IDL TP Final `scikit-learn` Projet

Charger les donnés

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

In [2]:
```
def load_reviews(data_dir):
        reviews = []
        labels = []
        for label in ["pos", "neg"]:
                directory = os.path.join(data_dir, label)
                for filename in os.listdir(directory):
                        if filename.endswith(".txt"):
                                file_path = os.path.join(directory, filename
                                with open(file_path, 'r', encoding='utf-8')
                                        reviews.append(file.read())
                                        labels.append(1 if label == "pos" el
        return reviews, labels
```

In [3]:
```
data_dir = 'imdb_smol'
reviews, labels = load_reviews(data_dir)
reviews_df = pd.DataFrame({'review': reviews, 'label': labels})
```

In [4]:
```
print(reviews_df.head())
```

```
                                              review  label
0  The production quality, cast, premise, authent...      1
1  This is no art-house film, it's mainstream ent...      1
2  Two great comedians in a great Neil Simon movi...      1
3  I'm a fan of TV movies in general and this was...      1
4  Once upon a time in a castle..... Two little ...      1
```

In [5]:
```
print("\nNombre de notes positives:", (reviews_df['label'] == 1).sum())
print("Nombre de notes négatives:", (reviews_df['label'] == 0).sum())
```

```
Nombre de notes positives: 301
Nombre de notes négatives: 301
```

In [6]:
```
print("\nInformations générales sur le DataFrame:")
print(reviews_df.info())
```

```
Informations générales sur le DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 602 entries, 0 to 601
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   review  602 non-null    object
 1   label   602 non-null    int64
dtypes: int64(1), object(1)
memory usage: 9.5+ KB
None
```

In [7]:
```python
for index, row in reviews_df.sample(5).iterrows():
        print("\nIndex de la note de review:", index, "\nTexte de la note de
        print("Étiquette:", "Positive" if row['label'] == 1 else "Négative")

        print("\nIndex de la note de review:", index, "\nTexte de la note de
        print("Étiquette:", "Positive" if row['label'] == 1 else "Négative")
```

Index de la note de review: 591
Texte de la note de review:
 The perfect murder is foiled when a wife(played by Mary Ellen Trainor, once the wife to director Robert Zemeckis, who helmed this episode), who murders her husband with a poker, has the misfortune of receiving a visitor as she i s about to move the body outside..an escaped insane madman dressed in a Sant a Claus suit(played by a deviously hideous Larry Drake). She fends for her l ife while trying to find a way of hiding her husband's corpse. She decides t o use an ax, once she downs the Santa killer who misses several chances to c hop off the woman's head, to frame the killer for her husband's murder. Sant a killer locks her in a closet and pursues the woman's daughter as she tries desperate to free herself to save the child.

This episode of TALE S FROM THE CRYPT just recycles tired material involving the old "Santa kill s" theme while also adding the oft-used(add nauseum)woman-murders-her-husban d-for-a-man-she's-been-cheating-with routine. It's essentially Trainor tryin g to find a way to avoid being caught with a dead body she kills while also keeping a safe distance from a maniac. There's nothing refreshing or new abo ut this plot which pretty much goes through the motions. Not one of the sho w's highlights.
Étiquette: Négative

Index de la note de review: 591
Texte de la note de review:
 The perfect murder is foiled when a wife(played by Mary Ellen Trainor, once the wife to director Robert Zemeckis, who helmed this episode), who murders her husband with a poker, has the misfortune of receiving a visitor as she i s about to move the body outside..an escaped insane madman dressed in a Sant a Claus suit(played by a deviously hideous Larry Drake). She fends for her l ife while trying to find a way of hiding her husband's corpse. She decides t o use an ax, once she downs the Santa killer who misses several chances to c hop off the woman's head, to frame the killer for her husband's murder. Sant a killer locks her in a closet and pursues the woman's daughter as she tries desperate to free herself to save the child.

This episode of TALE S FROM THE CRYPT just recycles tired material involving the old "Santa kill s" theme while also adding the oft-used(add nauseum)woman-murders-her-husban d-for-a-man-she's-been-cheating-with routine. It's essentially Trainor tryin g to find a way to avoid being caught with a dead body she kills while also keeping a safe distance from a maniac. There's nothing refreshing or new abo ut this plot which pretty much goes through the motions. Not one of the sho w's highlights.
Étiquette: Négative

Index de la note de review: 281
Texte de la note de review:
 The Sunshine Boys is a terrific comedy about two ex-vaudevillians who reluc tantly reunite for a TV special despite the fact that they despise each othe r.

