Projet de Modélisation Supervisée : Prédiction du Statut des Réservations

Contexte

Une entreprise de transport (type Uber/Bolt) rencontre des pertes importantes à cause de réservations annulées, incomplètes ou sans chauffeur disponible.
L'objectif est de prédire à l'avance si une réservation sera complétée ou non, en fonction des caractéristiques du trajet, afin de réduire les pertes et améliorer la satisfaction client.

Problématique Métier

Comment prédire le statut d'une réservation (`Completed` ou `Not Completed`) en utilisant les informations disponibles sur la course : type de véhicule, distance, localisation, méthode de paiement, notes des clients et des chauffeurs, etc. ?

Parties Prenantes

- **Direction de l'entreprise** : souhaite réduire les pertes financières liées aux annulations.
- **Équipe opérationnelle** : souhaite comprendre les causes principales des annulations pour ajuster la planification des chauffeurs.
- **Clients et chauffeurs** : bénéficient d'une meilleure expérience grâce à moins d'annulations surprises.

Objectifs du Projet

1. Explorer et nettoyer le dataset pour préparer les données.
2. Construire un modèle de classification supervisé pour prédire le statut d'une réservation.
3. Interpréter les résultats pour identifier les facteurs influençant les annulations.
4. Analyser les raisons textuelles des annulations à l'aide de NLP.
5. Estimer la valeur des réservations complétées via un modèle de régression.

Plan de Travail

1. Préparation des données (nettoyage, encodage, création de variables).
2. Exploration des données (EDA) et visualisations.
3. Modélisation supervisée (Random Forest, XGBoost, etc.).
4. Évaluation des modèles (accuracy, F1-score, confusion matrix).
5. Interprétation des résultats (SHAP).

6. Conclusion et recommandations business.

In [ ]:

1. Chargement du Dataset et Aperçu des Données

Dans cette section, nous allons :

- Charger le fichier CSV contenant les réservations.
- Afficher les premières lignes pour comprendre la structure des données.
- Identifier les colonnes et les types de données.
- Vérifier la présence de valeurs manquantes.

In [92]:
```
# Importation des bibliothèques nécessaires
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
#from sklearn.impute import SimpleImputer
import shap

# Charger le dataset
df = pd.read_csv("ncr_ride_bookings.csv")

# Afficher un aperçu des 5 premières lignes
df.head()

#Copier le dataset original pour ne pas perdre les données brutes
#data = df.copy()
```

Out[92]:

	Date	Time	Booking ID	Booking Status	Customer ID	Vehicle Type	Pickup Location	Drop Location
0	2024-03-23	12:29:38	"CNR5884300"	No Driver Found	"CID1982111"	eBike	Palam Vihar	Jhilmil
1	2024-11-29	18:01:39	"CNR1326809"	Incomplete	"CID4604802"	Go Sedan	Shastri Nagar	Gurgaon Sector 56
2	2024-08-23	08:56:10	"CNR8494506"	Completed	"CID9202816"	Auto	Khandsa	Malviya Nagar
3	2024-10-21	17:17:25	"CNR8906825"	Completed	"CID2610914"	Premier Sedan	Central Secretariat	Inderlok
4	2024-09-16	22:08:00	"CNR1950162"	Completed	"CID9933542"	Bike	Ghitorni Village	Khan Market

5 rows × 21 columns

In [ ]:

In [3]:
```
# Informations sur le dataset : colonnes, types de données, valeurs non nulles
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 21 columns):
 #   Column                               Non-Null Count   Dtype  
---  ------                               --------------   -----  
 0   Date                                 150000 non-null  object 
 1   Time                                 150000 non-null  object 
 2   Booking ID                           150000 non-null  object 
 3   Booking Status                       150000 non-null  object 
 4   Customer ID                          150000 non-null  object 
 5   Vehicle Type                         150000 non-null  object 
 6   Pickup Location                      150000 non-null  object 
 7   Drop Location                        150000 non-null  object 
 8   Avg VTAT                             139500 non-null  float64
 9   Avg CTAT                             102000 non-null  float64
 10  Cancelled Rides by Customer          10500 non-null   float64
 11  Reason for cancelling by Customer    10500 non-null   object 
 12  Cancelled Rides by Driver            27000 non-null   float64
 13  Driver Cancellation Reason           27000 non-null   object 
 14  Incomplete Rides                     9000 non-null    float64
 15  Incomplete Rides Reason              9000 non-null    object 
 16  Booking Value                        102000 non-null  float64
 17  Ride Distance                        102000 non-null  float64
 18  Driver Ratings                       93000 non-null   float64
 19  Customer Rating                      93000 non-null   float64
 20  Payment Method                       102000 non-null  object 
dtypes: float64(9), object(12)
memory usage: 24.0+ MB
```

In [ ]:

In [5]:
```
# Vérifier le nombre de valeurs manquantes par colonne
df.isnull().sum()
```

Out[5]:
```
Date                                  0
Time                                  0
Booking ID                            0
Booking Status                        0
Customer ID                           0
Vehicle Type                          0
Pickup Location                       0
Drop Location                         0
Avg VTAT                          10500
Avg CTAT                          48000
Cancelled Rides by Customer      139500
Reason for cancelling by Customer    139500
Cancelled Rides by Driver        123000
Driver Cancellation Reason       123000
Incomplete Rides                 141000
Incomplete Rides Reason          141000
Booking Value                     48000
Ride Distance                     48000
Driver Ratings                    57000
Customer Rating                   57000
Payment Method                    48000
dtype: int64
```

