# Devoir de Maison – Data Science

Analyse d'un jeu de données réel issu de la plateforme  Kaggle.

Le dataset utilisé est le suivant :

Dataset : Titanic – Machine Learning from Disaster

Titanic - Machine Learning from Disaster | Kaggle

L'objectif est de prédire la survie des passagers du Titanic en fonction de leurs caractéristiques.

AISSYA BOUKRAA L3SI

# Partie 1 – Chargement et compréhension des données

1. 
```
import pandas as pd
df = pd.read_csv('/content/train.csv')
df.head()
```

| | 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 |

2. Le dataset est compose de 12 colonnes, chaque ligne représente 1 passager. La colonne Cabin contient des valeurs manquantes.

3. La variable cible est Survived car elle représente la survie du passager, qui est l'élément que l'on cherche à prédire.

   Les variables explicatives sont les variables qui aident à expliquer ou prédire la variable cible. Donc nous avons Pclass, Sex, Age, SibSp, Parch, Fare, Cabin et Embarked.

# Partie 2 – Nettoyage et préparation des données

1. 
```
df.isnull().sum()
```

| | e |
|---|---|
| PassengerId | 0 |
| Survived | 0 |
| Pclass | 0 |
| Name | 0 |
| Sex | 0 |
| Age | 177 |
| SibSp | 0 |
| Parch | 0 |
| Ticket | 0 |
| Fare | 0 |
| Cabin | 687 |
| Embarked | 2 |

dtype: int64

   La commande df.isnull() return True si la valeur est manquante, en faisant df.isnull().sum() on additionne toutes les valeurs manquantes par colonne. Les valeurs manquantes sont donc dans les colonnes Age, Cabin et Embarked.

2. Pour la colonne Age, il nous manque 177 valeurs mais cette colonne est importante pour prédire la survie donc nous ne pouvons pas la supprimer. La stratégie consiste alors à remplir les valeurs manquantes par la médiane.
   df['Age'] = df['Age'].fillna(df['Age'].median())
   Df.fillna permet de remplir les valeurs manquantes avec une seule valeur. Ici avec la mediane de la colonne Age on remplit les valeurs manquantes de la colonne Age.
   Dans la documentation de pandas il est dit que "*Si vous souhaitez modifier le DataFrame d'origine, utilisez le paramètre inplace ( df.fillna(0, inplace=True)) ou attribuez-le au fichier de données d'origine ( df = df.fillna(0))"*. J'ai opté pour la deuxième option car google collab m'as dit que c'était préférable.

   Pour la colonne Cabin, il nous manque 687 valeurs. C'est plus de la moitié des informations qu'il nous manque c'est beaucoup trop et on peut retrouver cette info partiellement dans Pclass donc il est préférable de supprimer la colonne.
   df.drop('Cabin', axis=1, inplace=True)
   Df.drop permet de supprimer une colonne dans un DataFrame.
   Axis =1 correspond à axis = 'columns' Comme mentioner dans la documentation: '*df.drop(['B', 'E'], axis='columns', inplace=True) # or df = df.drop(['B', 'E'], axis=1) without the option inplace=True*'.

   Pour la colonne Embarked, il nous manque seulement 2 valeurs ce qui est très peu. Donc remplir les 2 valeurs manquantes par la valeur la plus fréquente suffit. L'analyse ne sera pas impactée.
   mode = df['Embarked'].mode()[0]
   Je voulais tout d'abord afficher la valeur la plus fréquente dans cette colonne à l'aide de la méthode mode() trouver dans la documentation.
   df['Embarked'] = df['Embarked'].fillna(mode)
   Puis remplacer les valeurs manquantes avec la même méthode utiliser dans la colonne Age.

```
df['Age'] = df['Age'].fillna(df['Age'].median())

df = df.drop('Cabin', axis=1)

mode = df['Embarked'].mode()[0]

df['Embarked'] = df['Embarked'].fillna(mode)
```

```
df.isnull().sum()
```

| | 0 |
|---|---|
| PassengerId | 0 |
| Survived | 0 |
| Pclass | 0 |
| Name | 0 |
| Sex | 0 |
| Age | 0 |
| SibSp | 0 |
| Parch | 0 |
| Ticket | 0 |
| Fare | 0 |
| Embarked | 0 |

dtype: int64

On peut voir que maintenant il ne manque plus aucune valeur à notre dataset.

3. Les algorithmes de Machine Learning ne comprennent que les nombres. Mais dans notre dataset, on a des textes.
L'encodage permet de transformer le texte en nombres pour que les algorithmes de ML puisse exploiter entièrement nos donnes.
Nous avons besoin d'encoder que les colonnes Sex et Embarked.

Pour la colonne Sex : Codage ordinal

```
from sklearn.preprocessing import OrdinalEncoder
encoder = OrdinalEncoder()
df[['Sex']] = encoder.fit_transform(df[['Sex']])
```

Pour la colonne Embarked : Encodage à chaud

```
df = pd.get_dummies(df, columns=['Embarked'], prefix='Embarked', dtype=int)
```

La variable Sex, qui ne contient que deux modalités (male/female), est encodée à l'aide d'OrdinalEncoder afin de la transformer en variable numérique (0 et 1). Cet encodage est adapté pour les variables binaires.

