CHURN PREDICTION

Anis HENTIT Ruben LEON 01 - EDA
02 - Models

04 - Presentation

03 - Analysis

01 - EDA

Performing a Explanatory Data Analysis

```
12 columns * 3150 rows with no null values But disproportional distribution:
```

- Don't Churn (0): 84% of the Database
- Churn (1): 16% of the Database

Correlation analysis

Call_Failure -	1	0.15	0.17	0.59	0.5	0.57	-0.022	0.5	0.05	0.19	-0.11	0.042	0.12	-0.009
Complains -	0.15	1	-0.02	-0.034	-0.1	-0.091	-0.11	-0.058	0.02	0.0011	0.27	0.0033	-0.13	0.53
Subscription_Length -	0.17	-0.02	1	0.079	0.12	0.11	0.076	0.092	0.021	-0.16	0.14	-0.0024	0.11	-0.033
Charge_Amount -	0.59	-0.034	0.079	1	0.45	0.38	0.092	0.42	0.28	0.32	-0.36	0.28	0.17	-0.2
Seconds_of_Use -	0.5	-0.1	0.12	0.45	1	0.95	0.1	0.68	0.02	0.13	-0.46	0.021	0.42	-0.3
Frequency_of_Use -	0.57	-0.091	0.11	0.38	0.95	1	0.1	0.74	-0.033	0.21	-0.45	-0.028	0.4	-0.3
Frequency_of_Sms -	-0.022	-0.11	0.076	0.092	0.1	0.1	1	0.08	-0.054	0.2	-0.3	-0.093	0.92	-0.22
Distinct_Called_Numbers -	0.5	-0.058	0.092	0.42	0.68	0.74	0.08	1	0.021	0.17	-0.41	0.051	0.28	-0.28
Age_Group -	0.05	0.02	0.021	0.28	0.02	-0.033	-0.054	0.021	1	-0.15	0.0025	0.96	-0.18	-0.015
Tariff_Plan -	0.19	0.0011	-0.16	0.32	0.13	0.21	0.2	0.17	-0.15	1	-0.16	-0.12	0.25	-0.11
Status -	-0.11	0.27	0.14	-0.36	-0.46	-0.45	-0.3	-0.41	0.0025	-0.16	1	-0.0013	-0.41	0.5
Age -	0.042	0.0033	-0.0024	0.28	0.021	-0.028	-0.093	0.051	0.96	-0.12	-0.0013	1	-0.22	-0.018
Customer_Value -	0.12	-0.13	0.11	0.17	0.42	0.4	0.92	0.28	-0.18	0.25	-0.41	-0.22	1	-0.29
Churn -	-0.009	0.53	-0.033	-0.2	-0.3	-0.3	-0.22	-0.28	-0.015	-0.11	0.5	-0.018	-0.29	1
	Call_Failure -	Complains -	Subscription_Length -	Charge_Amount -	Seconds_of_Use -	Frequency_of_Use -	Frequency_of_Sms -	Distinct_Called_Numbers -	Age_Group -	Tariff_Plan -	Status -	- Age -	Customer_Value -	Churn -

Correlation with Churn variable

Positive	Negative					
subscription_length	customer_value					
distinc_calls	distinc_calls					
age_group : 25-45	frequency_of_sms					
tarif_plan : 1	call_failurs					
	charge_amount					