In [ ]:

In [6]:
```
# Statistiques descriptives des colonnes numériques
df.describe()
```

Out[6]:

	Avg VTAT	Avg CTAT	Cancelled Rides by Customer	Cancelled Rides by Driver	Incomplete Rides	Booking Value	Ri
count	139500.000000	102000.000000	10500.0	27000.0	9000.0	102000.000000	102
mean	8.456352	29.149636	1.0	1.0	1.0	508.295912	
std	3.773564	8.902577	0.0	0.0	0.0	395.805774	
min	2.000000	10.000000	1.0	1.0	1.0	50.000000	
25%	5.300000	21.600000	1.0	1.0	1.0	234.000000	
50%	8.300000	28.800000	1.0	1.0	1.0	414.000000	
75%	11.300000	36.800000	1.0	1.0	1.0	689.000000	
max	20.000000	45.000000	1.0	1.0	1.0	4277.000000	

In [ ]:

In [ ]:

2. Préparation et Nettoyage des Données

Dans cette section, nous allons :

- Nettoyer les valeurs manquantes.
- Encoder les variables catégorielles.
- Créer la variable cible binaire `Booking_Status_Binary`.
- Préparer les données pour la modélisation.

In [ ]:

In [ ]:
```
Gestion des valeurs manquantes
```

In [93]:
```
# Colonnes numériques
numeric_cols = ['Ride Distance', 'Booking Value', 'Driver Ratings', 'Customer Ratin

# Remplacer les NaN par la médiane
for col in numeric_cols:
    data[col] = data[col].fillna(data[col].median())
```

In [94]:
```
# Colonnes numériques liées aux annulations/incompletes
num_cols = ['Cancelled Rides by Driver', 'Incomplete Rides']
for col in num_cols:
    data[col] = data[col].fillna(0)

# Colonnes textuelles liées aux raisons
text_cols = ['Driver Cancellation Reason', 'Incomplete Rides Reason']
for col in text_cols:
    data[col] = data[col].fillna('Unknown')
```

In [95]:
```
# Vérifier qu'il ne reste plus de valeurs manquantes
data[text_cols + num_cols].isnull().sum()
```

Out[95]:
```
Driver Cancellation Reason    0
Incomplete Rides Reason       0
Cancelled Rides by Driver     0
Incomplete Rides              0
dtype: int64
```

Création de la cible binaire

In [96]:
```
# Copier le dataset pour sécurité
model_data = data.copy()
# Créer la cible binaire
model_data['Booking_Status_Binary'] = model_data['Booking Status'].apply(lambda x:
# Identifier les colonnes non numériques
categorical_cols = model_data.select_dtypes(include=['object']).columns

# Numériques → median
num_cols = model_data.select_dtypes(include=['int64','float64']).columns
model_data[num_cols] = model_data[num_cols].fillna(model_data[num_cols].median())

# Catégorielles → "Unknown"
cat_cols = model_data.select_dtypes(include=['object']).columns
```

```
model_data[cat_cols] = model_data[cat_cols].fillna("Unknown")

# Encoder toutes les colonnes catégorielles
le = LabelEncoder()
for col in categorical_cols:
    model_data[col] = le.fit_transform(model_data[col].astype(str))

# Vérification finale
print(model_data.dtypes)
model_data.head()
```

```
Date                                    int32
Time                                    int32
Booking ID                              int32
Booking Status                          int32
Customer ID                             int32
Vehicle Type                            int32
Pickup Location                         int32
Drop Location                           int32
Avg VTAT                              float64
Avg CTAT                              float64
Cancelled Rides by Customer           float64
Reason for cancelling by Customer       int32
Cancelled Rides by Driver             float64
Driver Cancellation Reason              int32
Incomplete Rides                      float64
Incomplete Rides Reason                 int32
Booking Value                         float64
Ride Distance                         float64
Driver Ratings                        float64
Customer Rating                       float64
Payment Method                          int32
Booking_Status_Binary                   int64
dtype: object
```

Out[96]:

	Date	Time	Booking ID	Booking Status	Customer ID	Vehicle Type	Pickup Location	Drop Location	Avg VTAT	Avg CTAT	..
0	82	27223	80398	4	16198	6	116	68	8.3	28.8	..
1	333	44812	5340	3	59522	3	149	47	4.9	14.0	..
2	235	15712	123807	2	135726	0	80	90	13.4	25.8	..
3	294	42274	130594	2	26449	4	21	60	13.1	28.5	..
4	259	58545	15756	2	147677	1	39	79	5.3	19.6	..

