

We can collect all the estimated labels from SSL model and use them to fir a supervised learning model, in our case I fit the estimated labels to our baseline model. The hope is that the SL model fit on the entire training data with estimated labels from SSL will achieve even better performance than the SL model alone.

The results and comparison results are show below:

Model name	Test Accuracy
Baseline model (Logistic Regression)	84%
SSL1 (Label Spreading)	84.01%
Label Propagation + Logistic Regression	84.57%

We can see the combined SSL + SL performs slightly better than SL models alone, which meets our expectation.

5.3 SSL Model 3 (Self Training)

Label Propagation is a good approach to solve SSL problems. It uses Yarowsky's algorithm. The classifier can be called with a classifier that implements predict_proba, and in our case I use SVC as the called classifier. Self-training method is a wrapper method, it first a SL trained on labeled data only and then applied to the unlabeled data to generate more labeled data points.

The Self Training method is available in the scikit-learn Python machine learning library under semi_supervised. SelfTrainingClassifier class.

Pre-requisite: This algorithm requires the training data to include labeled examples and unlabeled examples. The unlabeled examples need to marked with a label of -1. I marked all the unlabeled training data to be -1 by "u_label = [-1 for _ in range(len(Y_train_U))]".

The classification result is shown below:

Model name	Test Accuracy
SSL2 (Self Training)	87.82%

We can see that SSL2 model gives a better Test accuracy than the baseline model.

Extensive approach:

We can collect all the estimated labels from SSL model and use them to fir a supervised learning model, in our case I fit the estimated labels to our baseline model. The hope is that the SL model fit on the entire training data with estimated labels from SSL will achieve even better performance than the SL model alone.

The results and comparison results are show below:

Model name	Test Accuracy
Baseline model (Logistic Regression)	84%
SSL1 (Self Training)	87.82%
Label Propagation + Logistic Regression	84.64%

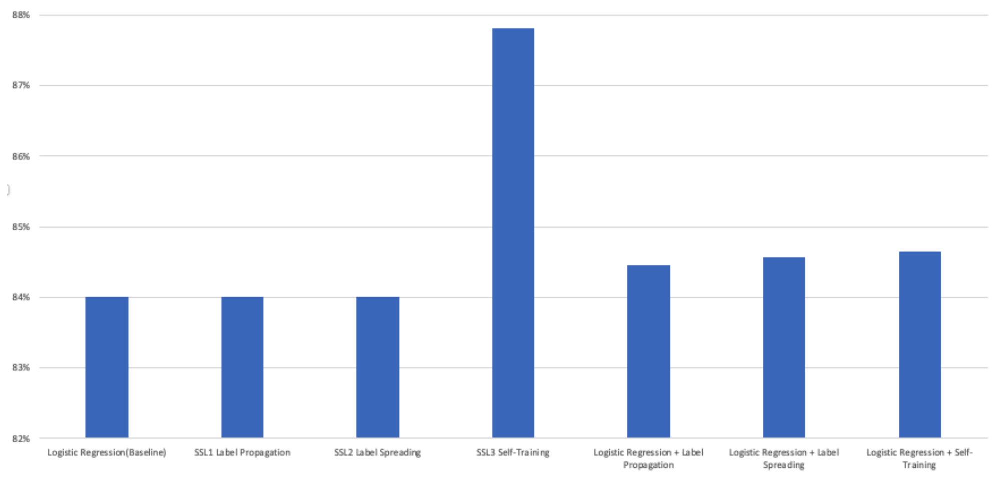
We can see the combined SSL + SL performs slightly better than SL models alone, which meets our expectation.

6. Final Results and Interpretation for the extension -SSL

I tried 3 different SSL method: Label Propagation, Label Spreading, and Self Training, and compared their performance based on the accuracy.

I also take another approach and research that how much improvement will we get if we combine SSL and SL together. The Results are shown below:

Scope	Model Name	Test Accuracy
Baseline	Logistic Regression	84%
SSL	Label Propagation	84.01%
	Label Spreading	84.01%
	Self-Training	87.82%
SL + SSL	Logistic Regression + Label Propagation	84.45%
	Logistic Regression + Label Spreading	84.56%
	Logistic Regression + Self-Training	84.64%



As we can see from the results, all the expectations have been met.

(1) SSL models performs slightly better than the SL baseline model that trained only on the labeled data.

(2) The combined SSL + SL model performs better than SL baseline model and also wins the SSL model alone

Best SSL model is Self-training Classifier, which gives the accuracy <u>87.82%</u>

7. Contributions of each team member

This is a one-person project.

8. Summary and conclusions

In this project, I explored 2 different schemes: Supervised learning and Unsupervised Learning

(1) <u>Supervised Learning</u>

In Supervised Learning, I explored 6 different classifiers including: 1.Logistic Regression; 2.SVC; 3.KNeighbors Classifier; 4.Decision Tree Classifier; 5.AdaBoost Classifier; 6.Random Forest Classifier. All of them are suitable for solving this 2 classes classification problem.

After exploring all the 6 different models, I conclude the best model is Random Forest Classifier, which gives the accuracy of 90.01%.

Something we can do still do better is a more robust way of pre-processing by analyze feature by feature independence and correlation with more detail, and it is possible to eliminate more than half of the features with still a good training and test accuracy.

(2) <u>Semi-Supervised Learning</u>

In Semi-Supervised Learning, I dropped half of the training set labels to make it an SSL problem. During my approach, I explored 3 different SSL models including Label Propagation, Label Spreading and Self Training. Also, I used the predicted labels from the above three models to fit on a supervised learning model, then verified that the performance is better than just use SL to fit on the labeled data alone.

The best SSL model from my investigation is Self-Training classifier, which gives the accuracy of 87.82%.

Something interesting to explore is to use the best SL model combined with the best SSL learning model to train and predict the result. I didn't cover this in my approach due to the time constraint since the training runtime will exceed my limit.

9. References

- *Sklearn.linear_model.logisticregression*. scikit. (n.d.). Retrieved December 7, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

- *Sklearn.svm.SVC*. scikit. (n.d.). Retrieved December 7, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.
- *Sklearn.neighbors.kneighborsclassifier*. scikit. (n.d.). Retrieved December 7, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.
- Landup, D. (2021, July 31). *One-hot encoding in python with pandas and Scikit-Learn*. Stack Abuse. Retrieved December 7, 2021, from https://stackabuse.com/one-hot-encoding-in-python-with-pandas-and-scikit-learn/.
- *Hyperparameter tuning*. GeeksforGeeks. (2020, October 16). Retrieved December 7, 2021, from https://www.geeksforgeeks.org/hyperparameter-tuning/.
- *Sklearn.ensemble.adaboostclassifier*. scikit. (n.d.). Retrieved December 7, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html.
- *Sklearn.semi_supervised.labelpropagation*. scikit. (n.d.). Retrieved December 7, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.semi_supervised.LabelPropagation.html.
- *1.14. semi-supervised learning*. scikit. (n.d.). Retrieved December 7, 2021, from https://scikit-learn.org/stable/modules/semi_supervised.html.
- *Pandas.get_dummies*. pandas.get_dummies - pandas 1.3.4 documentation. (n.d.). Retrieved December 7, 2021, from https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html.