02 - Models

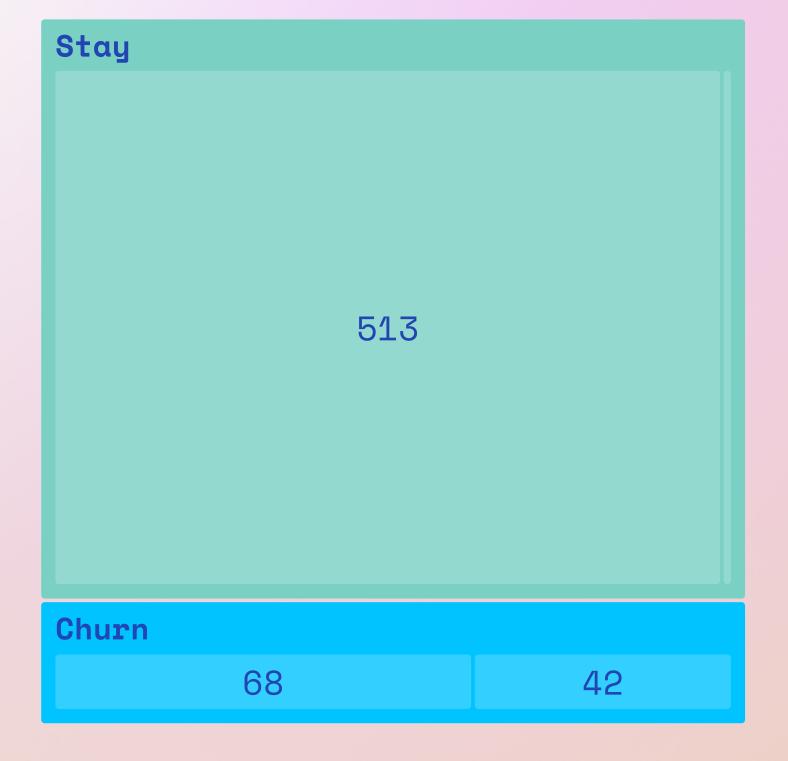
Binary Classification Problem

Three models to predict Churn

- Logistic regression
- XGBoost
- SVM

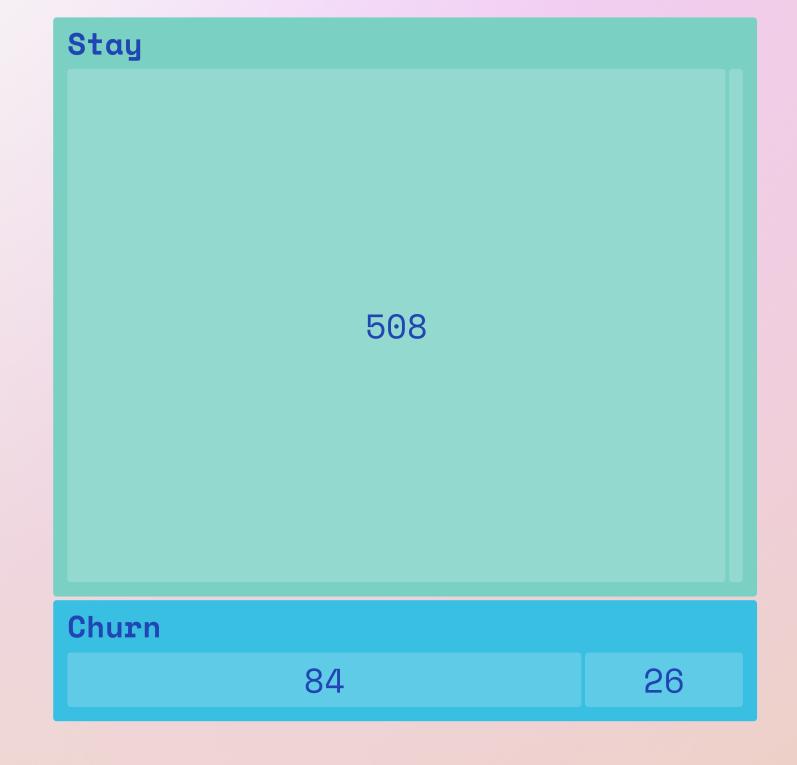
Logistic Regression

```
Estimates the probability of an event
occurring based on the values of
independent variables
Uses a logistic function to model the
relationship between the variables.
Best Hyperparameter : {
  'C': 0.1,
 'penalty': '11',
 'solver': 'liblinear'
```