La variable Embarked (3 modalités : C, Q, S) est encodée à l'aide d'un One-Hot Encoding via pd.get_dummies() afin d'éviter l'introduction d'un ordre artificiel entre les catégories. Cette méthode crée trois colonnes binaires indépendantes où chaque observation a la valeur 1 dans une seule colonne et 0 ailleurs.

Ressources importantes :
Encodages ordinaux et one-hot pour les données catégorielles
Qu'est-ce que l'encodage One Hot et comment l'implémenter en Python ? | DataCamp

```
from sklearn.preprocessing import OrdinalEncoder

encoder = OrdinalEncoder()
df[['Sex']] = encoder.fit_transform(df[['Sex']])

df = pd.get_dummies(df, columns=['Embarked'], prefix='Embarked', dtype=int)
```

df

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Embarked_C | Embarked_Q | Embarked_S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | Braund, Mr. Owen Harris | 1.0 | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | 0 | 0 | 1 |
| 1 | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | 0.0 | 38.0 | 1 | 0 | PC 17599 | 71.2833 | 1 | 0 | 0 |
| 2 | 3 | 1 | 3 | Heikkinen, Miss. Laina | 0.0 | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | 0 | 0 | 1 |
| 3 | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | 0.0 | 35.0 | 1 | 0 | 113803 | 53.1000 | 0 | 0 | 1 |
| 4 | 5 | 0 | 3 | Allen, Mr. William Henry | 1.0 | 35.0 | 0 | 0 | 373450 | 8.0500 | 0 | 0 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 886 | 887 | 0 | 2 | Montvila, Rev. Juozas | 1.0 | 27.0 | 0 | 0 | 211536 | 13.0000 | 0 | 0 | 1 |
| 887 | 888 | 1 | 1 | Graham, Miss. Margaret Edith | 0.0 | 19.0 | 0 | 0 | 112053 | 30.0000 | 0 | 0 | 1 |
| 888 | 889 | 0 | 3 | Johnston, Miss. Catherine Helen "Carrie" | 0.0 | 28.0 | 1 | 2 | W./C. 6607 | 23.4500 | 0 | 0 | 1 |
| 889 | 890 | 1 | 1 | Behr, Mr. Karl Howell | 1.0 | 26.0 | 0 | 0 | 111369 | 30.0000 | 1 | 0 | 0 |
| 890 | 891 | 0 | 3 | Dooley, Mr. Patrick | 1.0 | 32.0 | 0 | 0 | 370376 | 7.7500 | 0 | 1 | 0 |

891 rows × 13 columns

4. y = df['Survived']

X = df[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked_C', 'Embarked_Q', 'Embarked_S']]

y = Survived car c'est ce qu'on veut prédire. X contient les 9 features pertinentes (variables numériques et encodées) après exclusion des colonnes non prédictives (PassengerId, Name, Ticket)