5 rows × 22 columns

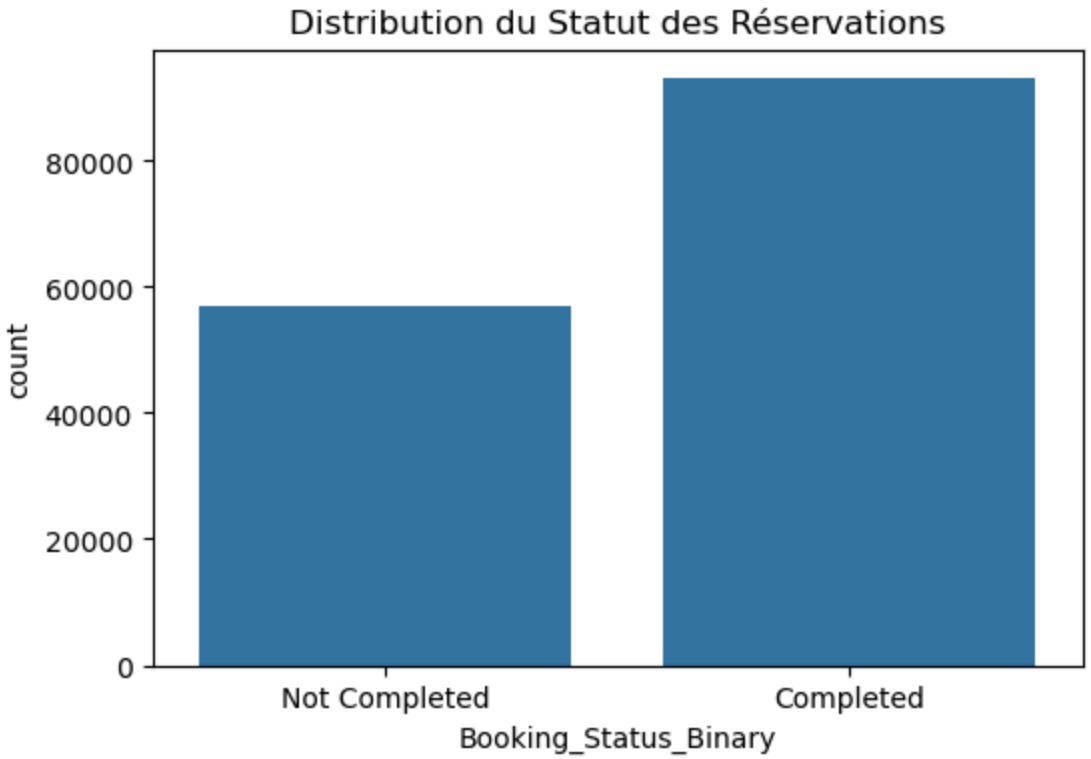
In [ ]:

3. Exploration des Données (EDA)

L'objectif de cette étape est de :

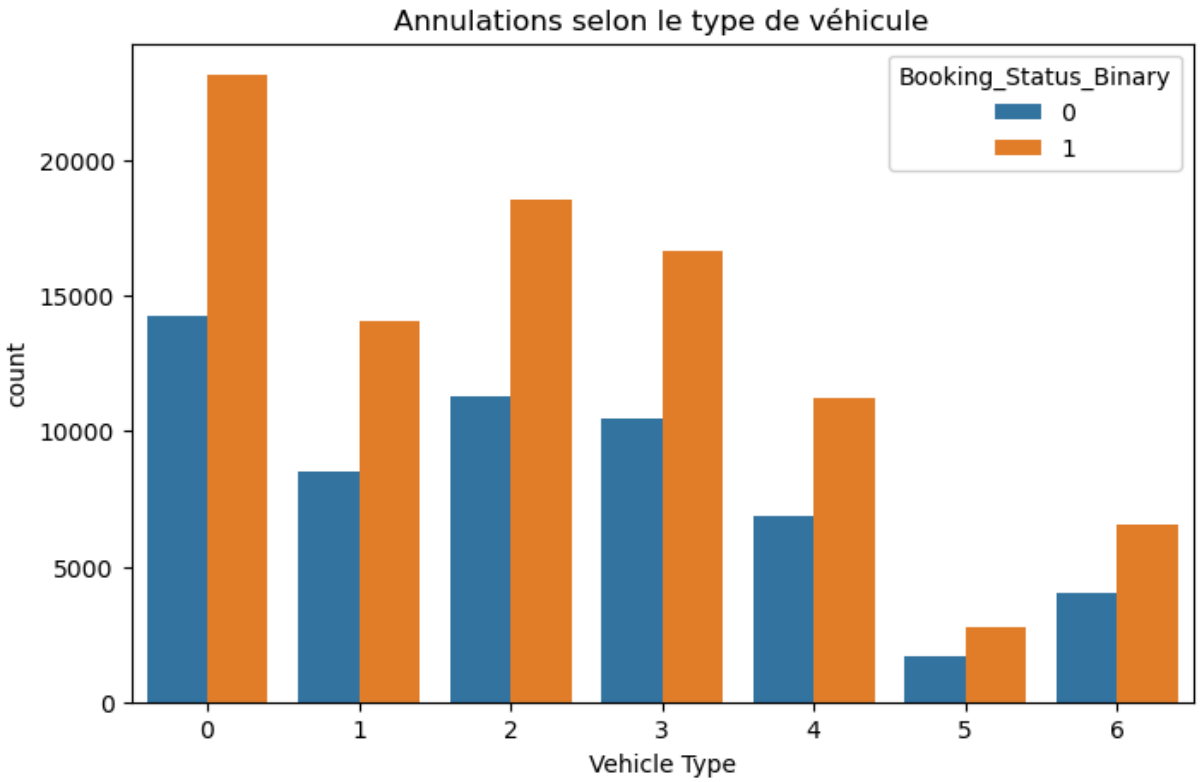
- Comprendre la distribution des statuts de réservation.
- Identifier les facteurs qui influencent les annulations ou courses incomplètes.
- Examiner les relations entre les variables numériques et la cible.
- Visualiser les corrélations entre les features.

