جـــامـعـة الـــشرق الأوســط
MIDDLE EAST UNIVERSITY
Amman - Jordan

Random Forest Classification Algorithm with Airline Passenger Satisfaction Dataset

Bassam Alboushi-202110352

Abstract

This research is mainly talking about Machine Learning Classification Algorithms, Random Forest Classification Algorithm as example and its performance with Airline Passenger Satisfaction dataset. Where this research will study in specific the Random Forest Algorithm such that the methodology, and what is make it good to use in such type of real-life classification problems.

Contents

List of Figures

1 Introduction

As an idea to start talking about random forest classification algorithm, let us talk about the machine learning classification process and how it actually work, and some types of classifiers.

Firstly, if you ask a human to clarify the difference between a dog and a cat, the answer will be easy and that will be back to the human mentality to distinguish things from each other based on certain characteristics that the Creator created to make this human possess the ability to use that skill, but the challenge was how to make a stupid machine to do that job. So for that machine learning appears to follow human made algorithms for clarify the difference between things in general. **Machine Learning** can be defined as a sub-field of computer science that evolved from studying pattern recognition and computational learning theory in artificial intelligence. Machine Learning explores the construction and study of algorithms that can learn from and make predictions on data. Such algorithms build a model from example inputs to make data-driven predictions or decisions rather than following strictly static program instructions. So machine learning, as a set of methods that can automatically detect patterns in data and then use the uncovered patterns to predict future data or to perform other kinds of decision-making under uncertainty (such as planning how to collect more data!)[Robert, 2014].

The difference between machine learning and the traditional approach appears in the algorithms that machine learning follows, such that in machine learning the algorithms consist of giving the model inputs and outputs and training it on them to be tested later by distinguishing patterns or discovering differences. But in traditional programming, algorithms consist of defining a program with lines of instructions executed sequentially to solve a specific problem and generate outputs (see figure 1).

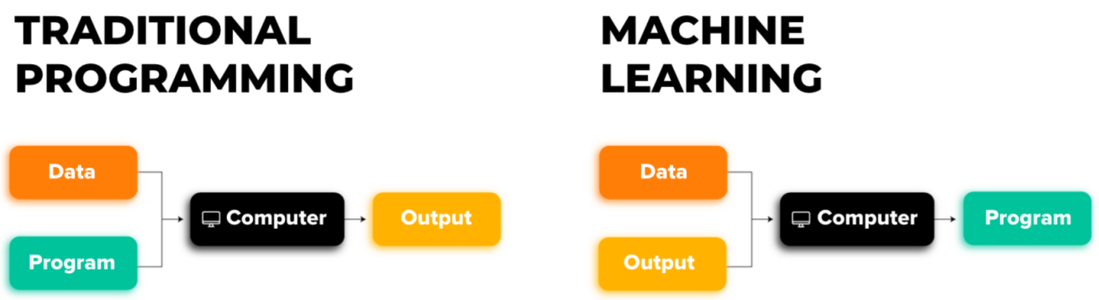
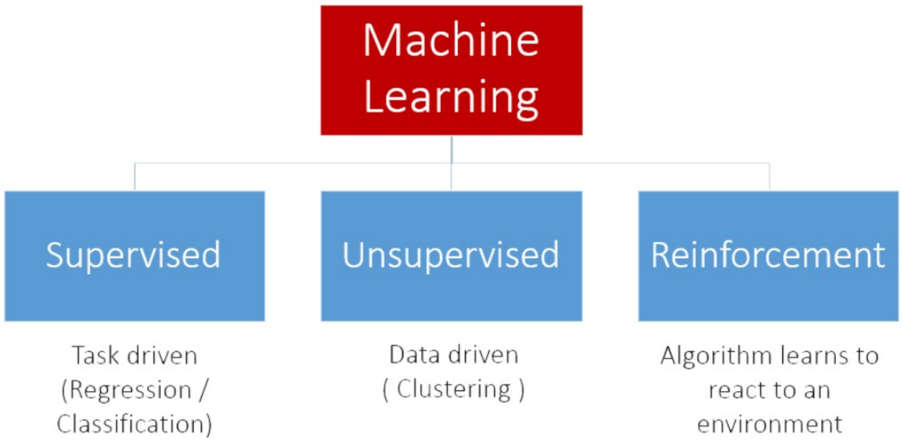


Figure 1: Traditional Programming vs Machine Learning.

