جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL

كلية العلوم والتقنيات
FACULTÉ DES SCIENCES ET TECHNIQUES

MASTER EN SCIENCES ET TECHNIQUES
SIDI
SYSTÈMES D'INFORMATION DÉCISIONNELS ET IMAGERIE
FACULTÉ DES SCIENCES ET TECHNIQUES ERRACHIDIA

Master Système d'informations décisionnels et imagerie

Module : Data Mining

Prévision du taux de désabonnement des voyageurs

Réaliser par :

- AHNNAOU Khalid

Responsable :

- Pr. Sabiri Mohammed

Plan

Intelligence Artificielle (IA) dans le Tourisme :

L'intelligence artificielle révolutionne le secteur du tourisme en permettant une analyse approfondie des données. Dans notre contexte, elle nous offre la capacité de prédire les comportements de désabonnement des clients en se basant sur des modèles sophistiqués, améliorant ainsi la capacité à anticiper les besoins et à personnaliser les stratégies de rétention.

Importance de la Rétention Client :

La rétention client demeure cruciale pour la stabilité financière de l'entreprise. En utilisant les avancées de l'intelligence artificielle, nous sommes en mesure de plonger plus profondément dans les motifs de désabonnement, ce qui nous permet d'optimiser les réservations d'hôtel et d'atteindre un niveau supérieur de satisfaction client.

01 Introduction et Problématique

Problématique :

Une entreprise de tourisme et de voyage souhaite prédire si un client va se désabonner ou non. Elle se base sur quelques caractéristiques du client, comme son âge, son statut de grand voyageur, sa classe de revenu annuel, les services qu'il a choisis, son compte sur les médias sociaux, l'hôtel qu'il a réservé ou non.

Caractéristiques Analytiques :

Nous analysons plusieurs caractéristiques pour prédire le désabonnement :

- Age du client.
- Statut de grand voyageur.
- Classe de revenu annuel.
- Services choisis.
- Présence sur les médias sociaux.
- Réservation d'hôtel.

	Age	FrequentFlyer	AnnualIncomeClass	ServicesOpted	AccountSyncedToSocialMedia	BookedHotelOrNot	Target
0	34	No	Middle Income	6	No	Yes	0
1	34	Yes	Low Income	5	Yes	No	1
2	37	No	Middle Income	3	Yes	No	0
3	30	No	Middle Income	2	No	No	0
4	30	No	Low Income	1	No	No	0

Visualisation :

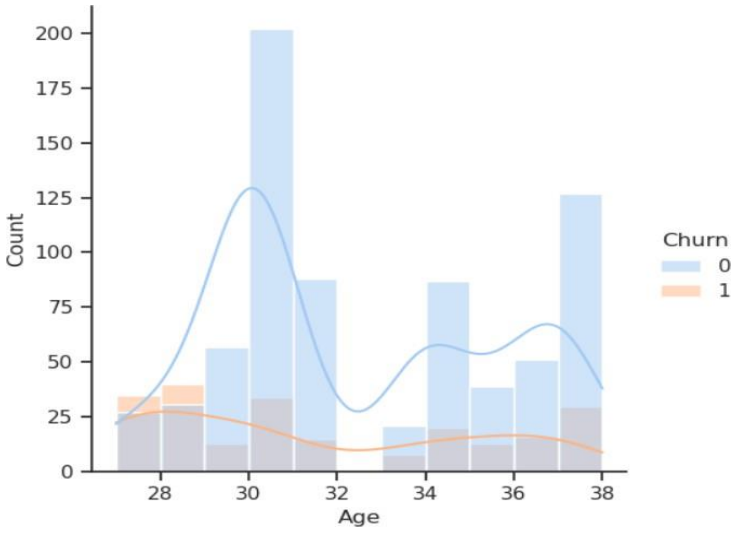
Visualisation des variables numériques :