The comic genius of two masters at work, George Burns and Walt er Matthau are stellar! Some of the best scenes are when the duo is fighting over the silliest little trivial things! The material is fast-paced and witt y, appealing to all ages.

MILD SPOILER ALERT: There are some mild ly sad moments toward the end of the movie that deal indirectly with the aff ects of aging that gives the film a soft, sincere, tenderness that shows to this reviewer that what the pair really need the most for success, are each other.

If anyone loves The Odd Couple, you'll adore this movie. A n excellent film!
Étiquette: Positive

Index de la note de review: 281
Texte de la note de review:
 The Sunshine Boys is a terrific comedy about two ex-vaudevillians who reluc
tantly reunite for a TV special despite the fact that they despise each othe
r.

The comic genius of two masters at work, George Burns and Walt
er Matthau are stellar! Some of the best scenes are when the duo is fighting
over the silliest little trivial things! The material is fast-paced and witt
y, appealing to all ages.

MILD SPOILER ALERT: There are some mild
ly sad moments toward the end of the movie that deal indirectly with the aff
ects of aging that gives the film a soft, sincere, tenderness that shows to
this reviewer that what the pair really need the most for success, are each
other.

If anyone loves The Odd Couple, you'll adore this movie. A
n excellent film!
Étiquette: Positive

Index de la note de review: 279
Texte de la note de review:
 For a danish movie, I have to say, that this is very good movie.
<br /
>It's in a class of its own, yet it has an international potential.

The movie has a big budget, and is starring famous danish actors, and a fe
w newcomers, who play very well. It can be watched by anyone who like advent
ures, and a little bit of 'ghost' movie.

Don't be afraid, be thri
lled!
Étiquette: Positive

Index de la note de review: 279
Texte de la note de review:
 For a danish movie, I have to say, that this is very good movie.
<br /
>It's in a class of its own, yet it has an international potential.

The movie has a big budget, and is starring famous danish actors, and a fe
w newcomers, who play very well. It can be watched by anyone who like advent
ures, and a little bit of 'ghost' movie.

Don't be afraid, be thri
lled!
Étiquette: Positive

Index de la note de review: 98
Texte de la note de review:
 in one of Neil Simon's best plays. Creaky, cranky ex-Vaudeville stars playe
d by Walter Matthau and George Burns are teaming up for a TV comedy special.
The problem is they haven't even SEEN each other in over a decade. Full of z
ippy one liners and inside showbiz jokes, this story flies along with a stea
dy stream of humor. Good work also by Richard Benjamin as the harried nephe
w, Rosetta LeNoire as the nurse, and Howard Hesseman as the TV commercial di
rector. Steve Allen and Phyllis Diller appear as themselves. Trivia note: Th
e opening montage contains footage from Hollywood Revue of 1929 and shows Ma
rie Dressler, Bessie Love, Polly Moran, Cliff Edwards, Charles King, Gus Edw
ards, and the singing Brox Sisters.
Étiquette: Positive

