CHURN PREDICTION

Anis HENTIT
Ruben LEON

01 - EDA

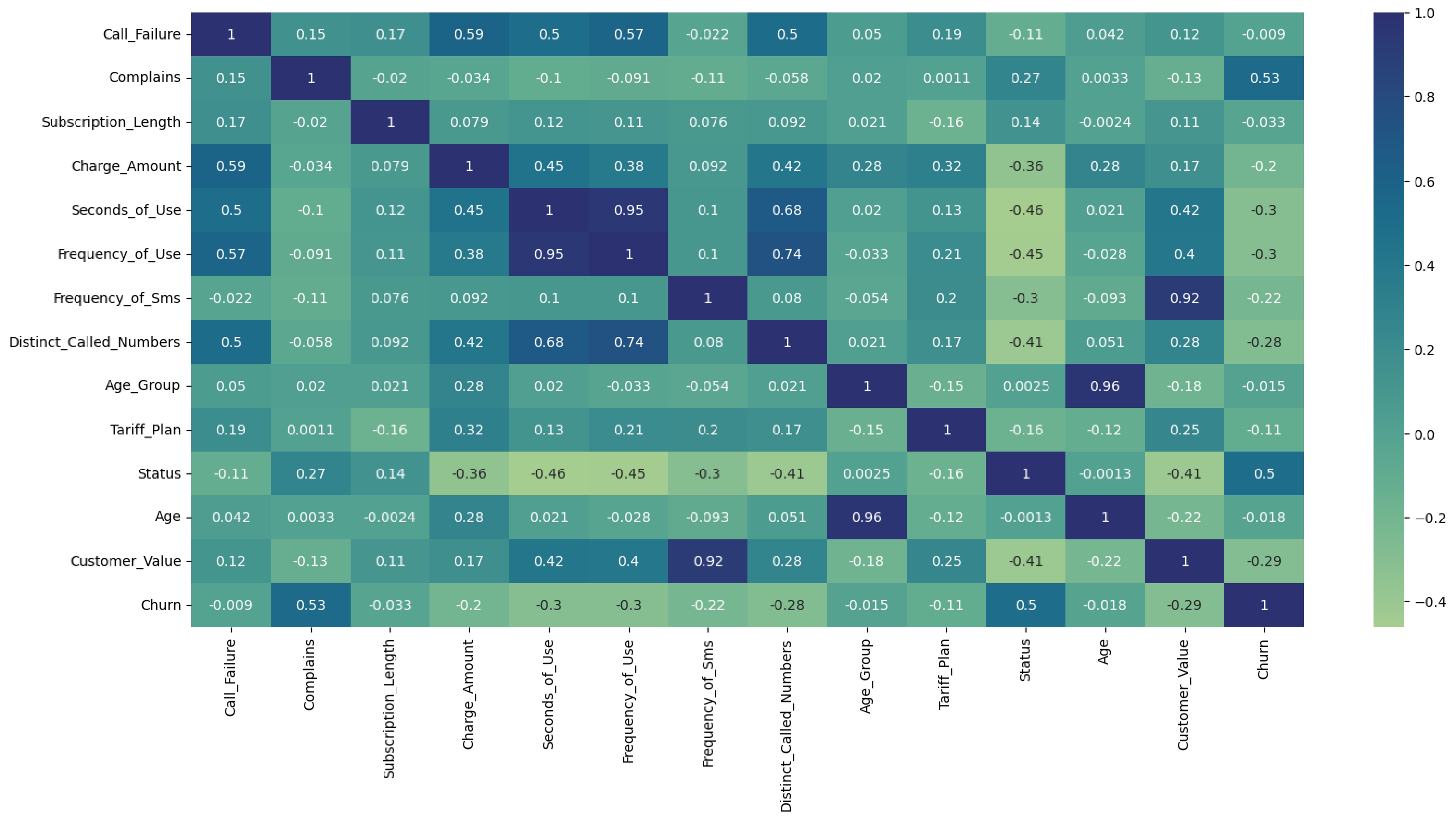
Performing a Explanatory Data Analysis

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



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