## Partie 3 – Visualisation des données

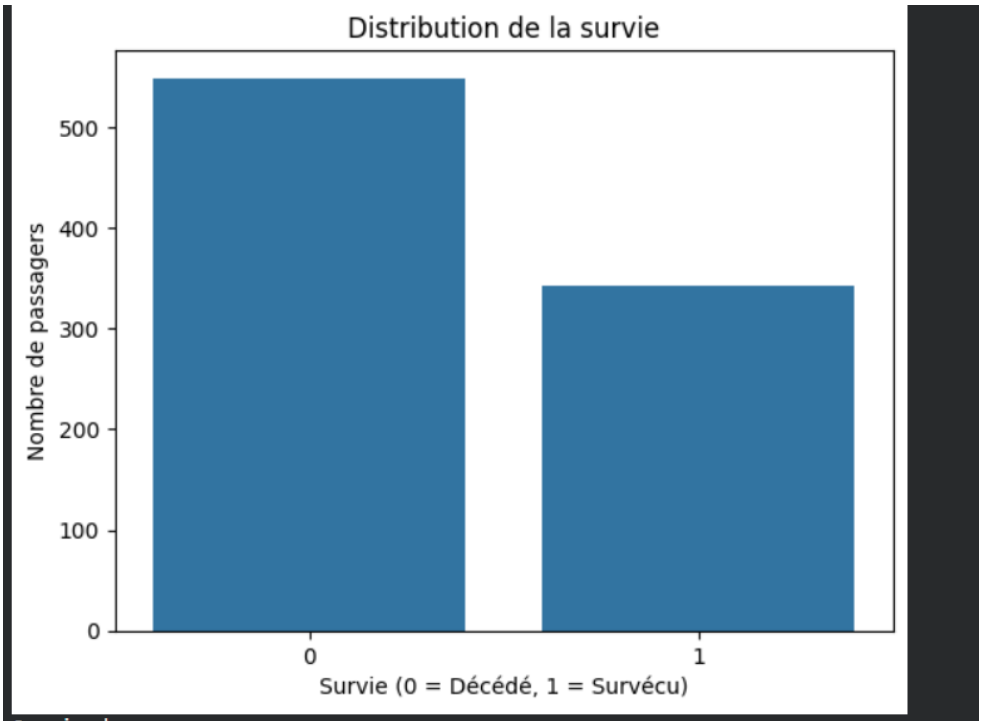
import matplotlib.pyplot as plt

import seaborn as sns

1.

```python
#cree un graphique a barres qui compte le nombr dobservation pour chaque categories
sns.countplot(data=df, x='Survived')
plt.title('Distribution de la survie')
plt.xlabel('Survie (0 = Décédé, 1 = Survécu)')
plt.ylabel('Nombre de passagers')
plt.show()

print(df['Survived'].value_counts())
print(f"Taux de survie : {df['Survived'].mean()*100:.2f}%")
```

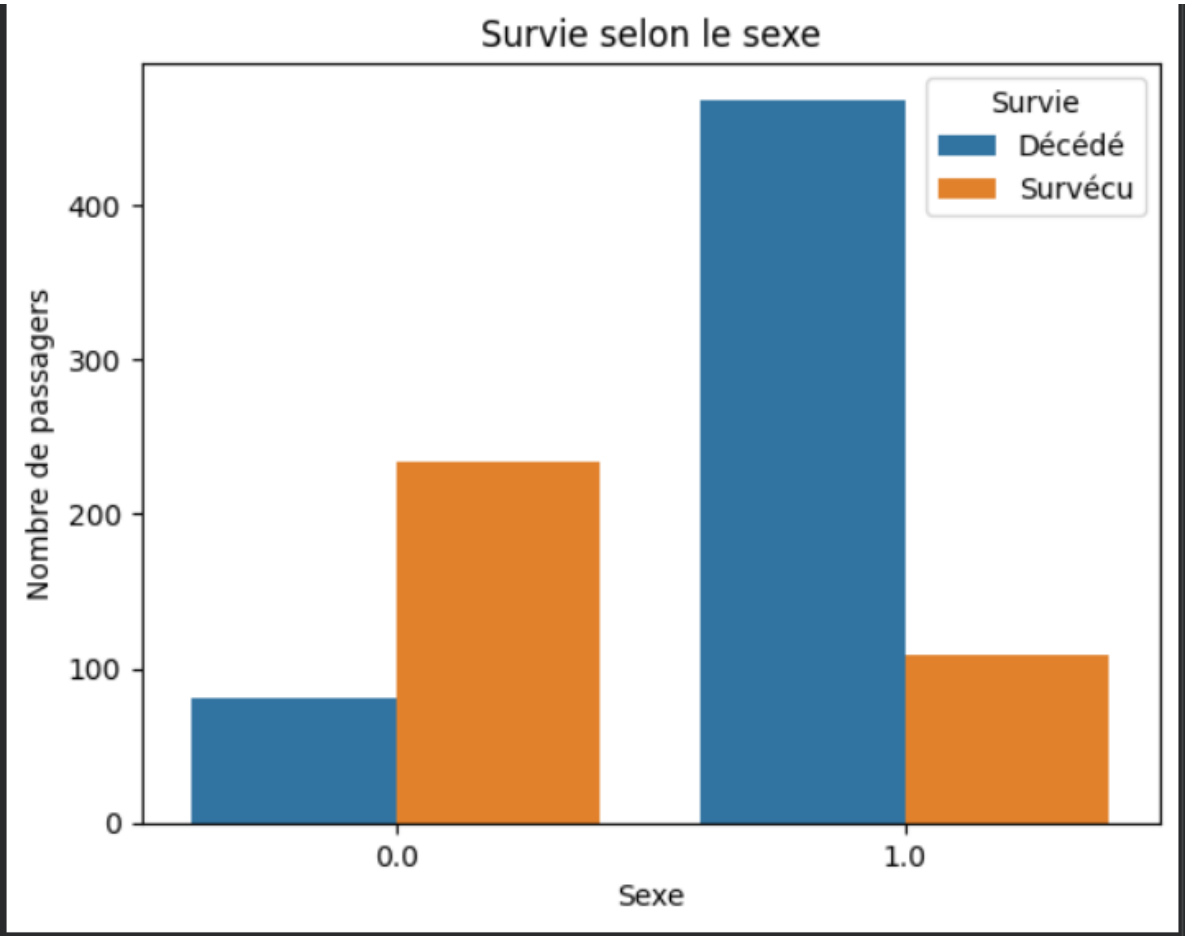



```
Survived
0    549
1    342
Name: count, dtype: int64
Taux de survie : 38.38%
```

J’ai utilisé un countplot car c'est le graphique le plus adapté pour visualiser la distribution d'une variable catégorielle binaire. On peut voir un taux très élevé de morts compare a celui de survie.