Index de la note de review: 98
Texte de la note de review:
 in one of Neil Simon's best plays. Creaky, cranky ex-Vaudeville stars playe
d by Walter Matthau and George Burns are teaming up for a TV comedy special.
The problem is they haven't even SEEN each other in over a decade. Full of z
ippy one liners and inside showbiz jokes, this story flies along with a stea

```
dy stream of humor. Good work also by Richard Benjamin as the harried nephe
w, Rosetta LeNoire as the nurse, and Howard Hesseman as the TV commercial di
rector. Steve Allen and Phyllis Diller appear as themselves. Trivia note: Th
e opening montage contains footage from Hollywood Revue of 1929 and shows Ma
rie Dressler, Bessie Love, Polly Moran, Cliff Edwards, Charles King, Gus Edw
ards, and the singing Brox Sisters.
Étiquette: Positive

Index de la note de review: 18
Texte de la note de review:
 Working-class romantic drama from director Martin Ritt is as unbelievable a
s they come, yet there are moments of pleasure due mostly to the charisma of
stars Jane Fonda and Robert De Niro (both terrific). She's a widow who can't
move on, he's illiterate and a closet-inventor--you can probably guess the r
est. Adaptation of Pat Barker's novel "Union Street" (a better title!) is so
laid-back it verges on bland, and the film's editing is a mess, but it's sti
ll pleasant; a rosy-hued blue-collar fantasy. There are no overtures to seri
ous issues (even the illiteracy angle is just a plot-tool for the ensuing lo
ve story) and no real fireworks, though the characters are intentionally a b
it colorless and the leads are toned down to an interesting degree. The fina
le is pure fluff--and cynics will find it difficult to swallow--though these
two characters deserve a happy ending and the picture wouldn't really be sat
isfying any other way. *** from ****
Étiquette: Positive

Index de la note de review: 18
Texte de la note de review:
 Working-class romantic drama from director Martin Ritt is as unbelievable a
s they come, yet there are moments of pleasure due mostly to the charisma of
stars Jane Fonda and Robert De Niro (both terrific). She's a widow who can't
move on, he's illiterate and a closet-inventor--you can probably guess the r
est. Adaptation of Pat Barker's novel "Union Street" (a better title!) is so
laid-back it verges on bland, and the film's editing is a mess, but it's sti
ll pleasant; a rosy-hued blue-collar fantasy. There are no overtures to seri
ous issues (even the illiteracy angle is just a plot-tool for the ensuing lo
ve story) and no real fireworks, though the characters are intentionally a b
it colorless and the leads are toned down to an interesting degree. The fina
le is pure fluff--and cynics will find it difficult to swallow--though these
two characters deserve a happy ending and the picture wouldn't really be sat
isfying any other way. *** from ****
Étiquette: Positive
```

Vectorisation

In [8]:
```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
```

In [9]:
```
def vectorize_text(data, vectorizer):
        """Vectorize text data using specified vectorizer."""
        vectorized_data = vectorizer.fit_transform(data)
        return vectorized_data, vectorizer
```

In [10]:
```
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
```

In [11]:
```
count_vectorizer = CountVectorizer(stop_words='english', max_features=5000)
```

In [12]:
```
features, current_vectorizer = vectorize_text(reviews_df['review'], tfidf_ve
print("Forme de la matrice TF-IDF :", features.shape)
```

```
Forme de la matrice TF-IDF : (602, 5000)
```

In [13]:
```
features_count, _ = vectorize_text(reviews_df['review'], count_vectorizer)
print("Forme de la matrice Count :", features_count.shape)
```

```
Forme de la matrice Count : (602, 5000)
```

Entraînement

1. Logistic Regression

In [14]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
```

In [15]:
```
def train_and_evaluate(features, labels):
        """Train and evaluate a logistic regression model."""
        X_train, X_test, y_train, y_test = train_test_split(features, labels
        model = LogisticRegression(random_state=42)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        print("Accuracy:", accuracy_score(y_test, y_pred))
        print("Classification Report:\n", classification_report(y_test, y_pr
```

In [16]:
```
train_and_evaluate(features, reviews_df['label'])
```

```
Accuracy: 0.8543046357615894
Classification Report:
               precision    recall  f1-score   support

           0       0.81      0.88      0.85        68
           1       0.90      0.83      0.86        83

    accuracy                           0.85       151
   macro avg       0.85      0.86      0.85       151
weighted avg       0.86      0.85      0.85       151
```

In [17]:
```
train_and_evaluate(features_count, reviews_df['label'])
```

