Supervised Machine Learning: Classification – Project

IBM Machine Learning Professional Certificate

By: Bernardita Štitić

Date: January 3, 2022

TABLE OF CONTENTS

1. OBJECTIVE AND BENEFITS 2

2. BRIEF DESCRIPTION OF THE DATA SET AND ITS ATTRIBUTES 2

3. DATA EXPLORATION, DATA CLEANING AND FEATURE ENGINEERING 4

3.1. Data exploration 4

3.2. Data cleaning and feature engineering 7

4. CLASSIFICATION MODELS 11

4.1. KNN classifier 11

4.2. Random Forest classifier 17

4.3. Extra Random Trees classifier 20

4.4. Model comparison 23

5. DISCUSSION AND MODEL RECOMMENDATION 24

6. KEY FINDINGS AND INSIGHTS 25

7. NEXT STEPS 26

REFERENCES 29

APPENDICES 30

Appendix A. Random Forest classifiers: OOB error 30

Appendix B. Extra Random Trees classifiers: OOB error 31

Appendix C. Random Forest and Extra Random Trees classifiers: OOB error 32

1. OBJECTIVE AND BENEFITS

In this project, the developed models will be focused on prediction so this will be the main objective of this analysis. Moreover, the benefits provided to the business or stakeholders of the churn phone data set, which was used in this project, correspond to models with satisfactory classification performance on the test set. Specifically, the developed models aim to achieve a minimum target value of a precision of 80%, a recall of 70% and an accuracy of 70%. In particular, results like these are promising and thus serve as an initial basis for further model optimization cycles to ideally obtain even higher quality results, especially concerning precision, which is extremely relevant in this case as it will be later addressed in this report.

Therefore, this strong initial basis regarding classification metrics is the most important benefit of this analysis. Furthermore, the developed models should also be able to classify data points in an unbalanced data set, which is another important benefit since such a scenario is likely in practice. In other words, it is desirable that the models show potential for properly addressing the classification problem of the project. However, it should be noted that testing data in this project will be used for validation purposes so additional data samples will be required to further evaluate the models. This topic will also be addressed later in this report.

2. BRIEF DESCRIPTION OF THE DATA SET AND ITS ATTRIBUTES

The chosen data set for this project corresponds to a subset of the customer churn data set. This subset was preprocessed in the file “03b_LAB_KNN.ipynb” (course laboratory) during the course. The customer churn data set belongs to the telecommunications industry, is based on a fictional company and includes information such as data usage, monthly revenues, offerings and customer data. Moreover, for this project, the data subset which was chosen is related to customers who have phone accounts as explained in the course laboratory. This subset will be referred to as the churn phone data set just like in the project instructions.

In particular, this data subset (or data set in this report) includes numeric and categorical variables, both nominal and ordinal. The specific variable categories will be further described later in Section 3, which addresses the preprocessing stage. In the meantime, as a summary, Table 2.1 shows the information obtained using the `.info()` Pandas method on a Pandas DataFrame which contains the phone data set. The table shows the names, non-null values and types of the data set attributes.

Row index	Column name	Non-null values	Column data type
0	Id	7043	Object
1	months	7043	int64
2	offer	7043	Object
3	Phone	7043	Object
4	Multiple	7043	Object
5	internet_type	7043	Object
6	gb_mon	7043	int64
7	Security	7043	Object
8	Backup	7043	Object
9	Protection	7043	Object
10	Support	7043	Object
11	Unlimited	7043	Object
12	Contract	7043	Object
13	Paperless	7043	Object
14	Payment	7043	Object
15	Monthly	7043	float64
16	total_revenue	7043	float64
17	Satisfaction	7043	int64
18	churn_value	7043	int64
19	churn_score	7043	int64
20	Cltv	7043	int64

Table 2.1. Relevant information of a Pandas DataFrame with the churn phone data set.

As Table 2.1 implies, this data set consists of 7043 observations with 21 variables and no missing values. There are, in total, 2 float64 columns, 6 int64 columns and 13 object columns in the Pandas DataFrame that contains the entire data set. Moreover, the target corresponds to the variable named churn_value, while the remaining 20 variables correspond to explanatory variables or features.

After specific preprocessing steps which will be described next, achieving good classification performance which uses a selection of the features in Table 2.1 is the goal of this analysis. Specifically, due to the nature of this classification problem, the aim is to reach a minimum precision of 80%, as well as satisfactory levels of accuracy and recall (70% minimum for both) as mentioned in Section 1. These target results aim to find models that will serve as a good initial basis for further model development to adequately solve the classification problem from a predictive perspective as highlighted in Section 1.

3. DATA EXPLORATION, DATA CLEANING AND FEATURE ENGINEERING

3.1. Data exploration

For the data exploration phase, the following actions were taken with respect to the data set from the previous section, which consisted of 7043 observations and 21 variables:

1. Relevant statistics: using the Pandas `.describe()` method, relevant statistics of all 20 features and the target were obtained.
    - For numeric variables (`float64` and `int64`), the statistics included the number of non-null values, the sample mean, the sample standard deviation, the minimum value, relevant percentiles (25^{th}, 50^{th} and 75^{th}) and the maximum value. The results are illustrated in Figure 3.1.1. As the figure shows, features are not on the same scale.
    - For `object` type columns, the statistics included the number of nun-null values, the unique values, the top value and its frequency. The results are shown in Figure 3.1.2.

	count	mean	std	min	25%	50%	75%	max
months	7043.0	32.386767	24.542061	1.00	9.00	29.00	55.000	72.00
gb_mon	7043.0	20.515405	20.418940	0.00	3.00	17.00	27.000	85.00
monthly	7043.0	64.761692	30.090047	18.25	35.50	70.35	89.850	118.75
total_revenue	7043.0	3034.379056	2865.204542	21.36	605.61	2108.64	4801.145	11979.34
satisfaction	7043.0	3.244924	1.201657	1.00	3.00	3.00	4.000	5.00
churn_value	7043.0	0.265370	0.441561	0.00	0.00	0.00	1.000	1.00
churn_score	7043.0	58.505040	21.170031	5.00	40.00	61.00	75.500	96.00
cltv	7043.0	4400.295755	1183.057152	2003.00	3469.00	4527.00	5380.500	6500.00

Figure 3.1.1. Relevant statistics of the numeric variables of the churn phone data set.

	count	unique	top	freq
id	7043	7043	7596-ZYWBB	1
offer	7043	6	None	3877
phone	7043	2	Yes	6361
multiple	7043	2	No	4072
internet_type	7043	4	Fiber Optic	3035
security	7043	2	No	5024
backup	7043	2	No	4614
protection	7043	2	No	4621
support	7043	2	No	4999
unlimited	7043	2	Yes	4745
contract	7043	3	Month-to-Month	3610
paperless	7043	2	Yes	4171
payment	7043	3	Bank Withdrawal	3909

Figure 3.1.2. Relevant statistics of the categorical variables of the churn phone data set.

2. Feature selection: as performed in the course laboratory on KNN, the features called id, phone, total_revenue, cltv and churn_score were discarded due to not being considered to be important enough for this analysis. Therefore, the columns related to these variables were dropped. These columns were related to customer ID information, phone customer status, total revenue, lifetime value and churn scores which were bank-estimated. In consequence, the feature space was reduced from a 20 dimensional to a 15 dimensional one.
3. Pair plot: a pair plot of the data set was obtained using the pairplot() function by Seaborn. It is shown in Figure 3.1.3. There are some linear dependencies, as well as some apparently skewed distributions, that should be addressed when necessary. This topic will be addressed later again in Section 7.

