UNIVERSITÀ DI PISA

Master’s degree in Artificial Intelligence and Data Engineering

Data Mining and Machine Learning project documentation

CHURNPREDICT

Denny Meini

INDICE

1. INTRODUCTION 3

2. DATASET 4

3. PREPROCESSING 7

4. CLASSIFICATION 10

5. IMPLEMENTATION OF THE APPLICATION 15

6. CONCLUSIONS 18

1. INTRODUCTION

ChurnPredict is an application which permits the user, an Ecommerce holder, to know if a customer is likeable or not to leave the Ecommerce.

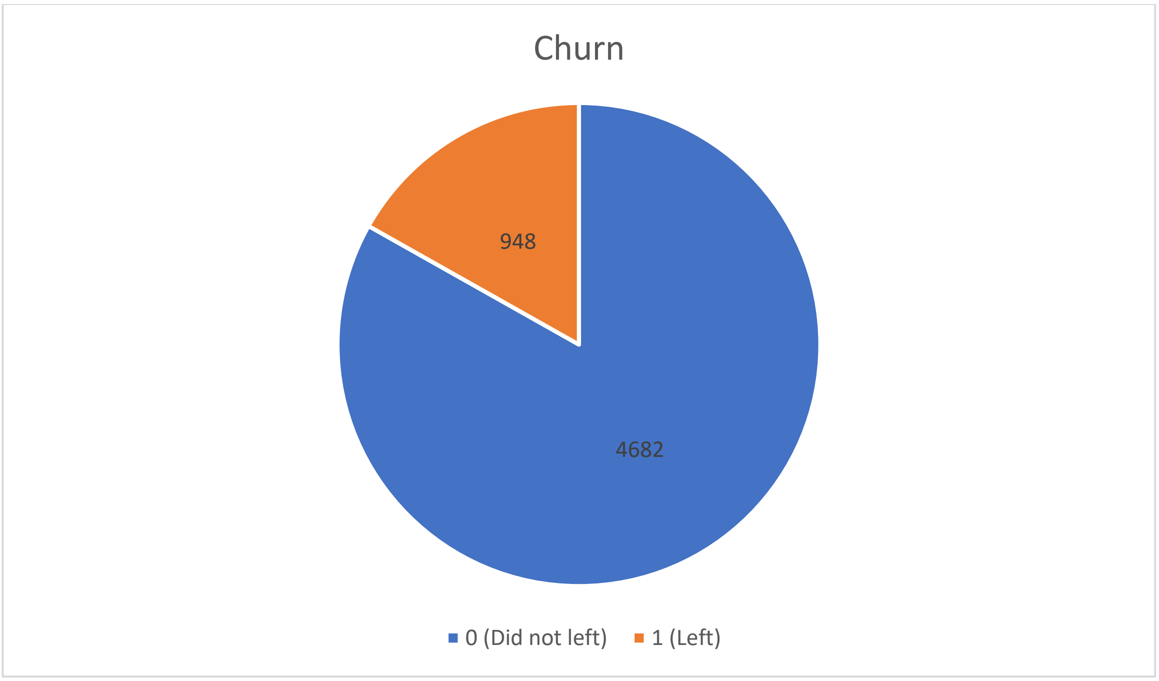
The main idea behind this project is to compare classifiers of different types, in order to evaluate their performance.

Before writing the application, I made some studies on the dataset, with the goal of preparing it to the classification and to use it with some classifiers. I compared the results I obtained, then, for my application, I used the classifier which gave me the best performances in the study phase.

The final application will give the user the possibility to insert customer’s data in the specified field and will return on the screen whether there is a risk or not that the customer will leave the Ecommerce.

2. DATASET

The dataset I chose for this project comes from Kaggle, and it is named “Ecommerce Customer Churn Analysis and Prediction”. It has 5630 rows, each of them represents a customer. Each row has 20 attributes, 19 of them are data relative to the customer, like tenure, preferred payment method, order count. The last feature is a binary attribute which tells us whether or not the customer has left the Ecommerce.



As we can see from the graphic, the dataset is not balanced, with a division that is 83.2%/16.8%. In fact I had to apply a rebalancing method before the classification phase.

The 20 features of the dataset are:

1. **CustomerID**
   The ID of the customer.

2. **Churn**
   Output class [0: did not left, 1 left].

3. **Tenure**
   The number of years the customer has been a customer.

4. **PreferredLoginDevice**
   The device the customer has used the most to access the Ecommerce.

5. **CityTier**
   The tier of the city (Chinese model) [1: big, 2: medium, 3: small].

6. **WarehouseToHome**
   Distance between the warehouse and the customer's house.

7. **PreferredPaymentMode**
   Customer’s preferred method of payment.

8. **Gender**
   The gender of the customer [Female/Male].

9. **HourSpendOnApp**
   Number of hours the customer has spent on the app.

10. **NumberOfDeviceRegistered**
    Number of devices the customer registered.

11. **PreferredOrderCat**
    The preferred item category of the customer based on his/her orders

12. **SatisfactionScore**
    Grade of satisfaction of the customer from 1 (not satisfied) to 5 (very satisfied).

13. **MaritalStatus**
    Marital status of the customer [Married/Single/Divorced].

14. **NumberOfAddress**
    Number of address added by a customer.

15. **Complain**
    Whether the customer complained or not during the last month [0: no, 1: yes].

16. **OrderAmountHikeFromlastYear**
    Percentage increase regarding the orders from the previous year.

17. **CouponUsed**
    Number of coupon the customer used in last month.

18. **OrderCount**
    Number of order the customer placed during last month.

19. **DaySinceLastOrder**
    Number of days since the customer’s last order.

20. **CashbackAmount**
    Average cashback of the customer during the last month.

What is interesting to see is that the dataset presents some null values.

3. PREPROCESSING

The preprocessing phase is composed by several operations, which have the goal of cleaning and adjusting the dataset in order to make it ready to the classification phase.

I. First analysis and division of the dataset

I started with an analysis of the values of the column. I found that there were no evident outliers in the dataset (eg. OrderCount<1), but there were some values in the categorical featured that was written in different ways even if they represent the same value:

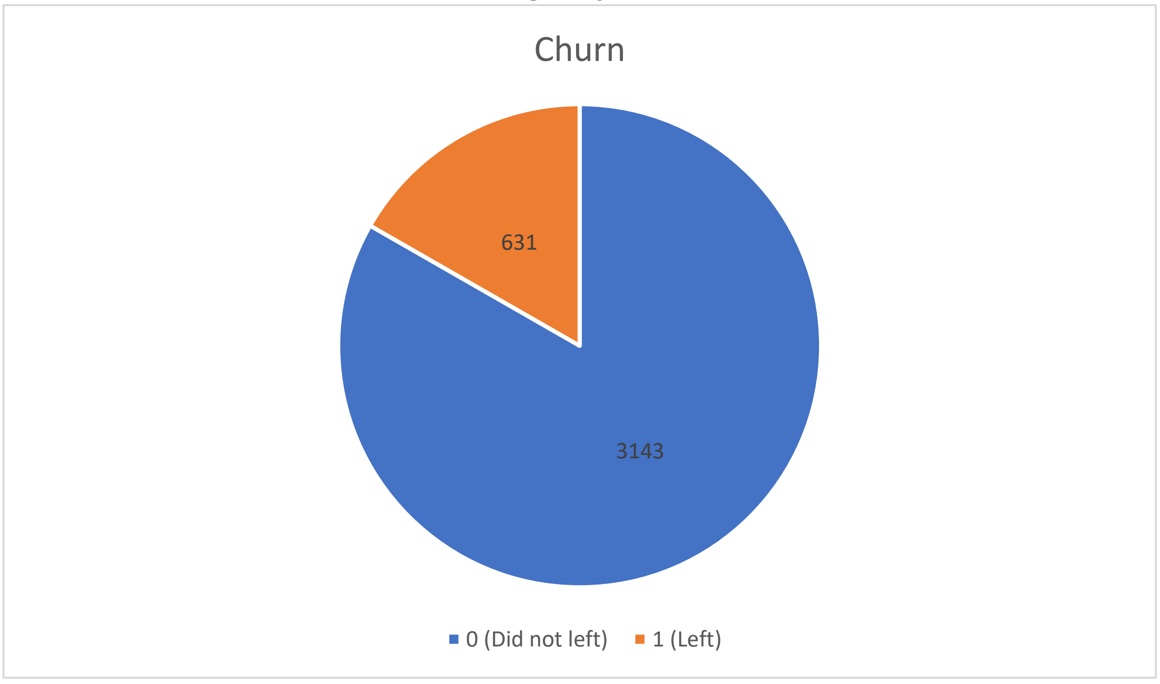
- 'Mobile phone' and 'Phone' in PreferredLoginDevice
- 'CC' and 'Credit Card' in PreferredPaymentMode
- 'COD' and 'Cash on Delivery' in PreferredPaymentMode
- 'Mobile' and 'Mobile Phone' in PreferedOrderCat

As a consequence I had to fix this problem changing the values "Phone" and "Mobile" in "Mobile Phone", "CC" in "Credit Card" and "COD" in "Cash on Delivery".