```
Accuracy: 0.8079470198675497
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.78      0.79        68
           1       0.82      0.83      0.83        83

    accuracy                           0.81       151
   macro avg       0.81      0.81      0.81       151
weighted avg       0.81      0.81      0.81       151
```

2. SVM & GridSearchCV

In [18]:
```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
```

In [19]:
```
def train_and_evaluate_svm(X_train, y_train, X_test, y_test):
        param_grid = {
                'C': [0.1, 1, 10, 100],
                'kernel': ['linear', 'rbf'],
                'gamma': ['scale', 'auto']
        }
        grid_search = GridSearchCV(SVC(), param_grid, cv=5, scoring='accurac
        grid_search.fit(X_train, y_train)
        print("Best parameters:", grid_search.best_params_)
        best_model = grid_search.best_estimator_
        y_pred = best_model.predict(X_test)
        print("Accuracy:", accuracy_score(y_test, y_pred))
        print("Classification Report:\n", classification_report(y_test, y_pr
```

In [20]:
```
train_and_evaluate_svm(features, reviews_df['label'], features, reviews_df['
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best parameters: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       301
           1       1.00      1.00      1.00       301

    accuracy                           1.00       602
   macro avg       1.00      1.00      1.00       602
weighted avg       1.00      1.00      1.00       602
```

In [21]:
```
train_and_evaluate_svm(features_count, reviews_df['label'], features_count,
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best parameters: {'C': 100, 'gamma': 'auto', 'kernel': 'rbf'}
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       301
           1       1.00      1.00      1.00       301

    accuracy                           1.00       602
   macro avg       1.00      1.00      1.00       602
weighted avg       1.00      1.00      1.00       602
```

3. Random Forest

In [22]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [23]:
```
def train_and_evaluate_rf(features, labels):
        X_train, X_test, y_train, y_test = train_test_split(features, labels
        model = RandomForestClassifier(n_estimators=100, random_state=42)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        print("Accuracy:", accuracy_score(y_test, y_pred))
        print("Classification Report:\n", classification_report(y_test, y_pr
```

In [24]:
```
train_and_evaluate_rf(features, reviews_df['label'])
```

```
Accuracy: 0.7549668874172185
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.81      0.75        68
           1       0.82      0.71      0.76        83

    accuracy                           0.75       151
   macro avg       0.76      0.76      0.75       151
weighted avg       0.76      0.75      0.76       151
```

In [25]:
```
train_and_evaluate_rf(features_count, reviews_df['label'])
```

```
Accuracy: 0.7549668874172185
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.87      0.76        68
           1       0.86      0.66      0.75        83

    accuracy                           0.75       151
   macro avg       0.77      0.77      0.75       151
weighted avg       0.78      0.75      0.75       151
```

4. Naive Bayes

In [26]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [27]:
```
def train_and_evaluate_nb(features, labels):
        X_train, X_test, y_train, y_test = train_test_split(features, labels
        model = MultinomialNB()
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        print("Accuracy:", accuracy_score(y_test, y_pred))
        print("Classification Report:\n", classification_report(y_test, y_pr
```

In [28]:
```
train_and_evaluate_nb(features, reviews_df['label'])
```

```
Accuracy: 0.8145695364238411
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.91      0.82        68
           1       0.91      0.73      0.81        83

    accuracy                           0.81       151
   macro avg       0.82      0.82      0.81       151
weighted avg       0.83      0.81      0.81       151
```

In [29]:
```
train_and_evaluate_nb(features_count, reviews_df['label'])
```

```
Accuracy: 0.8410596026490066
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.91      0.84        68
           1       0.92      0.78      0.84        83

    accuracy                           0.84       151
   macro avg       0.85      0.85      0.84       151
weighted avg       0.85      0.84      0.84       151
```

5. Decision Tree

In [30]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [31]:
```
def train_and_evaluate_dt(features, labels):
        X_train, X_test, y_train, y_test = train_test_split(features, labels
        model = DecisionTreeClassifier(random_state=42)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        print("Accuracy:", accuracy_score(y_test, y_pred))
        print("Classification Report:\n", classification_report(y_test, y_pr
```

In [32]:
```
train_and_evaluate_dt(features, reviews_df['label'])
```

```
Accuracy: 0.6622516556291391
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.66      0.64        68
           1       0.71      0.66      0.68        83

    accuracy                           0.66       151
   macro avg       0.66      0.66      0.66       151
weighted avg       0.67      0.66      0.66       151
```

In [33]:
```
train_and_evaluate_dt(features_count, reviews_df['label'])
```

```
Accuracy: 0.6291390728476821
Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.63      0.61        68
           1       0.68      0.63      0.65        83

    accuracy                           0.63       151
   macro avg       0.63      0.63      0.63       151
weighted avg       0.63      0.63      0.63       151
```

In [ ]:
```

```