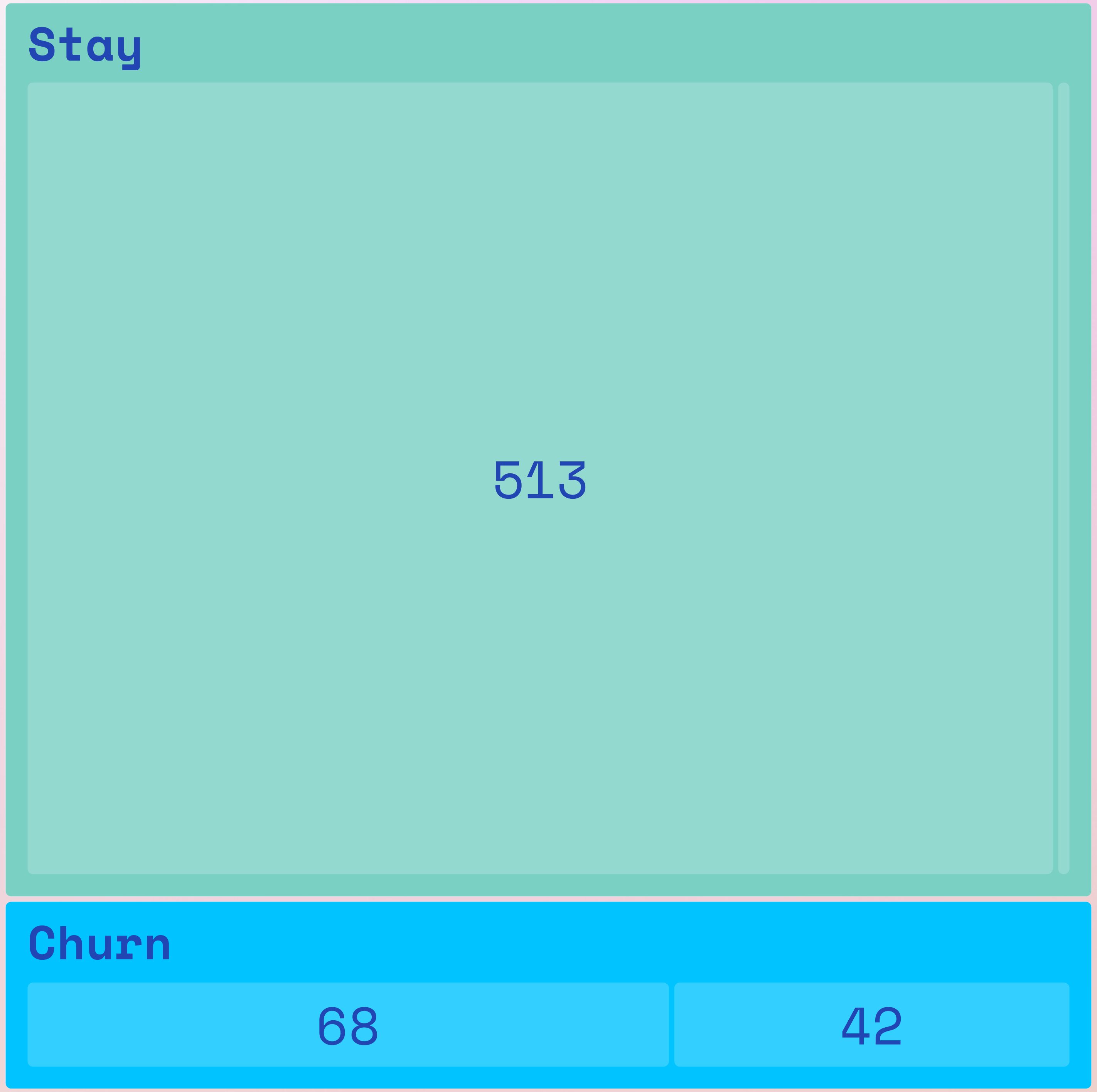
Three models to predict Churn

- 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.

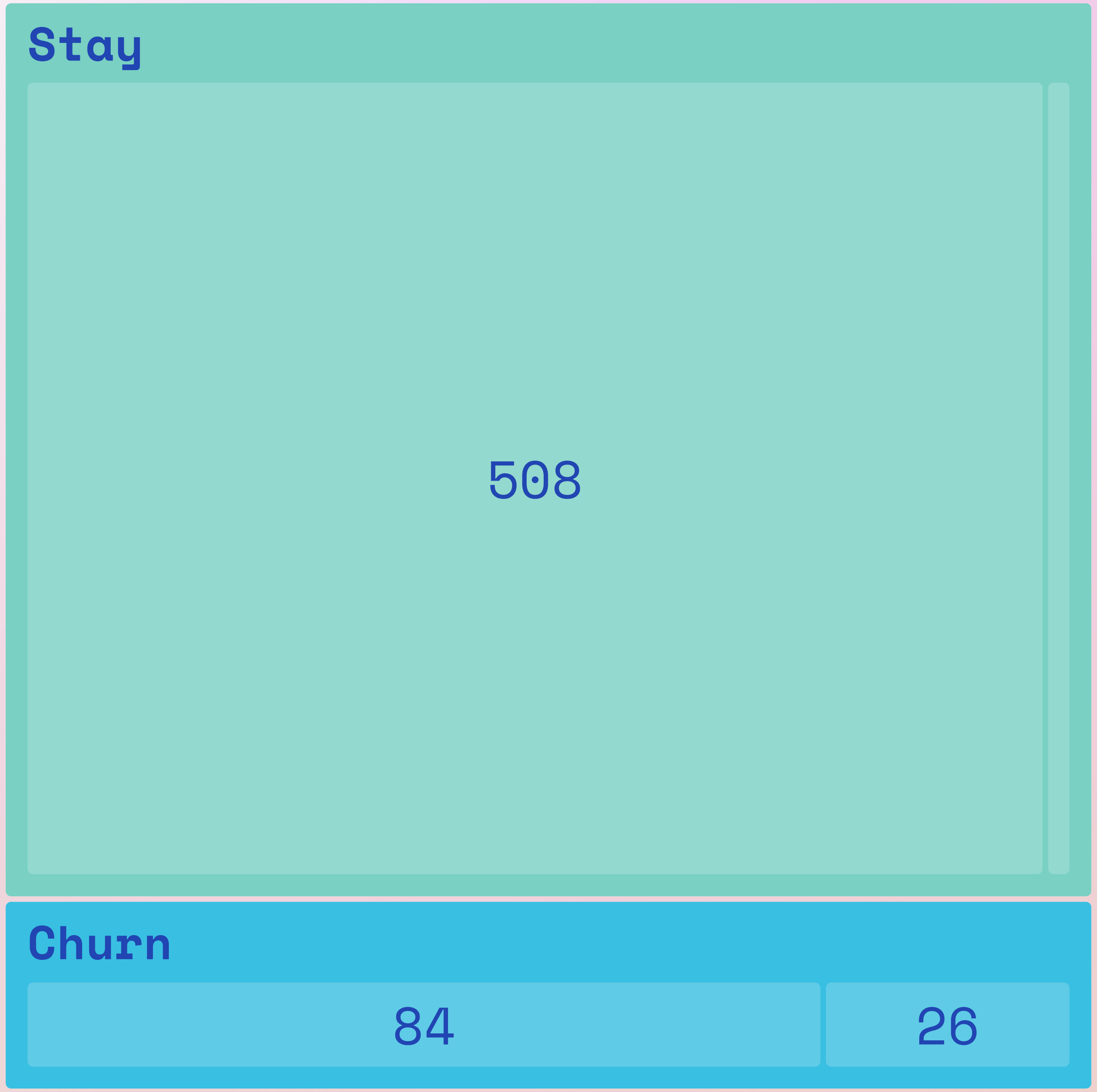
```
Best Hyperparameter : {
   'C': 0.1,
  'penalty': 'l1',
  'solver': 'liblinear'
}
```