2.

```
sns.countplot(data=df, x='Sex', hue='Survived')
plt.title('Survie selon le sexe')
plt.xlabel('Sexe')
plt.ylabel('Nombre de passagers')
plt.legend(title='Survie', labels=['Décédé', 'Survécu'])
plt.show()
# Calcule le taux de survie moyen par sexe
print(df.groupby('Sex')['Survived'].mean() * 100)
```



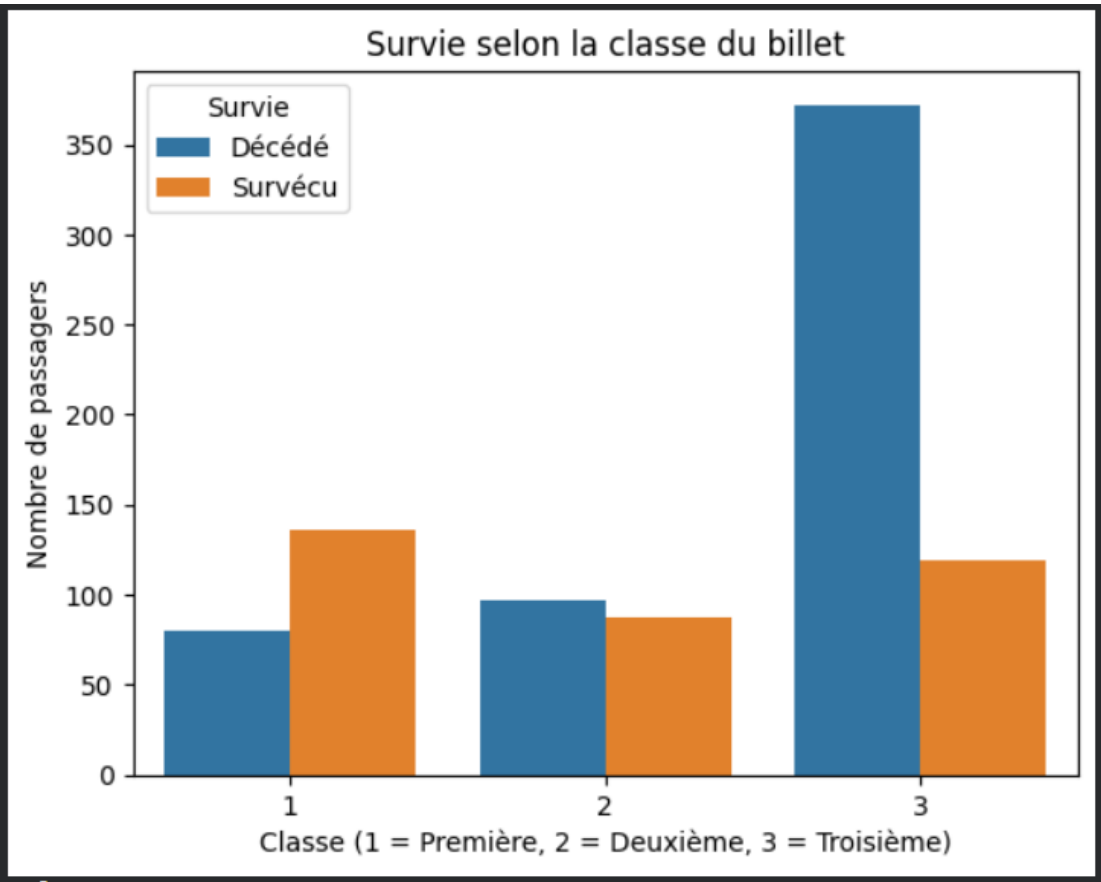

```
Sex
0.0    74.203822
1.0    18.890815
Name: Survived, dtype: float64
```

Il existe une forte corrélation entre le sexe et la survie. Les femmes avaient près de 4 fois plus de chances de survivre. Cela confirme la politique d'évacuation "les femmes et les enfants d'abord". Le sexe est une variable très prédictive.
(les hommes sont 1 et les femmes 0)

3.

```python
sns.countplot(data=df, x='Pclass', hue='Survived')
plt.title('Survie selon la classe du billet')
plt.xlabel('Classe (1 = Première, 2 = Deuxième, 3 = Troisième)')
plt.ylabel('Nombre de passagers')
plt.legend(title='Survie', labels=['Décédé', 'Survécu'])
plt.show()

print(df.groupby('Pclass')['Survived'].mean() * 100)
```



