Faire du Stacking avec Titanic

On va faire notre premier stacking ensemble. Voyons comment cela peut améliorer notre score sur le dataset du Titanic

1. Importez les librairies usuelles

In [1]:
```
import pandas as pd
```

2. Importez le dataset dans un DataFrame

In [2]:
```
df=pd.read_csv('C:/Users/dell/Desktop/titanic.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

3. Remplacez la colonne

`name` `Ticket`

Par le nombre de caractères qu'il y a dans la cellule

In [4]:
```
df['Name'] = df['Name'].apply(lambda x: len(str(x)))
df['Ticket'] = df['Ticket'].apply(lambda x: len(str(x)))
df.head()
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	23	male	22.0	1	0	9	7.2500	NaN	S
1	2	1	1	51	female	38.0	1	0	8	71.2833	C85	C
2	3	1	3	22	female	26.0	0	0	16	7.9250	NaN	S
3	4	1	1	44	female	35.0	1	0	6	53.1000	C123	S
4	5	0	3	24	male	35.0	0	0	6	8.0500	NaN	S

4. Créez une nouvelle colonne `FamilySize` qui sera la somme des colonnes `SibSp` & `Parch`

In [5]:
```
df['FamilySize'] = df['SibSp'] + df['Parch']
df.head()
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	FamilySize
0	1	0	3	23	male	22.0	1	0	9	7.2500	NaN	S	1
1	2	1	1	51	female	38.0	1	0	8	71.2833	C85	C	1
2	3	1	3	22	female	26.0	0	0	16	7.9250	NaN	S	0
3	4	1	1	44	female	35.0	1	0	6	53.1000	C123	S	1
4	5	0	3	24	male	35.0	0	0	6	8.0500	NaN	S	0

5. Créez une nouvelle colonne `IsAlone` qui indique si la personne a une famille ou non

In [6]:
```
df['IsAlone'] = (df['FamilySize'] == 1).astype(int)
df.head()
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	FamilySize	IsAlone
0	1	0	3	23	male	22.0	1	0	9	7.2500	NaN	S	1	1
1	2	1	1	51	female	38.0	1	0	8	71.2833	C85	C	1	1
2	3	1	3	22	female	26.0	0	0	16	7.9250	NaN	S	0	0
3	4	1	1	44	female	35.0	1	0	6	53.1000	C123	S	1	1
4	5	0	3	24	male	35.0	0	0	6	8.0500	NaN	S	0	0

6. Dans la colonne `Cabin` Faites en sorte de ne garder uniquement la première lettre de la cabine

In [7]:
```
df['Cabin'] = df['Cabin'].str[0]
df.head()
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	FamilySize	IsAlone
0	1	0	3	23	male	22.0	1	0	9	7.2500	NaN	S	1	1
1	2	1	1	51	female	38.0	1	0	8	71.2833	C	C	1	1
2	3	1	3	22	female	26.0	0	0	16	7.9250	NaN	S	0	0
3	4	1	1	44	female	35.0	1	0	6	53.1000	C	S	1	1
4	5	0	3	24	male	35.0	0	0	6	8.0500	NaN	S	0	0

7. Faites une interpolation linéaire pour remplacer les valeurs manquantes dans la colonne `Age`

In [8]:
```
df['Age'] = df['Age'].interpolate(method='linear')
df.head()
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	FamilySize	IsAlone
0	1	0	3	23	male	22.0	1	0	9	7.2500	NaN	S	1	1
1	2	1	1	51	female	38.0	1	0	8	71.2833	C	C	1	1
2	3	1	3	22	female	26.0	0	0	16	7.9250	NaN	S	0	0
3	4	1	1	44	female	35.0	1	0	6	53.1000	C	S	1	1
4	5	0	3	24	male	35.0	0	0	6	8.0500	NaN	S	0	0

8. Comptez le nombre de NaN qu'il y a dans la colonne Embarked et adoptez la stratégie qui vous semble le plus adapté pour gérer ces valeurs

In [9]:
```
nan_count = df['Embarked'].isna().sum()
print(f"Nombre de NaN dans la colonne 'Embarked': {nan_count}")
```

```
Nombre de NaN dans la colonne 'Embarked': 2
```

In [10]:
```
most_frequent_embarked = df['Embarked'].mode()[0]
df['Embarked'].fillna(most_frequent_embarked, inplace=True)
```

In [11]:
```
df.head()
```

Out[11]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	FamilySize	IsAlone
0	1	0	3	23	male	22.0	1	0	9	7.2500	NaN	S	1	1
1	2	1	1	51	female	38.0	1	0	8	71.2833	C	C	1	1
2	3	1	3	22	female	26.0	0	0	16	7.9250	NaN	S	0	0
3	4	1	1	44	female	35.0	1	0	6	53.1000	C	S	1	1
4	5	0	3	24	male	35.0	0	0	6	8.0500	NaN	S	0	0

9. Dummyfiez les variables catégoriques

In [12]:
```
df = pd.get_dummies(df, columns=['Sex', 'Embarked', 'Cabin'], drop_first=True)
df.head()
```

Out[12]:

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	FamilySize	...	Sex_male	Embarked_Q	Embarked
0	1	0	3	23	22.0	1	0	9	7.2500	1	...	1	0	
1	2	1	1	51	38.0	1	0	8	71.2833	1	...	0	0	
2	3	1	3	22	26.0	0	0	16	7.9250	0	...	0	0	
3	4	1	1	44	35.0	1	0	6	53.1000	1	...	0	0	
4	5	0	3	24	35.0	0	0	6	8.0500	0	...	1	0	

5 rows × 21 columns

10. Séparez votre dataset en X & y qui sont respectivement les variables explicatives et la variable cible

In [13]:
```
X = df.drop('Survived',axis=1)
y = df['Survived']
```

11. Normalisez votre dataset

In [14]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)
```

12. Faites un train_test_split

In [15]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_normalized, y, test_size=0.2, random_state=0)
```

13. Faites une première prédiction grâce à une régression logistique et regardez votre score

In [16]:
```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.metrics import accuracy_score
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
y_pred = logistic_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Score de la régression logistique : {accuracy}")
```

```
Score de la régression logistique : 0.8044692737430168
```

14. Faites une prédiction pour tout votre dataset X et concaténez ces prédiction dans un dataframe qu'on appellera X_new

In [17]:
```
all_predictions = logistic_model.predict(X_normalized)
```

In [18]:
```
X_new = pd.DataFrame({'Logistic_Predictions': all_predictions})
```

15. Faites un nouveau train_test_split sur X_new

In [19]:
```
X_new_train, X_new_test, y_train, y_test = train_test_split(X_new, y, test_size=0.2, random_state=0)
```

15. Importez Adaboost et entrainez votre nouveau modèle sur X_new

In [20]:
```
from sklearn.ensemble import AdaBoostClassifier
adaboost_model = AdaBoostClassifier()
adaboost_model.fit(X_new_train, y_train)
y_new_pred = adaboost_model.predict(X_new_test)
```

16. Regardez votre nouveau score

In [21]:
```
accuracy_new = accuracy_score(y_test, y_new_pred)
print(f"Score avec Adaboost : {accuracy_new}")
```

```
Score avec Adaboost : 0.8044692737430168
```