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

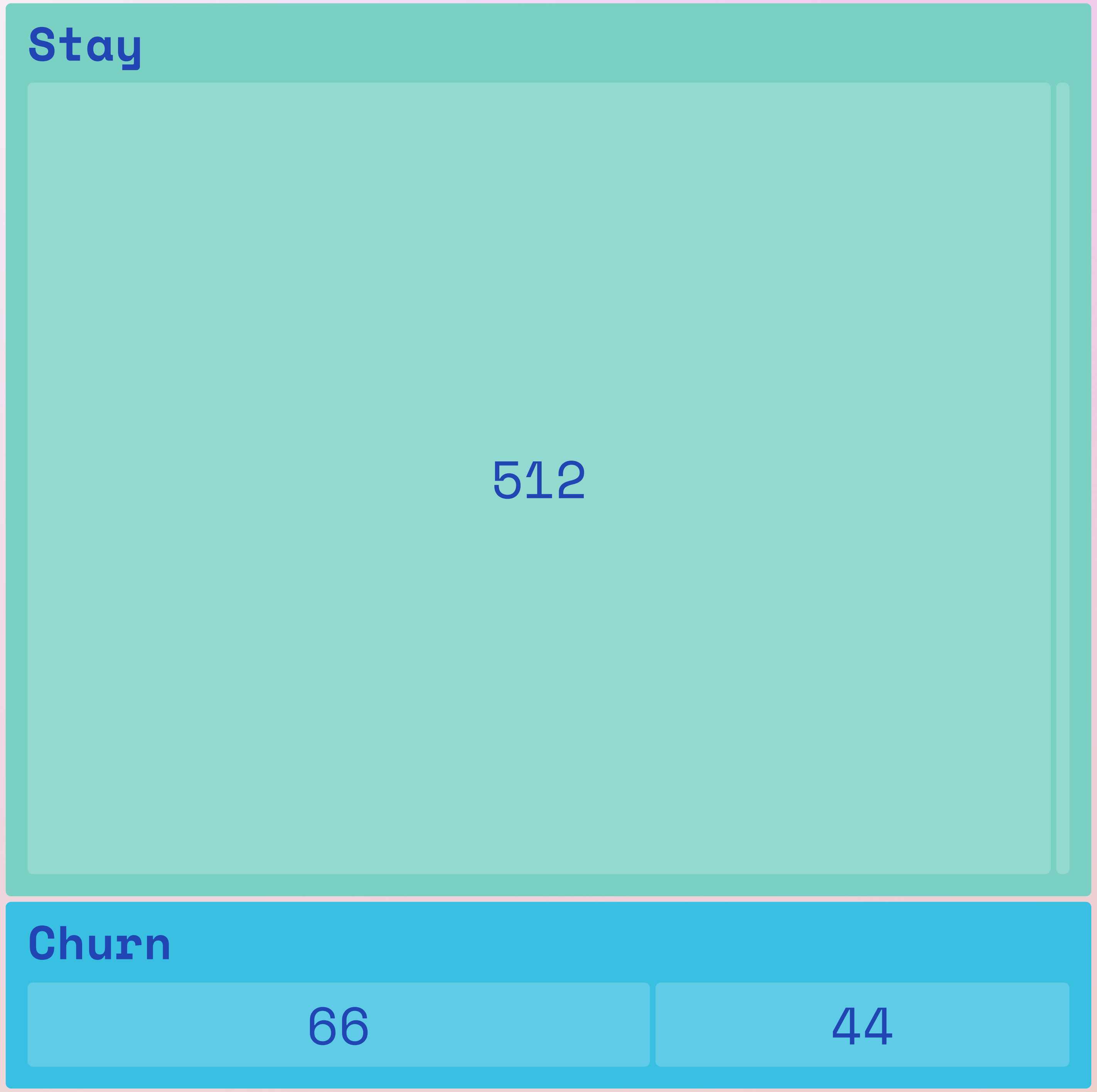
```
Best Hyperparameter: {
 'colsample_bytree': 0.8,
 'learning_rate': 0.2,
 'max_depth': 3,
 'n_estimators': 200,
 'subsample': 0.9
}
```