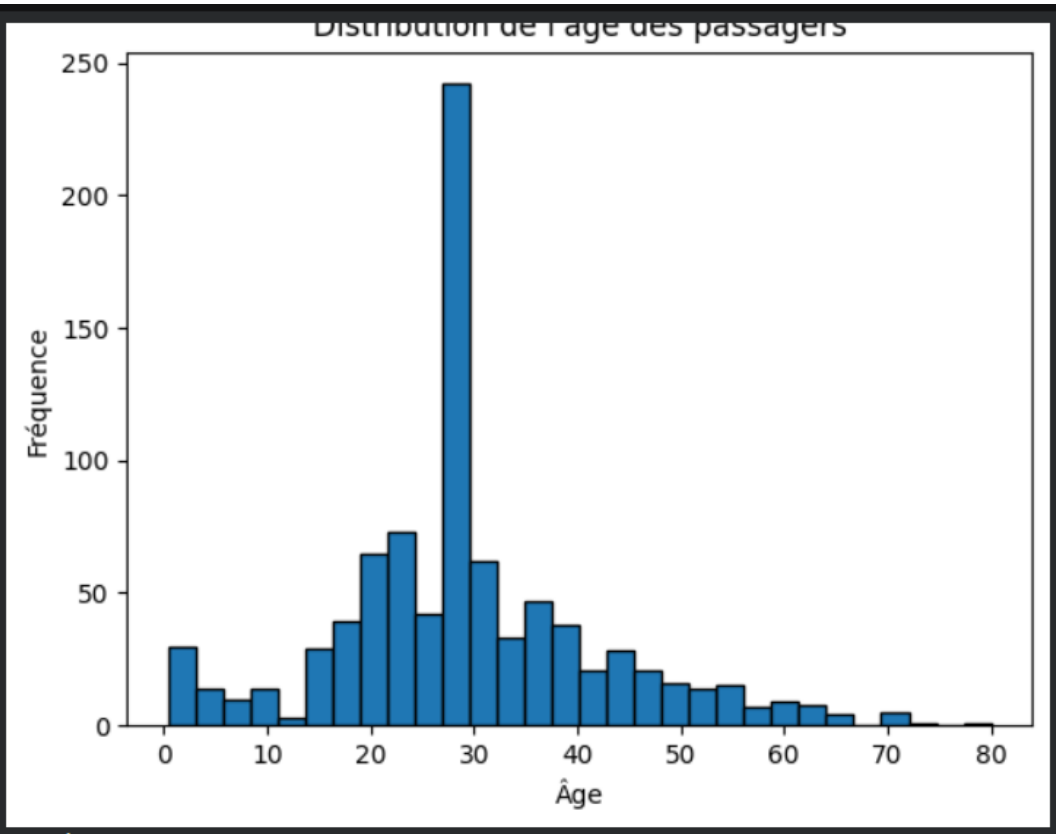

```
Pclass
1    62.962963
2    47.282609
3    24.236253
Name: Survived, dtype: float64
```

Il existe une corrélation négative entre la classe et la survie : plus la classe est élevée (numéro faible), plus les chances de survie sont élevées. Cela s'explique par l'emplacement des cabines (plus proches des canots de sauvetage pour la 1ère classe) et les privilèges d'accès. La classe sociale a fortement influencé les chances de survie.

4.

```
plt.hist(df['Age'], bins=30, edgecolor='black')
plt.title('Distribution de l\'âge des passagers')
plt.xlabel('Âge')
plt.ylabel('Fréquence')
plt.show()

print(df['Age'].describe())
```




```
count    891.000000
mean      29.361582
std       13.019697
min        0.420000
25%       22.000000
50%       28.000000
75%       35.000000
max       80.000000
```

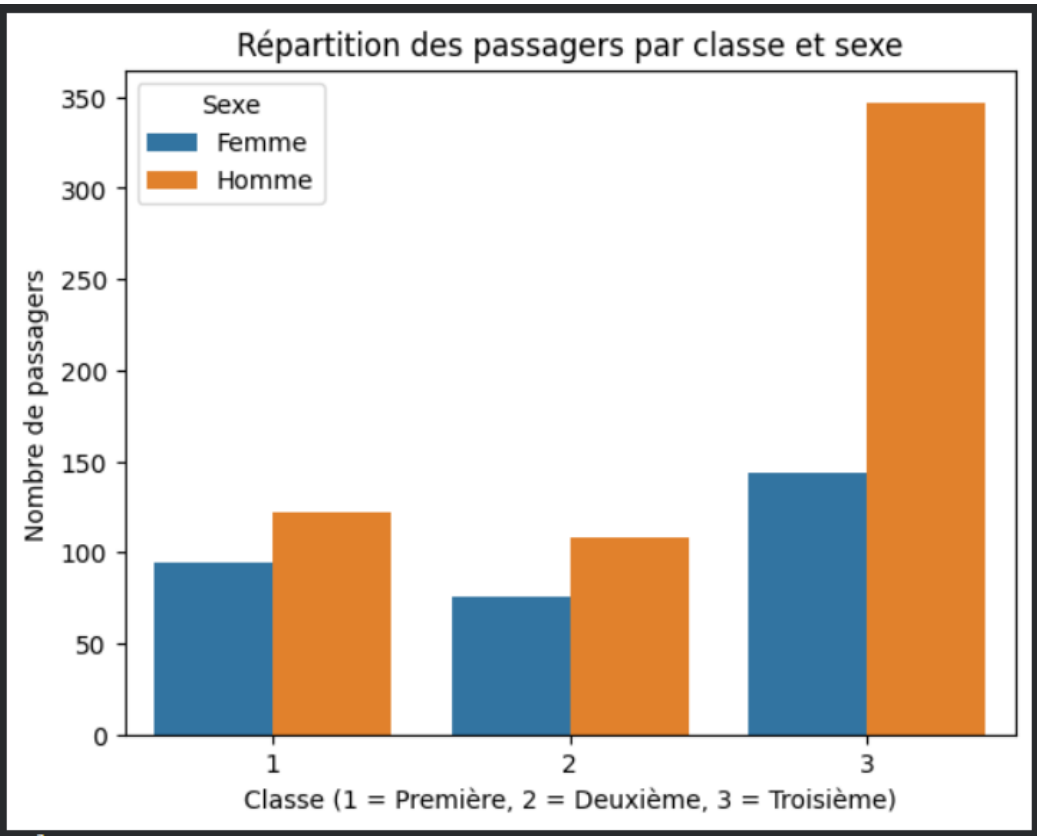
L'histogramme est le graphique standard pour visualiser la distribution d'une variable continue.