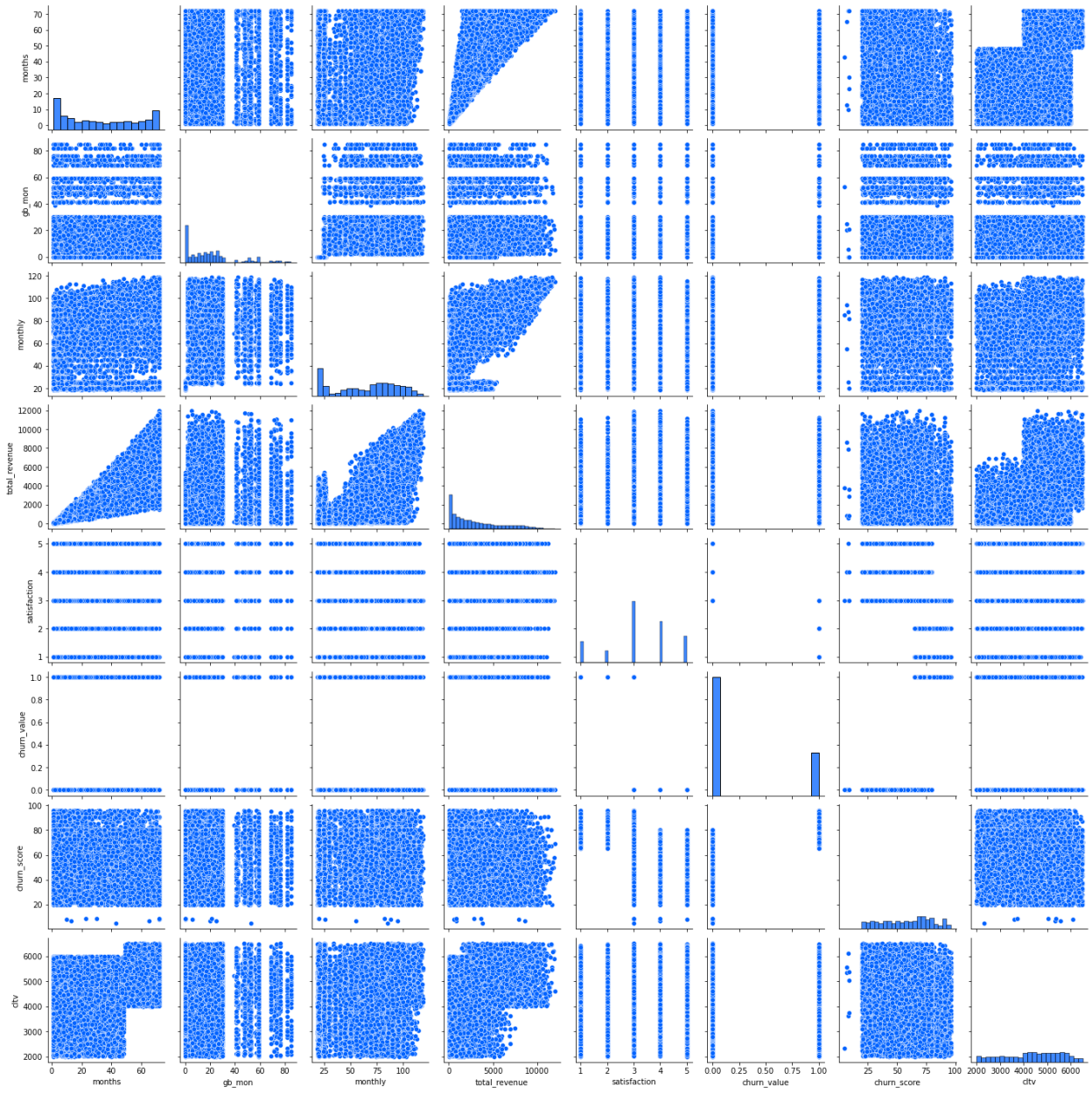

Figure 3.1.3. Pair plot of the churn phone data set.

3.2. Data cleaning and feature engineering

Data cleaning and feature engineering steps were, in this project, strongly related to the categories of the variables. Therefore, to take adequate steps with respect to the chosen data set, the same approach from the course laboratory on KNN algorithms was followed. In particular, the following was performed:

1. Unique values: the number of unique values of each variable was calculated using the `.unique()` Pandas method. The results are shown in Figure 3.2.1.

Variable	Unique Values
months	72
offer	6
multiple	2
internet_type	4
gb_mon	50
security	2
backup	2
protection	2
support	2
unlimited	2
contract	3
paperless	2
payment	3
monthly	1585
satisfaction	5
churn_value	2

Figure 3.2.1. Unique values of a selection of variables of the churn phone data set.

2. Classification of variables: following the same approach and criteria of the KNN laboratory, variables were separated into four groups: binary, categorical but not ordinal, categorical and ordinal and numeric.
    - Binary (8): variables with only 2 unique values (multiple, security, backup, protection, support, unlimited, paperless, churn_value).
    - Categorical but not ordinal (3): nominal categorical variables, which were considered to be those variables with a number of unique values in the range (2,6] and with no order just like in the course laboratory on KNN (offer, internet_type, payment).
    - Categorical and ordinal (3): from a list containing the 5 categorical variables whose number of unique values fell in the range (2,6], 2 variables were identified as ordinal (contract, satisfaction). Furthermore, the variable months, which could be treated as numerical, was instead added to this group of categorical and ordinal variables like in the laboratory.
    - Numeric (2): these are the remaining variables, which correspond to gb_mon (monthly gigabytes) and monthly. The histograms of these variables are shown in Figure 3.2.2.

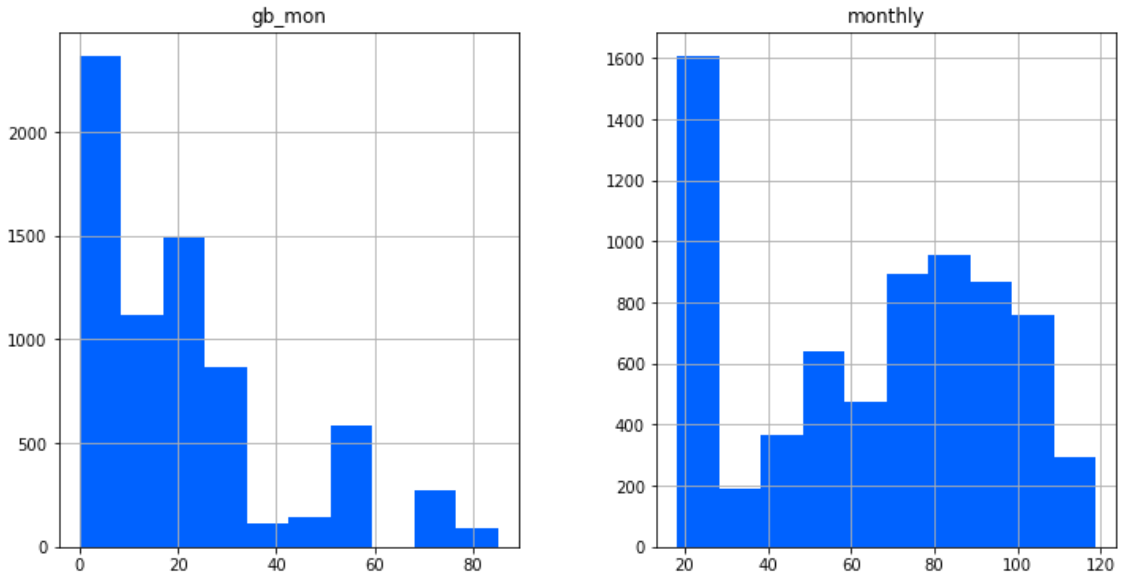

Figure 3.2.2. Histograms of the numerical variables of the churn phone data set.

3. Bins for the ordinal variable called months: like in the course laboratory (KNN), the ordinal variable months was split into 5 bins using the .cut Pandas function (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.describe.html) prior to encoding. As a result, the following 5 bins or categories were generated: (0.929, 15.2], (15.2, 29.4], (29.4, 43.6], (43.6, 57.8], (57.8, 72.0].
4. Encoding: encoding was performed according to feature type, following the same steps from the KNN laboratory.
    - Ordinal variables: used LabelEncoder() by Scikit-Learn.
    - Binary variables: used LabelBinarizer() by Scikit-Learn.
    - Categorical variables (nominal): used the .get_dummies() Pandas function with drop_first=True to avoid potential multicollinearity issues, which could negatively impact algorithms like Logistic Regression or KNN.
5. Scaling: like in the course laboratory, MinMaxScaler() by Scikit-Learn was used to scale numerical and ordinal features. This was performed to make all features belong to the range [0,1]. This was required since a KNN classifier, which is distance based and is thus affected by numerical scales, was used in this project and will be later presented in Section 4.