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

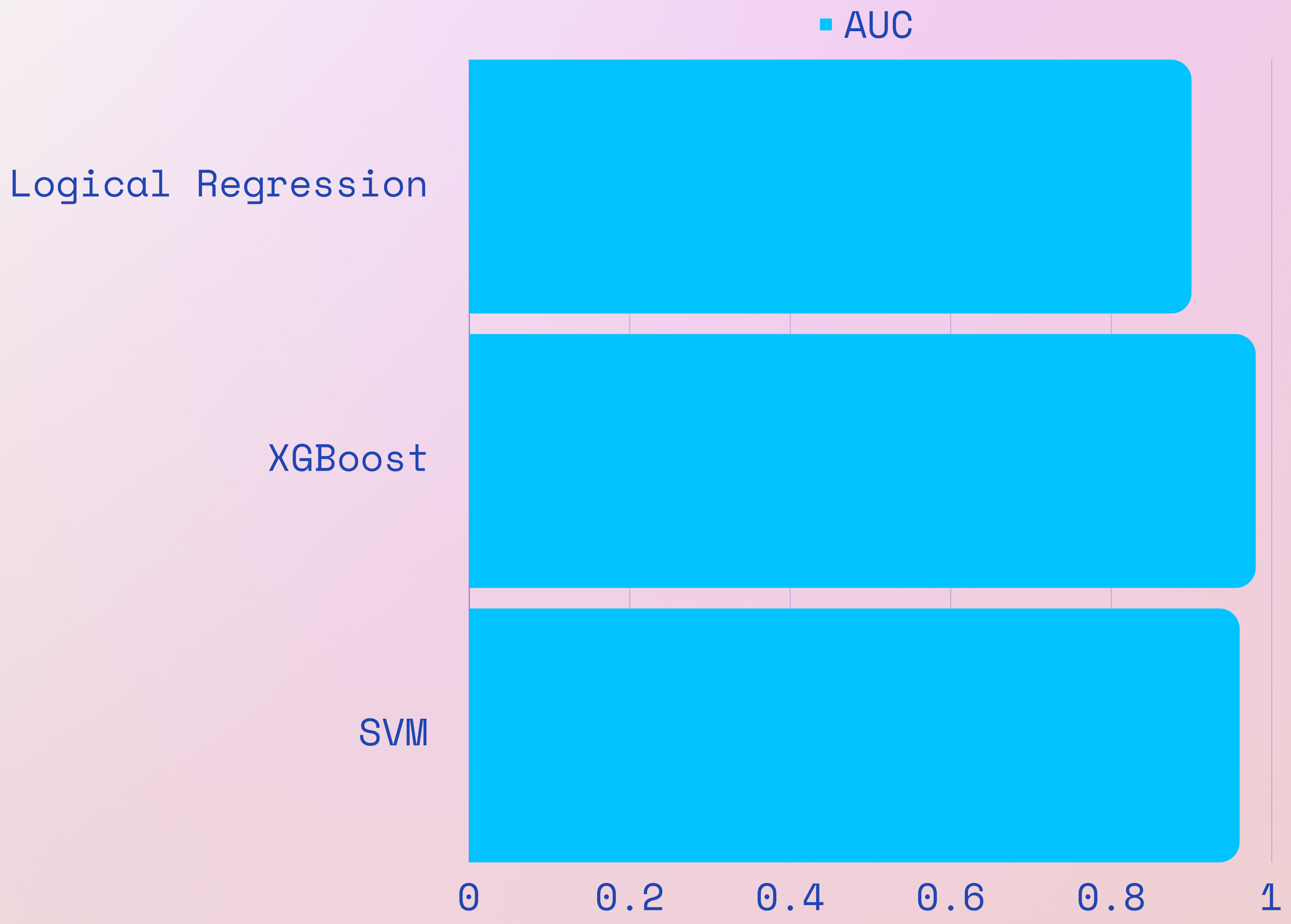
ROC and AUC

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