Closing this part, to understand the machine learning mechanism, types of machine learning should be mentioned, to figure out the machine learning types, see the following schema below.



Machine Learning
Supervised
Unsupervised
Reinforcement
Task driven (Regression / Classification)
Data driven (Clustering)
Algorithm learns to react to an environment

Figure 2: Types Machine Learning.

As mentioned before, this research will only talk about classification which belongs to Supervised Machine Learning. In addition, the difference between supervised machine learning and the other types is that supervised machine learning asking to enter the data or patterns with its labels to train with, the following figure will clarify the mechanism of Supervised Machine Learning.

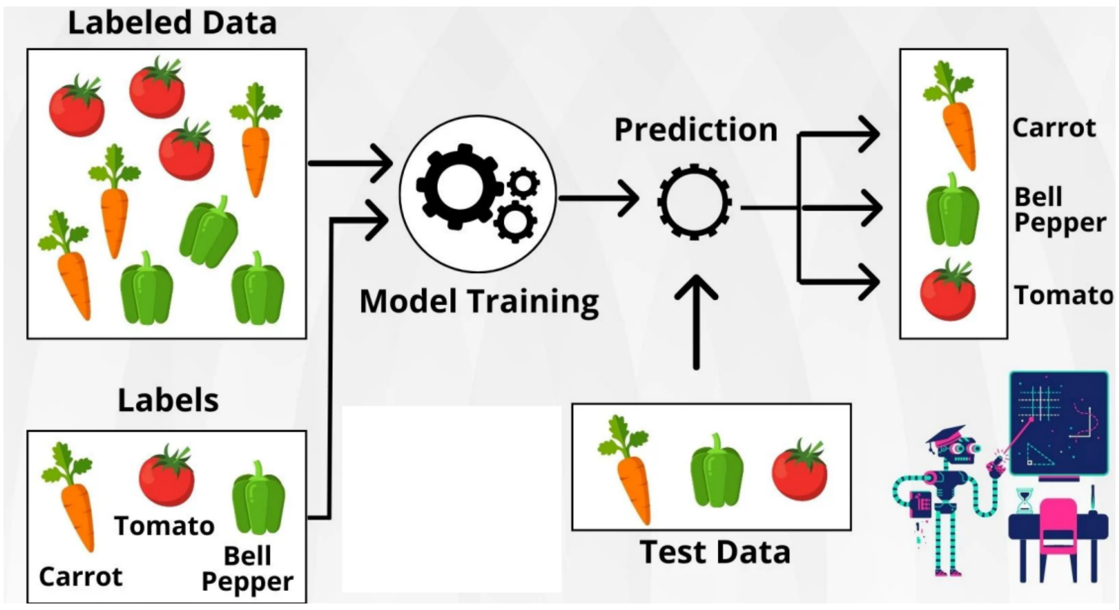


Figure 3: Supervised Machine Learning workflow.

Secondly in this section, after understanding the machine learning concepts and algorithms mentioned above, let us talk about the classification process. The classification process consists of training machines on a pattern (this is also called fitting a model and where model is mentioned, it also means machine in this research) by giving its labels that are desired to be classified making the model ready to understand the patters later even if there are no labels entered. In another simple meaning as example, train the model and give it the answers so you can test the model, and the model will predict the answers based on what it is trained for. But, it is not that simple, the overfitting and underfitting of the models should be avoided. So, let us illustrate what overfitting and underfitting mean in two simple definitions:

- **Overfitting:** means that when the models are trained too much so that make them understand only the patterns that they trained for, without making any sense to ask that models to predict any other patterns or data that desired to be classified.

- **Underfitting:** means that when the models are not trained enough to make them distinguish the patterns in a proper way so that when the data or patterns are entered to be classified, the models will make a lot of mistakes in prediction.

Here will be noticed that the models should be trained in a proper way that makes them understand the patterns and data that want to be classified, simple naming of this is set to be just ”fit a model”. Fitting a model by dividing the dataset into training and testing sets is a very crucial part of the classification process, it is playing a huge role when non-labeled patterns or data (testing set) are entered to be classified. This means, that dividing the dataset into training and testing sets, then training the models on the training set, should make the models able to classify and understand the testing set so they can predict the result with minimizing the misses in predictions of labels, and this is simply the goal of machine learning Classification process. As an idea to make machines classify objects, surely there are some classification algorithms that should be worked with, the most popular classification algorithms are:

- Decision Tree
- Support Vector Machine
- Random Forest
- K-Nearest Neighbour
- Naive Bayes
- Linear Discriminate Analysis

So, it is mentioned before that this research will illustrate the Random Forest Classification Algorithm which provides the Random Forest Classifier.