L'age moyen est d'environ 28 ans. La majorité des passagers ont entre 20 et 40 ans, avec quelques enfants et personnes âgées aux extrêmes.

5.

```
sns.countplot(data=df, x='Pclass', hue='Sex')
plt.title('Répartition des passagers par classe et sexe')
plt.xlabel('Classe (1 = Première, 2 = Deuxième, 3 = Troisième)')
plt.ylabel('Nombre de passagers')
plt.legend(title='Sexe', labels=['Femme', 'Homme'])
plt.show()

print(df.groupby('Pclass')['Sex'].value_counts())
```




```
Pclass  Sex
1       1.0    122
        0.0     94
2       1.0    108
        0.0     76
3       1.0    347
        0.0    144
```

Cette visualisation est pertinente car elle révèle la composition démographique du Titanic. Comme nous avons vu que le sexe et la classe sont les deux facteurs les plus importants pour la survie, il est essentiel de comprendre combien de personnes étaient dans chaque catégorie. Cela permet de contextualiser les taux de survie : même si les hommes de 3ème classe avaient un faible taux de survie, ils représentaient la plus grande partie des passagers.

La 3ème classe est la plus nombreuse (~500 passagers), avec une forte majorité d'hommes (~350 hommes vs ~150 femmes)

Les 1ère et 2ème classe sont moins peuplées et ont une répartition plus équilibrée entre sexes
Au total, il y a environ 2 fois plus d'hommes que de femmes à bord
Cette répartition explique pourquoi le taux de mortalité global est élevé : la majorité des passagers (hommes de 3ème classe) appartenaient au groupe avec le plus faible taux de survie. Cela met en évidence les inégalités sociales et de genre face à la catastrophe.

Pour cette partie, je me suis appuyée sur la documentation Seaborn et j'ai utilisé Claude pour améliorer la formulation de mes explications. Le code et les analyses restent mon travail personnel. Je n'ai pas détaillé Matplotlib car je maîtrise déjà cette bibliothèque.

Ressources :
[seaborn.countplot() in Python - GeeksforGeeks]

## Partie 4 – Analyse exploratoire des données (EDA)

1.
```
print(df.describe())
```

```
       PassengerId    Survived      Pclass         Sex         Age  \
count   891.000000  891.000000  891.000000  891.000000  891.000000
mean    446.000000    0.383838    2.308642    0.647587   29.361582
std     257.353842    0.486592    0.836071    0.477990   13.019697
min       1.000000    0.000000    1.000000    0.000000    0.420000
25%     223.500000    0.000000    2.000000    0.000000   22.000000
50%     446.000000    0.000000    3.000000    1.000000   28.000000
75%     668.500000    1.000000    3.000000    1.000000   35.000000
max     891.000000    1.000000    3.000000    1.000000   80.000000

            SibSp       Parch        Fare  Embarked_C  Embarked_Q  Embarked_S
count  891.000000  891.000000  891.000000  891.000000  891.000000  891.000000
mean     0.523008    0.381594   32.204208    0.188552    0.086420    0.725028
std      1.102743    0.806057   49.693429    0.391372    0.281141    0.446751
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000    0.000000    7.910400    0.000000    0.000000    0.000000
50%      0.000000    0.000000   14.454200    0.000000    0.000000    1.000000
75%      1.000000    0.000000   31.000000    0.000000    0.000000    1.000000
max      8.000000    6.000000  512.329200    1.000000    1.000000    1.000000
```

On peut diviser ces statistiques descriptives en plusieurs catégories.

Les variables démographiques, donc avec l'age et le sex, on voit que l'age moyen est d'environ 29,4. L'age des passagers va de 0.42 donc bebe a 80 ans. On voit aussi que la moyenne du sex est de 0,65 soit 65% des passagers sont des hommes.

Les variables socio-economiques tel que Pclass ou Fare. Pclass nous montre que 75% des passagers voyageaient en 3eme classes. L'ecart type de Fare qui est d'environ 49.7 nous montre une forte inégalité économique, le prix max du billet est de 512 donc 16 fois le prix de la moyenne.