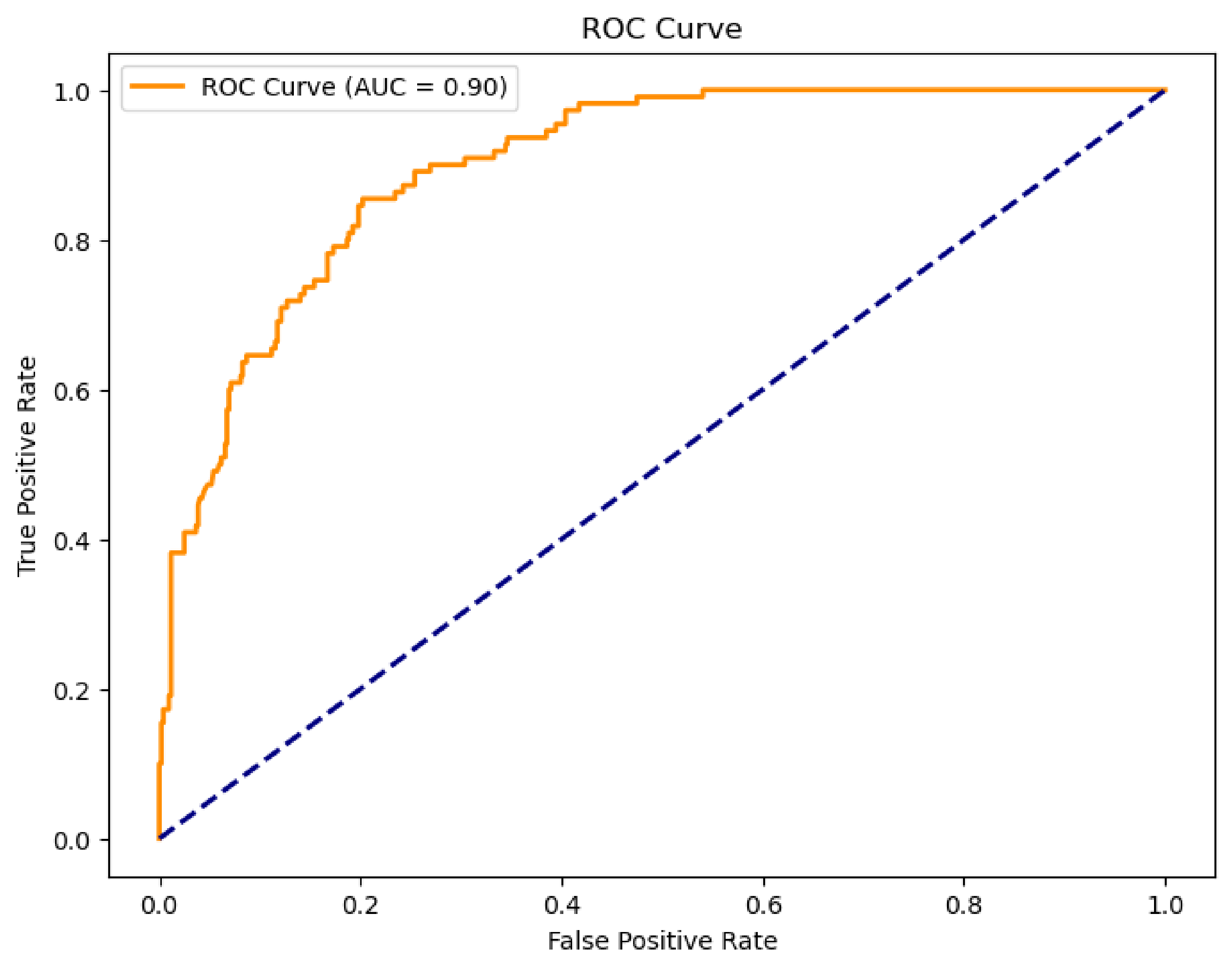


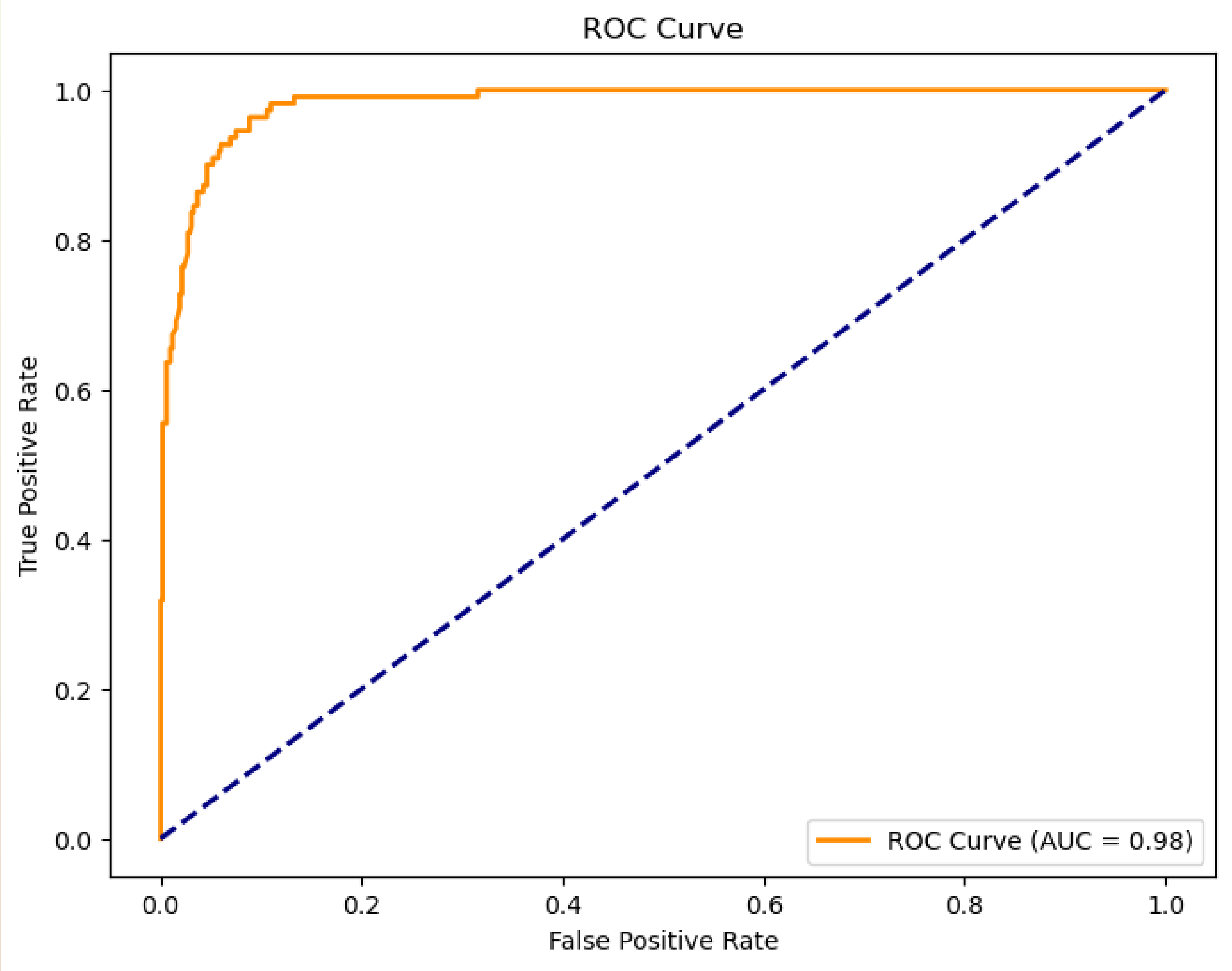
More details

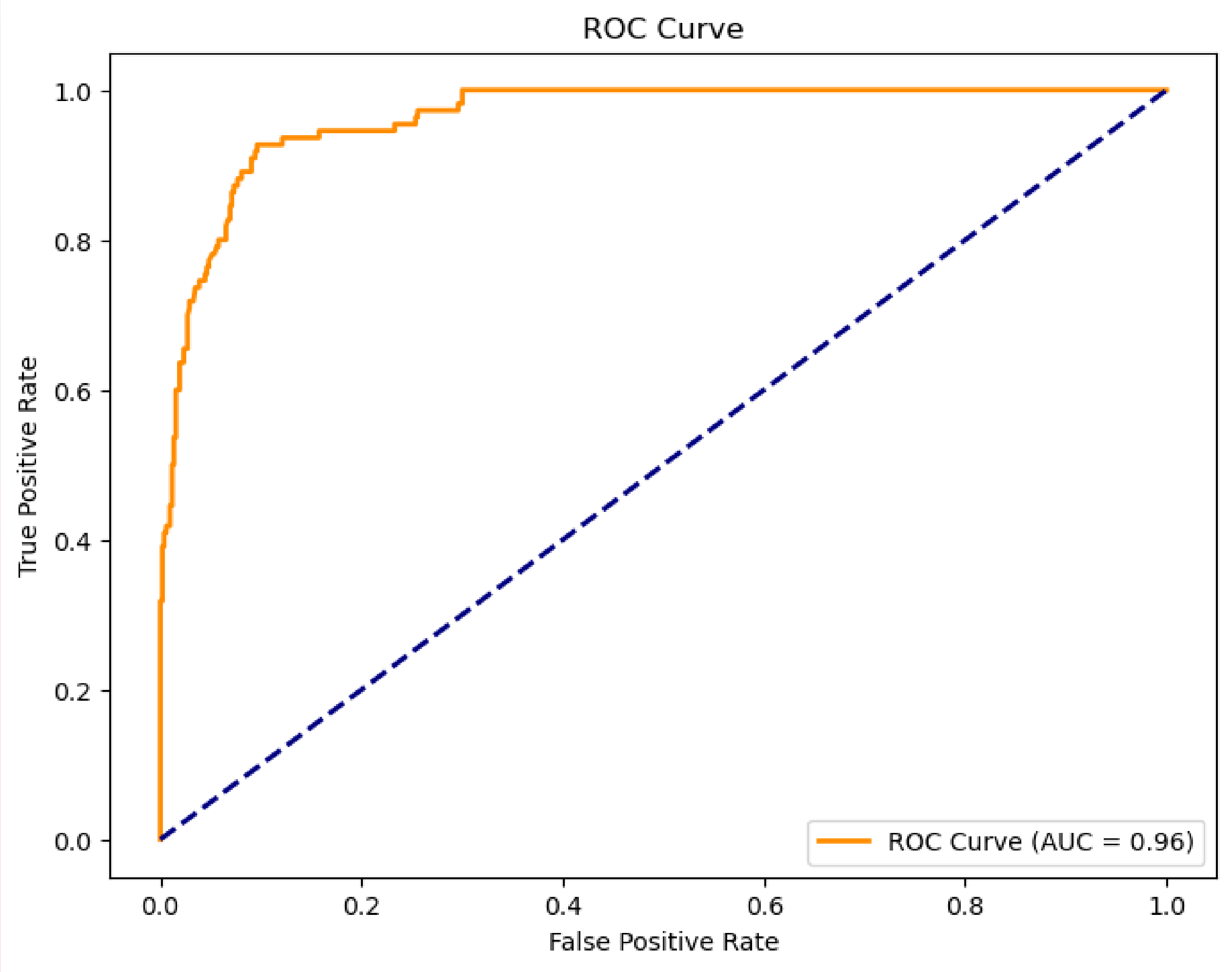
Logistic Regression

XGBoost