Finally in this section, after talking, understanding, and clarifying the classification process, let us talk about the model itself, which will be called a ”classifier”. The history of classifiers will take so long to talk about or talk about each classifier’s pros or cons. Therefore, some types of classifiers will be mentioned as follows:

1. **Perceptron Classifier:** The earliest and simplest classifier that deals with linearly separable datasets by drawing one linear line to separate two classes.

2. **Support Vector Classifier:** Classifies data by finding an optimal line or hyperplane that maximizes the distance between each class in an N-dimensional space.

3. **K-Nearest Neighbour Classifier:** Non-parametric classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

4. **Random Forest Classifier:** A classifier that creates a set of decision trees from a randomly selected subset of the training set.

In the end, of course, there are more and more classifiers to talk about, but the four mentioned above are the most popular and this research will take a Random Forest Classifier algorithm as a main subject to talk about its different characteristics.

2 Literature Review

The random forest algorithm, proposed by L. Breiman in 2001, has been extremely successful as a general-purpose classification and regression method, where the approach, which is represented by combining several decision trees and aggregates their prediction by averaging, has shown excellent performance in large-scale problems because of the diversity of its abilities. Random Forest Supervised Learning algorithm works with a simple but effective ”divide and conquer” principle that works as taking a sample of data, growing a randomized decision tree predictor on each small piece, then aggregating these predictors together. The popularity of random forests comes from that they can be applied to a wide range of prediction (classification) problems with a high-dimensional number of features beside that ease to use and deal with as well as the small number of parameters to tune [Biau and Scornet, 2016].

The most popular theoretical result of how random forest works reached by Breiman (see [Breiman, 2001]), which offers an upper bound on the generalization error of forests in terms of correlation and strength of the individual trees. This was followed by a technical note mentioned in [2004], which focuses on a stylized version of the original algorithm [2000]. Speaking of random forest ingredients, the bagging and Classification And Regressing Trees (CART)-split criterion, which are essential components, play critical roles. Bagging (a contraction of bootstrap-aggregating) is a general aggregation scheme, which generates bootstrap samples from the original data set, constructs a predictor from each sample, and makes decisions by averaging the constructed predictors results. Bagging is one of the most effective computationally intensive procedures to improve on unstable estimates, especially for large, high-dimensional data sets, where finding a good model in one step is impossible because of the complexity and scale of the problem. As for the CART-split criterion, which is used in the construction of the individual trees to choose the best cuts perpendicular to the axes. At each node of each tree, the best cut is selected by optimizing the CART-split criterion, based on the so-called Gini impurity (for classification) or the prediction squared error (for regression)[Bühlmann and Yu, 2002].

As the Airline Passengers Satisfaction dataset is a binary represented classification problem with a high-dimensional number of features and a huge number of records, a random forest classifier will be used, keeping in mind that random forests are intrinsically capable of dealing with multi-class problems. However, the random forest mechanism appears simple, but it involves many different driving forces which makes it difficult to analyze. In fact, its mathematical properties remain to date largely unknown, and, up to now, most theoretical studies have concentrated on isolated parts or stylized versions of the algorithm[Biau, 2012].

3 Data Description

The Airline Passenger Satisfaction dataset is a dataset that contains the most popular important factors that affect passenger satisfaction. Here the dataset has 22 features in addition to its label that classifies the passenger to be satisfied (or dissatisfied) according to his answers in the survey.

In the columns that contain numeric answers (except Delay in minutes features), it should be known that the maximum number is 5, which means that the passenger is said to be satisfied with that factor. On the other hand, the minimum number is 1, which means that the passenger is said to be dissatisfied with that factors, and surly there are some answers in the middle like 3. So, where should that answers be classified? Is passenger said to be satisfied or dissatisfied? To solve this point of confusion, the labels in the dataset are classified into two categories, passengers will be "satisfied" or "neutral or dissatisfied". In detail, the dataset is going to be described in the next few points.