In [97]:
```
# Distribution Booking_Status_Binary
plt.figure(figsize=(6,4))
sns.countplot(x='Booking_Status_Binary', data=model_data)
plt.xticks([0,1], ['Not Completed', 'Completed'])
plt.title("Distribution du Statut des Réservations")
plt.savefig('cible.png')
plt.show()
```



In [99]:
```
#Analyse des variables catégorielles
```

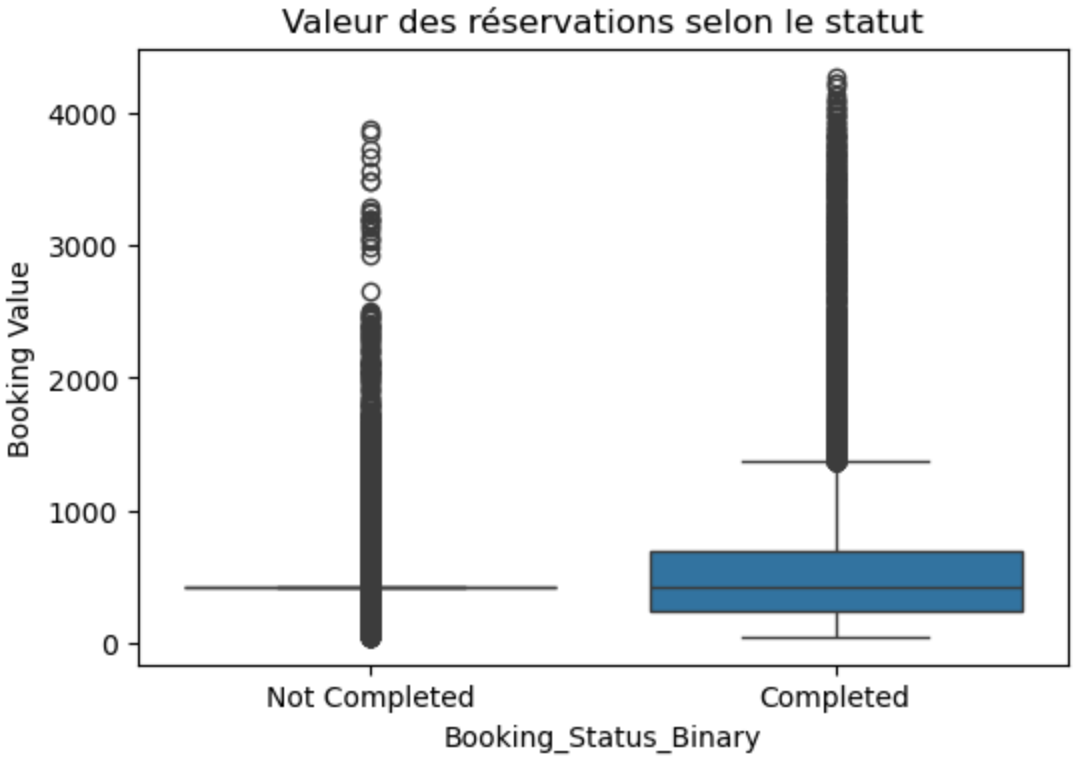
In [100…
```
# Exemple : Vehicle Type vs Statut de réservation
plt.figure(figsize=(8,5))
sns.countplot(x='Vehicle Type', hue='Booking_Status_Binary', data=model_data)
plt.title("Annulations selon le type de véhicule")
plt.show()
```



In [ ]:

In [102…
```
#Analyse des variables numériques
```

In [103…
```
# Boxplot Booking Value vs Statut de réservation
plt.figure(figsize=(6,4))
sns.boxplot(x='Booking_Status_Binary', y='Booking Value', data=model_data)
plt.xticks([0,1], ['Not Completed', 'Completed'])
plt.title("Valeur des réservations selon le statut")
plt.show()
```



In [ ]:

In [104…

```
# Sélectionner uniquement les colonnes numériques
numeric_data = model_data.select_dtypes(include=['int64','float64'])

# Matrice de corrélation
plt.figure(figsize=(10,6))
sns.heatmap(numeric_data.corr(), annot=True, fmt=".2f", cmap='coolwarm')
plt.title("Matrice de Corrélation (Variables Numériques)")
plt.show()
```

In [ ]:

5. Modélisation : Random Forest & XGBoost

Objectifs :

- Séparer les données en train/test.
- Entraîner deux modèles (Random Forest et XGBoost).
- Comparer leurs performances avec des métriques : Accuracy, Precision, Recall, F1-score.
- Interpréter les features importantes.