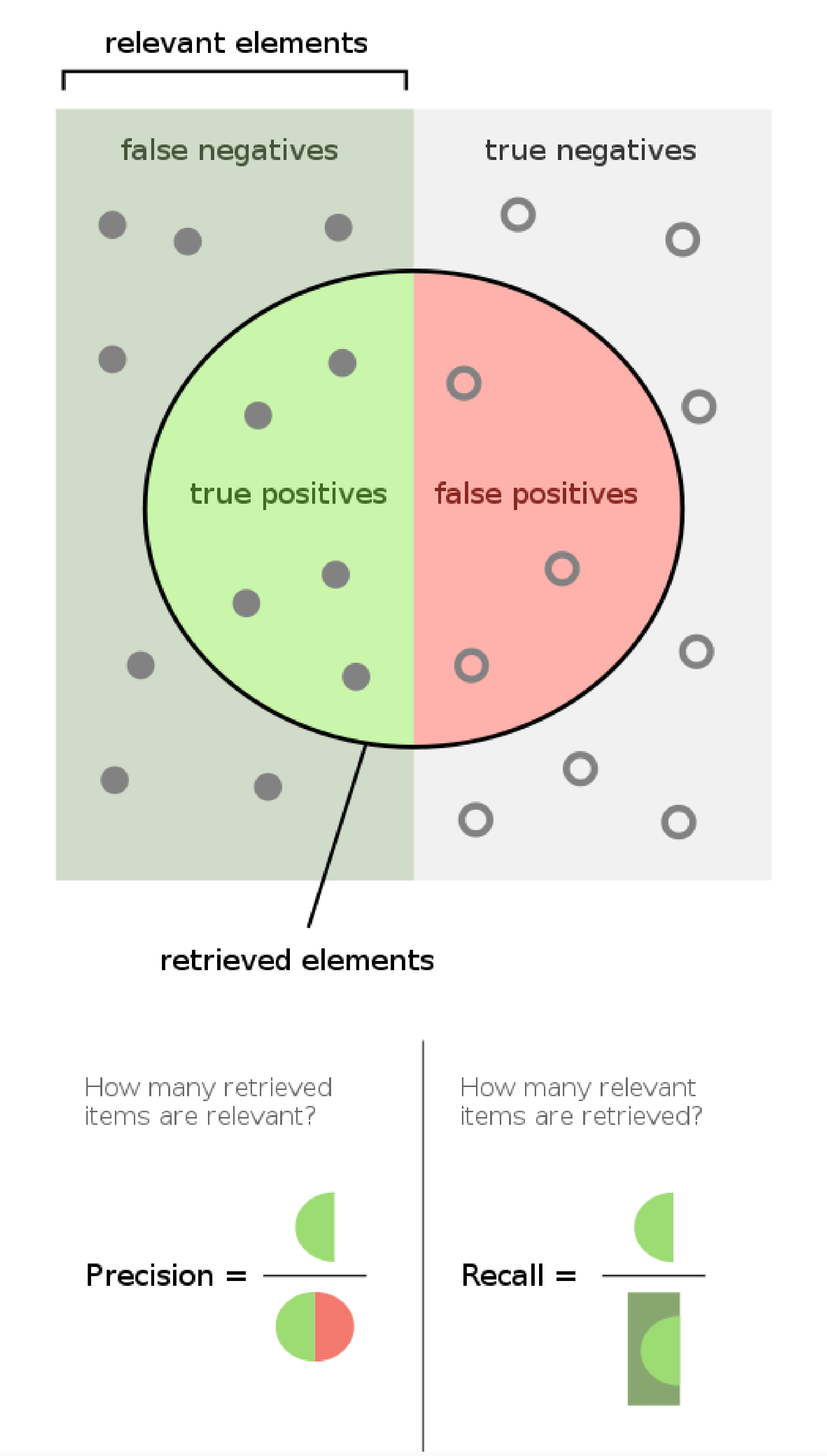
SVM



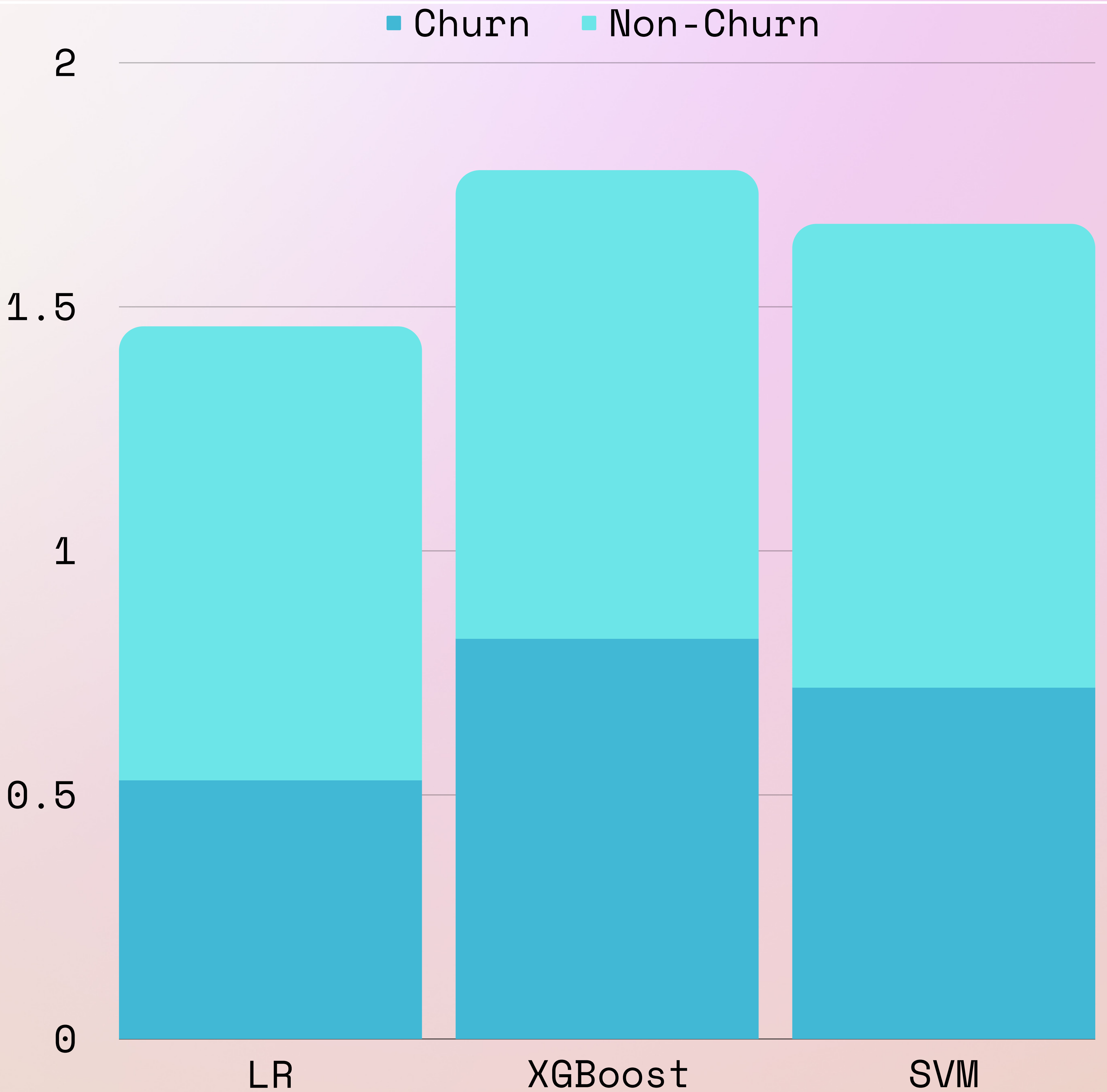




F1-Score



$$F_1 = \frac{2}{precision^{-1} + recall^{-1}}$$



04 - Presentation