- **Sources:** Airline Passenger Satisfaction dataset is collected from USA airline passengers by survey and then uploaded on Kaggle[Kggle, n.d.]from John D[john d, n.d.]. Note that the dataset was modified from TJ Klein[Klein, n.d.]. It has been cleaned up for the purposes of classification.
- **Size:** Divided into two datasets for training and testing. Basically the training dataset contains 103,905 records, and the testing dataset contains 25,977 records, both with 22 features and one observer(label).

- **Attributes:** The dataset contains two types of attributes, categorical and numerical. Most of numerical attributes contains satisfaction scale out of 5 except the ID, age, delay in minutes for both arrival and departure time. As well as, there are some categorical attributes which contain passenger information such that gender, customer type, type of travel , and flight class. In this work, the focus will be on those attributes that contains satisfaction scale and all categorical attributes that affects the classification decision.

- **Pre-processing Steps:** The both training and testing datasets are said to almost clean, but of course there are one dropped column and some missing values in some columns that were processed with Python built-in libraries that provide functions. The datasets preprocessing is done as following:

  1. **Drop 'Unnamed' column which is meaningless int the datasets**

```
#Drop the unnamed column from the train and test dataset
APS_test_data = APS_test_data.drop(columns=['Unnamed: 0'])
APS_train_data = APS_train_data.drop(columns=['Unnamed: 0'])
```

  2. **Check for null values**

```
#Check for Null values if exist in datasets
APS_train_data.isnull().sum()

Output:
id                                     0
Gender                                 0
Customer Type                          0
Age                                    0
Type of Travel                         0
Class                                  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
On-board 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             310
satisfaction                           0
dtype: int64
```

```
APS_test_data.isna().sum()
Output:
id                                       0
Gender                                   0
Customer Type                            0
Age                                      0
Type of Travel                           0
Class                                    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
On-board 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                83
satisfaction                             0
dtype: int64
```

3. **Fill the null values**

```
#Filling the NULL values in both train and test datasets, here I
                                     will use the mean of
                                    column that contain NULL
                                    values
mean_train = APS_train_data['Arrival Delay in Minutes'].mean()
mean_test = APS_test_data['Arrival Delay in Minutes'].mean()

APS_train_data['Arrival Delay in Minutes'].fillna(mean_train,
                                    inplace = True)
APS_test_data['Arrival Delay in Minutes'].fillna(mean_test,
                                    inplace = True)
```

4. **Check again if the datasets are cleaned successfully**

```
#Recheck for NULL values
APS_train_data.isnull().sum()
Output:
id                                      0
Gender                                  0
Customer Type                           0
Age                                     0
Type of Travel                          0
Class                                   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
On-board 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             0
satisfaction                         0
dtype: int64
```

```
APS_test_data.isnull().sum()
Output:
id                                   0
Gender                               0
Customer Type                        0
Age                                  0
Type of Travel                       0
Class                                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
On-board 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             0
satisfaction                         0
dtype: int64
```

4 Methodology

After Airline Passenger Satisfaction dataset is cleaned successfully, it is ready to use for machine learning classification models. As mentioned above, for this huge dataset with high-dimensional number of features, the random forest classifier has been decided to use. So for that, to make interesting results in classification, some feature engineering was used. However, this section will discuss all three classification results that are:

1. **Results that appear from Recursive Features Elimination (RFE) method**
2. **Results that appear from author selected features**
3. **Results that appear from selecting all features**

Firstly, some common steps will be mentioned that will lead to extracting the previously mentioned results. let us write down the steps:

- **Include all libraries that will be used**

```
#All libraries that will be used
import numpy as np
import pandas as pd
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import accuracy_score, precision_score,
                                  recall_score, f1_score,
                                  confusion_matrix,
                                  classification_report
```

- **Replace categorical values with numerical values for classification models**

```
#Convert the dataset in both train and test to numeric data that the
                                    classifier can treat with
                                   easily
APS_train_data['Gender'] = APS_train_data['Gender'].replace({'Male':
                                    0, 'Female': 1})
APS_train_data['Customer Type'] = APS_train_data['Customer Type'].
                                   replace({'Loyal Customer': 0, '
                                   disloyal Customer': 1})
APS_train_data['Type of Travel'] = APS_train_data['Type of Travel'].
                                   replace({'Personal Travel': 0, '
                                   Business travel': 1})
APS_train_data['Class'] = APS_train_data['Class'].replace({'Eco': 0,
                                    'Eco Plus': 1, 'Business' : 2})
APS_train_data['satisfaction'] = APS_train_data['satisfaction'].
                                   replace({'neutral or
                                   dissatisfied': 1, 'satisfied' :
                                   2})
```

```
APS_test_data['Gender'] = APS_test_data['Gender'].replace({'Male': 0
                                    , 'Female': 1})
APS_test_data['Customer Type'] = APS_test_data['Customer Type'].
                                    replace({'Loyal Customer': 0, '
                                    disloyal Customer': 1})
APS_test_data['Type of Travel'] = APS_test_data['Type of Travel'].
                                    replace({'Personal Travel': 0, '
                                    Business travel': 1})
APS_test_data['Class'] = APS_test_data['Class'].replace({'Eco': 0, '
                                    Eco Plus': 1, 'Business' : 2})
APS_test_data['satisfaction'] = APS_test_data['satisfaction'].
                                    replace({'neutral or
                                    dissatisfied': 1, 'satisfied' :
                                    2})
```