Then I faced the problem of the null values and I saw that there were two possible ways:

- Dropping the records that contained a null value
- Filling the null value

I decided to use both ways and to compare them in the classification phase. By dropping the record with null values the dataset was reduced to 3774 rows, distributed in the following way:

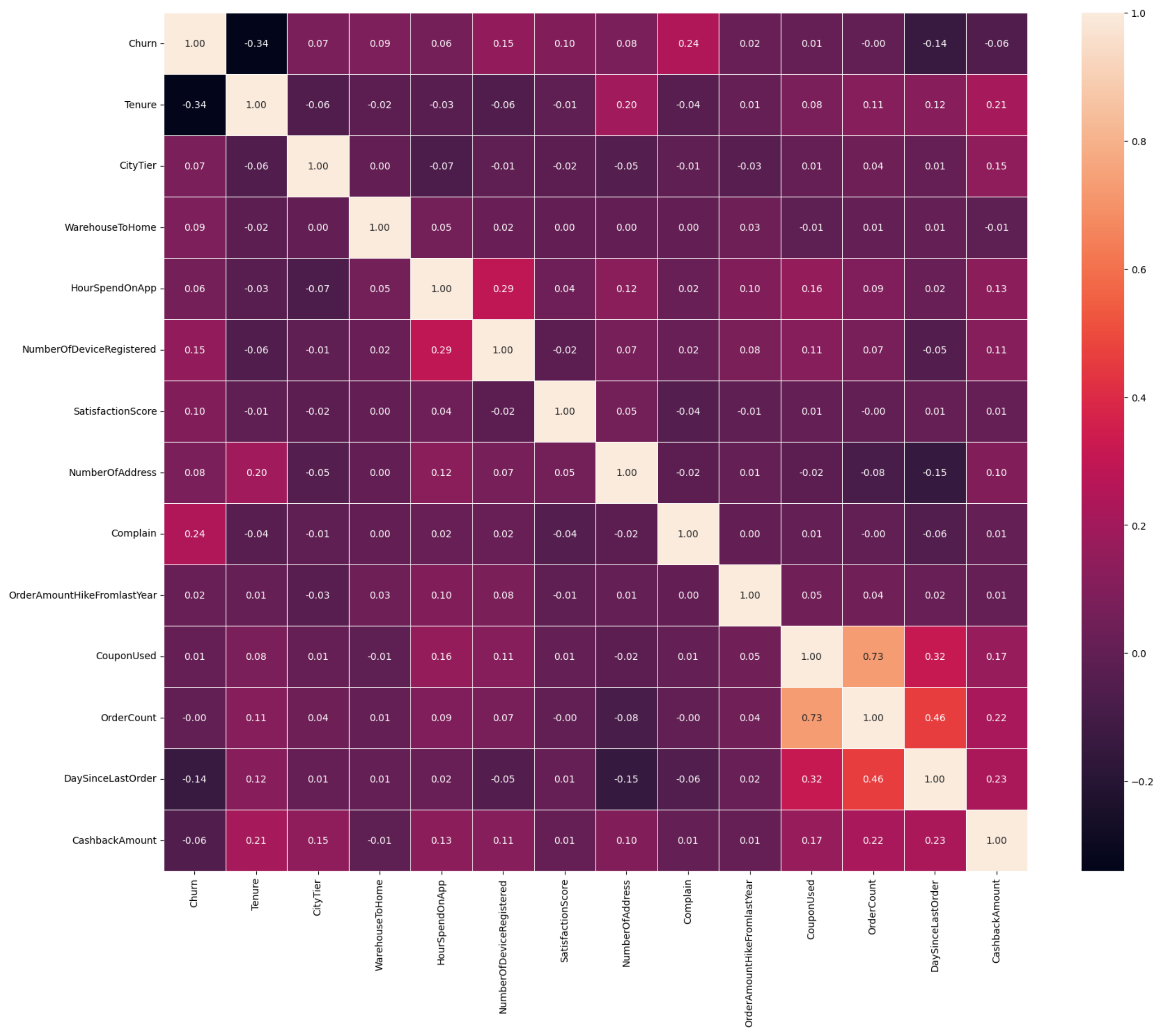


It’s clear that the proportion between the two classess remains the same.
For what concerns the dataset with the null values filled, I decided to replace every null value with the mean of the feature (all the null values was about a numerical feature), however I considered only the records belonging to the same class. The proportion between the two classes obviously remains the same.
Finally I dropped from both datasets the column “CustomerID” because the ID of the customer is unique and does not have effects on the classification.

II. Correlation analysis and feature extraction

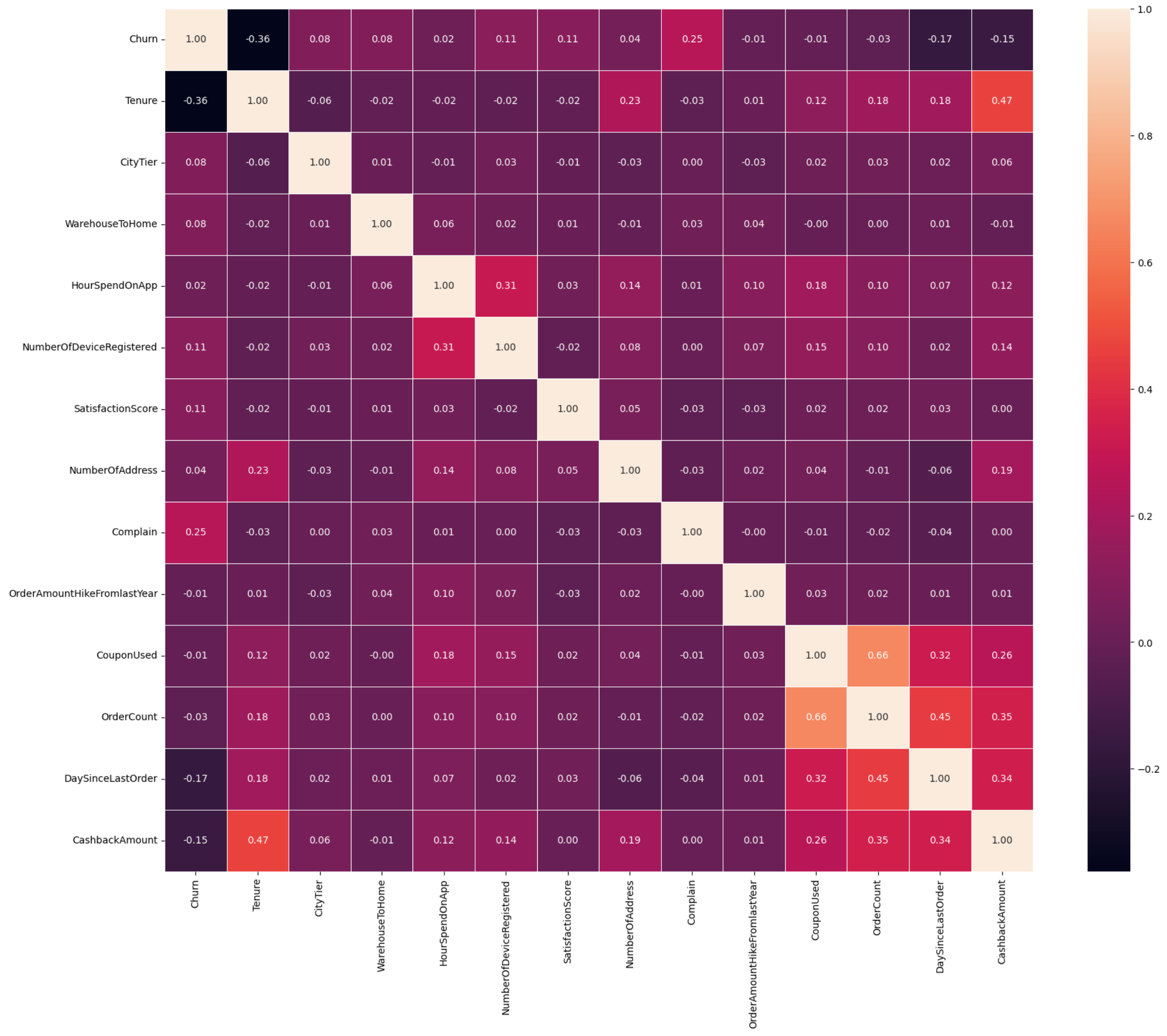
I performed a correlation analysis in order to see if there were some features that were strongly correlated (at least 0.9).
I obtained this two correlation tables:

Dropped values:



	Churn	Tenure	CityTier	WarehouseToHome	HourSpendOnApp	NumberOfDeviceRegistered	SatisfactionScore	NumberOfAddress	Complain	OrderAmountHikeFromlastYear	CouponUsed	OrderCount	DaySinceLastOrder	CashbackAmount
Churn	1.00	-0.34	0.07	0.09	0.06	0.15	0.10	0.08	0.24	0.02	0.01	-0.00	-0.14	-0.06
Tenure	-0.34	1.00	-0.06	-0.02	-0.03	-0.06	-0.01	0.20	-0.04	0.01	0.08	0.11	0.12	0.21
CityTier	0.07	-0.06	1.00	0.00	-0.07	-0.01	-0.02	-0.05	-0.01	-0.03	0.01	0.04	0.01	0.15
WarehouseToHome	0.09	-0.02	0.00	1.00	0.05	0.02	0.00	0.00	0.00	0.03	-0.01	0.01	0.01	-0.01
HourSpendOnApp	0.06	-0.03	-0.07	0.05	1.00	0.29	0.04	0.12	0.02	0.10	0.16	0.09	0.02	0.13
NumberOfDeviceRegistered	0.15	-0.06	-0.01	0.02	0.29	1.00	-0.02	0.07	0.02	0.08	0.11	0.07	-0.05	0.11
SatisfactionScore	0.10	-0.01	-0.02	0.00	0.04	-0.02	1.00	0.05	-0.04	-0.01	0.01	-0.00	0.01	0.01
NumberOfAddress	0.08	0.20	-0.05	0.00	0.12	0.07	0.05	1.00	-0.02	0.01	-0.02	-0.08	-0.15	0.10
Complain	0.24	-0.04	-0.01	0.00	0.02	0.02	-0.04	-0.02	1.00	0.00	0.01	-0.00	-0.06	0.01
OrderAmountHikeFromlastYear	0.02	0.01	-0.03	0.03	0.10	0.08	-0.01	0.01	0.00	1.00	0.05	0.04	0.02	0.01
CouponUsed	0.01	0.08	0.01	-0.01	0.16	0.11	0.01	-0.02	0.01	0.05	1.00	0.73	0.32	0.17
OrderCount	-0.00	0.11	0.04	0.01	0.09	0.07	-0.00	-0.08	-0.00	0.04	0.73	1.00	0.46	0.22
DaySinceLastOrder	-0.14	0.12	0.01	0.01	0.02	-0.05	0.01	-0.15	-0.06	0.02	0.32	0.46	1.00	0.23
CashbackAmount	-0.06	0.21	0.15	-0.01	0.13	0.11	0.01	0.10	0.01	0.01	0.17	0.22	0.23	1.00

Filled values



	Churn	Tenure	CityTier	WarehouseToHome	HourSpendOnApp	NumberOfDeviceRegistered	SatisfactionScore	NumberOfAddress	Complain	OrderAmountHikeFromlastYear	CouponUsed	OrderCount	DaySinceLastOrder	CashbackAmount
Churn	1.00	-0.36	0.08	0.08	0.02	0.11	0.11	0.04	0.25	-0.01	-0.01	-0.03	-0.17	-0.15
Tenure	-0.36	1.00	-0.06	-0.02	-0.02	-0.02	-0.02	0.23	-0.03	0.01	0.12	0.18	0.18	0.47
CityTier	0.08	-0.06	1.00	0.01	-0.01	0.03	-0.01	-0.03	0.00	-0.03	0.02	0.03	0.02	0.06
WarehouseToHome	0.08	-0.02	0.01	1.00	0.06	0.02	0.01	-0.01	0.03	0.04	-0.00	0.00	0.01	-0.01
HourSpendOnApp	0.02	-0.02	-0.01	0.06	1.00	0.31	0.03	0.14	0.01	0.10	0.18	0.10	0.07	0.12
NumberOfDeviceRegistered	0.11	-0.02	0.03	0.02	0.31	1.00	-0.02	0.08	0.00	0.07	0.15	0.10	0.02	0.14
SatisfactionScore	0.11	-0.02	-0.01	0.01	0.03	-0.02	1.00	0.05	-0.03	-0.03	0.02	0.02	0.03	0.00
NumberOfAddress	0.04	0.23	-0.03	-0.01	0.14	0.08	0.05	1.00	-0.03	0.02	0.04	-0.01	-0.06	0.19
Complain	0.25	-0.03	0.00	0.03	0.01	0.00	-0.03	-0.03	1.00	-0.00	-0.01	-0.02	-0.04	0.00
OrderAmountHikeFromlastYear	-0.01	0.01	-0.03	0.04	0.10	0.07	-0.03	0.02	-0.00	1.00	0.03	0.02	0.01	0.01
CouponUsed	-0.01	0.12	0.02	-0.00	0.18	0.15	0.02	0.04	-0.01	0.03	1.00	0.66	0.32	0.26
OrderCount	-0.03	0.18	0.03	0.00	0.10	0.10	0.02	-0.01	-0.02	0.02	0.66	1.00	0.45	0.35
DaySinceLastOrder	-0.17	0.18	0.02	0.01	0.07	0.02	0.03	-0.06	-0.04	0.01	0.32	0.45	1.00	0.34
CashbackAmount	-0.15	0.47	0.06	-0.01	0.12	0.14	0.00	0.19	0.00	0.01	0.26	0.35	0.34	1.00

In both cases the correlation between the output feature (Churn) and the others is really weak (at maximum -0.36); in general the strongest correlation in both datasets is between OrderCount and CouponUsed (0.73 in dropped, 0.66 in filled). Since they were not greater than 0.9 I decided not to drop one of them and I maintained them both.
Then I used the get_dummies in order to transform the categorical values in a set of numerical features (0/1). At that point I had only numerical features, so I gave another look at the correlation table. I did not obtain any relevant difference with respect to the two values I highlighted in the previous case, as a matter of fact I could simply extract the two datasets and end the preprocessing phase (I did not report the last two correlation tables because

they would not be clearly readable in this document. However, they are still present in the “Preprocessing” Notebook).