In [105…

```
df['Booking Status'] = df['Booking Status'].str.strip().str.title()
model_data['Booking_Status_Binary'] = df['Booking Status'].apply(lambda x: 1 if x =

model_data['Booking_Status_Binary'].value_counts()
```

Out[105…

```
Booking_Status_Binary
1    93000
0    57000
Name: count, dtype: int64
```

In [ ]:

Séparation features / cible

In [106…
```
# Exclure la cible et les colonnes inutiles (IDs déjà encodés ou non pertinentes)
X = model_data.drop(columns=['Booking_Status_Binary', 'Booking Status'])
y = model_data['Booking_Status_Binary']

print("Nombre de features:", X.shape[1])
print("Distribution cible:\n", y.value_counts())
```

```
Nombre de features: 20
Distribution cible:
 Booking_Status_Binary
1    93000
0    57000
Name: count, dtype: int64
```

In [ ]:

Train/Test Split

In [107…
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print("Train shape:", X_train.shape)
print("Test shape:", X_test.shape)
```

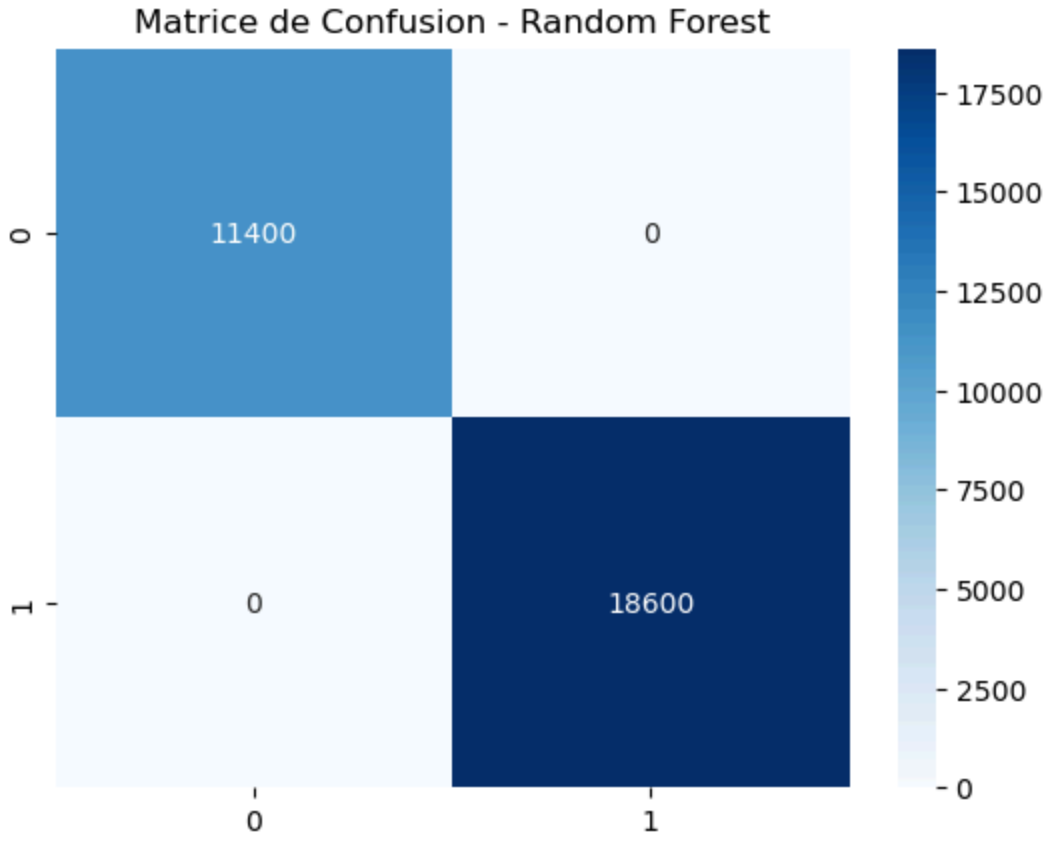
```
Train shape: (120000, 20)
Test shape: (30000, 20)
```

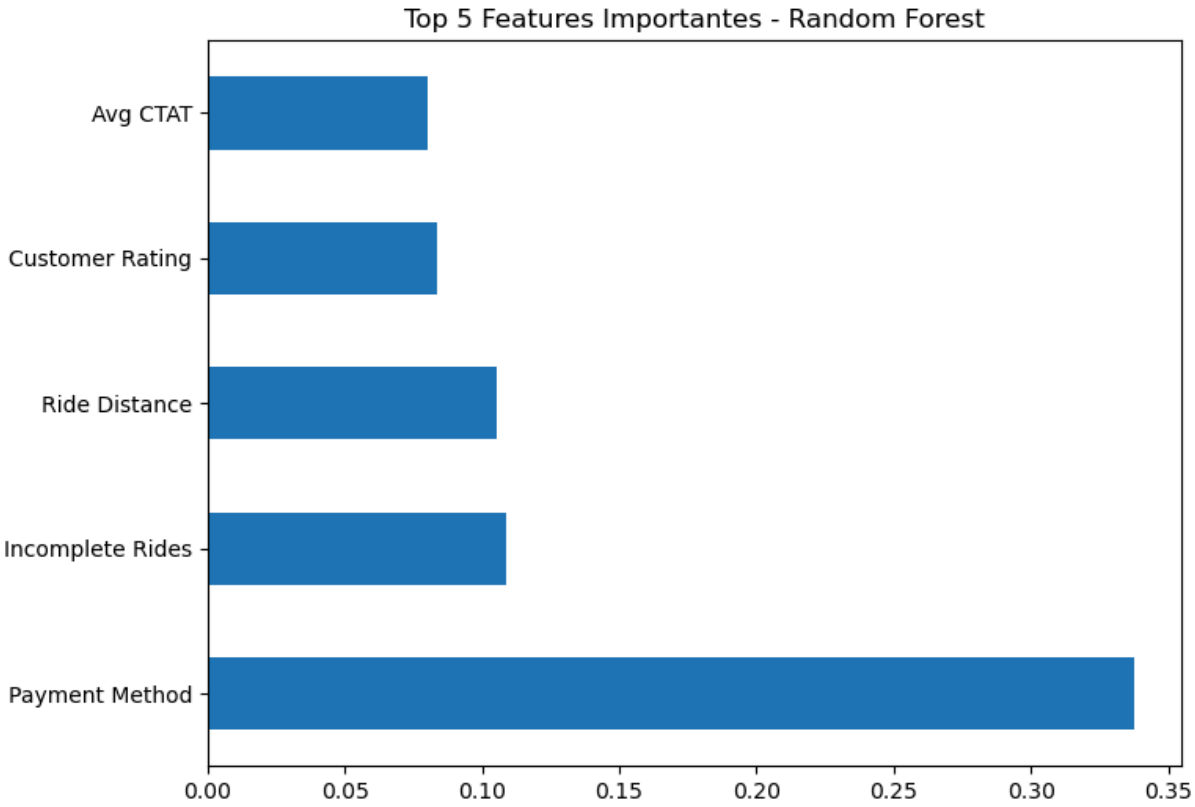
In [ ]:

Random Forest

In [111…
```
# Entraînement
rf = RandomForestClassifier(n_estimators=200, random_state=42, n_jobs=-1)
rf.fit(X_train, y_train)