- **Split the dataset attributes into 'features' and 'label'**

```
#As example, these lines of code will show how to split the datasets
                                     into X (features) and Y (label)
X = APS_train_data.iloc[:, :-1] #select all columns except the last
                                    one
Y = APS_train_data['satisfaction'] #select the labels column
```

As well as all common steps are discussed, let us start new discussion for how three-mentioned results are obtained.

4.1 Results that appear from Recursive Features Elimination (RFE) method

In this part, the discussion will be about results that obtained from well-known feature engineering method, which is Recursive Features Elimination (RFE). In simple and basic way, RFE method eliminate the least important features that affect the classification process results and keep the most important features by fitting a model that studies the features and its effectiveness on the results. After extracting the most important features, random forest classification model with RFE selected features will be fitted and tested as well as evaluation and validation results will be obtained. Here is step by step Python code:

- **Select the number of features that you want to extract then train the model to extract it**

```
no_features_to_select = 12
model = RandomForestClassifier ()
rfe = RFE(estimator = model , no_features_to_select)
X_rfe = rfe.fit_transform (X, Y)
```

- **Get the selected features from RFE**

```
# Get the selected features
selected_features = [X.columns[i] for i in range(len(X.columns)) if
                                    rfe.support_ [i]]
print("Selected features: ", selected_features)
Output:
Selected features:  ['id', 'Customer Type', 'Age', 'Type of Travel',
                                     'Class', 'Flight Distance', '
                                    Inflight wifi service', 'Ease of
                                     Online booking', 'Online
                                    boarding', 'Seat comfort', '
                                    Inflight entertainment', 'Leg
                                    room service']
```

- **Drop the non-selected features from the dataset**

```
#Drop the non-selected features to train the model with the selected
                                     features
columns_to_drop = ['Gender', 'Customer Type', 'Departure/Arrival
                                    time convenient', 'Gate location
                                    ', 'Food and drink', 'Departure
                                    Delay in Minutes', 'Arrival
                                    Delay in Minutes', 'Checkin
                                    service', 'Inflight service', '
                                    Baggage handling', 'Cleanliness'
                                    ]
test_of_FE = test_of_FE.drop(columns = columns_to_drop, axis = 1,
                                    inplace=False) #test_of_FE is a
                                    copy data set for feature
                                    engineering
```

- **Train and test the model with the training and testing datasets that contain the selected features**

```
#Train the random forest classifier
rf_clf = RandomForestClassifier()
rf_clf.fit(X_train, Y_train)

#Predict the labeles for the testing dataset
Y_pred = rf_clf.predict(X_test)
```

- **Validate the predicted results via cross-validation approach**

```
clf_cross = RandomForestClassifier()
scores = cross_val_score(clf_cross, X_train, Y_train, cv = 10,
                                    scoring='accuracy')
scores
Output:
array([0.94966798, 0.94822442, 0.94803195, 0.95236262, 0.94754572,
       0.94889317, 0.95178056, 0.95120308, 0.95216554, 0.95216554])
```

```
clf_y_pred = cross_val_predict(clf_cross, X_test, Y_test, cv=10)
report = classification_report(Y_test, clf_y_pred)
print(report)
Output:
                precision    recall  f1-score   support

           1       0.94      0.96      0.95     14573
           2       0.95      0.92      0.94     11403

    accuracy                           0.94     25976
   macro avg       0.95      0.94      0.94     25976
weighted avg       0.94      0.94      0.94     25976
```