Use Website under Flask to show XGBoost result

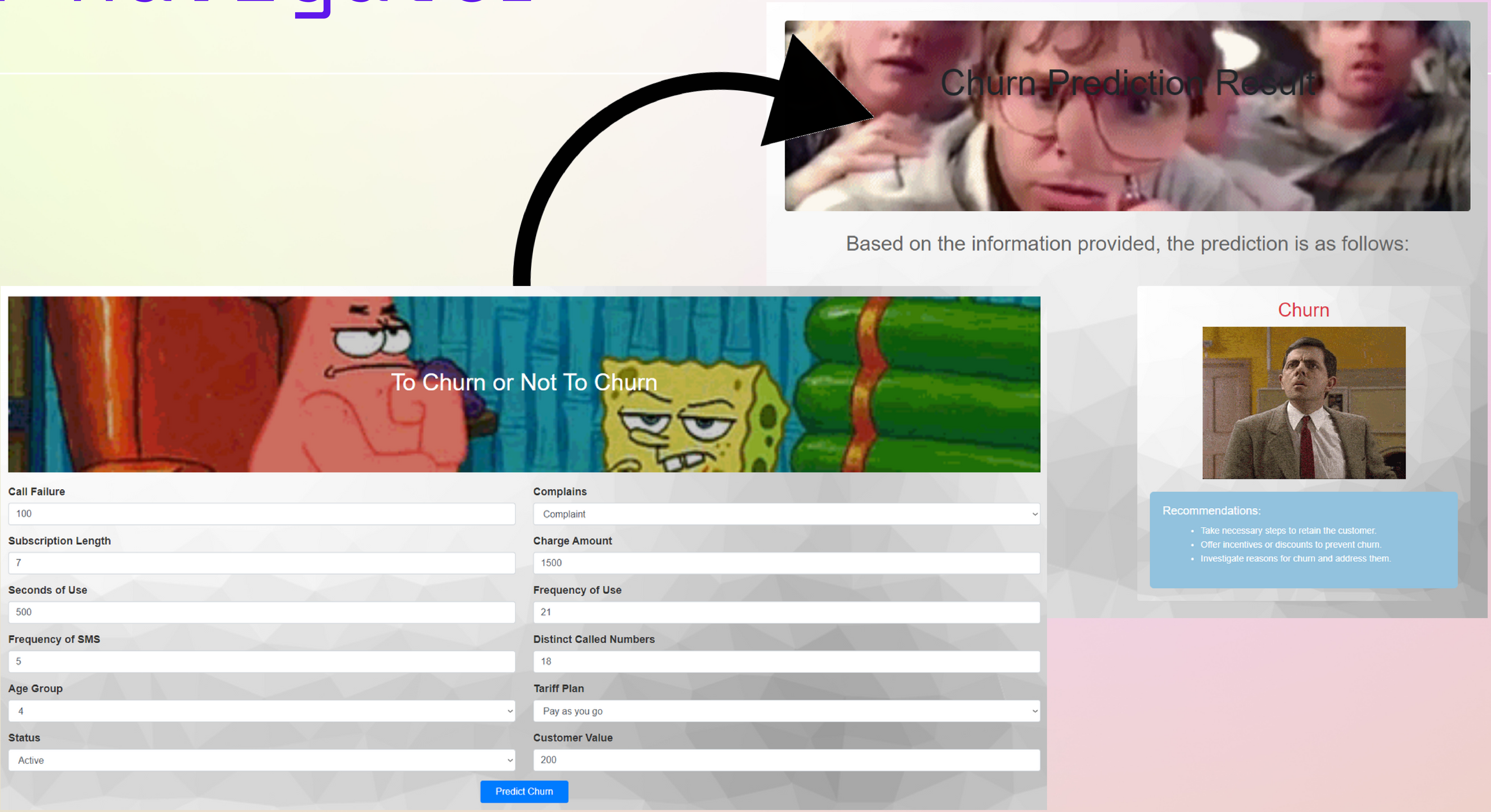
Use of Flask for it convenience for a simple website with a page for the form and an other one for the result

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

Main function

```python
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



Thanks