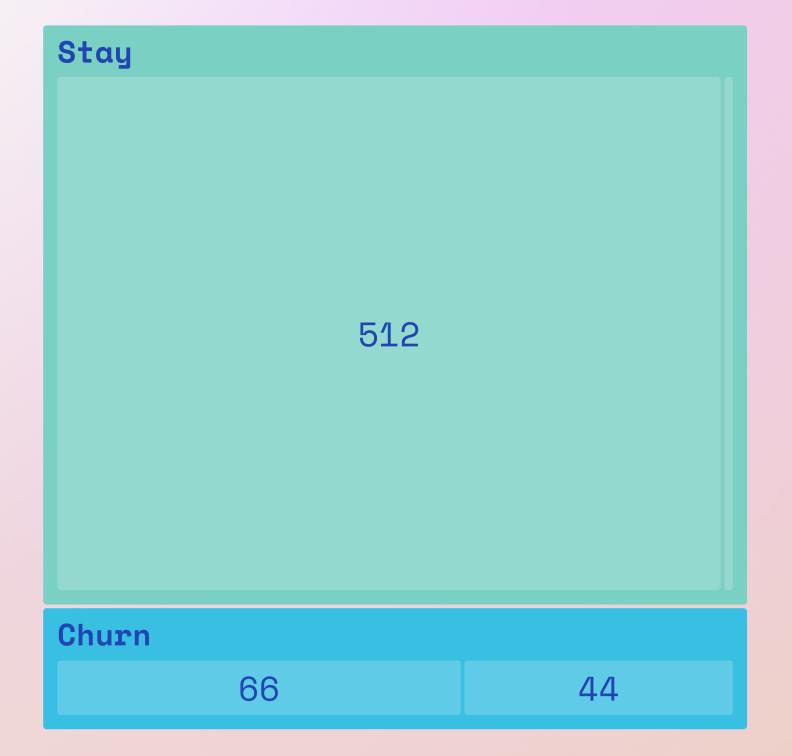
XGBoost

```
XGBoost (Extreme Gradient Boosting)
combines the predictions of multiple weak
decision trees trained sequentially to
correct the mistakes made by the previous
ones. Usefull when the is a class
unbalance
Best Hyperparameter: {
 'colsample_bytree': 0.8,
 'learning_rate': 0.2,
 'max_depth': 3,
 'n_estimators': 200,
 'subsample': 0.9
```



SVM

```
SVM (Support Vector Machines) find an optimal
hyperplane that separates the data points of
different classes with the maximum margin
(distance between the hyperplane and the
nearest data points of each class).
Can handle high-dimensional data and have a
flexibility in for non-linear relationships
through the kernel trick
Best Hyperparameters: {
  'C': 10,
 'kernel': 'rbf'
```



03 - Analyse

Which model is the most performant?

Our Database :

- 12 columns * 3150 clean rows
- High unbalance (84/16) between the classes
- "Easy" to predict churn class
- Goal: minimize the false no-churn category

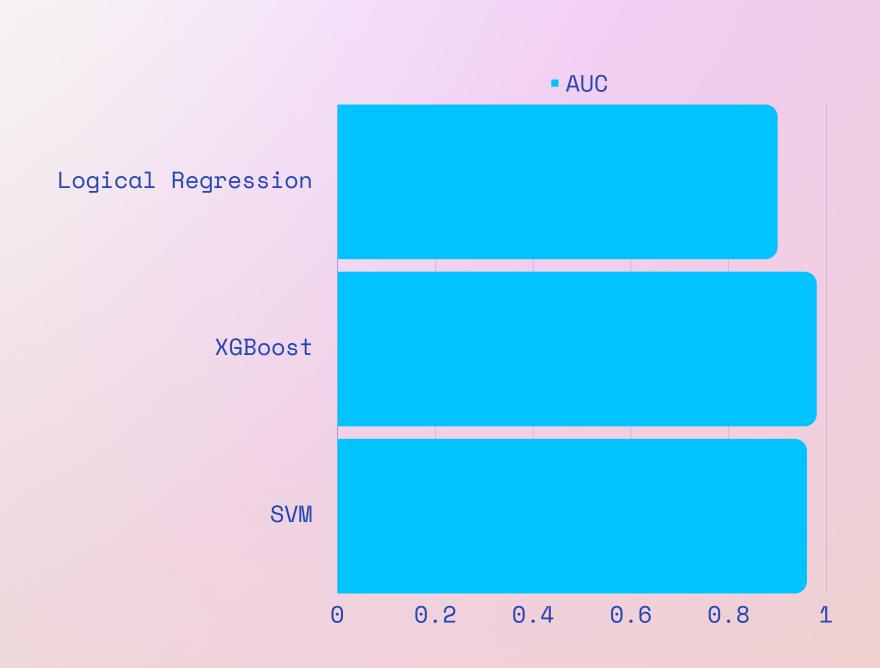
ROC and AUC

ROC (Receiver Operating Characteristic) is a graphical representation of the performance of a classification model :