- **Evaluate the results with the evaluation metrics**

```
        cm = confusion_matrix(Y_test, Y_pred)
accuracy = accuracy_score(Y_test, Y_pred)
precision = precision_score(Y_test, Y_pred, average='weighted')
recall = recall_score(Y_test, Y_pred, average='weighted')
f1 = f1_score(Y_test, Y_pred, average='weighted')
print("Confusion Matrix for the measurment of classification:")
print(cm)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)

Output:
Confusion Matrix for the measurment of classification:
[[14090   483]
 [  871 10532]]
Accuracy: 0.9478749615029258
Precision: 0.9480895853869068
Recall: 0.9478749615029258
F1 Score: 0.9477662776758736
```

4.2 Results that appear from author selected features

In this part, the discussion will be about results that obtained from author selected features based on simple experience with customer service and what are the factors that may make the customer happy more than the others, so that features will be included in the classification porcess. Here is step by step Python code for how the results are obtained and evaluated:

- **Select the features**

```
#Select 12 features from the dataset
manually_selected_features = ['Age', 'Gender', 'Customer Type', '
                                    Class','Flight Distance', '
                                    Inflight wifi sevice', 'Depature
                                    /Arrival convenient','Ease of
                                    Online booking',
                                                                    '
                                    Online boarding', 'Seat comfort'
                                    ,'Inflight enterainment', '
                                    Inflight sevice']
```

- **Drop the non-selected features from the dataset**

```
#Drop the non-selected column from the training and testing datasets
Colmuns_to_drop = ['id','Gate location', 'Food and drink', 'On-board
                                     service', 'Leg room service', '
                                    Baggage handling','Checkin
                                    service', 'Cleanliness', '
                                    Departure Delay in Minutes', '
                                    Arrival Delay in Minutes']
manual_test_dataset = manual_test_dataset.drop(columns =
                                    Colmuns_to_drop, axis = 1,
                                    inplace=False)
manual_train_dataset = manual_train_dataset.drop(columns =
                                    Colmuns_to_drop, axis = 1,
                                    inplace=False)
#manual_train/test_datasets is another copy from Airline Passenger
                                    Satisfaction dataset to work
                                    with
```

- **Train and test the model with the training and testing datasets that contain the selected features**

```
#Call classification model and train it with the selected features
manual_model = RandomForestClassifier()
manual_X_train = manual_train_dataset.iloc[:, :-1]
manual_Y_train = manual_train_dataset['satisfaction']
manual_model.fit(manual_X_train, manual_Y_train)

#Predict the results with the testing datasets
manual_X_test =  manual_test_dataset.iloc[:, :-1]
manual_Y_test = manual_test_dataset['satisfaction']
manual_y_pred = manual_model.predict(manual_X_test)
```

- **Validate the predicted results via cross-validation approach**

```
clf_cross_for_my_test = RandomForestClassifier()
scores = cross_val_score(clf_cross_for_my_test, manual_X_train,
                                    manual_Y_train, cv = 10, scoring
                                    ='accuracy')
scores
Output:
array([0.9517852 , 0.95313252, 0.95159272, 0.95544221, 0.94966314,
       0.95341675, 0.95178056, 0.95370549, 0.95534167, 0.9533205 ])
```

```
cross_y_pred_for_my_test = cross_val_predict(clf_cross_for_my_test,
                                    manual_X_test, manual_Y_test, cv
                                     = 10)
report = classification_report(manual_Y_test,
                                    cross_y_pred_for_my_test)
print(report)
Output:
                precision    recall  f1-score   support

           1       0.95      0.97      0.96     14573
           2       0.95      0.93      0.94     11403

    accuracy                           0.95     25976
   macro avg       0.95      0.95      0.95     25976
weighted avg       0.95      0.95      0.95     25976
```

- **Evaluate the results with the evaluation metrics**

```
manual_cm_for_my_test = confusion_matrix(manual_Y_test,
                                    manual_y_pred)
manual_test_accuracy = accuracy_score(manual_Y_test, manual_y_pred)
manual_test_precision = precision_score(manual_Y_test, manual_y_pred
                                    , average='weighted')
manual_test_recall = recall_score(manual_Y_test, manual_y_pred,
                                    average='weighted')
manual_test_f1 = f1_score(manual_Y_test, manual_y_pred, average='
                                    weighted')
print("Confusion Matrix for the measurment of classification:")
print(manual_cm_for_my_test)
print("Accuracy:", manual_test_accuracy)
print("Precision:", manual_test_precision)
print("Recall:", manual_test_recall)
print("F1 Score:", manual_test_f1)

Output:
Confusion Matrix for the measurment of classification:
[[14150   423]
 [  769 10634]]
Accuracy: 0.954111487526948
Precision: 0.9542884928671984
Recall: 0.954111487526948
F1 Score: 0.9540272093819867
```