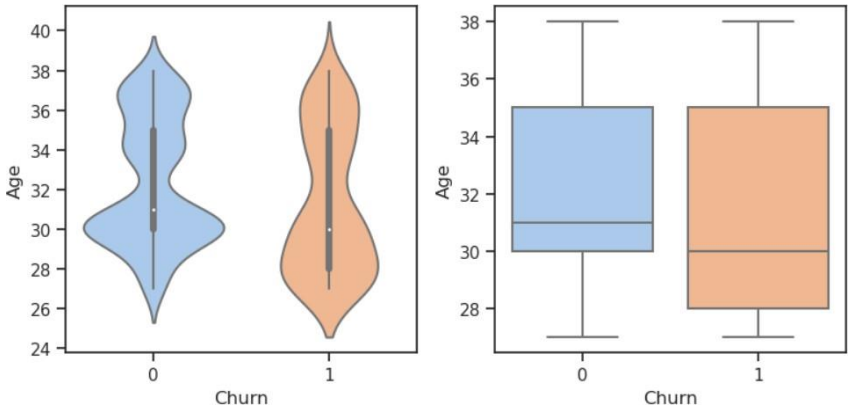
```
# Split the numeric and categorical features
num_features = ['Age']
ordinal_features = ['AnnualIncomeClass', 'ServicesOpted']
cat_features = ['FrequentFlyer', 'AccountSyncedToSocialMedia', 'BookedHotelOrNot', 'Churn']
```

```
sns.displot(data=df, x='Age', hue='Churn', kde=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x192eebfabe0>
```

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 4))
ax1 = sns.violinplot(data=df, x='Churn', y='Age', ax=ax1)
ax2 = sns.boxplot(data=df, x='Churn', y='Age', ax=ax2)
plt.tight_layout();
```





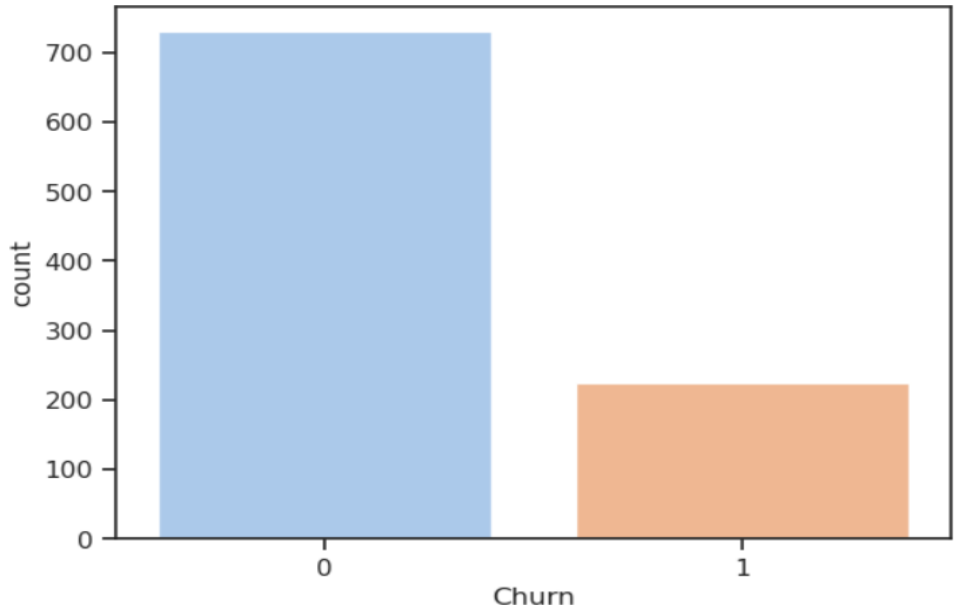
Visualisation :

Visualisation des variables Catégorielles :

```
df['Churn'].value_counts()
```

```
0    730
1    224
Name: Churn, dtype: int64
```

```
sns.countplot(data=df, x='Churn')
```

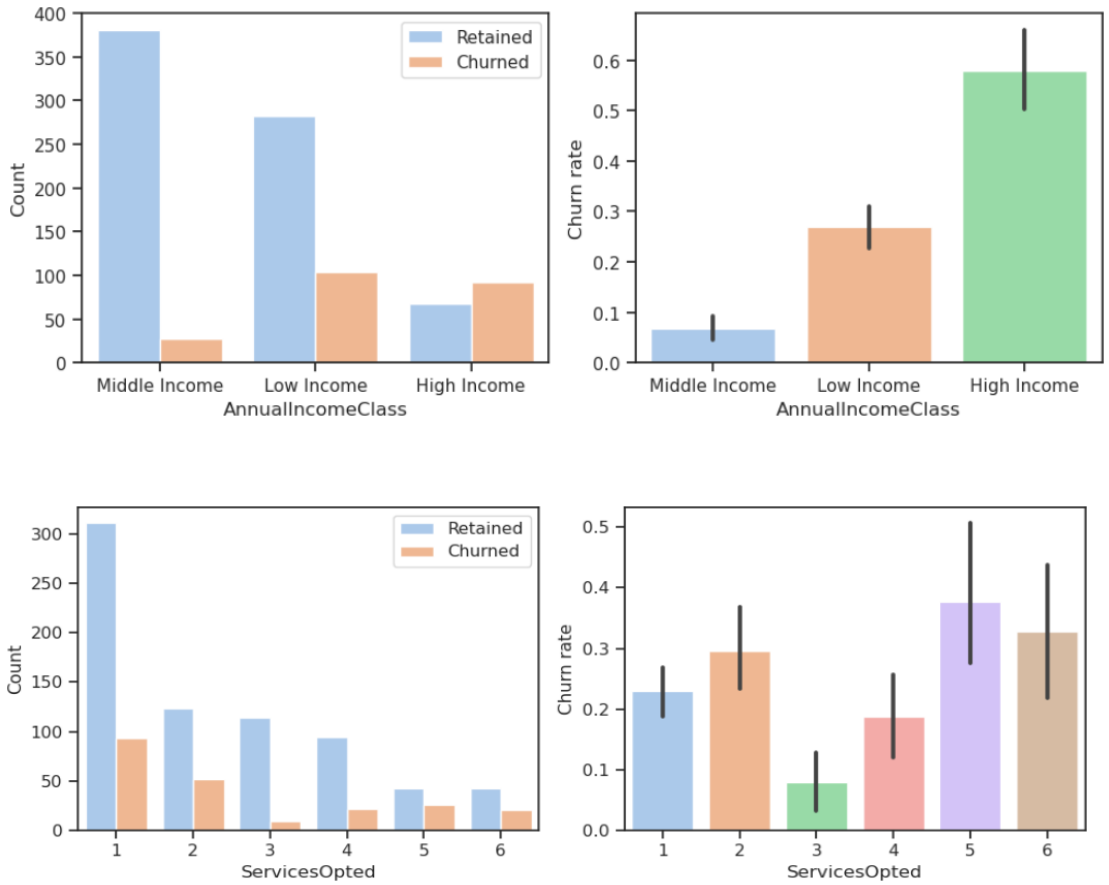


Visualisation :

Visualisation des variables Catégorielles :

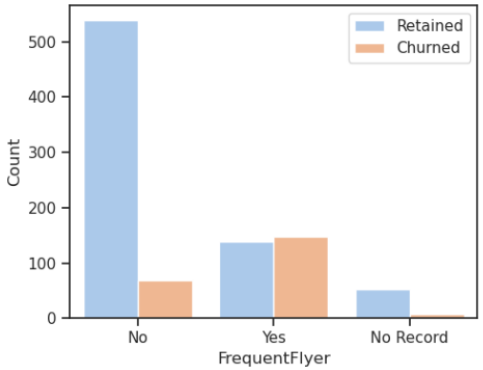
```
def plot_categorical(feature):
    '''For a categorical feature, plot a seaborn.countplot for the total counts of each category next to a barplot for the
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 4))

    sns.countplot(x=feature,
                  hue='Churn',
                  data=df,
                  ax=ax1)
    ax1.set_ylabel('Count')
    ax1.legend(labels=['Retained', 'Churned'])

    sns.barplot(x=feature,
                y='Churn',
                data=df,
                ax=ax2)
    ax2.set_ylabel('Churn rate')


    plt.tight_layout();
```

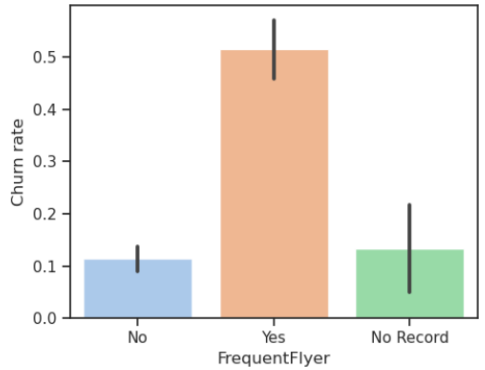
```
for feature in (ordinal_features + cat_features[:-1]):
    plot_categorical(feature)
```

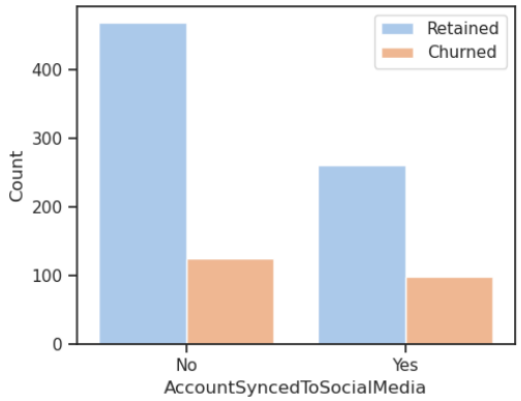


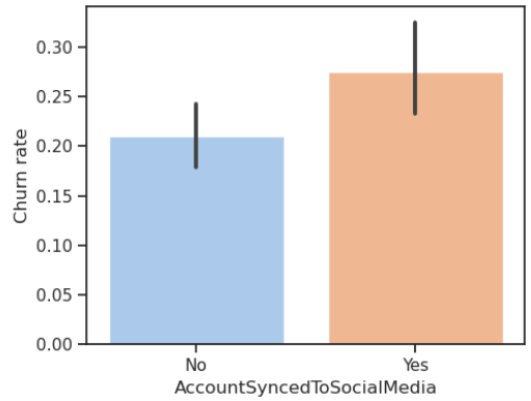
Visualisation :

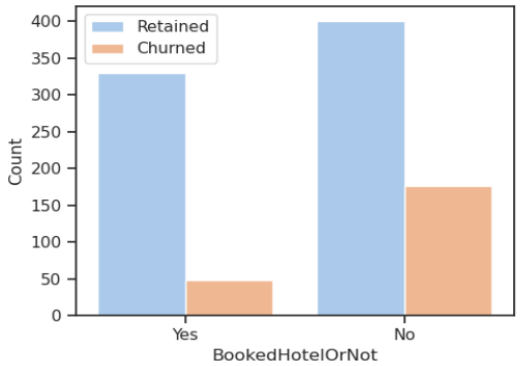
Visualisation des variables Catégorielles :

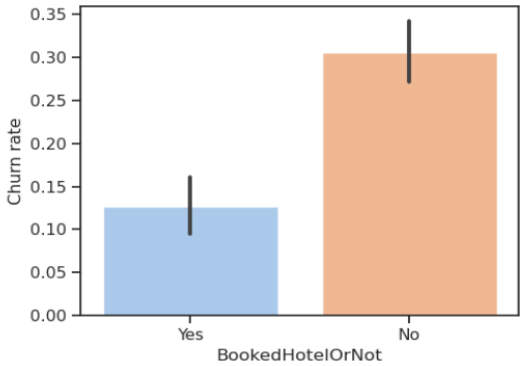












Prétraitement des données :

One-hot encoding pour les variables catégorielles :

```
cat_data = pd.DataFrame()
for feature in cat_features[:-1]:
    temp = pd.get_dummies(df[feature], prefix=feature)
    cat_data = pd.concat([cat_data, temp], axis=1)
```

Ordinal-encoding pour les variables ordinales :

```
df['AnnualIncomeClass'] = df['AnnualIncomeClass'].map({'Low Income':0,
                                                       'Middle Income':1,
                                                       'High Income':2})
```

Min-max scale pour les variables numériques :

```
features_to_scale = ['Age']
df[features_to_scale] = MinMaxScaler().fit_transform(df[features_to_scale])

scaler =MinMaxScaler()

df[num_features] = scaler.fit_transform(df[num_features])
```

Entraînement de modèle :

```
# merging the data
X = pd.concat([cat_data, df[ordinal_features], df[num_features]], axis=1)
```

```
y = df['Churn']
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8)
```

Modélisation :

```
%%capture
!pip install lazypredict
```

```
from lazypredict.Supervised import LazyClassifier
```

```
clf = LazyClassifier(verbose=0,ignore_warnings=True, custom_metric=None, random_state=100)
models,predictions = clf.fit(X_train, X_test, y_train, y_test)
```

```
100%|██████████| 29/29 [00:03<00:00,  8.80it/s]

[LightGBM] [Info] Number of positive: 178, number of negative: 585
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.000494 seconds.
```

Comparaison des modèles :

```
models.sort_values('F1 Score', ascending=False)
```

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
DecisionTreeClassifier	0.91	0.85	0.85	0.91	0.03
BaggingClassifier	0.91	0.84	0.84	0.90	0.09
XGBClassifier	0.90	0.83	0.83	0.89	0.19
RandomForestClassifier	0.90	0.83	0.83	0.89	0.54
LGBMClassifier	0.89	0.82	0.82	0.89	0.17
ExtraTreesClassifier	0.89	0.82	0.82	0.89	0.52
LabelSpreading	0.87	0.81	0.81	0.87	0.13
LabelPropagation	0.87	0.81	0.81	0.87	0.07
ExtraTreeClassifier	0.87	0.80	0.80	0.87	0.02
KNeighborsClassifier	0.85	0.72	0.72	0.83	0.07
AdaBoostClassifier	0.82	0.70	0.70	0.81	0.41
GaussianNB	0.78	0.71	0.71	0.78	0.03
BernoulliNB	0.78	0.70	0.70	0.78	0.05
SVC	0.80	0.65	0.65	0.78	0.12
SGDClassifier	0.79	0.67	0.67	0.78	0.04
LinearDiscriminantAnalysis	0.79	0.66	0.66	0.77	0.05
LogisticRegression	0.80	0.62	0.62	0.76	0.06
NearestCentroid	0.75	0.70	0.70	0.76	0.03
Perceptron	0.75	0.69	0.69	0.76	0.03

Comparaison des modèles :

```
models = []
models.append(('LR', LogisticRegression(random_state = 100)))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier(random_state = 100)))
models.append(('RF', RandomForestClassifier(random_state = 100)))
models.append(('SVM', SVC(gamma='auto', random_state = 100)))
models.append(('XGB', GradientBoostingClassifier(random_state = 100)))
models.append(("LightGBM", LGBMClassifier(random_state = 100)))
models.append(("CatBoost", CatBoostClassifier(random_state = 100, verbose = False)))

# evaluate each model in turn
results = []
names = []
```

```
for name, model in models:
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        msg = "%s: (%f)" % (name, accuracy)
        print(msg)
```

```
LR: (0.790576)
KNN: (0.869110)
CART: (0.910995)
RF: (0.895288)
SVM: (0.806283)
XGB: (0.905759)
```

Random forest le plus performant :

```
model = RandomForestClassifier(random_state=100)
param_grid = {
    'n_estimators': [100],
    'criterion': ['entropy', 'gini'],
    'bootstrap': [True, False],
    'max_depth': [6],
    'max_features': ['auto', 'sqrt'],
    'min_samples_leaf': [2, 3, 5],
    'min_samples_split': [2, 3, 5]
}
```

```
from sklearn.model_selection import GridSearchCV
rf_clf = GridSearchCV(estimator=model,
                      param_grid=param_grid,
                      scoring='balanced_accuracy',
                      cv=5,
                      verbose=False,
                      n_jobs=-1)

best_rf_clf = rf_clf.fit(X_train, y_train)
```

```
best_rf_clf
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=100),
             n_jobs=-1,
             param_grid={'bootstrap': [True, False],
                         'criterion': ['entropy', 'gini'], 'max_depth': [6],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [2, 3, 5],
                         'min_samples_split': [2, 3, 5],
                         'n_estimators': [100]},
             scoring='balanced_accuracy', verbose=False)
```

Résultats de modèle Random forest :

```
y_pred = best_rf_clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

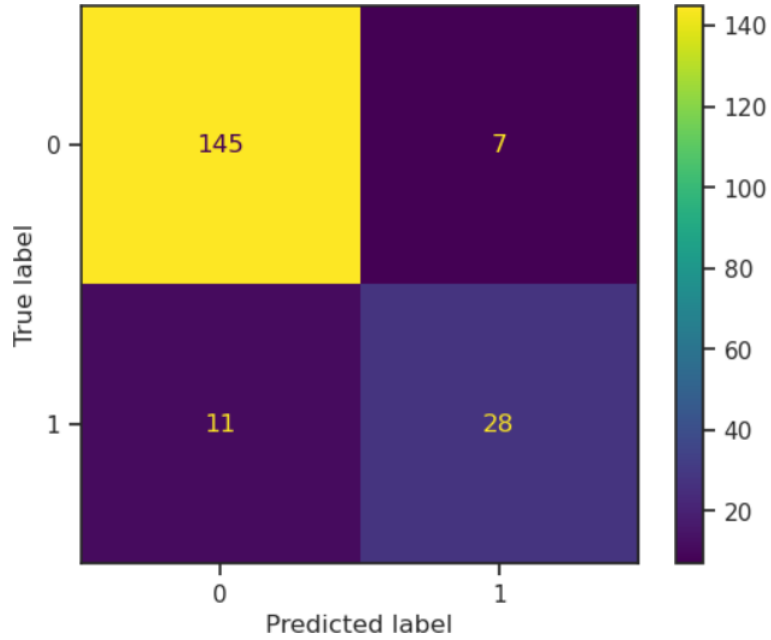
```
Accuracy: 0.8691099476439791
```

```
cm = confusion_matrix(y_test, y_pred, labels=best_rf_clf.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=best_rf_clf.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x192ebc59a60>
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.97      0.92       145
           1       0.86      0.54      0.67        46

    accuracy                           0.87       191
   macro avg       0.87      0.76      0.79       191
weighted avg       0.87      0.87      0.86       191
```



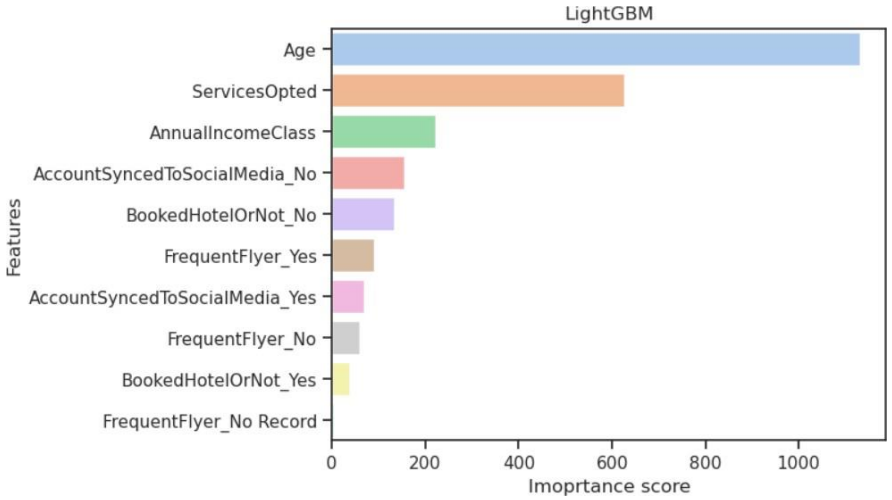
Analyse des factures :

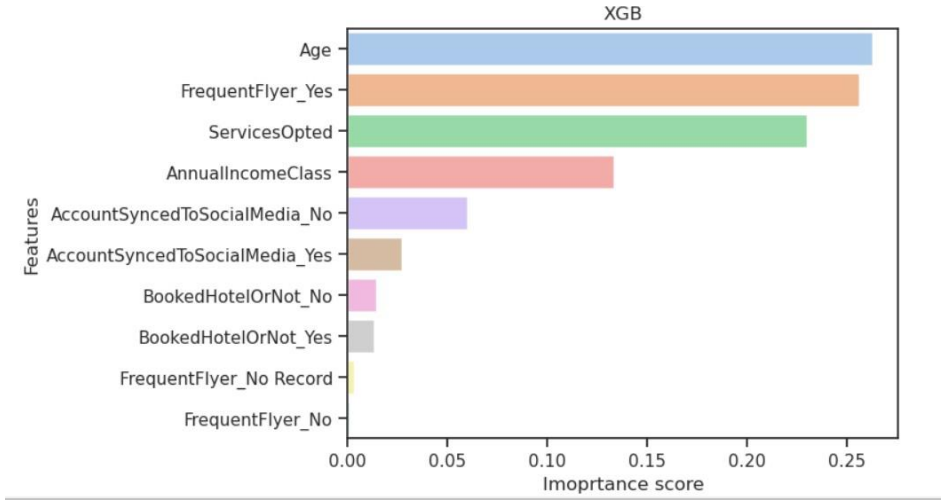
```
models2 = []
models2.append(('DecisionTree', DecisionTreeClassifier(random_state = 42)))
models2.append(('RF', RandomForestClassifier( random_state = 42)))
models2.append(('XGB', GradientBoostingClassifier( random_state = 42)))
models2.append(("LightGBM", LGBMClassifier( random_state = 42)))
models2.append(("CatBoost", CatBoostClassifier(random_state = 42, verbose = False)))
```

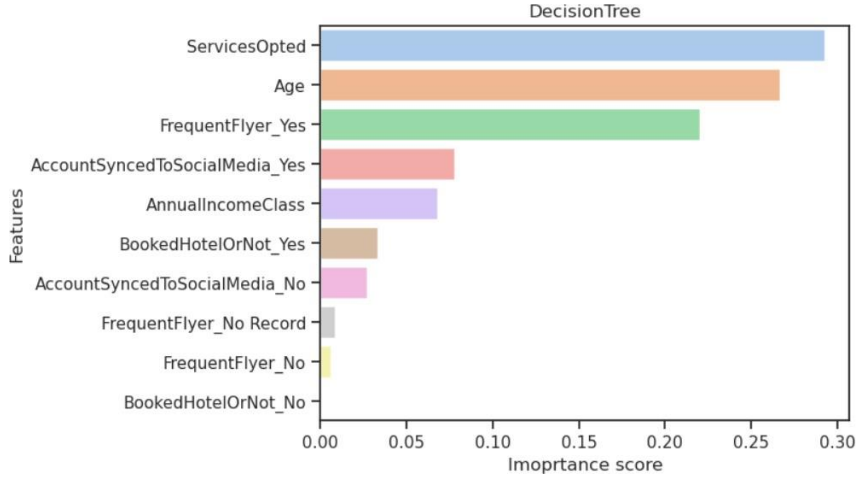
```
for name, model in models2:
        base = model.fit(X_train,y_train)
        y_pred = base.predict(X_test)
        acc_score = accuracy_score(y_test, y_pred)
        feature_imp = pd.Series(base.feature_importances_,
                        index=X.columns).sort_values(ascending=False)

        sns.barplot(x=feature_imp, y=feature_imp.index)
        plt.xlabel('Imoprtance score')
        plt.ylabel('Features')
        plt.title(name)
        plt.show()
```

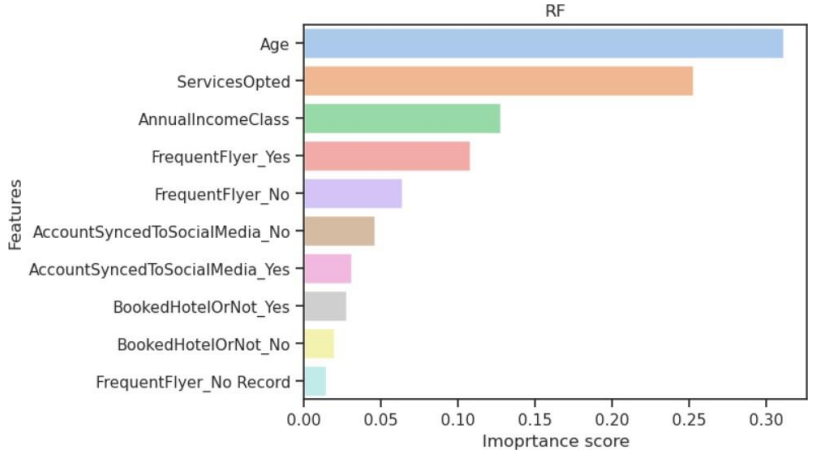
Résultats

Analyse des factures :









07 Conclusion

En conclusion, anticiper et réduire le désabonnement client est essentiel pour le succès financier à long terme. En intégrant ces pratiques dans la stratégie globale, nous sommes mieux positionnés pour offrir une expérience client exceptionnelle tout en maximisant la rentabilité de l’entreprise.