- True positive rate TPR=TP/(TP+FN)
- False positive rate FPR=FP/(FP+TN)

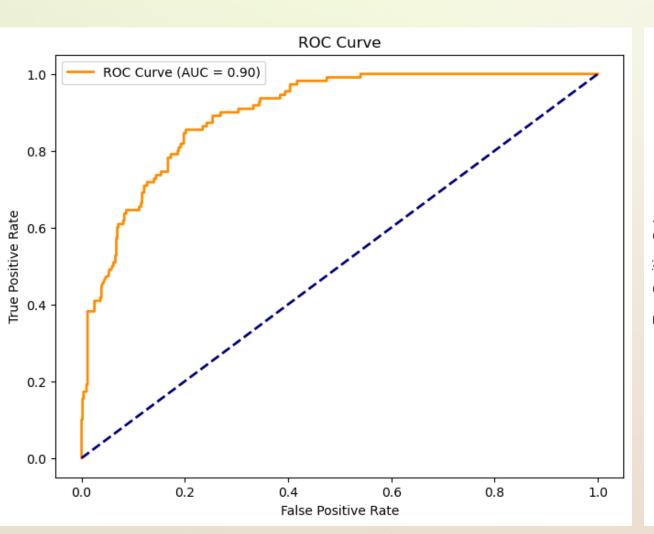
AUC (Area Under the Curve) quantifies the overall performance of the model:

- AUC=0.5 : random classifier
- AUC=1 : perfect classifier

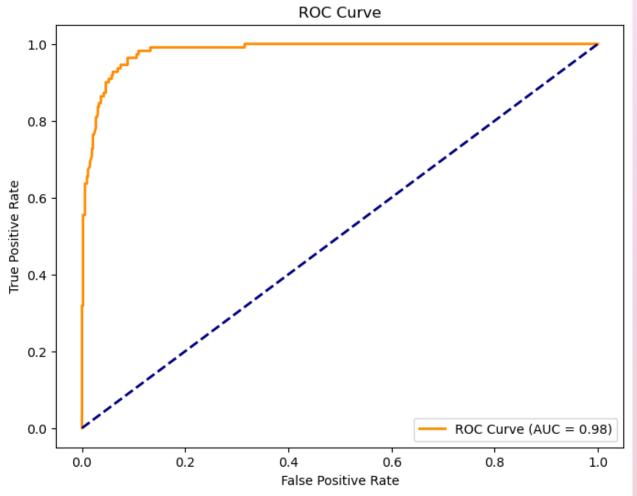


More details

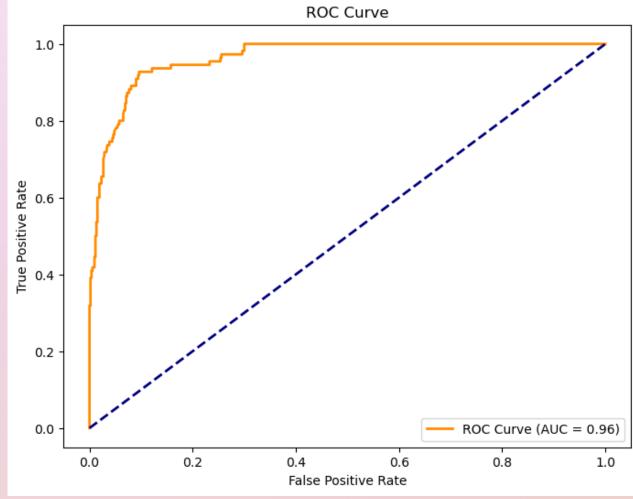
Logistic Regression



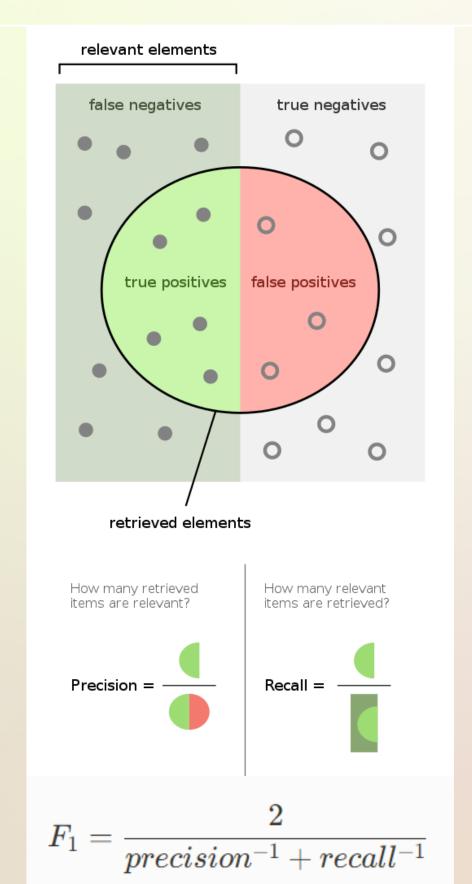
XGBoost

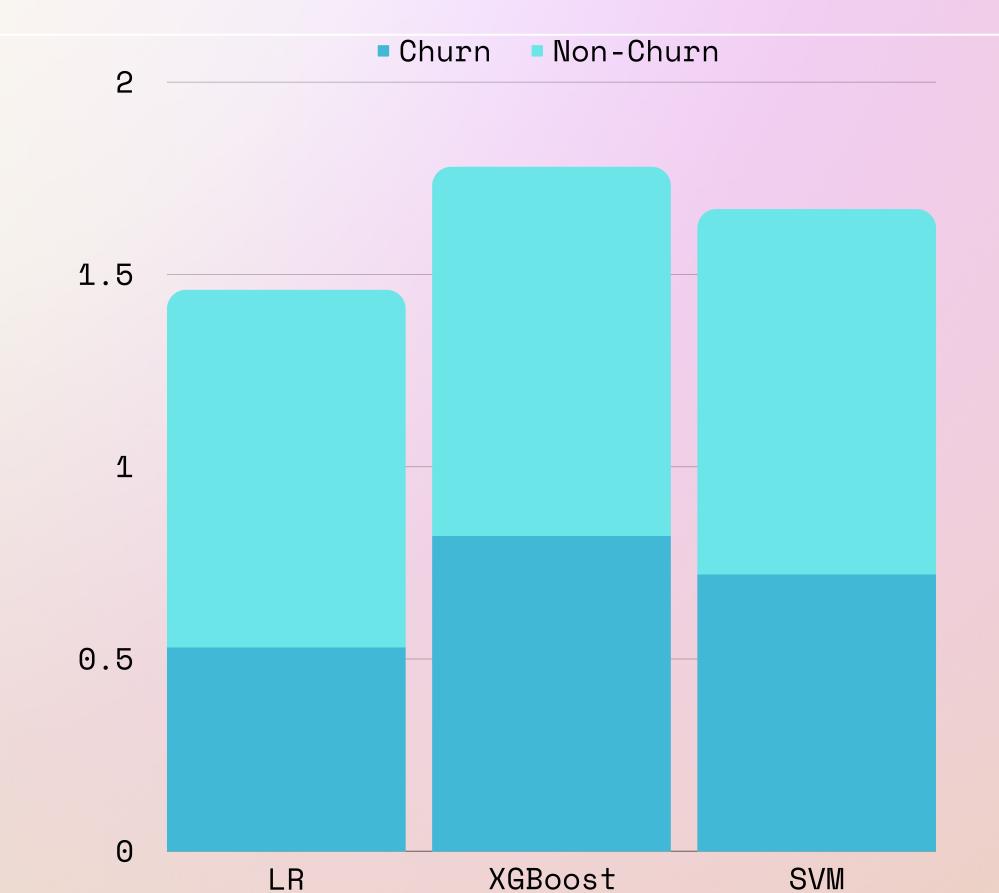


SVM



F1-Score





04 - Presentation

Use Website under Flask to show XGBoost result

Use of Flask for it convenience for a simple website with a page for the form and an other one for the result

Choose XGBoost because of its better performance on the training dataset when having unbalanced classes