4.3 Results that appear from selecting all features

In this part, the discussion will be about results that obtained from select all features to make it included the classification process. Here is step by step Python code for how the results are obtained and evaluated:

- **Upload a new copy with all features**

```
#Load a new copy and split training, testing features inplace
all_features_train = APS_train_data
all_features_test = APS_test_data
all_features_train_X = all_features_train.iloc[:, :-1]
all_features_train_Y = all_features_train['satisfaction']
all_features_test_X = all_features_test.iloc[:, :-1]
all_features_test_Y = all_features_test['satisfaction']
```

- **Train and test the model with the training and testing datasets**

```
#Train the model with all features and its labels
all_features_model = RandomForestClassifier()
all_features_model.fit(all_features_train_X, all_features_train_Y)

#Predict the result of training
all_features_predict = all_features_model.predict(
                                     all_features_test_X)
```

- **Evaluate the results with the evaluation metrics**

```
#Simple evaluation on the predicted results
all_features_cm = confusion_matrix(all_features_test_Y,
                                     all_features_predict)
all_features_accuracy = accuracy_score(all_features_test_Y,
                                     all_features_predict)
all_features_precision = precision_score(all_features_test_Y,
                                     all_features_predict, average='
                                     weighted')
all_features_recall = recall_score(all_features_test_Y,
                                     all_features_predict, average='
                                     weighted')
all_features_f1 = f1_score(all_features_test_Y, all_features_predict
                                     , average='weighted')
print("Confusion Matrix for the measurment of classification:")
print(all_features_cm)
print("Accuracy:", all_features_accuracy)
print("Precision:", all_features_precision)
print("Recall:", all_features_recall)
print("F1 Score:",all_features_f1)

Output:
Confusion Matrix for the measurement of classification:
[[14301   272]
 [  636 10767]]
Accuracy: 0.965044656606098
Precision: 0.9652960334990764
Recall: 0.965044656606098
F1 Score: 0.9649767595828657
```

5 Results and Discussion

So, the results have been obtained in the previous section, make us aware that when the dataset is large enough to be classified, the random forest classifier is one of the best choices to treat with. In this work, it was aimed in a special case to compare the results from the three-mentioned above criteria, and the results were clear. What is wanted to be proved, that the feature engineering method (RFE), is it a good way to reduce the dataset dimensionality Which in turn will reduce complexity?
The answer in our case is simply yes, since the accuracy from the RFE selected features is very close to that one appeared from selecting all features or the author selected features in general. One more time, Recursive Feature Elimination is such an efficient way to reduce the complexity in the classification process with a high-dimensional number of features and huge number of records.

6 Conclusion

Choosing the algorithm, model, and do some feature engineering that deals with the dataset that wants to be classified is an important and very effective decision to make, since machine learning, in general, uses fancy algorithms. The decision you make, is the accuracy you have.

Certainly, all factors should be taken into account when some classification algorithm is said to be chosen, it is not easy or simple when you choose the wrong classification algorithm for certain datasets and certain features, the machine learning classification process is much more complex to deal with those mistakes.

References

Biau, G. (2012). Analysis of a random forests model. *The Journal of Machine Learning Research*, *13*(1), 1063–1095.

Biau, G., & Scornet, E. (2016). A random forest guided tour. *Test*, *25*, 197–227.

Breiman, L. (2000). Randomizing outputs to increase prediction accuracy. *Machine Learning*, *40*, 229–242.

Breiman, L. (2001). Random forests. *Machine learning*, *45*, 5–32.

Breiman, L. (2004). Consistency for a simple model of random forests. *Machine Learning*.

Bühlmann, P., & Yu, B. (2002). Analyzing bagging. *The annals of Statistics*, *30*(4), 927–961.

john d. (n.d.). https://www.kaggle.com/johnddddd

Kggle. (n.d.). https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction

Klein, T. (n.d.). https://www.kaggle.com/teejmahal20

Robert, C. (2014). Machine learning, a probabilistic perspective. *CHANCE*, *27*(2), 62–63. https://doi.org/10.1080/09332480.2014.914768