The description of the preprocessed data set is illustrated in Figure 3.2.3. This description was obtained using the .describe() Pandas method. As it can be observed, all features now fall in the range [0,1]. Furthermore, the resulting feature space, which was used to train the classifiers, contains 23 features instead of 16 like at the beginning.

	count	mean	std	min	25%	50%	75%	max
months	7043.0	0.43	0.40	0.0	0.00	0.25	0.75	1.0
multiple	7043.0	0.42	0.49	0.0	0.00	0.00	1.00	1.0
gb_mon	7043.0	0.24	0.24	0.0	0.04	0.20	0.32	1.0
security	7043.0	0.29	0.45	0.0	0.00	0.00	1.00	1.0
backup	7043.0	0.34	0.48	0.0	0.00	0.00	1.00	1.0
protection	7043.0	0.34	0.48	0.0	0.00	0.00	1.00	1.0
support	7043.0	0.29	0.45	0.0	0.00	0.00	1.00	1.0
unlimited	7043.0	0.67	0.47	0.0	0.00	1.00	1.00	1.0
contract	7043.0	0.38	0.42	0.0	0.00	0.00	1.00	1.0
paperless	7043.0	0.59	0.49	0.0	0.00	1.00	1.00	1.0
monthly	7043.0	0.46	0.30	0.0	0.17	0.52	0.71	1.0
satisfaction	7043.0	0.56	0.30	0.0	0.50	0.50	0.75	1.0
churn_value	7043.0	0.27	0.44	0.0	0.00	0.00	1.00	1.0
internet_type_DSL	7043.0	0.23	0.42	0.0	0.00	0.00	0.00	1.0
internet_type_Fiber Optic	7043.0	0.43	0.50	0.0	0.00	0.00	1.00	1.0
internet_type_None	7043.0	0.22	0.41	0.0	0.00	0.00	0.00	1.0
payment_Credit Card	7043.0	0.39	0.49	0.0	0.00	0.00	1.00	1.0
payment_Mailed Check	7043.0	0.05	0.23	0.0	0.00	0.00	0.00	1.0
offer_Offer A	7043.0	0.07	0.26	0.0	0.00	0.00	0.00	1.0
offer_Offer B	7043.0	0.12	0.32	0.0	0.00	0.00	0.00	1.0
offer_Offer C	7043.0	0.06	0.24	0.0	0.00	0.00	0.00	1.0
offer_Offer D	7043.0	0.09	0.28	0.0	0.00	0.00	0.00	1.0
offer_Offer E	7043.0	0.11	0.32	0.0	0.00	0.00	0.00	1.0

Figure 3.2.3. Description of the preprocessed churn phone data set.

4. CLASSIFICATION MODELS

In this section, the developed models will be addressed. In particular, 3 classifiers were developed using the following algorithms: KNN, Random Forest and Extra Random Trees. For all cases, 60% of the data set (4225 samples) was used for training while the remaining 40% (2818 samples) was used for testing purposes. For future models, it is recommended that a separate subset should be left aside for further evaluation of generalization capabilities, a topic which will be discussed later in this report.

Furthermore, it should be noted that when performing the split with train_test_split() by Scikit-Learn the data was stratified. This was decided since it was observed that approximately 73% of the data corresponded to samples labeled as "not churn" while the remaining observations were labeled as "churn". Consequently, maintaining this proportion was deemed necessary for proper training.

4.1. KNN classifier

For the KNN classifier, the steps from the course laboratory on KNN were followed with some changes. Therefore, initially a good value of K was sought. Toward this purpose, for $K \in [1, 40]$ KNN classifiers were created using the following configuration: KNeighborsClassifier(n_neighbors=K, weight='distance'). It should be noted that KNeighborsClassifier objects can be instantiated after importing KNeighborsClassifier from the sklearn.neighbors module. For each classifier, and on the test set, its values of F1-score, error (1-accuracy), precision and recall were saved.

In particular, for this business case, precision is considered to be highly important since it's relevant to guarantee, as much as possible, a high level of confidence when predicting the positive cases or "churn" labels. In other words, if a data point is predicted to belong to the positive class it's relevant that this is truly the case, so the quality of predictions should be high. The remaining metrics were considered to be of interest in the sense that these should be as high as possible to have a generally good quality classifier.

Once this iterative process was finished, results (the metrics against the values of K) were plotted using Matplotlib. These figures are shown in Figure 4.1.1 (F1-score), Figure 4.1.2 (error or the elbow curve), Figure 4.1.3 (precision) and Figure 4.1.4 (recall).

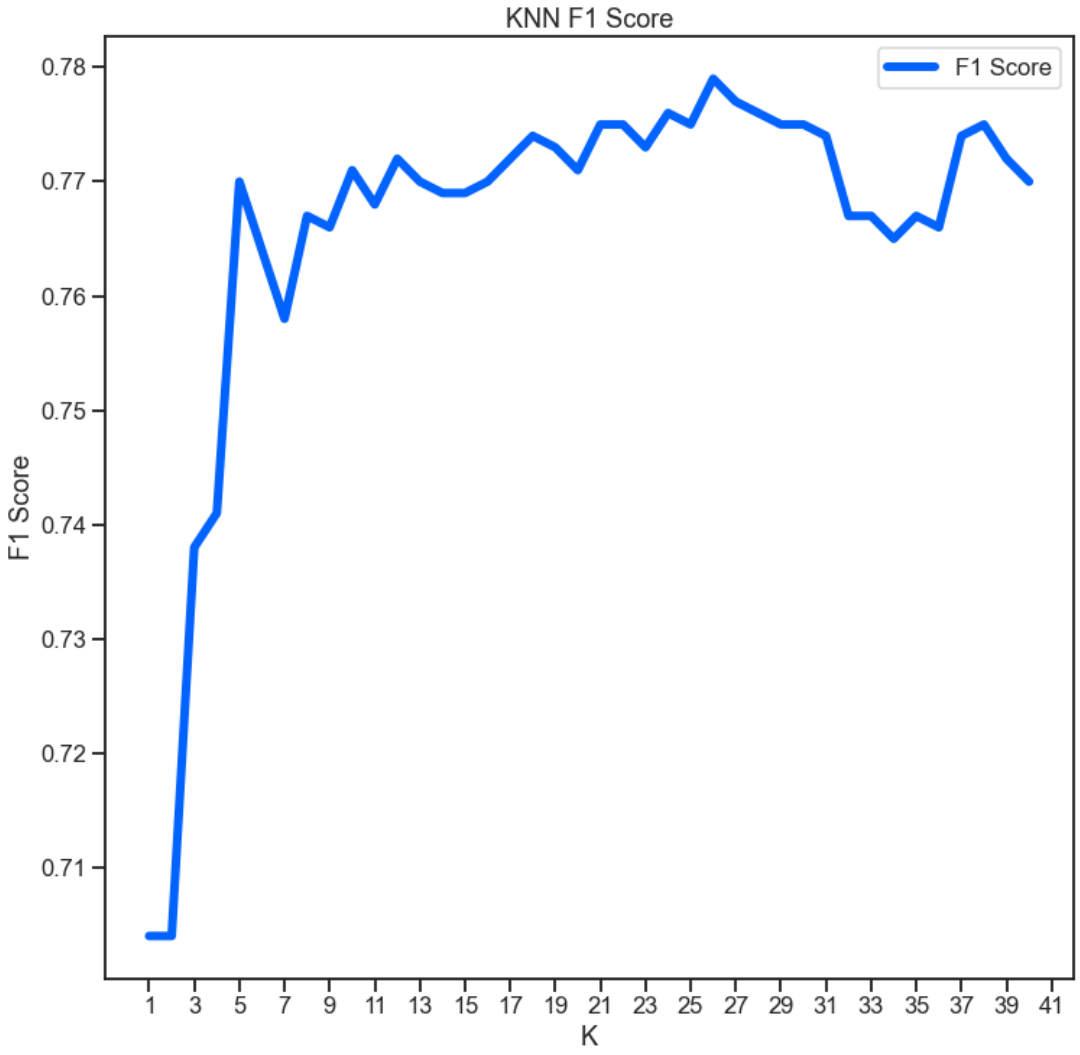

Figure 4.1.1. F1-scores for each of the considered values of K (test set).