Les variables de structure familiale comme SibSp ou Parch montrent que peu de passagers voyageaient accompagne.

On peut comparer les 3 variables Embarked et conclurent que 73% des passagers ont embarque dans S.

Notre variable cible survived avec une moyenne de 0.38 donc qui indique un taux de survie global de 38% donc 62 % des passagers sont décédé.

Ces statistiques montrent un dataset avec de fortes inégalités sociales, beaucoup de passagers en 3eme classes, beaucoup plus d'hommes que de femmes et le faible taux de survie montre l'ampleur de cet incident.

2. La corrélation entre les variables numériques c'est se demander si les variables sont liée entre elles.
   C'est un nombre entre -1 et +1 qui indique si deux variables évoluent ensemble.

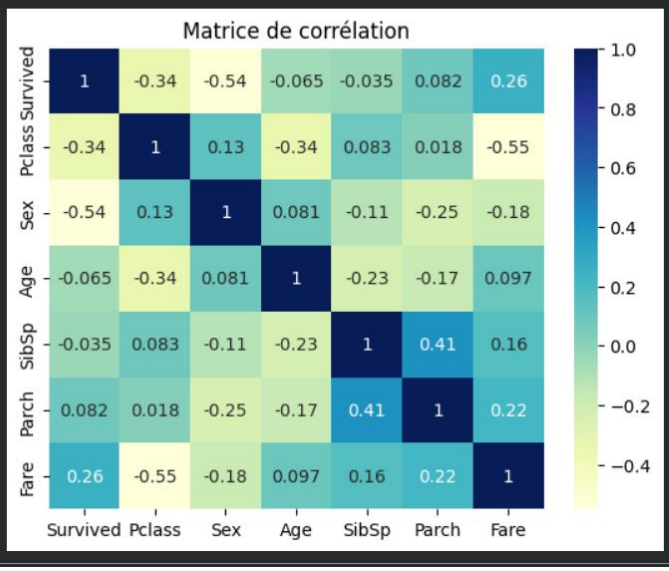
```
correlation = df[['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']].corr()
```

```
correlation
```

| | Survived | Pclass | Sex | Age | SibSp | Parch | Fare |
|---|---|---|---|---|---|---|---|
| **Survived** | 1.000000 | -0.338481 | -0.543351 | -0.064910 | -0.035322 | 0.081629 | 0.257307 |
| **Pclass** | -0.338481 | 1.000000 | 0.131900 | -0.339898 | 0.083081 | 0.018443 | -0.549500 |
| **Sex** | -0.543351 | 0.131900 | 1.000000 | 0.081163 | -0.114631 | -0.245489 | -0.182333 |
| **Age** | -0.064910 | -0.339898 | 0.081163 | 1.000000 | -0.233296 | -0.172482 | 0.096688 |
| **SibSp** | -0.035322 | 0.083081 | -0.114631 | -0.233296 | 1.000000 | 0.414838 | 0.159651 |
| **Parch** | 0.081629 | 0.018443 | -0.245489 | -0.172482 | 0.414838 | 1.000000 | 0.216225 |
| **Fare** | 0.257307 | -0.549500 | -0.182333 | 0.096688 | 0.159651 | 0.216225 | 1.000000 |

Pour faciliter la visualisation des donnes et analyser la corrélation entre les variables numériques. J'ai fait des recherches et trouvee une correlation heatmap.

```
correlation = df[['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']].corr()
sns.heatmap(correlation, annot=True, cmap='YlGnBu')
plt.title('Matrice de corrélation')
plt.show()
```




On peut se concentrer sur le premiere ligne pour voir les correlations avec la variable Survived.

Les corrélations négatives (défavorable a la survie): on voit qu'avec sex il y a une forte corrélation -0.54, effectivement les hommes ont beaucoup moins survécu que les femmes. On a aussi –0.34 pour Pclass, une corrélation moyenne, plus le numero de classe est eleve (donc moins bonne classe) moins la chance de survie est eleve. Pour age et SibSp, corrélation faible voir pratiquement pas.

Les correlations positives (favorable a la survie) : Avec Fare +0.26, on a une correlation positive moyenne donc malgres tout plus le billet est cher plus les chances de survie augmentent. Avec Parch on a +0.082 correlation tres faible.

Pour conclure les variables Sex et Pclass présentent les corrélations les plus fortes avec la survie. Le Fare (prix du billet) est également significatif car il reflète indirectement la classe sociale. L'âge a une influence directe faible, mais peut avoir un effet en combinaison avec d'autres variables (principe "femmes et enfants d'abord").

Ressources importantes :
How to create a correlation heatmap in Python? - GeeksforGeeks

3. Je pense que la première observation qu'on puisse faire c'est que les femmes des classes superieures ont un taux de survie tres eleve contrairement aux hommes

de 3eme classes qui eux ont le plus faible taux on en deduit que le sex et la classe sociale ont agi sur le taux de survi donc etre une femme ET en classe superieur maximisait les chances de survie tandis qu'etre un homme ET en 3eme classe minimisait.