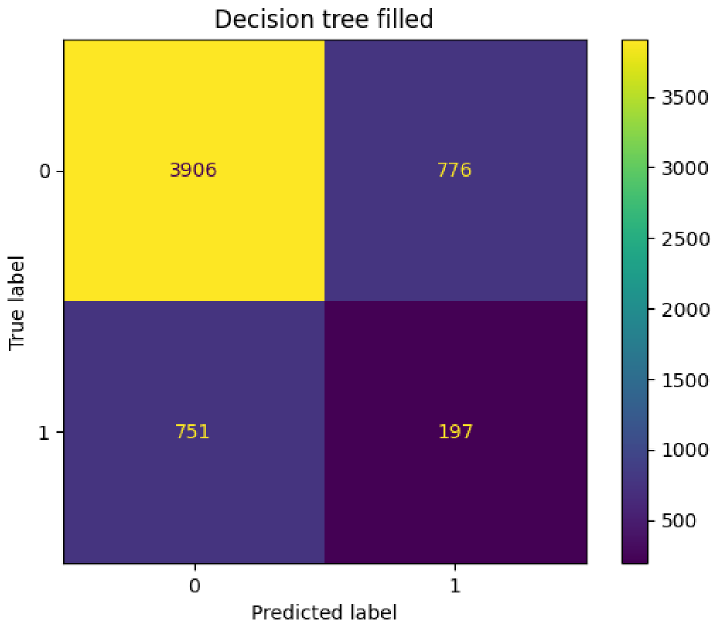
4. CLASSIFICATION

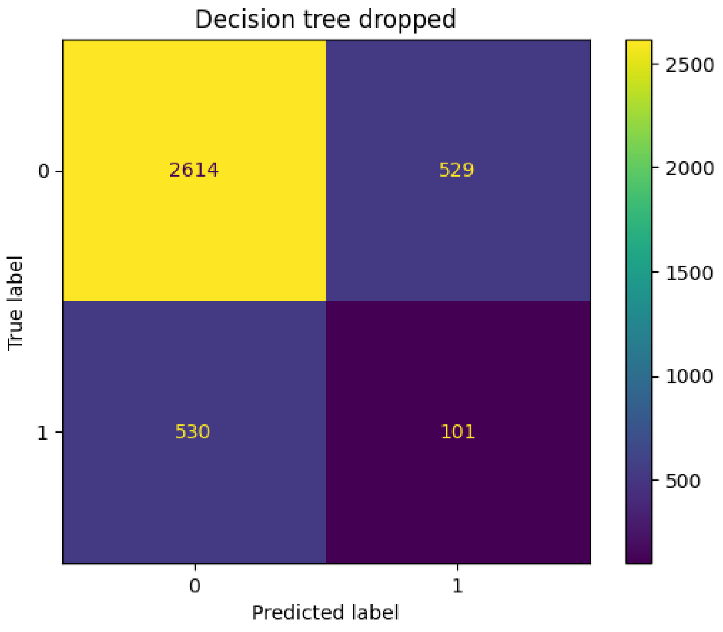
For the classification phase the objective was to find the classifier which will correctly predict the output of an instance. In order to do this, I dropped the Churn column from the dataset and I used it as target column.

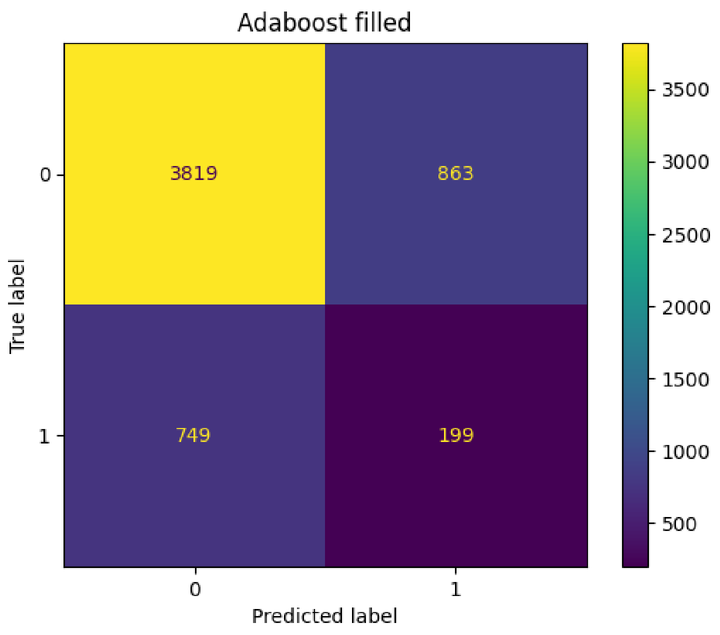
I decided to evaluate the accuracy of the classifiers using a k-fold cross validation (with k=5) in order to obtain results that were the least dependent by the choice of test and training set; specifically I used Stratified K fold. The parameters that I used to evaluate the models were f-score and average accuracy.

Due the fact that the two classes are imbalanced, I used SMOTE before every classification in order to rebalance the dataset.

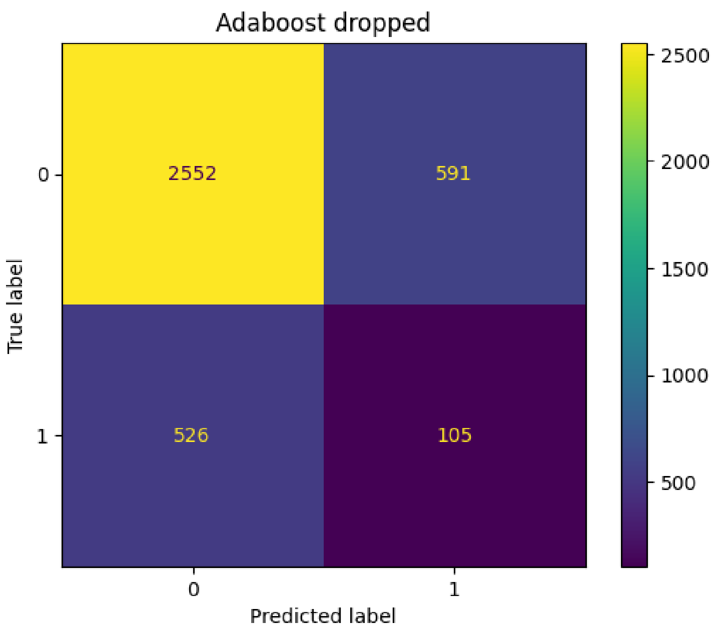
I used each classifier two times: first with the filled dataset and then with the dropped one. For K Nearest Neighbors I tried with K berween 3 and 10 and I obtained the best results for K=4 (I do not report all the tries for simplicity). Here I show the confusion matrixes of the classification and then I will report the results in terms of f-score and average accuracy for each try.



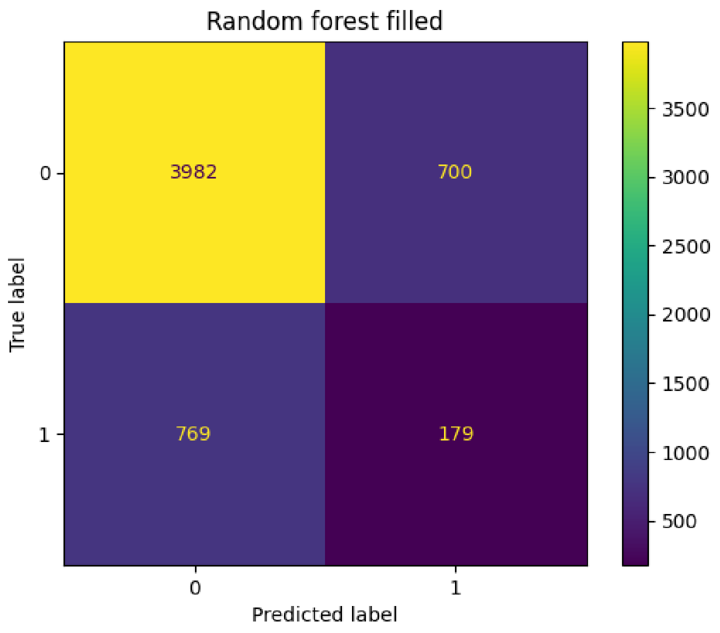




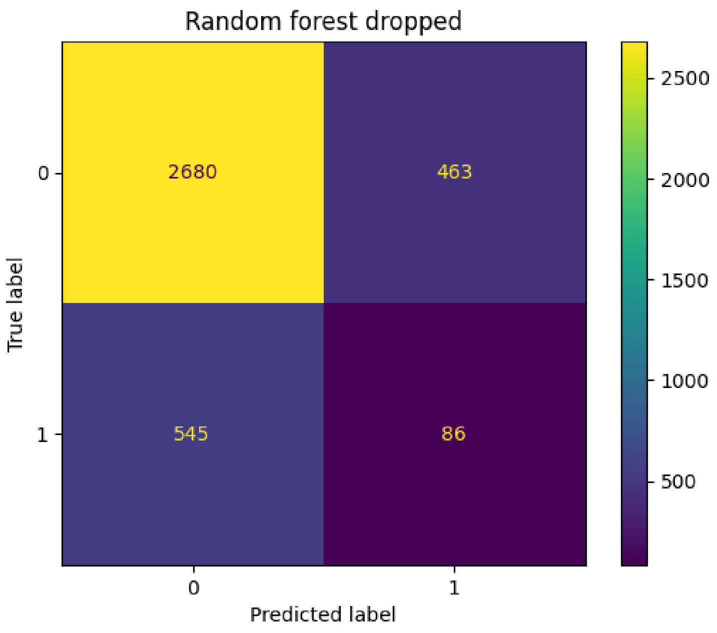
Adaboost filled
3819
863
749
199
True label
Predicted label
0
1
3500
3000
2500
2000
1500
1000
500



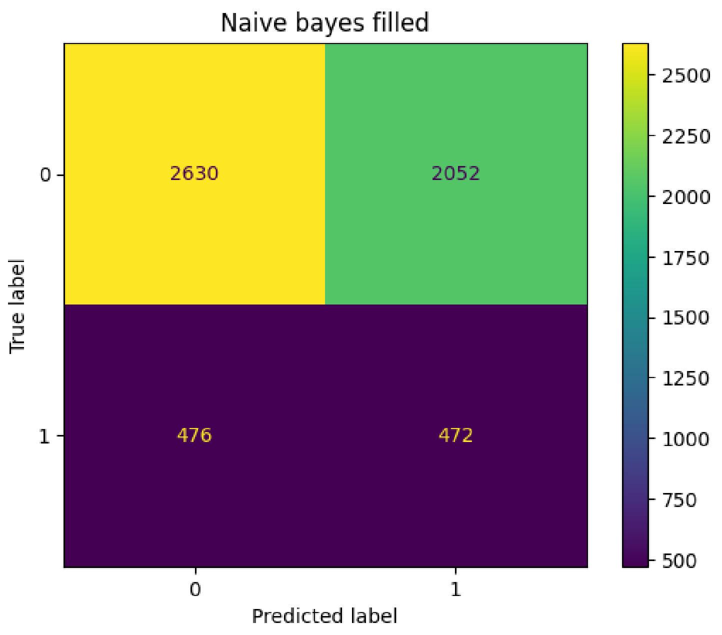
Adaboost dropped
2552
591
526
105
True label
Predicted label
0
1
2500
2000
1500
1000
500

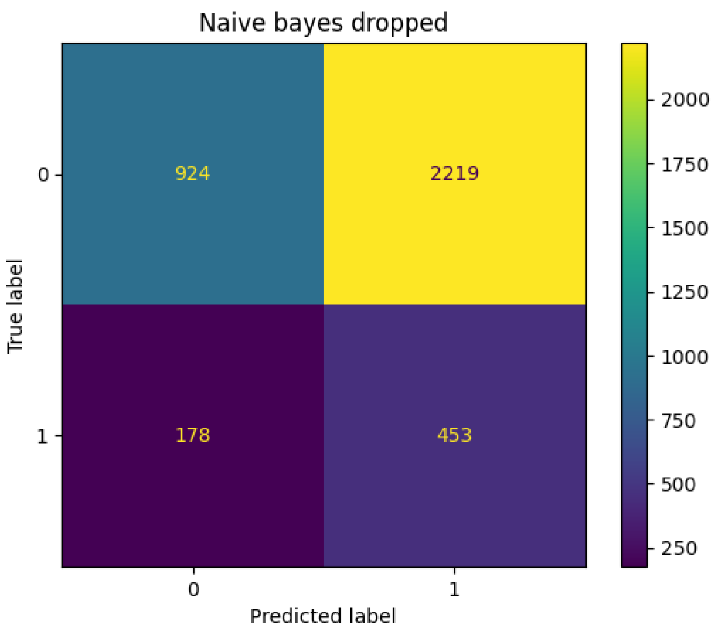


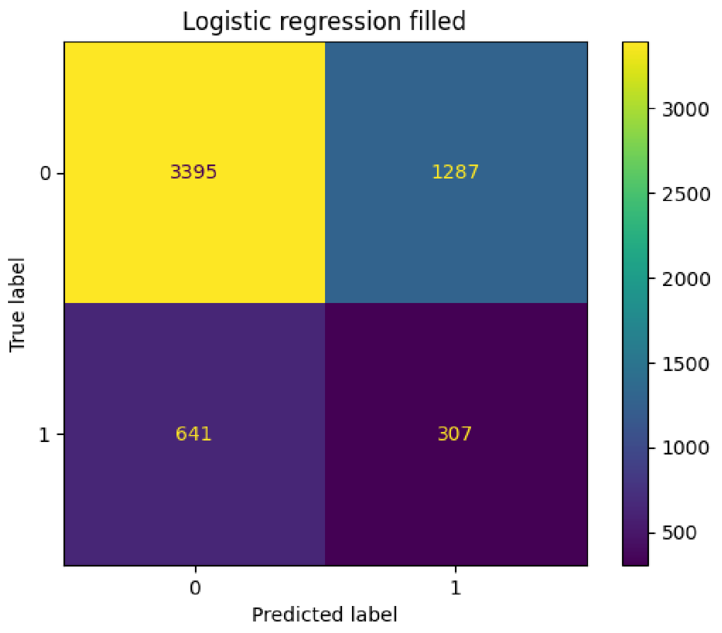
Random forest filled
3982
700
769
179
True label
Predicted label
0
1
3500
3000
2500
2000
1500
1000
500

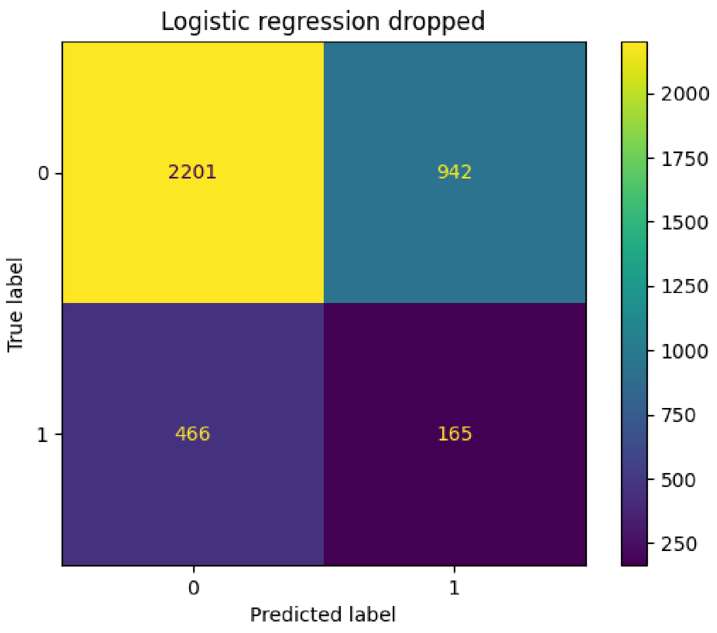


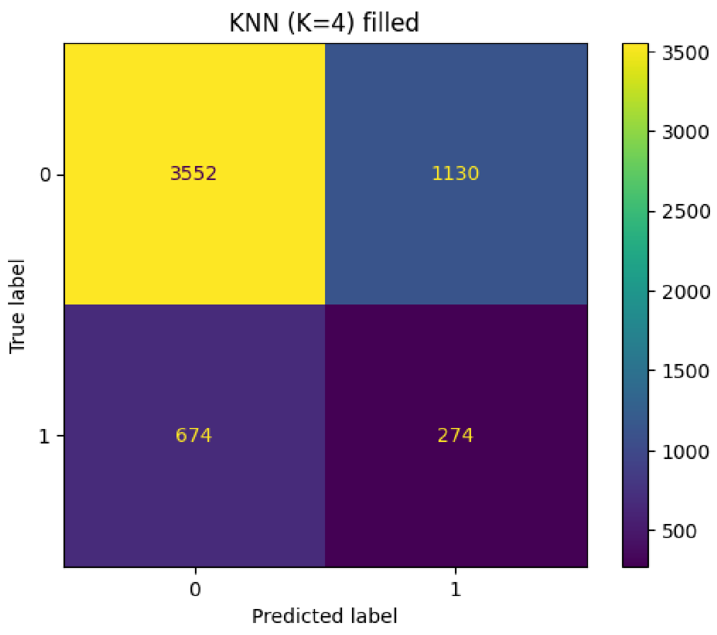
Random forest dropped
2680
463
545
86
True label
Predicted label
0
1
2500
2000
1500
1000
500

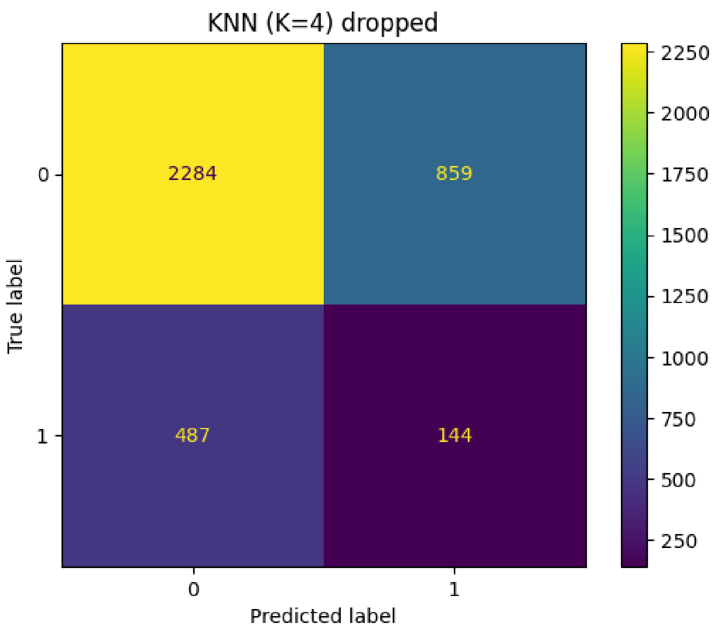

Naive bayes filled
True label
Predicted label
0
1
2630
2052
476
472
2500
2250
2000
1750
1500
1250
1000
750
500

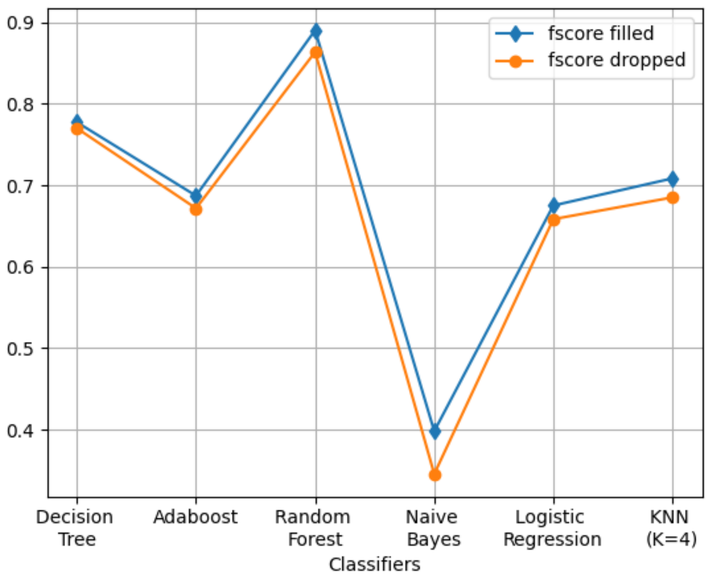

Naive bayes dropped
True label
Predicted label
0
1
924
2219
178
453
2000
1750
1500
1250
1000
750
500
250

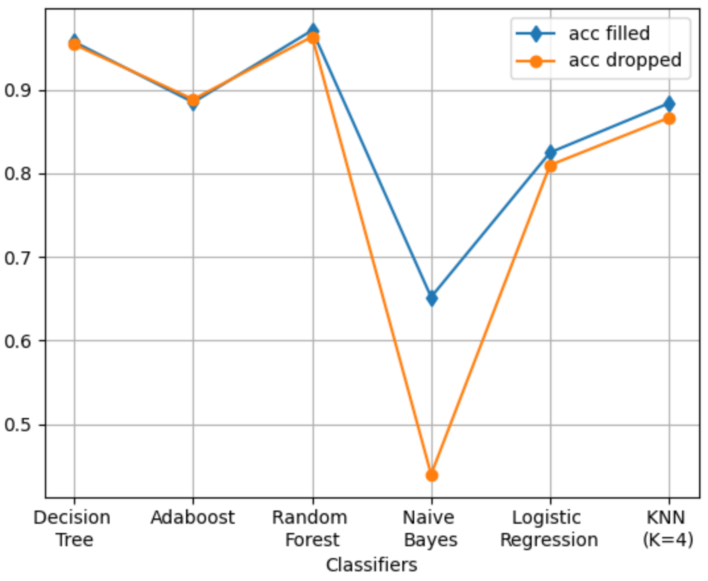

Logistic regression filled
True label
Predicted label
0
1
3395
1287
641
307
3000
2500
2000
1500
1000
500


Logistic regression dropped
True label
Predicted label
0
1
2201
942
466
165
2000
1750
1500
1250
1000
750
500
250









Classifiers		**f-score**	**Avg accuracy**
Decision Tree	Filled	0.777	0.957
	Dropped	0.770	0.954
AdaBoost	Filled	0.687	0.885
	Dropped	0.672	0.888
Random Forest	Filled	0.890	0.971
	Dropped	0.864	0.963
Gaussian Naive Bayes	Filled	0.398	0.652
	Dropped	0.345	0.440
Logistic Regression	Filled	0.675	0.825
	Dropped	0.658	0.810
KNN (K=4)	Filled	0.708	0.884
	Dropped	0.685	0.866

From these results I could draw two conclusions:

- The dataset with the null values filled always gave best results (the only exception is avg accuracy with AdaBoost, but the difference is just of 0.003)
- Random Forest is globally the best classifier with respect to both f-score and avg accuracy

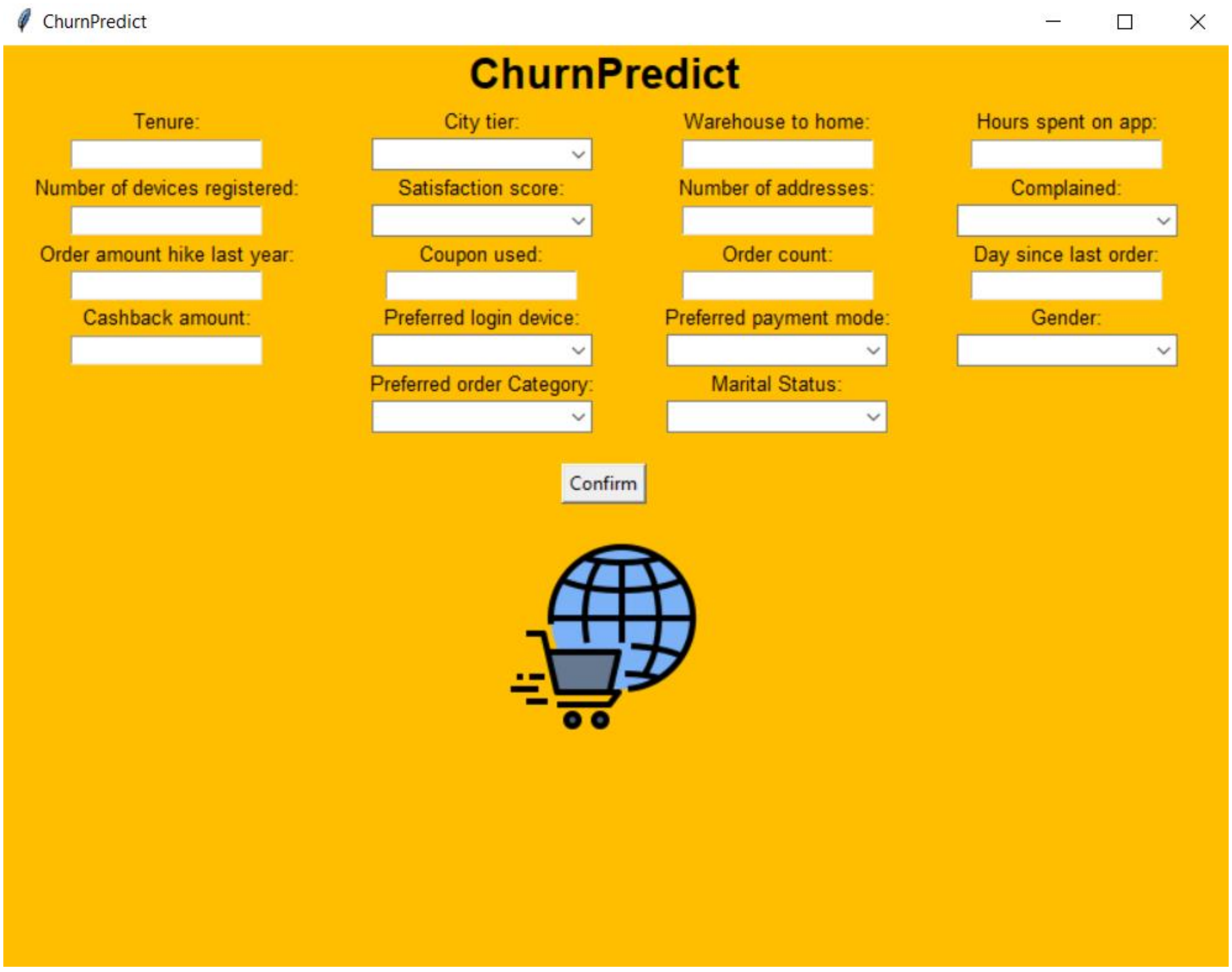
The last analysis I performed was the Wilcoxon test because I wanted to know if the difference between the filled and the dropped results was statistically relevant. The test for Random Forest gave a pvalue of 0.0625. Since this value is greater than 0.05 I could say that the difference had no statistical relevance, therefore I cannot conclude that to fill the null value is globally a better choice than drop the relative records. Due to the fact that filling the value is not so complicated, compared to drop the records, and since the difference of f-score for Random Forest is not so small (0.026), I could conclude that Random Forest with the null values filled was the best option in order to obtain a better classification.

5. IMPLEMENTATION OF THE APPLICATION

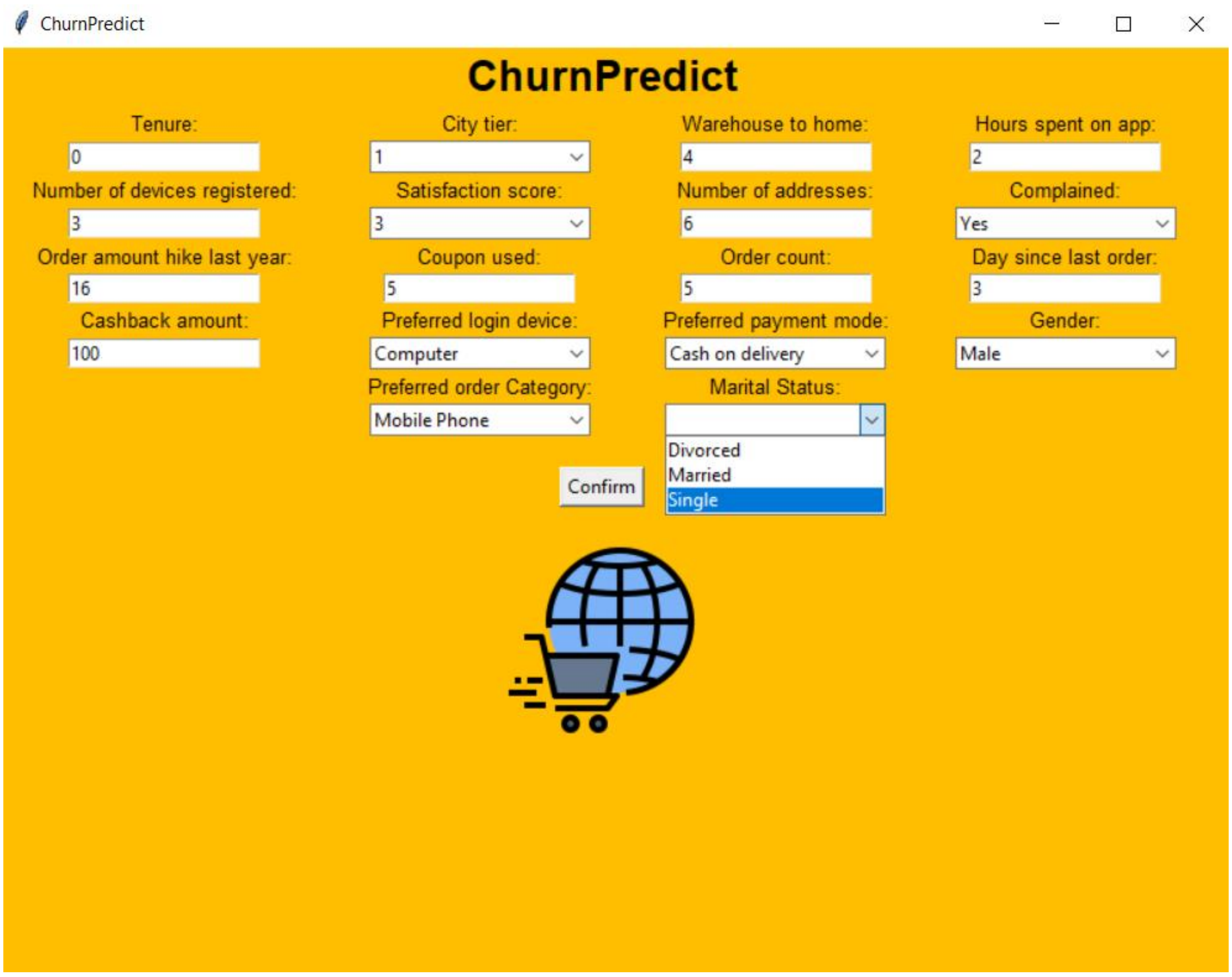
The application I created, in order to use the dataset and Random Forest, is called "ChurnPredict". The goal of the application is to give the user the possibility to know if a specific customer is likable to leave the Ecommerce, thanks to a list of data asked by the application itself. The application is written in Python and makes use of the following libraries:

- Tkinter, for the GUI.
- Sklearn, to perform Random Forest.
- Pandas, to read the dataset.
- Imblearn, for using the pipeline and SMOTE
- Numpy, in order to transform the list of inputs in an array.

The homepage looks like this:



The user has to insert the values into the fields; some of them give the user a list of possible choices (like Preferred order category), others are just text fields.

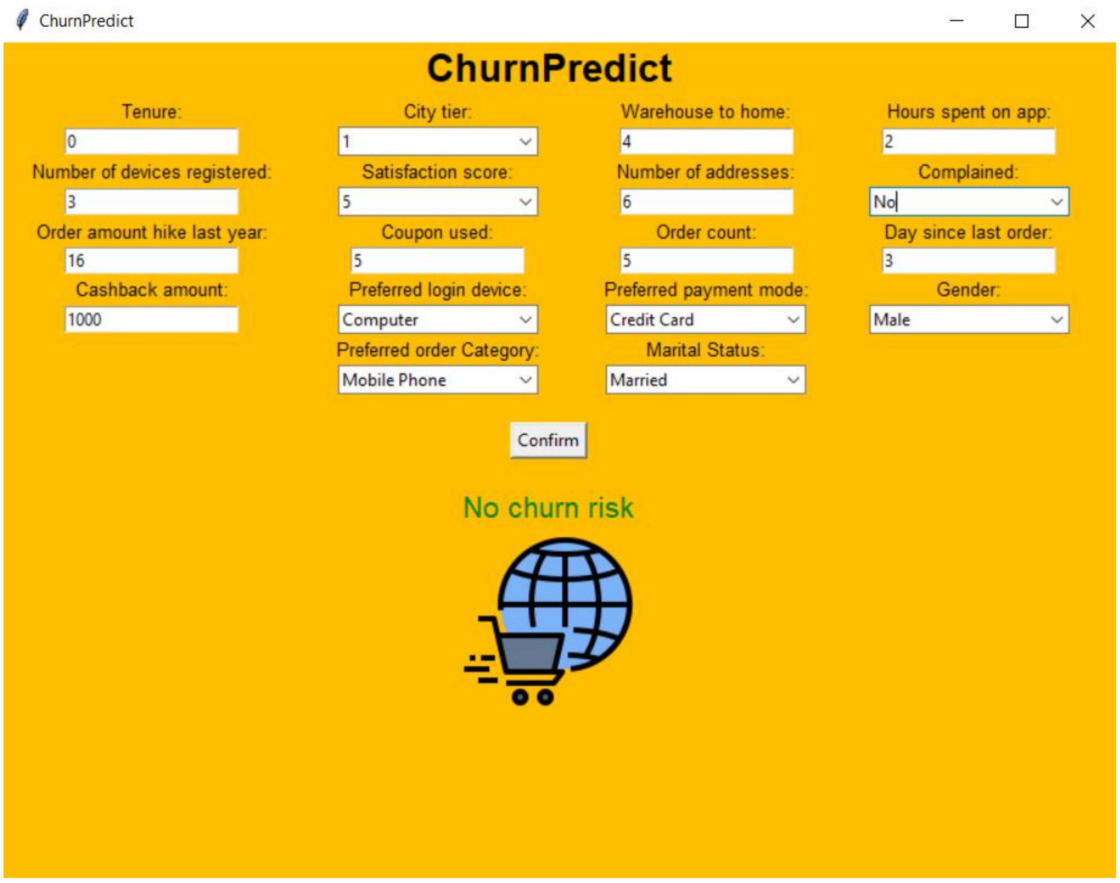


ChurnPredict

ChurnPredict

Tenure: 0	City tier: 1	Warehouse to home: 4	Hours spent on app: 2
Number of devices registered: 3	Satisfaction score: 3	Number of addresses: 6	Complained: Yes
Order amount hike last year: 16	Coupon used: 5	Order count: 5	Day since last order: 3
Cashback amount: 100	Preferred login device: Computer	Preferred payment mode: Cash on delivery	Gender: Male
	Preferred order Category: Mobile Phone	Marital Status:	

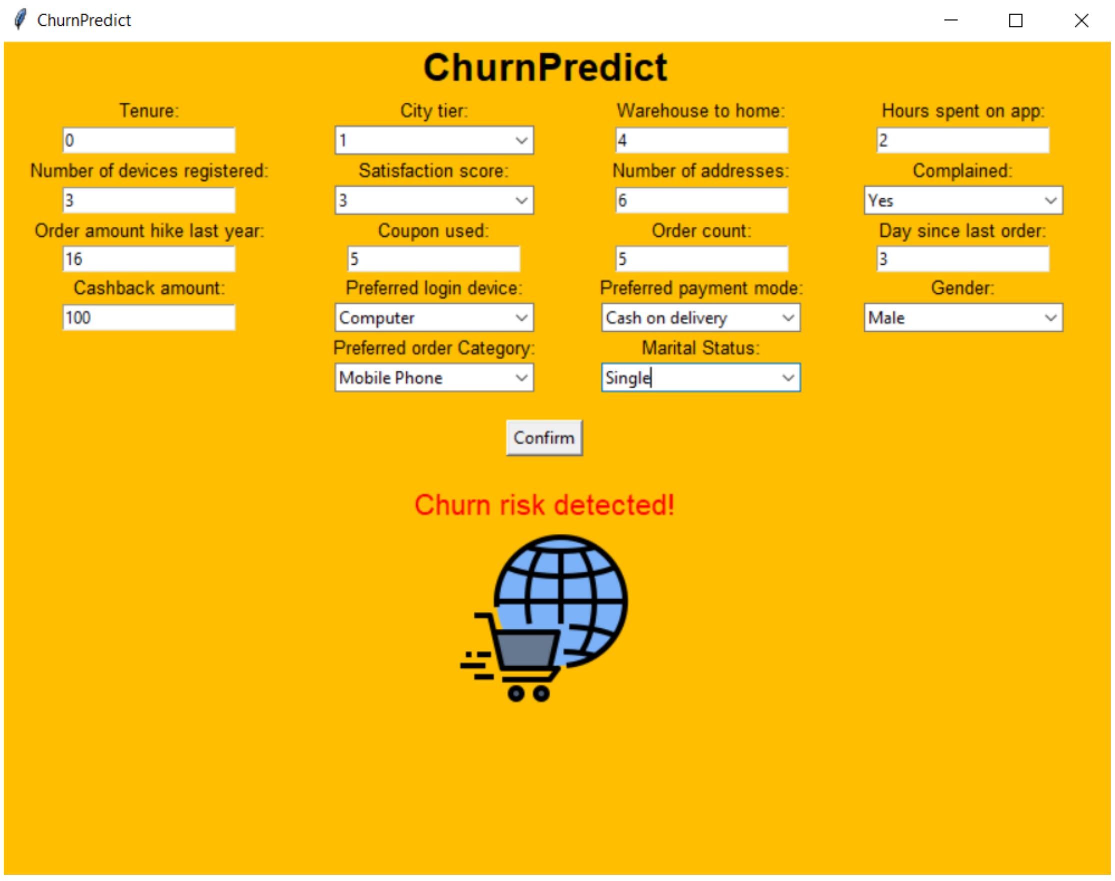
Finally, the user will click on "Confirm" and the app will show a message on the screen telling whether there is a risk or not.


ChurnPredict

ChurnPredict

Tenure: 0	City tier: 1	Warehouse to home: 4	Hours spent on app: 2
Number of devices registered: 3	Satisfaction score: 5	Number of addresses: 6	Complained: No
Order amount hike last year: 16	Coupon used: 5	Order count: 5	Day since last order: 3
Cashback amount: 1000	Preferred login device: Computer	Preferred payment mode: Credit Card	Gender: Male
	Preferred order Category: Mobile Phone	Marital Status: Married	

No churn risk


ChurnPredict

ChurnPredict

Tenure: 0	City tier: 1	Warehouse to home: 4	Hours spent on app: 2
Number of devices registered: 3	Satisfaction score: 3	Number of addresses: 6	Complained: Yes
Order amount hike last year: 16	Coupon used: 5	Order count: 5	Day since last order: 3
Cashback amount: 100	Preferred login device: Computer	Preferred payment mode: Cash on delivery	Gender: Male
	Preferred order Category: Mobile Phone	Marital Status: Single	

Churn risk detected!

6. CONCLUSIONS

In this study I compared a list of classifiers using a dataset with a binary classification. The tests were done using a Stratified K Fold as cross validator and SMOTE, in order to deal with the fact that the dataset was imbalaced. Due to the fact that there were some null values, I created a dataset without the records that had a null value and a dataset with the null values filled with the mean of the feature (only considering the same class). All the classifiers, with the exception of Gaussian Naive Bayes, had a better performance on the filled dataset with respect to f-score and average accuracy. Particularly good was Random Forest, considering that it gave me a very high performance, even if Wilcoxon test told us that there is no statistical relevance in the performance difference. In conclusion, the last step was building a simple application in Python which uses the filled dataset and Random Forest.