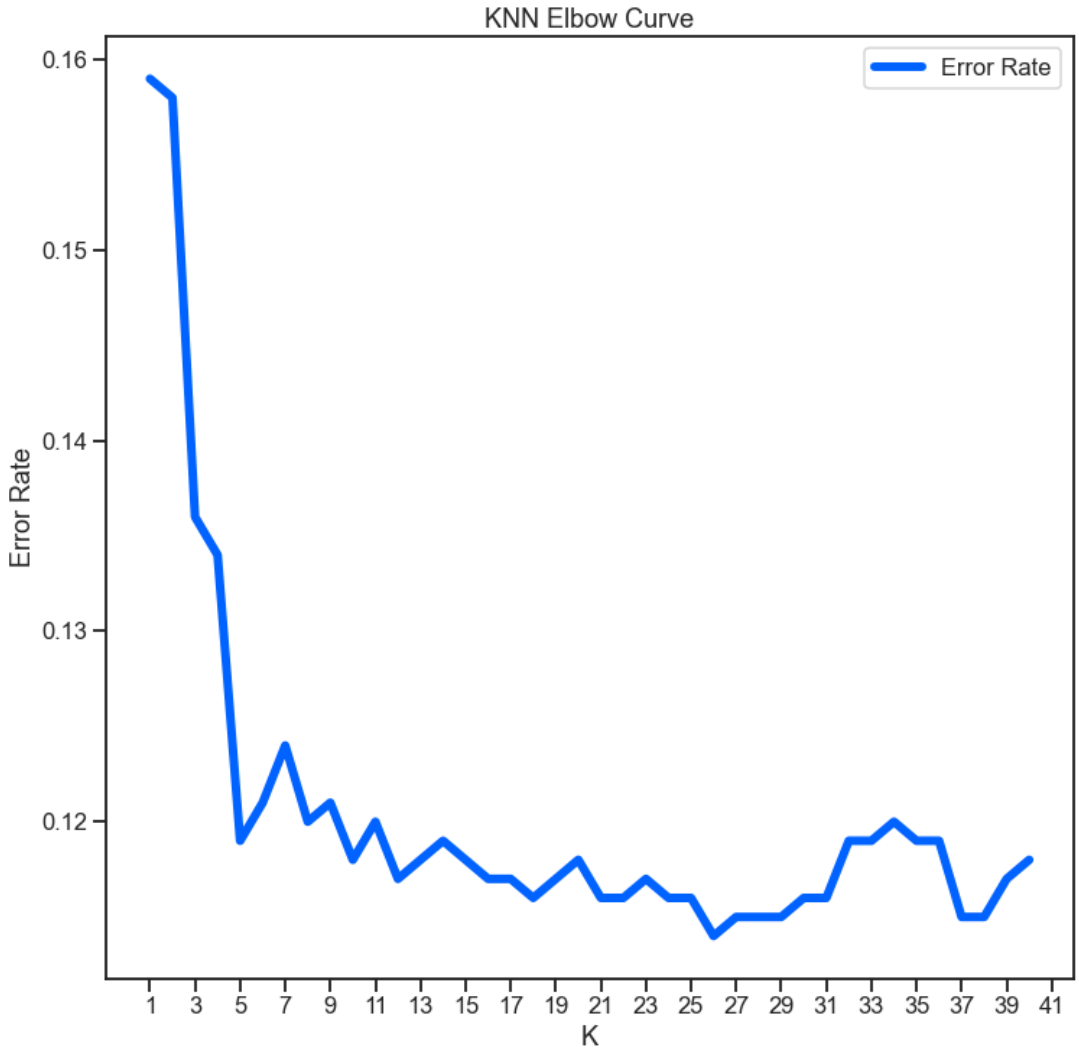

Figure 4.1.2. Error obtained for each of the considered values of K (test set).

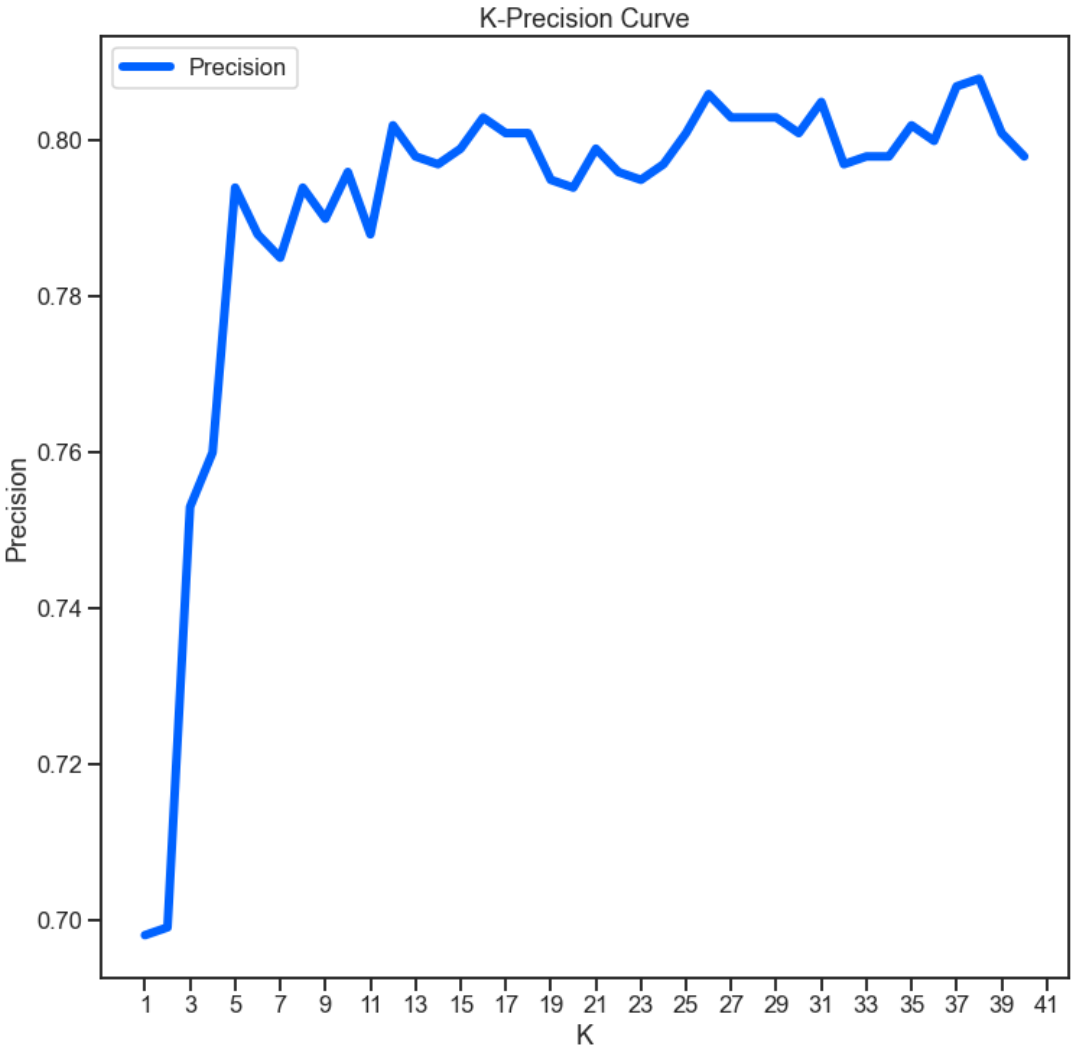

Figure 4.1.3. Precision for each of the considered values of K (test set).

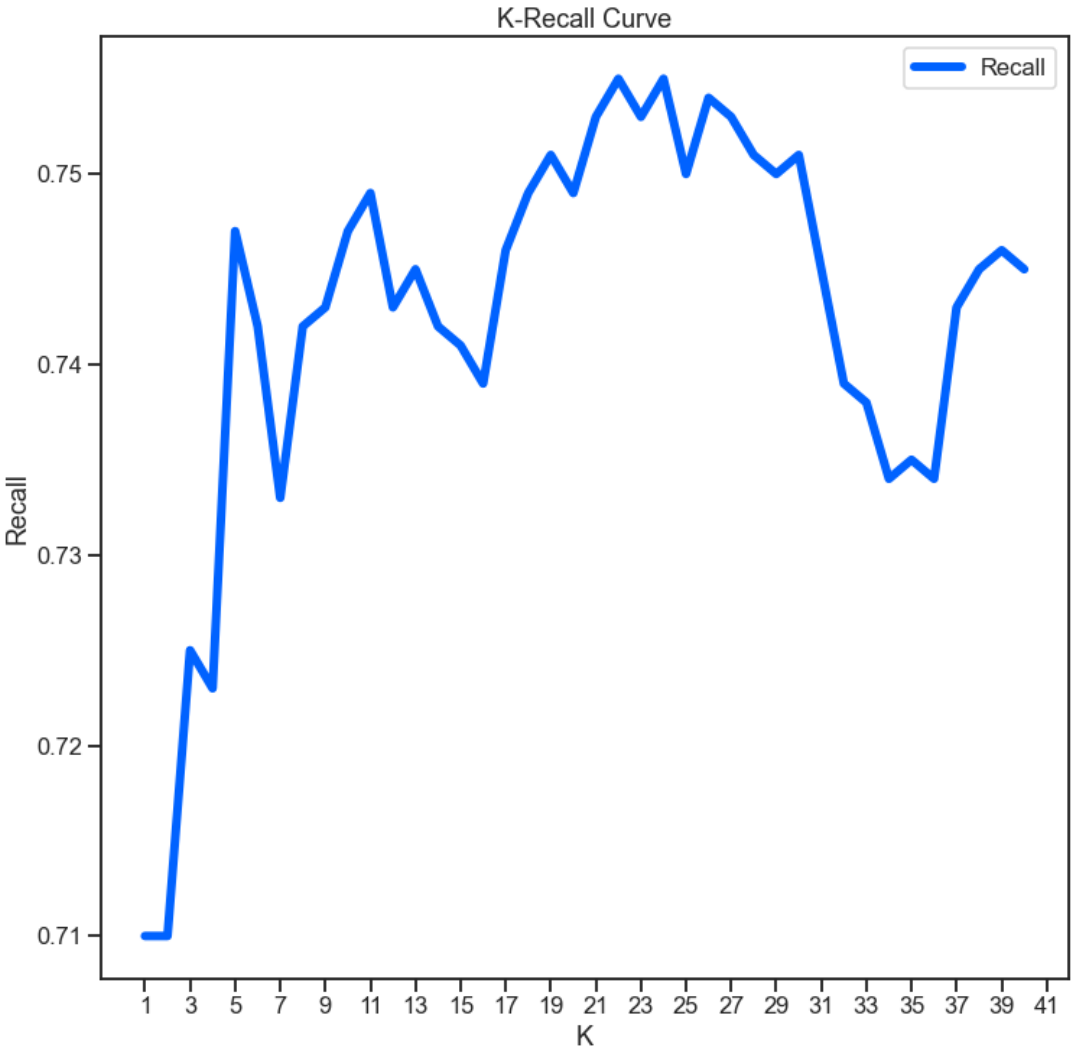

Figure 4.1.4. Recall for each of the considered values of K (test set).