La deuxième analyse c'est vraiment sur l'inégalité économique qui est refleter par l'ecart type du prix des billets qui est tres eleve ce qui montre une distribution tres inegale avec quelques billets extrement chers donc ca reflete les inegalites economiques importantes entre les passagers.

4. Pour moi l'hypothese qu'on pourrait poser pour expliquer la survie des passagers a partir des donnees observees c'est que les femmes et les passagers des classes supérieures ont été prioritaires lors de l'évacuation.
   Cette hypothèse repose sur les observations précédentes.

## Partie 5 – Machine Learning supervisé

1. Je choisi l'algorithme de Régression Logistique, car nous avons ici un problème de classification binaire donc 'survie ou mort'. La régression logistique est le model le plus adapte pour ce genre de problème.

```
y = df['Survived']
X = df[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare',
        'Embarked_C', 'Embarked_Q', 'Embarked_S']]
```

```
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=16)
```

```
# import the class
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression(random_state=16, max_iter=1000)

# fit the model with data
logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)
print(f"\nExemple de prédictions (10 premières) : {y_pred[:10]}")
print(f"Valeurs réelles correspondantes : {y_test[:10].values}")
```

```
...
Exemple de prédictions (10 premières) : [1 0 0 1 1 1 0 0 1 0]
Valeurs réelles correspondantes : [1 1 1 1 1 1 0 0 1 0]
```

2. J'ai séparé les données en 75% pour l'entraînement et 25% pour le test. Le modèle a été entraîné sur X_train et y_train, puis j'ai effectué des prédictions sur X_test.

Les résultats montrent les prédictions du modèle (0 ou 1) comparées aux valeurs réelles.

```
# import the metrics class
from sklearn import metrics

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

```
array([[118,  19],
       [ 32,  54]])
```

```
accuracy = metrics.accuracy_score(y_test, y_pred)
precision = metrics.precision_score(y_test, y_pred)
recall = metrics.recall_score(y_test, y_pred)

print(f"\nAccuracy : {accuracy:.2%}")
print(f"Precision : {precision:.2%}")
print(f"Recall : {recall:.2%}")
```

```
...

Accuracy : 77.13%
Precision : 73.97%
Recall : 62.79%
```

3. Ici, vous pouvez voir la matrice de confusion sous la forme d'un tableau. La dimension de cette matrice est de 2*2 car ce modèle est une classification binaire. Vous avez deux classes 0 et 1. Les valeurs diagonales représentent des prédictions exactes, tandis que les éléments non diagonaux représentent des prédictions inexactes. Dans le résultat, 118 et 54 sont des prédictions réelles, et 32 et 19 sont des prédictions incorrectes.

Accuracy (77.13%) : Le modèle prédit correctement la survie ou le décès dans 77% des cas.
Precision (73.97%) : Parmi les passagers que le modèle prédit comme survivants, environ 74% ont réellement survécu. Cela signifie que le modèle a un taux de faux positifs modéré.
Recall (62.79%) : Le modèle identifie correctement 63% des passagers qui ont réellement survécu. Cela indique qu'il manque environ 37% des survivants (faux négatifs).

Ressources importantes :
Tutoriel de régression logistique en Python avec Sklearn et Scikit | DataCamp
Recall, Precision, F1 Score - Explication Simple Métrique en ML

Métriques en Machine Learning : Tout ce qu'il faut savoir

## Partie 6 – Questions de réflexion

1. Le modèle présente un recall faible (63%), ce qui signifie qu'il manque 37% des survivants réels. De plus, il ne capture pas les interactions entre variables comme l'effet combiné du sexe et de la classe observée dans l'EDA.
2. Pour améliorer les performances, on pourrait créer des variables d'interaction (comme Sex × Pclass) pour capturer les effets combinés. On pourrait également tester des algorithmes plus complexes comme Random Forest ou les arbres de décision qui capturent mieux les relations non-linéaires.

AUTRES RESSOURCES :

Logistic Regression vs K Nearest Neighbors in Machine Learning - GeeksforGeeks

Un guide pratique pour choisir le bon algorithme pour votre problème : de la régression aux réseaux de neurones

Comment afficher la valeur la plus fréquente dans une série Pandas ?

pandas-fr.pdf

Encodages ordinaux et one-hot pour les données catégorielles

Qu'est-ce que l'encodage One Hot et comment l'implémenter en Python ? | DataCamp

Machine Learning Fundamentals in Python | Apprendre le ML avec Python | DataCamp

pandas.DataFrame.value_counts — pandas 2.3.3 documentation

seaborn.countplot() in Python - GeeksforGeeks

NumPy, SciPy et pandas : corrélation avec Python

How to create a correlation heatmap in Python? - GeeksforGeeks

La régression logistique, qu’est-ce que c’est ?

Algorithme de classification : Définition et principaux modèles