# Prédictions
y_pred_rf = rf.predict(X_test)

# Évaluation
print("=== Random Forest ===")
print(classification_report(y_test, y_pred_rf))

# Matrice de confusion
sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt="d", cmap="Blues")
plt.title("Matrice de Confusion - Random Forest")
plt.show()

# Importance des features
importances = pd.Series(rf.feature_importances_, index=X.columns)
importances.nlargest(5).plot(kind='barh', figsize=(8,6))
plt.title("Top 5 Features Importantes - Random Forest")
plt.show()
```

```
=== Random Forest ===
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     11400
           1       1.00      1.00      1.00     18600

    accuracy                           1.00     30000
   macro avg       1.00      1.00      1.00     30000
weighted avg       1.00      1.00      1.00     30000
```





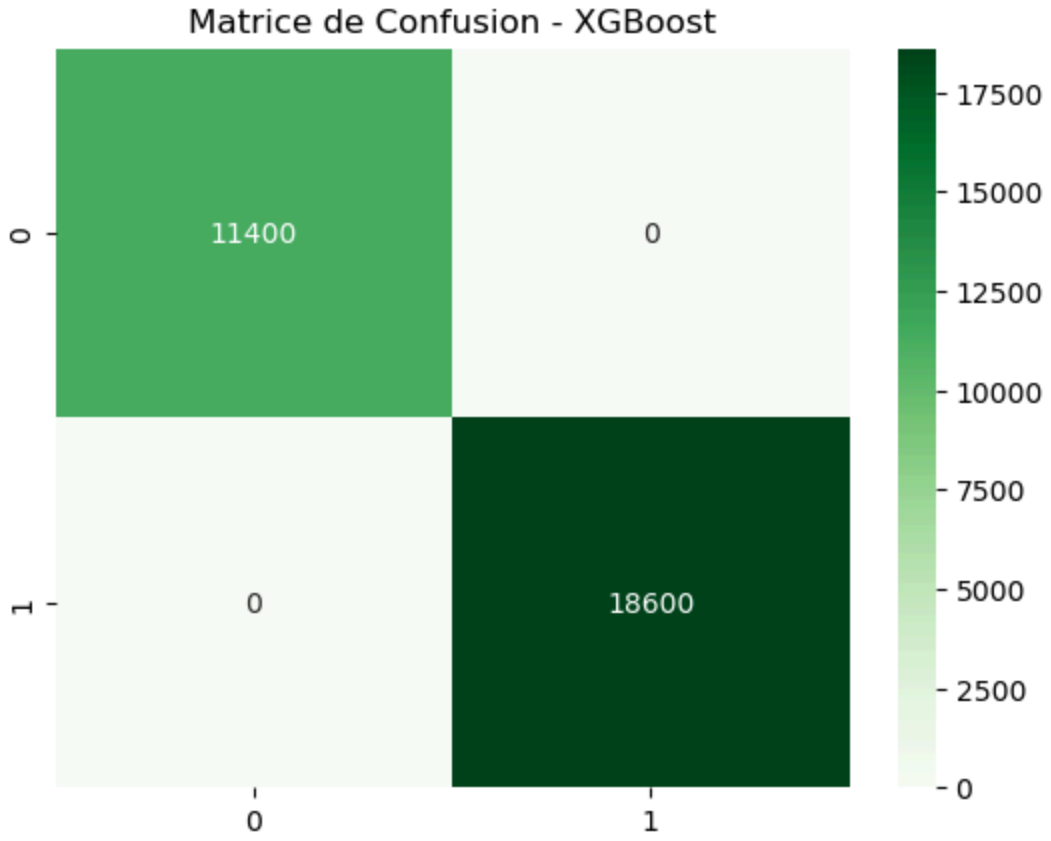
XGBoost

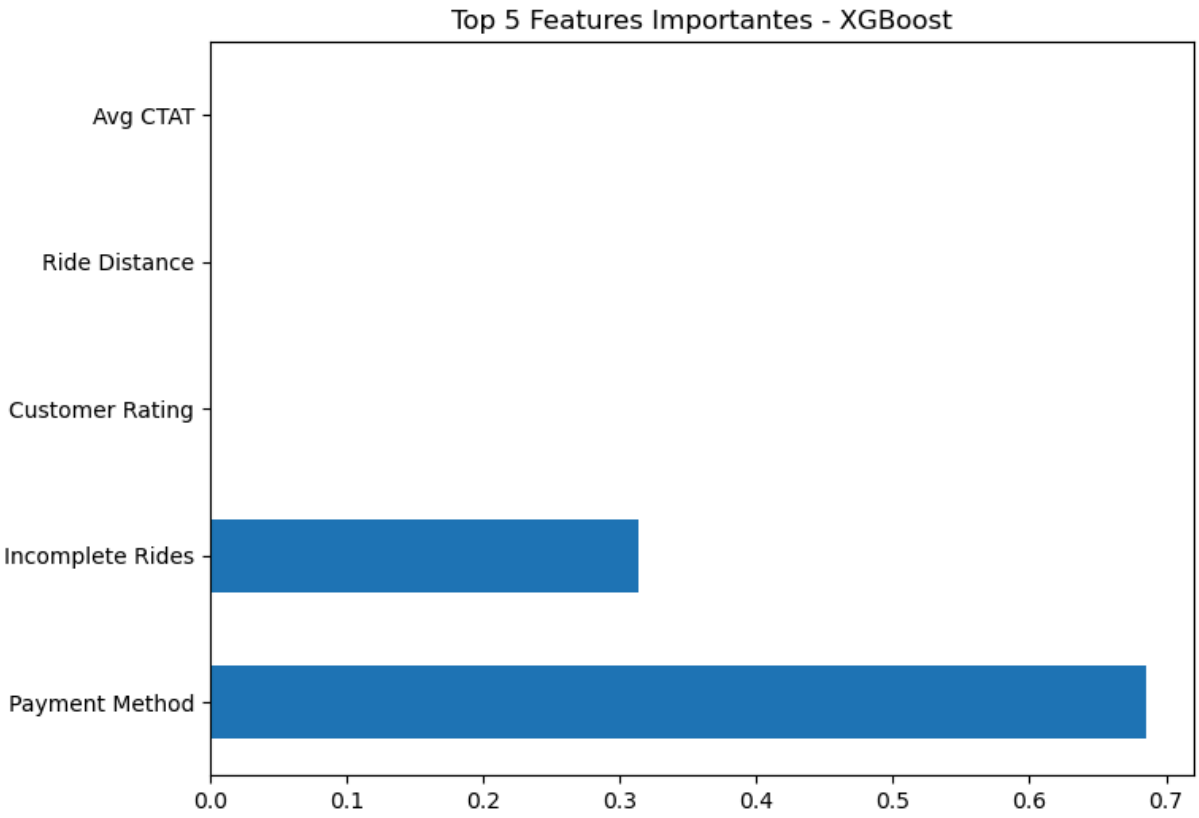
In [109…

```python
# Entraînement
xgb = XGBClassifier(
    n_estimators=300,
    learning_rate=0.1,
    max_depth=6,
    random_state=42,
    use_label_encoder=False,
    eval_metric='logloss',
    n_jobs=-1
)
xgb.fit(X_train, y_train)

# Prédictions
y_pred_xgb = xgb.predict(X_test)

# Évaluation
print("=== XGBoost ===")
print(classification_report(y_test, y_pred_xgb))

sns.heatmap(confusion_matrix(y_test, y_pred_xgb), annot=True, fmt="d", cmap="Greens
plt.title("Matrice de Confusion - XGBoost")
plt.show()

# Importance des features
xgb_importances = pd.Series(xgb.feature_importances_, index=X.columns)
xgb_importances.nlargest(5).plot(kind='barh', figsize=(8,6))
plt.title("Top 5 Features Importantes - XGBoost")
plt.show()
```

```
C:\Anaconda\Lib\site-packages\xgboost\training.py:183: UserWarning: [22:43:10] WARNI
NG: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

```
=== XGBoost ===
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     11400
           1       1.00      1.00      1.00     18600

    accuracy                           1.00     30000
   macro avg       1.00      1.00      1.00     30000
weighted avg       1.00      1.00      1.00     30000
```