In consequence, based on the results shown in Figure 4.1.1, Figure 4.1.2, Figure 4.1.3 and Figure 4.1.4, a value of K=26 was chosen. Therefore, a new KNN model was trained using this value of K and the same configuration which was indicated at the beginning of this section regarding the weights parameter. Due to the fact that the data set is not balanced, the classification report, confusion matrix and precision-recall curve were obtained. These were gathered using the approach from the laboratory on KNN. The classification report is illustrated in Table 4.1.1, the confusion matrix in Figure 4.1.5 and the precision-recall curve in Figure 4.1.6.

Class or metric	Precision	Recall	F1-score	Support
0 (“Not churn”)	0.91	0.93	0.92	2070
1 (“Churn”)	0.80	0.74	0.77	748
Accuracy	-	-	0.88	2818
Macro average	0.85	0.84	0.85	2818
Weighted average	0.88	0.88	0.88	2818

Table 4.1.1. KNN classifier with K=26: classification report as elaborated by Scikit-Learn (test set).

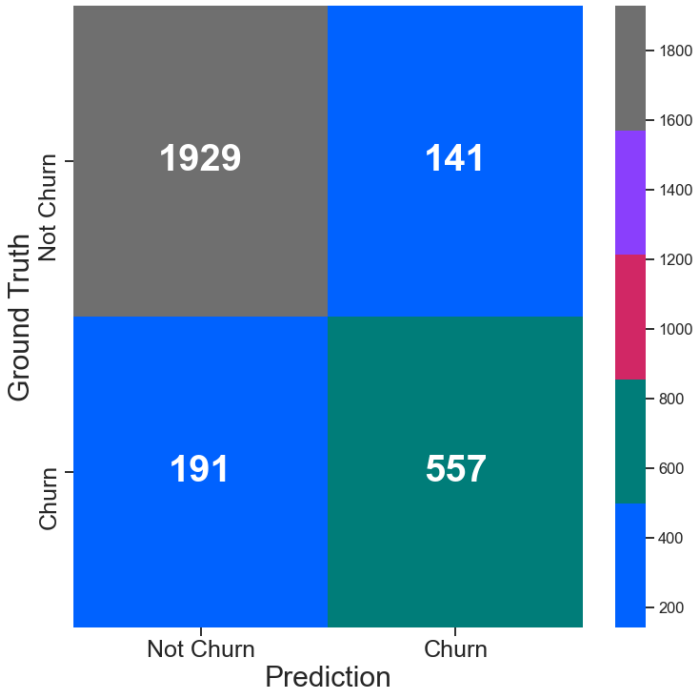

Figure 4.1.5. KNN classifier with K=26: confusion matrix as elaborated by Scikit-Learn (test set).

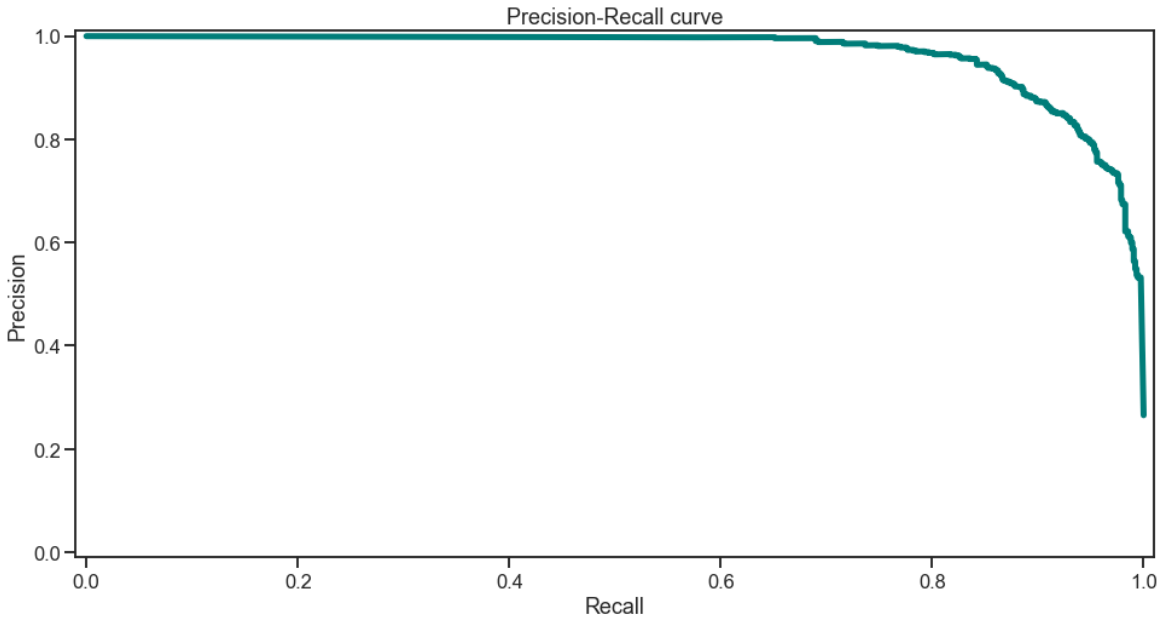

Figure 4.1.6. KNN classifier with K=26: precision-recall curve (test set).

4.2. Random Forest classifier

For the Random Forest classifier, the steps from the course laboratory on Bagging were followed with some modifications. Similarly to the laboratory, the first step was to calculate the Out-of-bag (OOB) error for Random Forest classifiers with different numbers of Decision Trees. In consequence, and iteratively, Random Forest models were developed with a number of trees equal to `n_estimators` (a parameter of the class). In this project, this parameter could take a value in the following array: [15, 20, 30, 40, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800] (the integers in this array were selected based on iterative testing).

Furthermore, said classifiers used the following configuration: `RandomForestClassifier(oob_score=True, random_state=42, warm_start=True, n_jobs=-1)` to optimize resources and speed up the training process. It should be noted that `RandomForestClassifier` objects can be instantiated after importing `RandomForestClassifier` from the `sklearn.ensemble` module. In Appendix A, a `Pandas DataFrame` with the results is shown. Based on these results, it was observed that a good value for the number of Decision Trees ranged from 200-300 trees. Consequently, a middle value was selected, in particular 250 trees.

Specifically, to do so, after the iterative training process was finished, the following was done: `RF.set_params(n_estimators=250)`. Here, `RF` corresponds to the last Random Forest object of the training loop where multiple classifiers were trained with different numbers of trees. Afterward and just like for the KNN classifier, the classification report, confusion matrix and precision-recall curve were obtained using the same approach from Section 4.1. These are shown in Table 4.2.1, Figure 4.2.1 and Figure 4.2.2 respectively.

Class or metric	Precision	Recall	F1-score	Support
0 (“Not churn”)	0.96	0.98	0.97	2070
1 (“Churn”)	0.94	0.89	0.92	748
Accuracy	-	-	0.96	2818
Macro average	0.95	0.94	0.94	2818
Weighted average	0.96	0.96	0.96	2818

Table 4.2.1. Random Forest model (250 trees): classification report as elaborated by Scikit-Learn (test set).

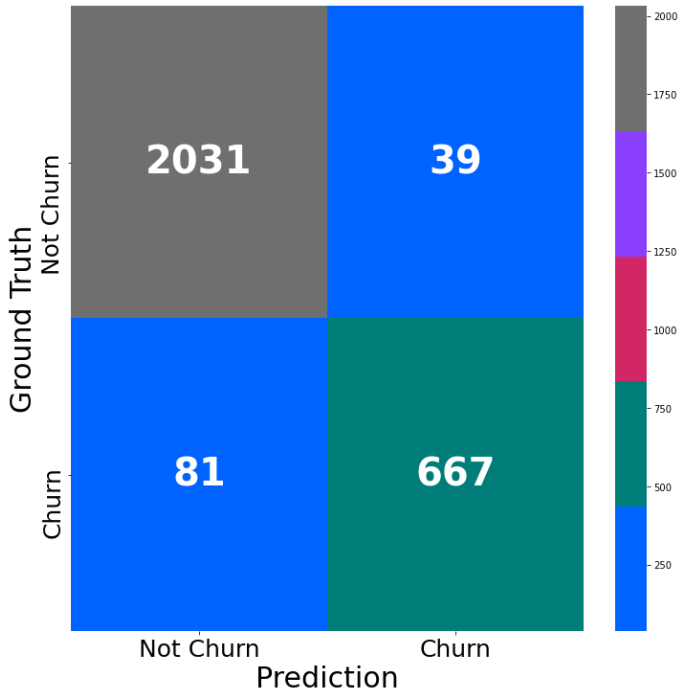

Figure 4.2.1. Random Forest model (250 trees): confusion matrix as elaborated by Scikit-Learn (test set).

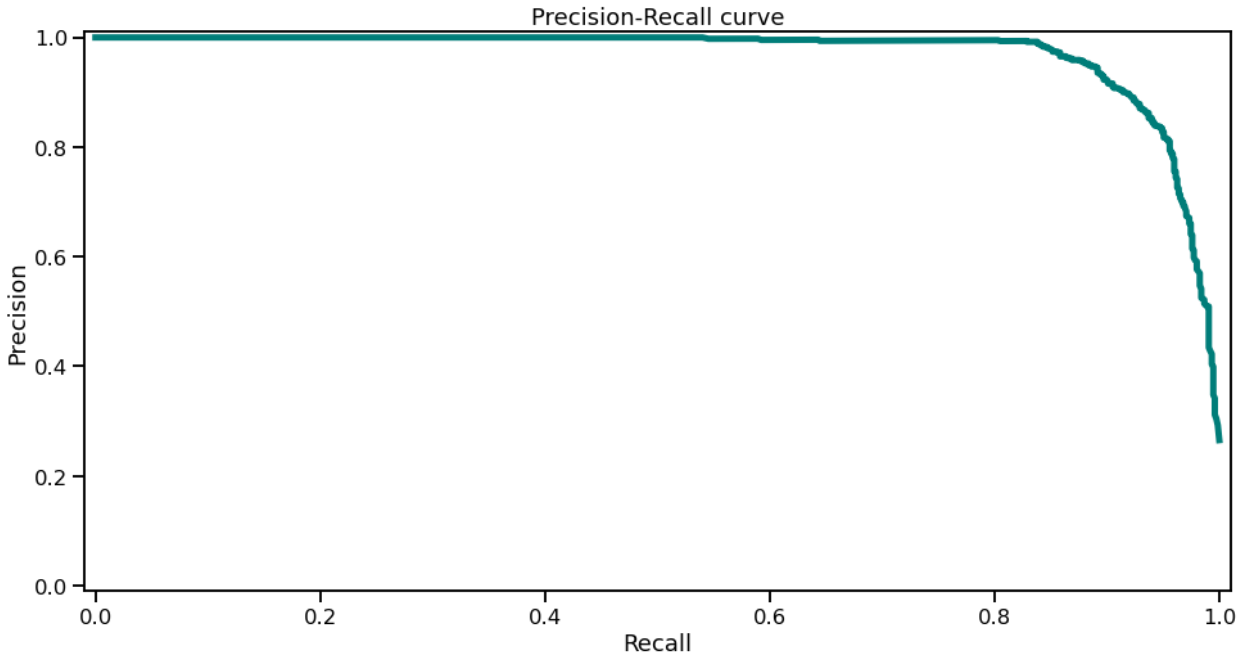

Figure 4.2.2. Random Forest model (250 trees): precision-recall curve (test set).

In addition, for this algorithm it was possible to obtain the feature importance, just like in the laboratory on Bagging. Similarly to the course notebook, the results were plotted using the Pandas `.plot()` function. The results are shown in Figure 4.2.3. As the figure shows, the most important feature, which is notably more important than the rest, is `satisfaction`, which makes sense since it would be expected for highly satisfied customers to stay. This would also very likely imply that the company is providing good services to its customers.

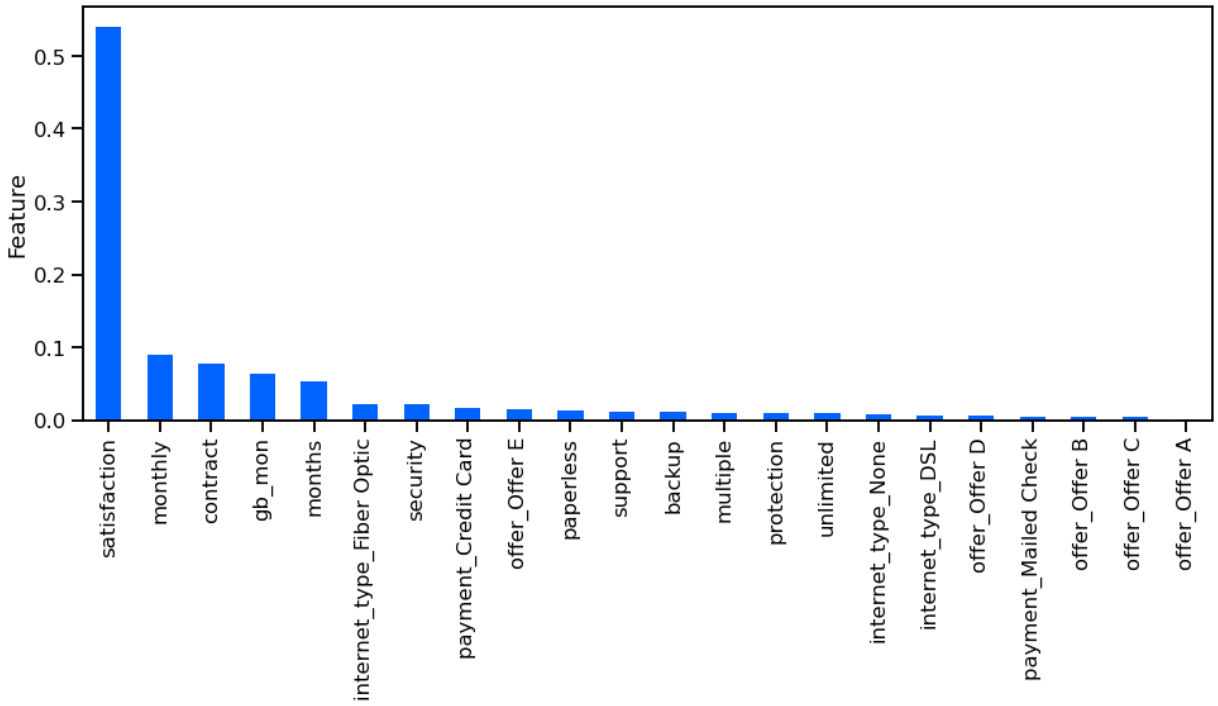

Figure 4.2.3. Random Forest model (250 trees): feature importance.

4.3. Extra Random Trees classifier

Similarly to the Random Forest classifier, the first step also consisted on calculating the Out-of-bag (OOB) error for different Extra Random Trees classifiers with different numbers of Decision Trees. Consequently, various models were developed with a number of trees equal to n_estimators (a parameter of the class). Once again, this parameter could take a value in the array: [15, 20, 30, 40, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800] (the same array was chosen to properly compare this model to the Random Forest one).

Moreover, the Extra Random Trees classifiers used the following configuration: ExtraTreesClassifier(oob_score=True, random_state=42, warm_start=True, boostrap=True, n_jobs=-1) to speed up the training process once more. It should be highlighted that ExtraTreesClassifier objects can be instantiated after importing ExtraTreesClassifier from the sklearn.ensemble module. In Appendix B, a Pandas DataFrame with the results is illustrated. Based on these results, a good value for the number of Decision Trees for this case ranged from 600-750 trees. Specifically, the value 600 was chosen to reduce complexity.

Furthermore, prior to training the final Extra Random Trees model, just like in the laboratory on Bagging the OOB errors of the Random Forest and Extra Random Trees classifiers were compared. A Pandas DataFrame that compares both algorithms is shown in Appendix C. It can be seen that the best performing algorithm, with respect to the training set of this project and the chosen range of trees, corresponds to the Random Forest algorithm.

After this step, similarly to what was described in Section 4.2, the following was performed: EF.set_params(n_estimators=600). In this case, EF corresponds to the last Extra Trees model of the training loop where various Extra Trees classifiers were trained (with different numbers of trees). Then, like for the KNN and Random Forest classifiers, the classification report, confusion matrix and precision-recall curve were obtained using the approach of Section 4.1. These are shown in Table 4.3.1, Figure 4.3.1 and Figure 4.3.2 respectively.

Class or metric	Precision	Recall	F1-score	Support
0 (“Not churn”)	0.96	0.97	0.97	2070
1 (“Churn”)	0.92	0.89	0.91	748
Accuracy	-	-	0.95	2818
Macro average	0.94	0.93	0.94	2818
Weighted average	0.95	0.95	0.95	2818

Table 4.3.1. Extra Random Trees model (600 trees): classification report as elaborated by Scikit-Learn (test set).

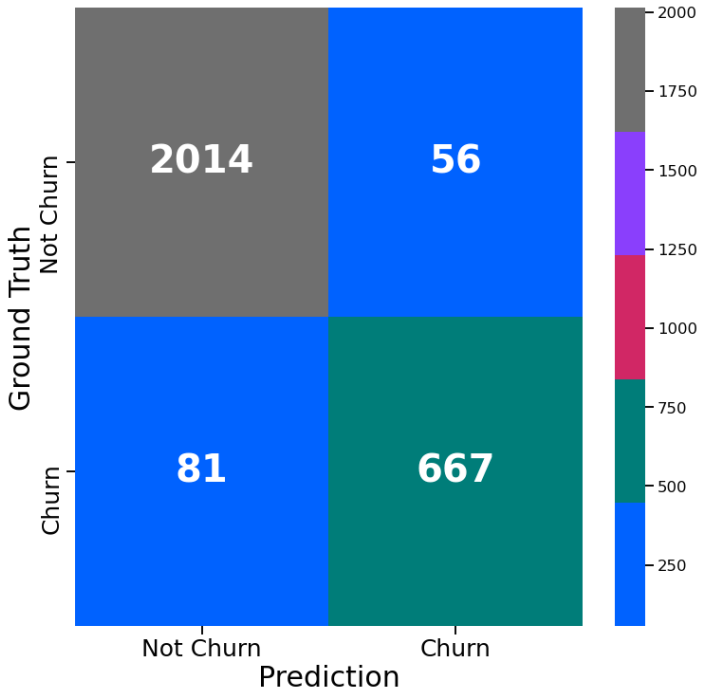

Figure 4.3.1. Extra Random Trees model (600 trees): confusion matrix as elaborated by Scikit-Learn (test set).

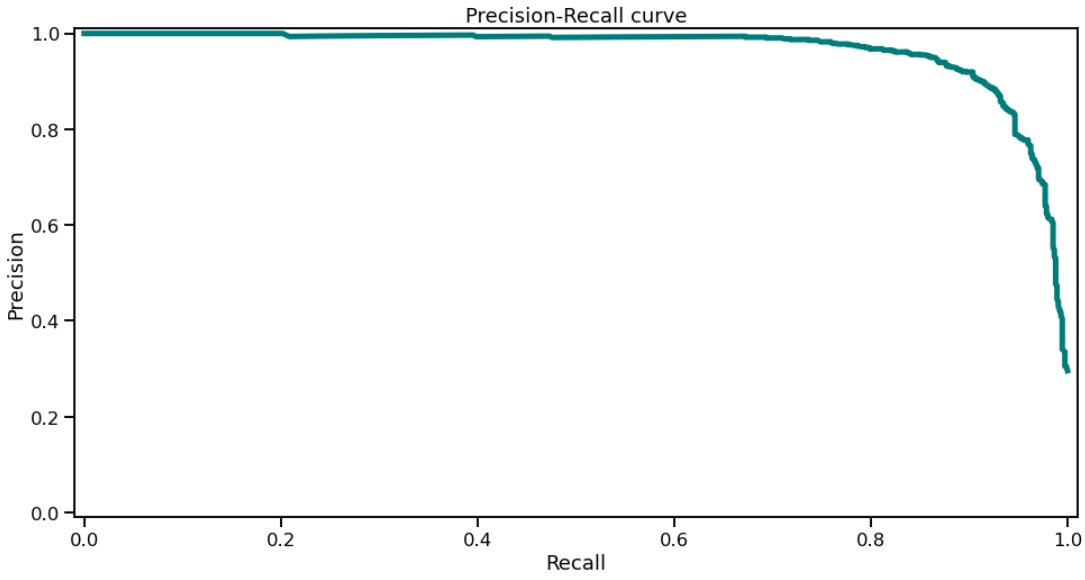

Figure 4.3.2. Extra Random Trees model (600 trees): precision-recall curve (test set).

Additionally, like for Random Forest, for this algorithm it was possible to obtain the feature importance. As described in Section 4.2, the results were plotted using the Pandas `.plot()` function and are illustrated in Figure 4.3.3. As shown, the most important feature is satisfaction once again.

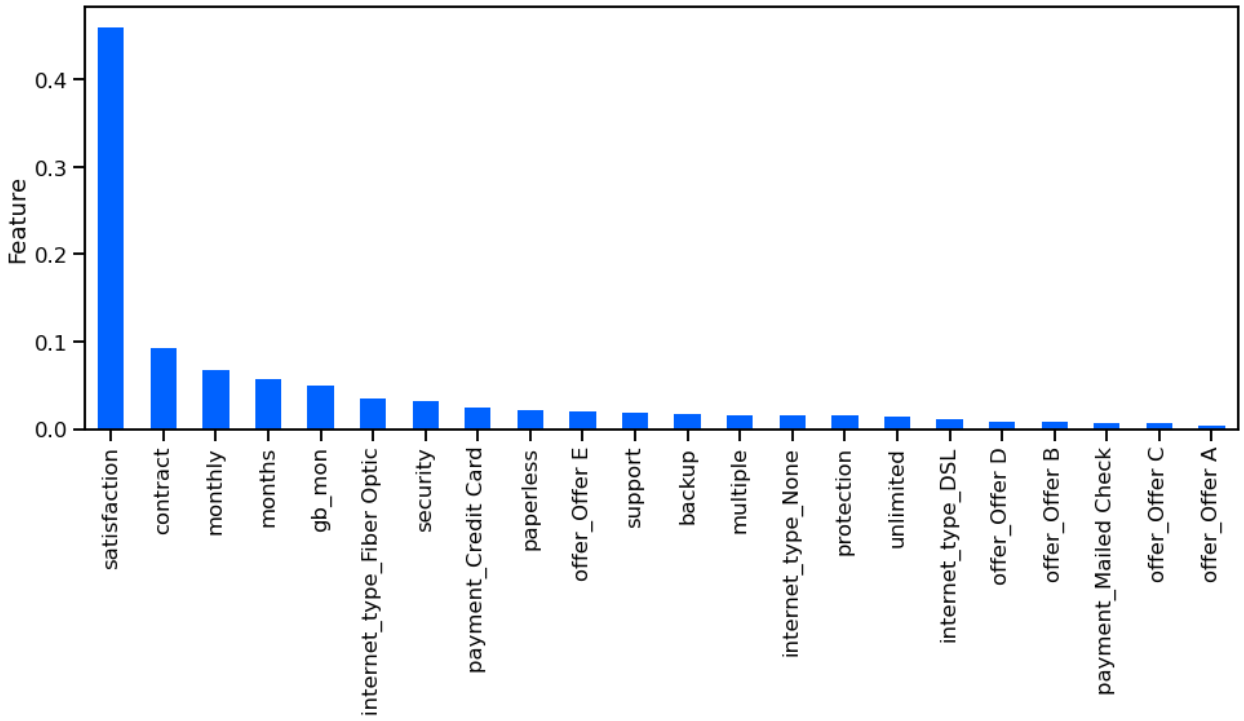

Figure 4.3.3. Extra Random Trees model (600 trees): feature importance.

4.4. Model comparison

Table 4.4.1 shows a summary and comparison of the results of the 3 developed models. In particular, the table focuses on the resulting values of precision, recall, F1-score and accuracy with respect to the test set. Note that Table 4.4.1 includes the precision, recall and F1-score by class. Since accuracy is not a metric which is calculated by class but rather for the classifier, the value was repeated for each algorithm and for both classes only due to the way the table was designed.

Classifier	Class	Precision	Recall	F1-score	Accuracy
KNN	0	0.91	0.93	0.92	0.88
KNN	1	0.80	0.74	0.77	0.88
Random Forest	0	0.96	0.98	0.97	0.96
Random Forest	1	0.94	0.89	0.92	0.96
Extra Trees	0	0.96	0.97	0.97	0.95
Extra Trees	1	0.92	0.89	0.91	0.95

Table 4.4.1. Developed classifiers: comparison of key results (test set).

Regarding precision, which is considered to be highly important in this analysis, the Random Forest classifier achieved the highest value for both classes. Specifically, it scored a precision of 96% for Class 0 (“not churn”) and 94% for Class 1 (“churn”). It also achieved the highest accuracy (96%), the highest recall values (98% for Class 0 and 89% for Class 1) and F1-scores (97% for Class 0 and 92% for Class 1). The second best performing model corresponds to the Extra Random Trees classifier, whose results are higher than the results of the KNN model. Moreover, the results in Table 4.4.1 are consistent with the confusion matrices and all precision-recall curves showed, overall, well performing behavior. A deeper discussion of these models will be presented later in this report.

5. DISCUSSION AND MODEL RECOMMENDATION

Based on the results reported in Table 4.4.1 in Section 4.4, the recommended model corresponds to the Random Forest classifier since it obtained the highest values for all metrics on the test set. In particular, with respect to the objectives of this project regarding predictability and explainability, this model also surpassed the minimum threshold of a precision of 80%, a recall of 70% and an accuracy of 70%. In fact, all results were either close to 90% or above this value. Moreover, all models met these minimum requirements so the best performing one can be suggested for further development in this project.

However, it should be noted that to properly evaluate the generalization capabilities of the suggested model more data is required. The test set used in this project in reality serves the purpose of a validation set, where the data samples are kept constant and the results of different models are compared. Therefore, the suggested Random Forest model should be

understood as a basis for a future model optimization and evaluation phase, which will allow for a better estimation of classifier performance in production or in a business setting.

Another aspect to consider when only comparing models using a single test set (or truly validation set in this case) is the possible issue of overfitting to the validation or test data. Nonetheless, based on the available data set, the chosen classifier appears promising. Finally, it is interesting to note that the lowest performing metric in all cases corresponds to the recall of the positive class (or samples labeled as "churn"). This topic will be addressed in Section 6 and further in Section 7 since it is related to possible next steps to improve this analysis.

6. KEY FINDINGS AND INSIGHTS

In summary, after performing this analysis, the key findings and insights are:

- Considering the results in Table 4.4.1 and the 3 chosen algorithms (KNN, Random Forest, Extra Trees), it would appear like ensemble models like Random Forest and Extra Random Trees are able to better learn from the features of this preprocessed data set.
    - Specifically, and with respect to the particular objectives of this analysis, in this context this means that ensemble methods based on Decision Trees are capable of achieving higher quality prediction. Furthermore, it means that this is achieved with high precision and satisfactory values of recall and accuracy.
        - It should be noted that since the data set is imbalanced, accuracy is not the best metric for this analysis. However, it is desirable to have an overall good classifier with a good enough level of accuracy.
    - Moreover, all models scored higher than the minimum target thresholds established for this analysis, which as summarized before correspond to a precision of 80%, a recall of 70% and an accuracy of 70%. However, Random Forest was the best performing classifier on the test set, which was used for validation purposes in this project.
        - The overall good performance of the models is supported by the confusion matrices and the precision-recall curves, which showed satisfactory behavior.

        - Between Random Forest and Extra Trees, as shown by the results on the test set (Table 4.4.1) and the OOB error (Appendix C), Random Forest appears like the better choice for this case since it performed better than the Extra Trees algorithm.
    - Additionally, the ensemble models also present a good degree of explainability and are easy to understand. For instance, the bar plots about feature importance (Figure 4.2.3 and Figure 4.3.3) revealed that the most important feature to determining customer churn is the level of satisfaction of the customers, which is reasonable.
    - As highlighted in the previous section, it will be necessary to use additional, separate data to evaluate model generalization performance.
        - In this sense, the chosen model (Random Forest) should be considered as a basis for future model optimization phases.
- Based on the results of this project, although some additional steps could be taken it would appear like the chosen preprocessing steps, including the way variables were classified, were appropriate.
    - These additional steps will be described in the next section.
    - As observed in Figure 3.1.3 (pair plot before preprocessing), the data set shows some linear dependencies and skewed distributions.
- As emphasized in the previous section, the recall of the positive class is the worst performing metric in this whole analysis. This could be due to class imbalance, a topic which will be addressed in the next section.
    - Considering that one of the purposes of this project was to look for an initial, satisfactory classifier, the results obtained so far are good enough. In particular, it appears like the chosen algorithms are able to properly learn from the data after using stratification to maintain class proportions. Consequently, this approach should be repeated in the future.

7. NEXT STEPS

As mentioned before, the suggested Random Forest classifier should be considered as a first step toward developing an even better model with more accurately estimated generalization

performance. Some next steps toward achieving this purpose, with respect to the data set of this project, include:

- Increase the data set to include a separate test set.
    - Since it resulted in promising results, the stratification approach used in this project should be maintained as already highlighted.
    - Retain the “test set” used in this project as a validation set to compare and validate model performance as done in this analysis.
    - A good proportion corresponds to a test set which is roughly equal in size to the validation set.
    - Acquiring and using new data samples, which should be preprocessed in the same way as the training and validation sets, will allow for proper estimation of generalization performance and possible overfitting issues (variance of the models).
        - This aspect is a potential weakness or limitation of the suggested model (Random Forest) since a separate data subset was not left aside to study this.
        - It will also be possible to study the learning curves of the models during training to better assess any potential overfitting behavior.
- Similarly, to address class imbalance further and improve training quality, it could be a good idea to experiment further by increasing the number of samples of the positive class. However, this should be done to an appropriate extent to make sure the model trains under relatively realistic conditions with respect to class proportions. This topic is an aspect of the models of this project which can be improved.
- It could be interesting to study the performance of other types of algorithms, such as Logistic Regression, Boosting and/or Stacking.
- The data set can be further preprocessed; this could potentially improve performance.
    - As mentioned earlier, the behavior (skewness and dependencies) observed in the pair plot (Figure 3.1.3) should be further addressed when appropriate depending on the chosen algorithm(s). For instance, it might be necessary to transform certain features depending on the case.
        - Similarly, depending on the algorithm, after the preprocessing phase it would be appropriate to obtain the pair plot of the new, preprocessed features to check for algorithm related assumptions.

- It could also be a good idea to check for outliers using box plots, since algorithms like KNN are affected by outlier behavior. Furthermore, it will be necessary to determine whether any potential outliers are truly outliers (regarding value).
    - If these outliers cannot be replaced by a given value or eliminated from the data set, perhaps considering other scalers like the RobustScaler by Scikit-Learn would be a good idea.
- Potentially, studying correlation coefficients like in the course (for instance, with a correlation matrix) could be useful if attempting to improve the explainability of the models in future analyses. Of course, this is assuming this aspect becomes an objective of any future analysis related to this project. This could also impact the predictive capabilities of the models.
    - Considering this, it will be required to first check the assumptions necessary for the calculation of said correlation coefficients. A good source which explains these assumptions for the case of the Pearson correlation coefficients, like the absence of outliers, can be found in (Zach, 2021).
    - Furthermore, to this end, it might be necessary to carry out appropriate hypothesis testing to check for normality.

REFERENCES

Zach. (November 17, 2021). The Five Assumptions for Pearson Correlation [Blog entry]. Statology. https://www.statology.org/pearson-correlation-assumptions/

APPENDICES

Appendix A. Random Forest classifiers: OOB error

Figure A.1 shows a Pandas DataFrame with the OOB error of several Random Forest classifiers with different numbers of trees. This topic was addressed in Section 4.2.

n_trees	oob
15.0	0.060118
20.0	0.056331
30.0	0.054675
40.0	0.051834
50.0	0.052308
100.0	0.051124
150.0	0.051598
200.0	0.050888
250.0	0.050888
300.0	0.050178
350.0	0.051361
400.0	0.050888
450.0	0.051361
500.0	0.049941
550.0	0.050414
600.0	0.050178
650.0	0.050651
700.0	0.049941
750.0	0.050651
800.0	0.050651

Figure A.1. Random Forest classifiers: Out-of-bag error for different numbers of trees (training set).

Appendix B. Extra Random Trees classifiers: OOB error

Figure B.1 shows a Pandas DataFrame with the OOB error of several Extra Trees classifiers with different numbers of trees. This topic was addressed in Section 4.3.

n_trees	oob
15.0	0.077633
20.0	0.069822
30.0	0.066036
40.0	0.066509
50.0	0.062959
100.0	0.059172
150.0	0.056568
200.0	0.056095
250.0	0.053491
300.0	0.053964
350.0	0.054201
400.0	0.053728
450.0	0.053491
500.0	0.053018
550.0	0.052308
600.0	0.051598
650.0	0.051834
700.0	0.051598
750.0	0.051598
800.0	0.052544

Figure B.1. Extra Random Trees classifiers: Out-of-bag error for different numbers of trees (training set).

Appendix C. Random Forest and Extra Random Trees classifiers: OOB error

Figure C.1 shows a Pandas DataFrame with the OOB error of several Random Forest and Extra Trees classifiers with different numbers of trees for comparison purposes. This topic was addressed in Section 4.3.

n_trees	RandomForest	ExtraTrees
15.0	0.060118	0.077633
20.0	0.056331	0.069822
30.0	0.054675	0.066036
40.0	0.051834	0.066509
50.0	0.052308	0.062959
100.0	0.051124	0.059172
150.0	0.051598	0.056568
200.0	0.050888	0.056095
250.0	0.050888	0.053491
300.0	0.050178	0.053964
350.0	0.051361	0.054201
400.0	0.050888	0.053728
450.0	0.051361	0.053491
500.0	0.049941	0.053018
550.0	0.050414	0.052308
600.0	0.050178	0.051598
650.0	0.050651	0.051834
700.0	0.049941	0.051598
750.0	0.050651	0.051598
800.0	0.050651	0.052544

Figure C.1. Random Forest and Extra Random Trees classifiers: Out-of-bag error for different numbers of trees (training set).