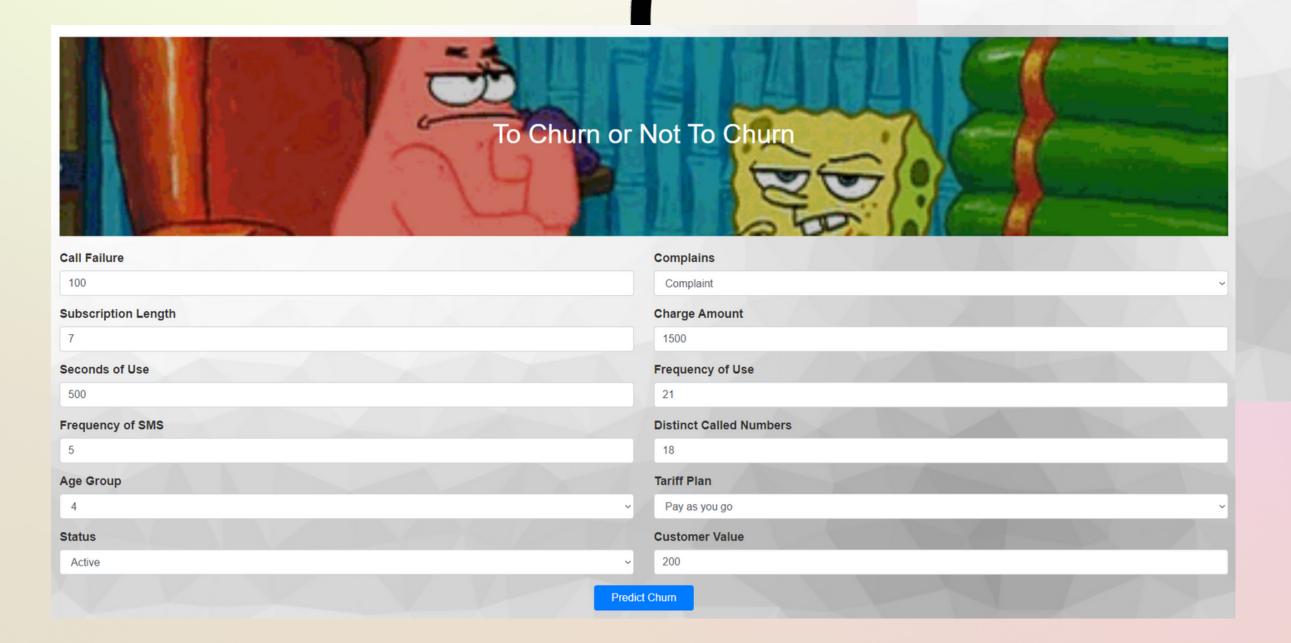
Main function

```
def preprocessDataAndPredict(feature dict):
    # Create a DataFrame from the input data
    test_data = pd.DataFrame({k: [float(v)] for k, v in feature dict.items()})
    test_data = pd.DataFrame(test_data)
    print(test data.dtypes)
    # Features to standardize
    features_to_standardize = ["Customer_Value", "Frequency_of_Sms", "Frequency_of_Use", "Seconds_of_Use",
                               "Subscription_Length", "Call Failure", "Distinct Called Numbers"]
    # Means and standard deviations for standardization
    means = [470.97291587, 73.17492063, 69.46063492, 4472.45968254, 32.54190476, 7.62793651, 23.50984127]
    stds = [516.93336034, 112.21974279, 57.4041938, 4197.2422989, 8.57212108, 7.26273248, 17.21460431]
    # Standardize the specified features
    for feature, mean, std in zip(features to standardize, means, stds):
        test data[feature] = (test data[feature] - mean) / std
    # Use XGBoost model to make predictions
    prediction = model.predict(xgb.DMatrix(test data))[0]
      # Apply a threshold to get the binary prediction (0 or 1)
    threshold = 0.5
    prediction = 1 if prediction >= threshold else 0
    print(prediction)
    return prediction
```

On navigator



Based on the information provided, the prediction is as follows:



Churn



Recommendation

- Take necessary steps to retain the customer.
- Offer incentives or discounts to prevent churn.
- Investigate reasons for churn and address them

Thanks