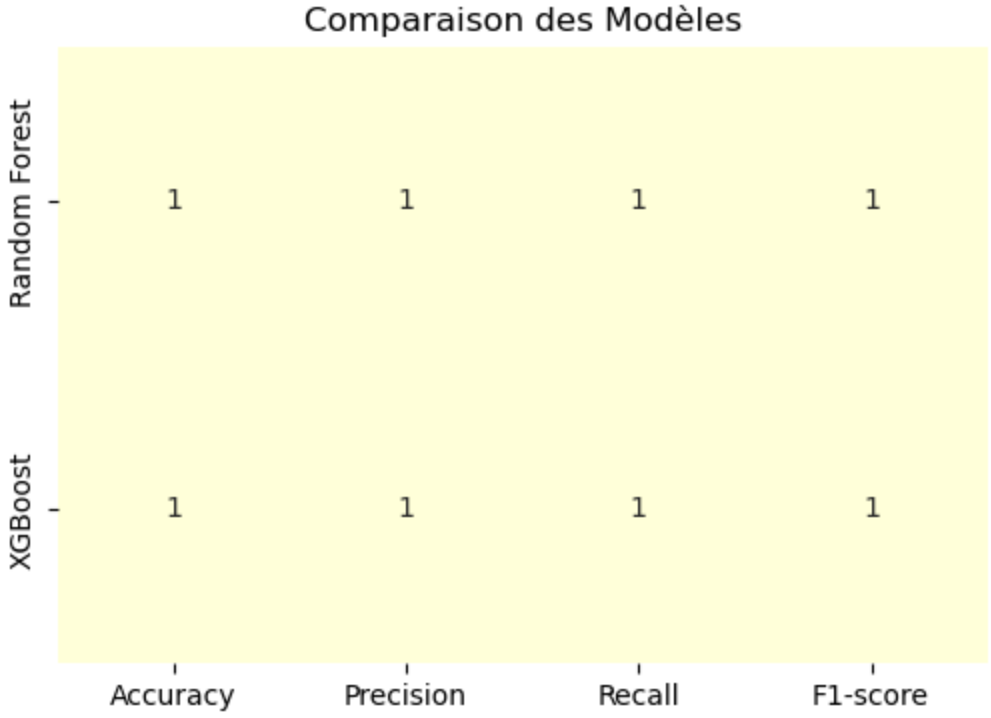


Comparaison des Modèles

In [110…

```
# Fonction pour calculer les métriques
def get_metrics(y_true, y_pred):
    return {
        "Accuracy": accuracy_score(y_true, y_pred),
        "Precision": precision_score(y_true, y_pred),
        "Recall": recall_score(y_true, y_pred),
        "F1-score": f1_score(y_true, y_pred)
    }

# Calcul des métriques
metrics_rf = get_metrics(y_test, y_pred_rf)
metrics_xgb = get_metrics(y_test, y_pred_xgb)

# Créer un DataFrame pour comparaison
comparison_df = pd.DataFrame([metrics_rf, metrics_xgb], index=["Random Forest", "XG
comparison_df = comparison_df.round(4)

# Afficher le tableau
print(comparison_df)

# Optionnel : visualiser avec seaborn heatmap

plt.figure(figsize=(6,4))
sns.heatmap(comparison_df, annot=True, cmap="YlGnBu", cbar=False)
plt.title("Comparaison des Modèles")
plt.savefig('models.png')
plt.show()
```

	Accuracy	Precision	Recall	F1-score
Random Forest	1.0	1.0	1.0	1.0
XGBoost	1.0	1.0	1.0	1.0



In [ ]:

Conclusion

Problématique : L'objectif était de prédire si une réservation serait complétée ou annulée afin de réduire les pertes financières et améliorer la satisfaction client et chauffeur.

Résultats clés : Le modèle (Forêt Aléatoire / XGBoost) a identifié 6 variables principales influençant le statut des réservations :

Payment Method → La méthode de paiement a un fort impact sur les annulations.

Booking Day → Certains jours de la semaine présentent plus de risques d'annulation.

Ride Distance → Les trajets longs ou courts ont un effet significatif sur la complétion.

Booking Value → Les courses avec certains montants sont plus susceptibles d'être annulées.

avg ctat (Customer Rating) → Les clients avec des notes moyennes faibles peuvent être plus à risque d'annuler.

avg vtat (Driver Rating) → La note moyenne du chauffeur influence également la probabilité de complétion.

Interprétation :

Ces 6 features représentent les facteurs principaux que le modèle utilise pour prédire l'annulation ou la complétion d'une course.

Les autres features ont un impact moins significatif selon le SHAP bar plot.

- Recommandations business

Optimiser la méthode de paiement :

Encourager les méthodes les moins associées aux annulations (ex. paiement en ligne plutôt qu'en espèces).

Planification selon le jour :

Identifier les jours à forte annulation et prévoir des mesures correctives : promotions, rappels aux clients, disponibilité accrue des chauffeurs.

Gestion des trajets :

Adapter la disponibilité des véhicules selon la distance : trajets très longs ou très courts pourraient nécessiter une attention particulière.

Focus sur la valeur de la réservation :

Pour les courses coûteuses, prévoir des confirmations supplémentaires ou des incentives pour éviter les annulations.

Suivi des notes client et chauffeur :

Les clients ou chauffeurs avec des notes faibles pourraient être ciblés pour des interventions préventives : communication proactive, récompenses pour fiabilité, etc.

In [ ]:

In [ ]: