IMDb Movie Reviews Dataset

TABLE OF CONTENTS

THE

Dataset

Dataset

IMDB Dataset

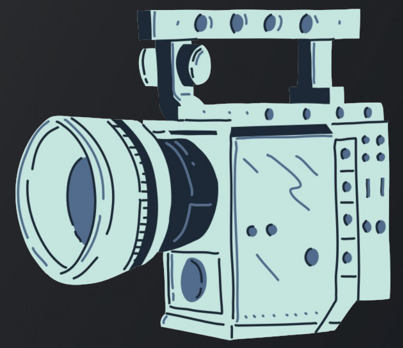
IMDB資料集包含了50,000筆電影評論，可用於自然語言處理或文本分析。 這是一個用於二元情感分類的資料集，比以往的基準資料集含有更多資料。其提供了一組25,000筆極性強烈的電影評論作為訓練資料，另外25,000筆用於測試。

[Kaggle]

定制行銷活動

應用
IMDB Dataset

- 市場定位和預測
- 定制行銷活動
- 評估電影回饋



Dataset

IMDB Dataset

處理空值

```
[5] # 檢查資料集中是否有空值
    if df.isnull().any().any():
        print("Missing values in the dataset:")
        print(df.isnull())
        df = df.dropna()
    else:
        print("No missing values in the dataset.")
```

```
No missing values in the dataset.
```

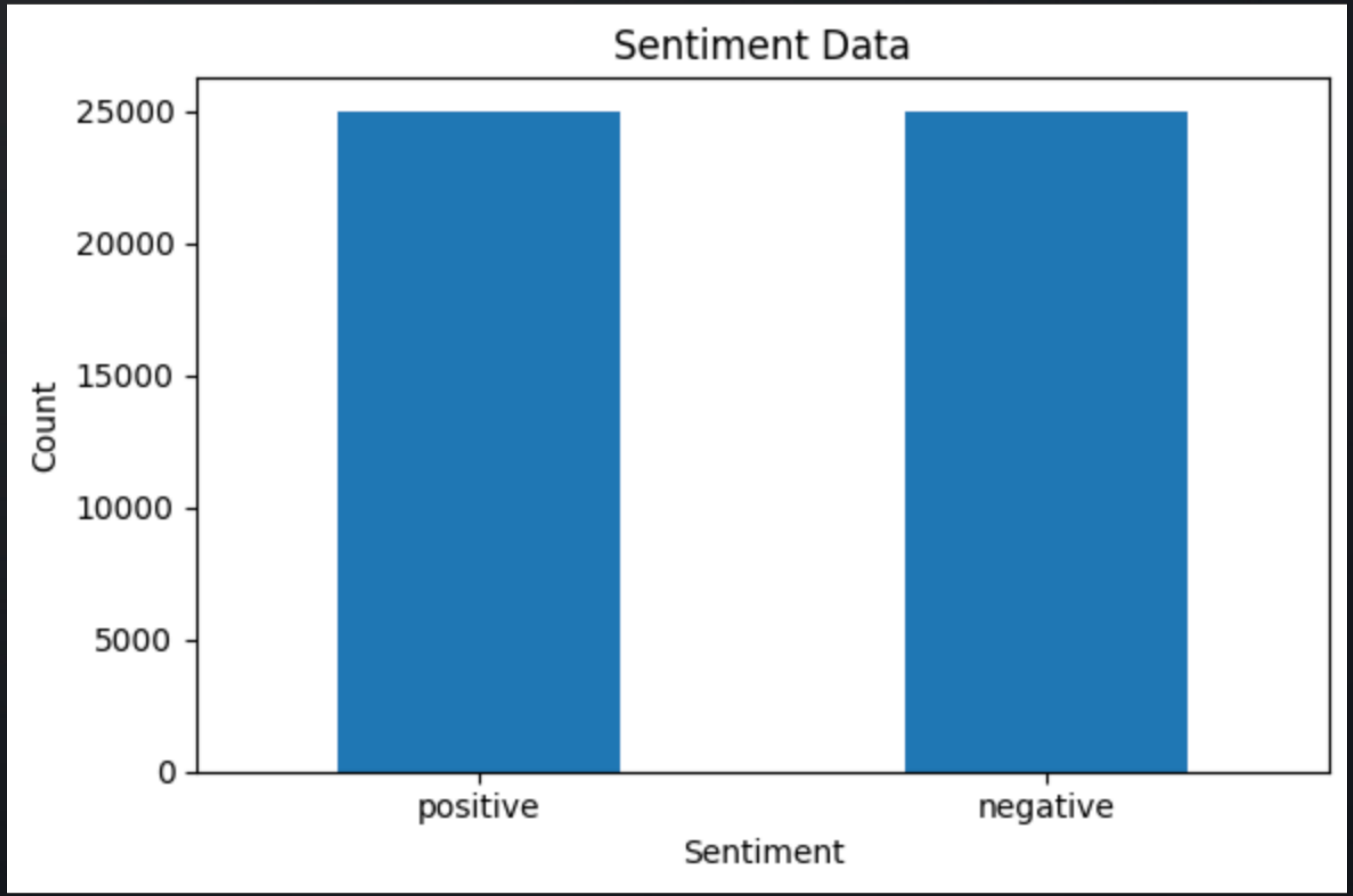
```
[6] print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8500 entries, 10854 to 34711
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   review     8500 non-null   object
 1   sentiment  8500 non-null   object
dtypes: object(2)
memory usage: 199.2+ KB
None
```

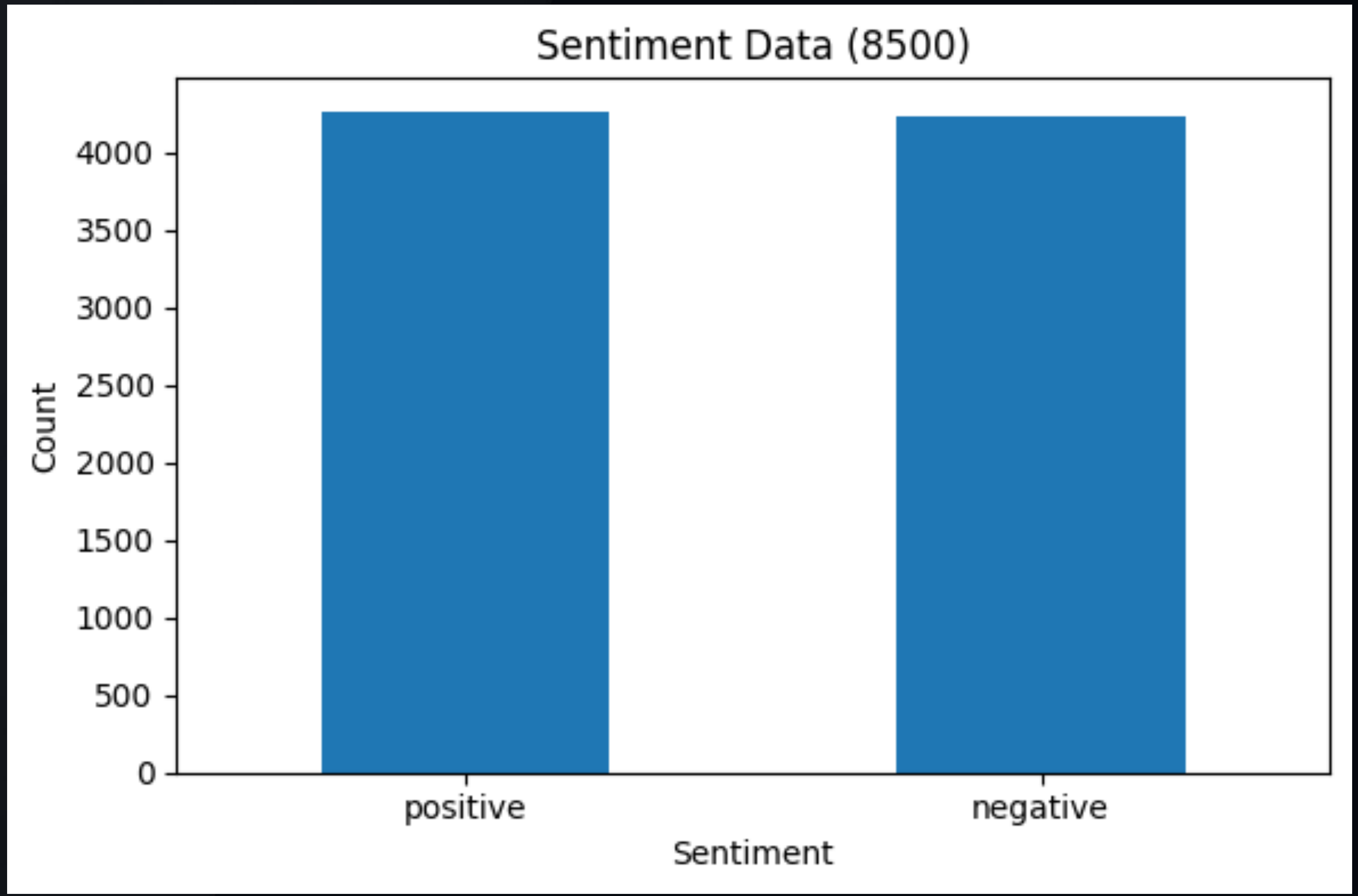
Dataset

IMDB Dataset

原始資料集

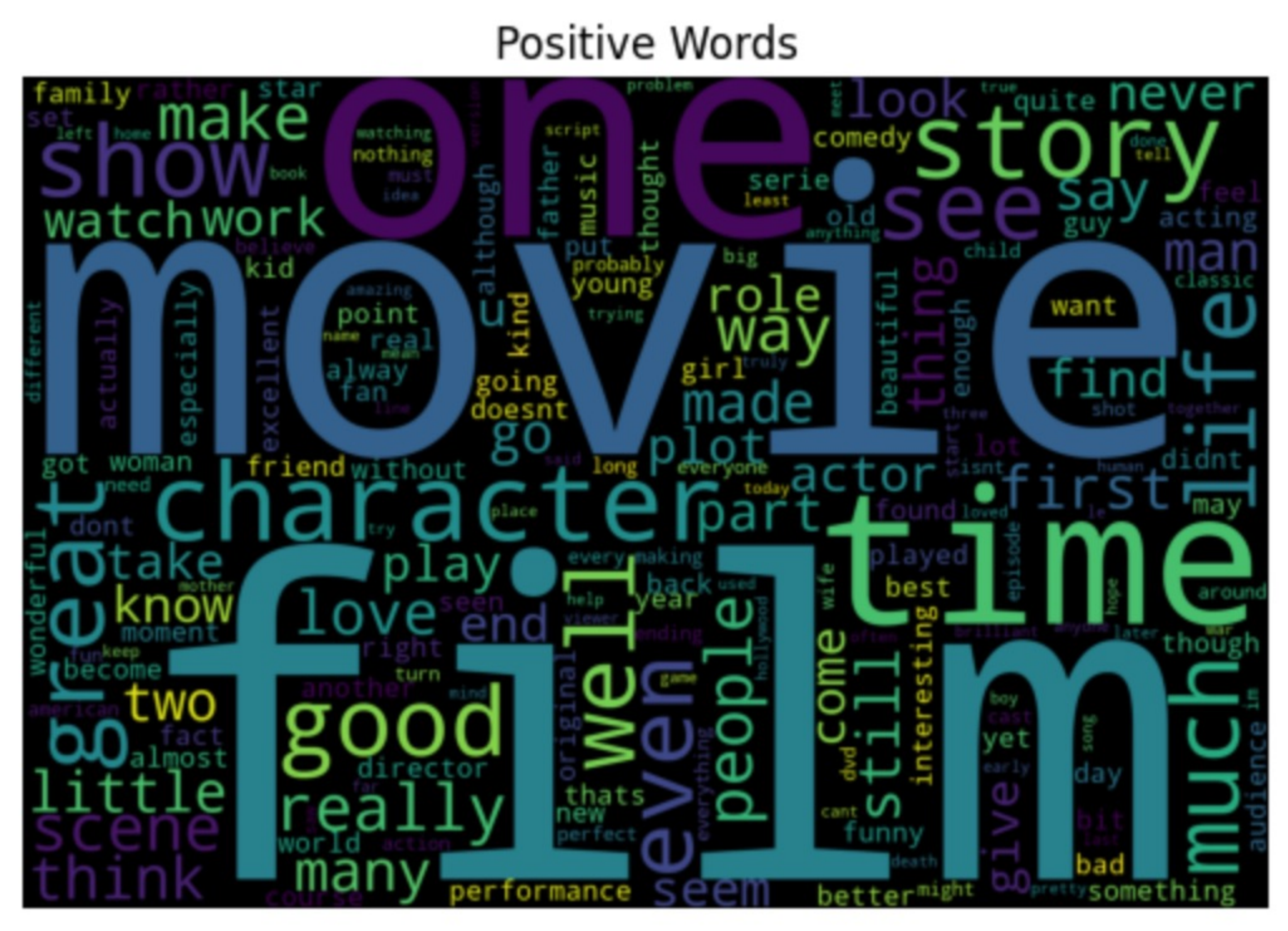


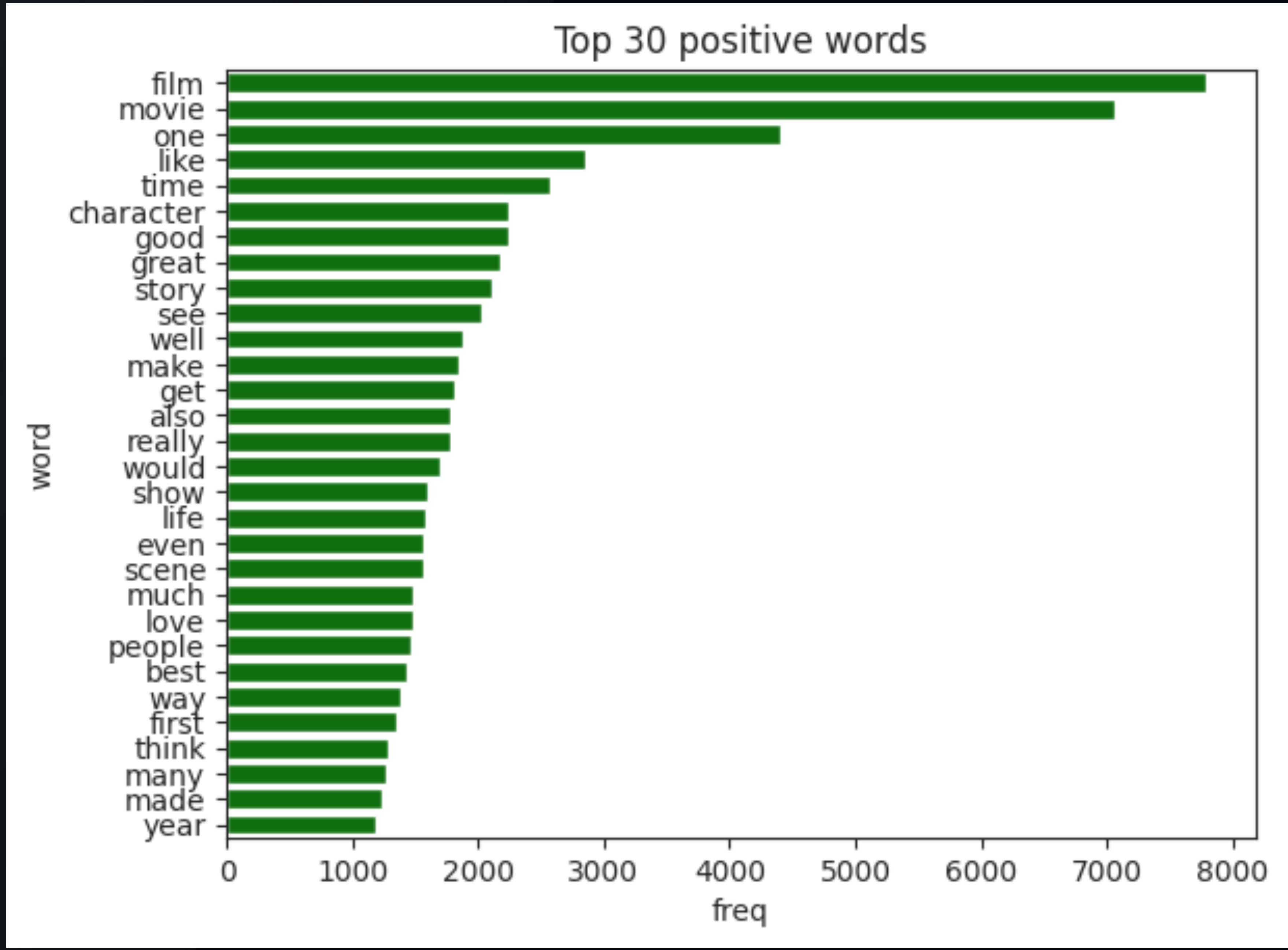
切割後的資料集



Dataset

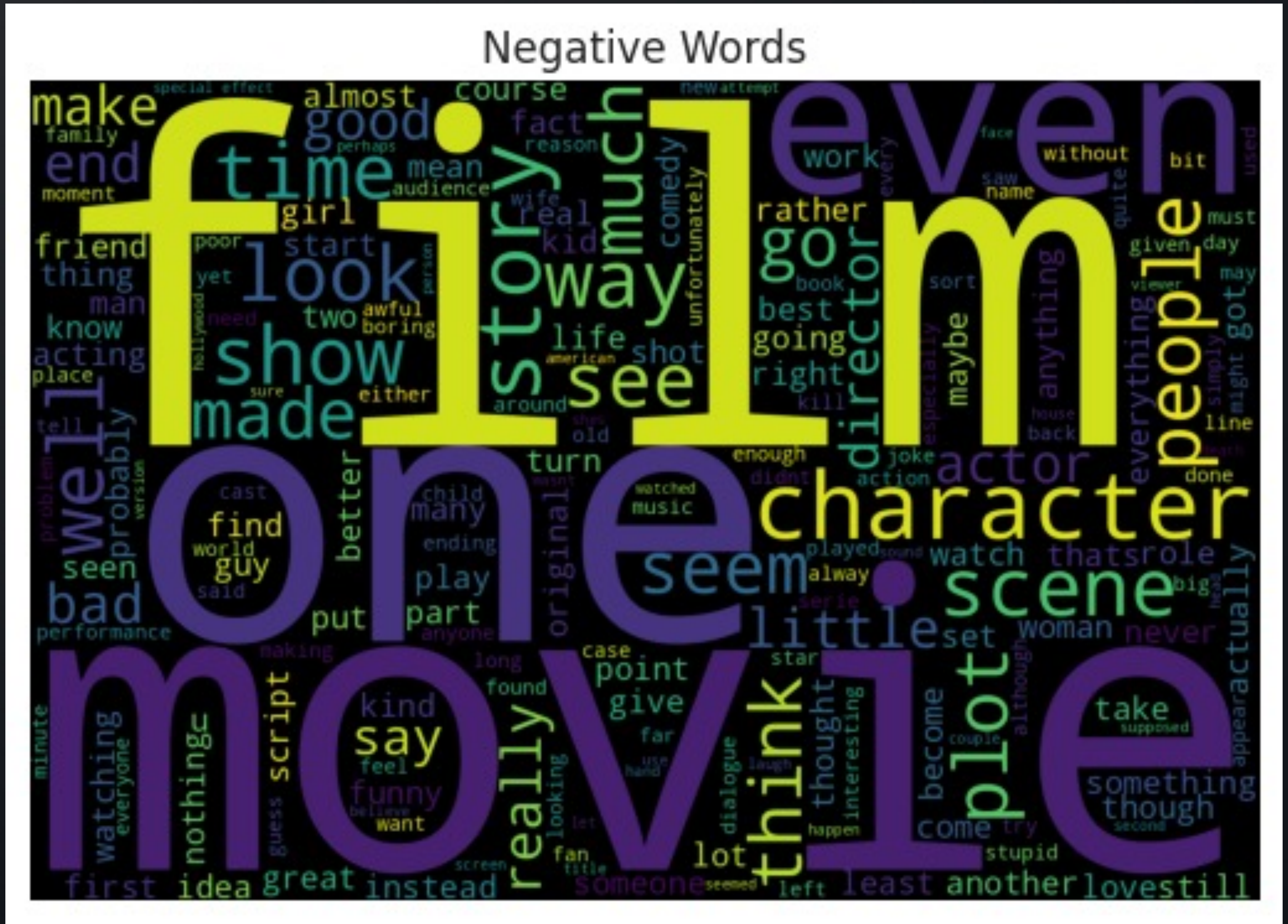
IMDB Dataset

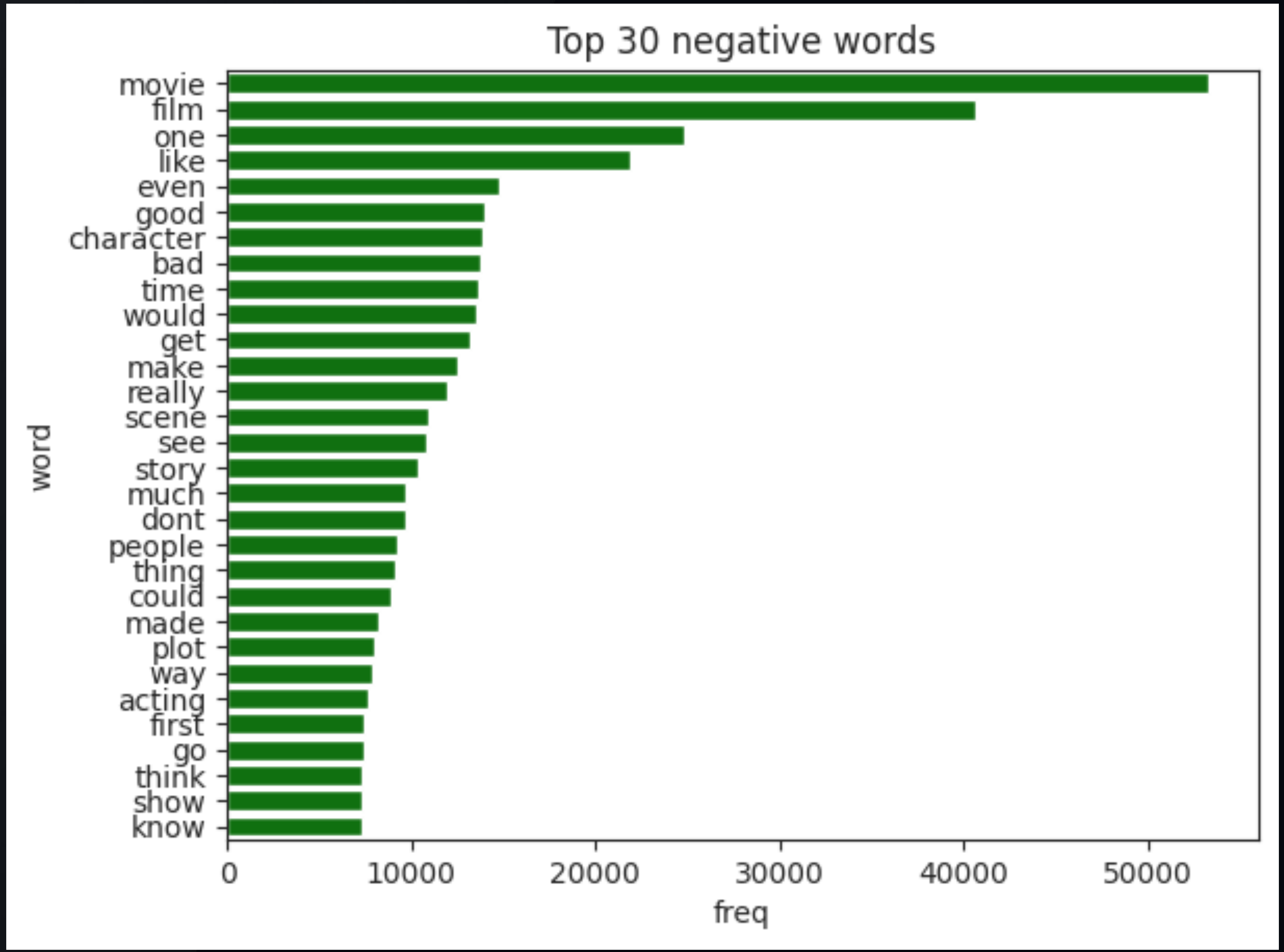




Dataset

IMDB Dataset





THE

Model

Model

IMDB Dataset

文本預處理

```
[ ] # 初始化 WordNet 詞形還原器與停用詞
    lm = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))
```

```
[ ] # 文本預處理函數，將評論轉換為小寫、去除標點符號、標記化、詞形還原等操作
    def transform_data(review):
        # 使用BeautifulSoup移除HTML標記
        review = BeautifulSoup(review, "html.parser").get_text()
        # 將文本轉換為小寫
        review = review.lower()
        # 移除非字母字符
        review = re.sub(r'[^a-zA-Z\s]', '', review)
        # 將文本分詞
        tokens = nltk.word_tokenize(review)
        # 進行詞形還原並移除停用詞
        review = [lm.lemmatize(token) for token in tokens if token not in stop_words]
        # 將處理後的單詞組合成一個文本字串
        review = " ".join(review)
        return review
```

```
[ ] # 將文本資料進行預處理
    tranformed_rev = df.review.apply(transform_data)
```

Model

IMDB Dataset

文本預處理

處理前

```
[ ] print(df.head())
```

```
                                                   review sentiment
10854  This is an hybrid creature born at Carl Macek ...  negative
25169  This isn't one of Arbuckle's or Keaton's bette...  negative
25810  James Aaron, a chubby actor living in Chicago,...  positive
13591  I'll admit that I've never seen "Waiting for G...  positive
26717  NATURAL BORN KILLERS (1994)<br /><br />Cinema ...  negative
```

處理後

```
[ ] print(tranformed_rev.head())
```

```
10854    hybrid creature born carl macek mind robotech ...
25169    isnt one arbuckles keaton better film thats su...
25810    james aaron chubby actor living chicago man lo...
13591    ill admit ive never seen waiting guffman criti...
26717    natural born killer cinema cut r director cut ...
Name: review, dtype: object
```

Model

IMDB Dataset

資料集分割

```
[ ] # 切分訓練集和測試集
    X = tranformed_rev
    y = df.sentiment.replace({'positive': 1, 'negative': 0})
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

把 sentiment (目標屬性) 的 negative 跟 positive 轉成0跟1

Model

IMDB Dataset

資料集分割

訓練資料跟測試資料裡的positive跟negative分布

```
[19] import numpy as np

     # 使用 numpy 的 unique 函數來獲取每個類別的數量及其對應的標籤
     unique_train, counts_train = np.unique(y_train, return_counts=True)
     unique_test, counts_test = np.unique(y_test, return_counts=True)

     # 轉換為字典格式方便查看
     train_distribution = dict(zip(unique_train, counts_train))
     test_distribution = dict(zip(unique_test, counts_test))

     print('y_train 分布:', train_distribution)
     print('y_test 分布:', test_distribution)
```

```
y_train 分布: {0: 3387, 1: 3413}
y_test 分布: {0: 847, 1: 853}
```

Model

IMDB Dataset

特徵向量化 / 初始訓練

```python
# 特徵提取
tf = TfidfVectorizer()#計算詞彙的重要性，TF-IDF特徵提取方法
cv = CountVectorizer()#文本->詞頻特徵矩陣
X_train = tf.fit_transform(X_train).toarray()#轉成TF-IDF特徵矩陣，計算TF-IDF值後回傳稠密矩陣
X_test = tf.transform(X_test).toarray()#使用TF-IDF特徵提取方法
```

```python
# 模型選擇
models = {
    'lr': LogisticRegression(),#邏輯回歸模型
    'rf': RandomForestClassifier(),#隨機森林分類器
    'gs': GaussianNB(),#樸素貝氏分類
    'knn': KNeighborsClassifier(),
    'xgb': XGBClassifier()#XGBoost，基於梯度提升樹算法，連續訓練多個弱分類器提升性能
}
```

```python
# 訓練多個模型並評估其表現
# 回傳預測結果以及模型名稱
def fit_predict(models, X_train, y_trian, X_test, y_test):
    y_pred = []
    models_name = []
    for model_name, model_obj in models.items():#遍歷每個模型
        model_obj.fit(X_train, y_trian)
        print(f'{model_name} done....')
        y_pred.append(model_obj.predict(X_test))
        models_name.append(model_name)
    return y_pred, models_name
```

```python
# 計算模型的準確率
def get_score(y_pred, y_test):
    score = [accuracy_score(y_test, y) for y in y_pred]
    return score
```

Model

IMDB Dataset

初始訓練

```
[ ] print(X_train.shape)
    print(y_train.shape)

    y_pred, models_name = fit_predict(models, X_train, y_train, X_test, y_test)
```

```
(6800, 58860)
(6800,)
lr done....
rf done....
gs done....
knn done....
xgb done....
```

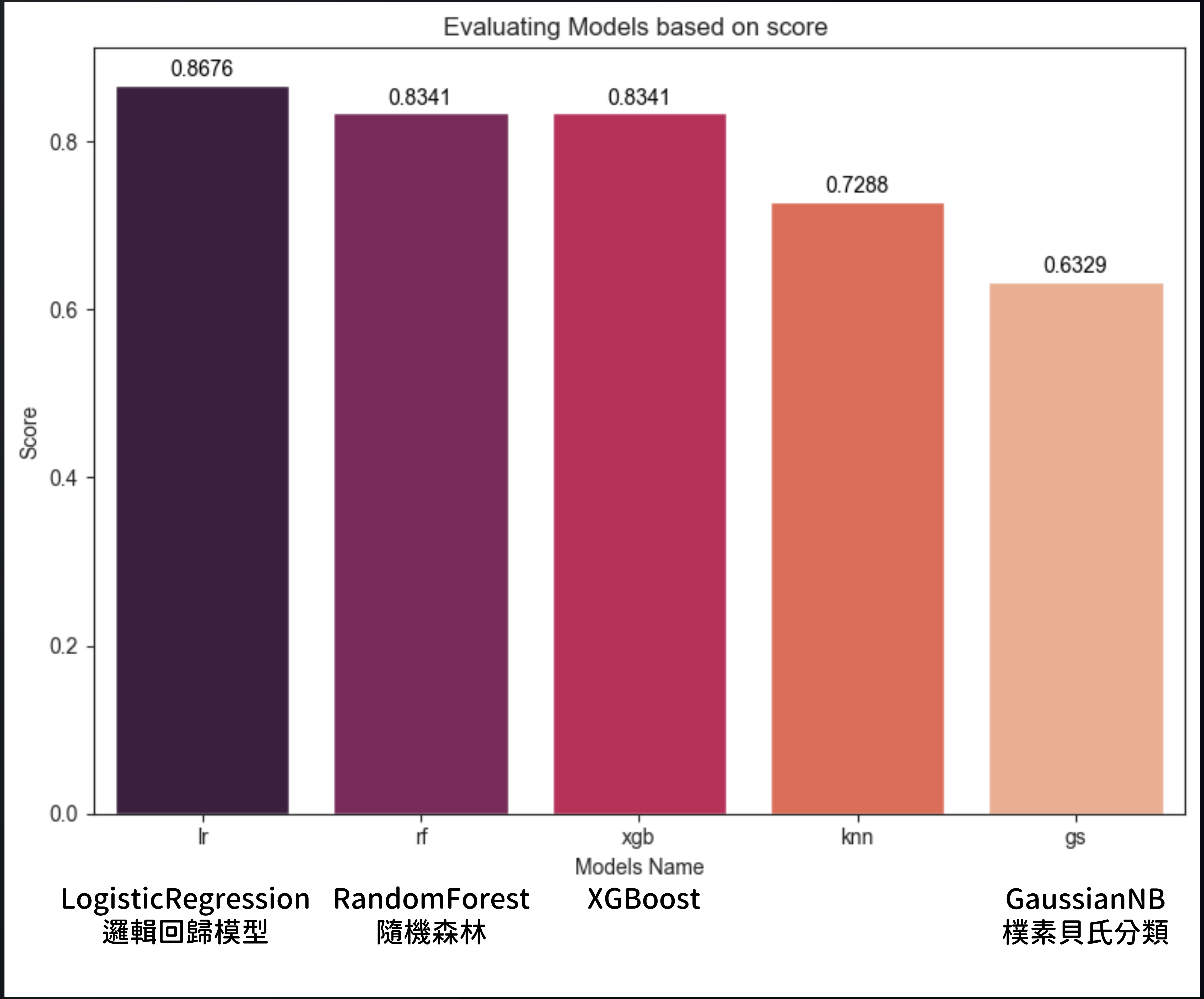
THE

Result

Result

IMDB Dataset

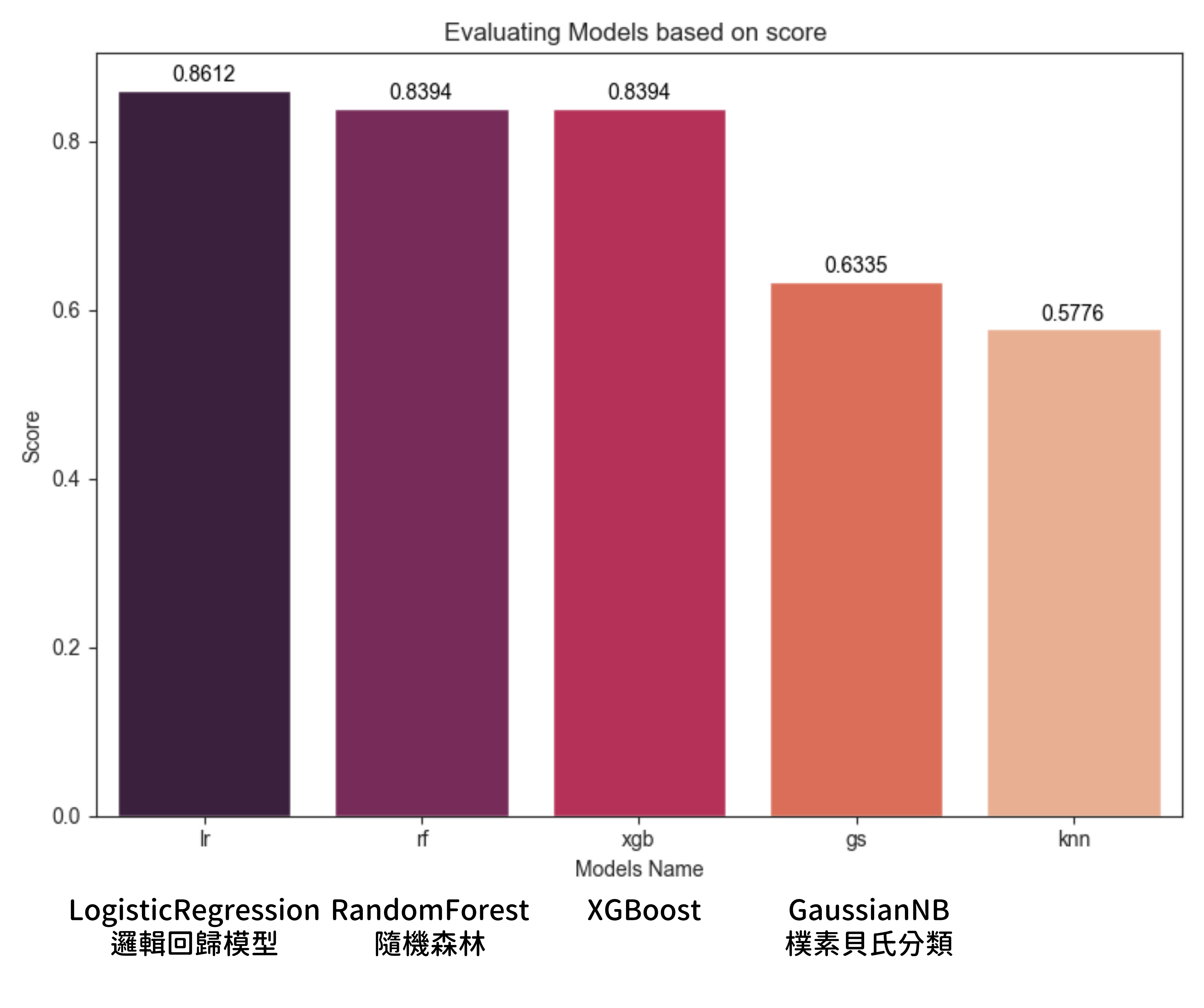
TF-IDF 向量化



Result

IMDB Dataset

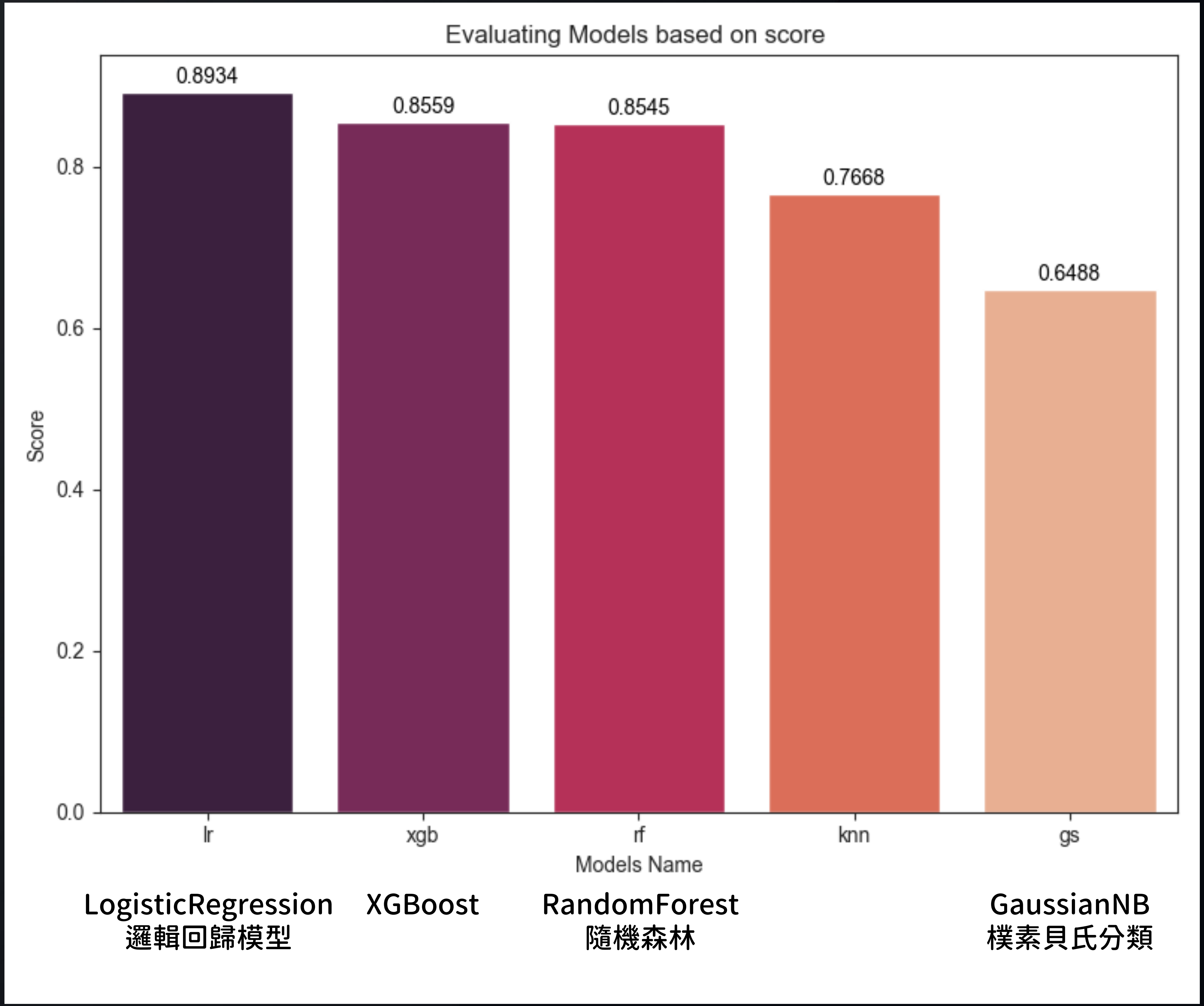
CountVectorizer 向量化



Result

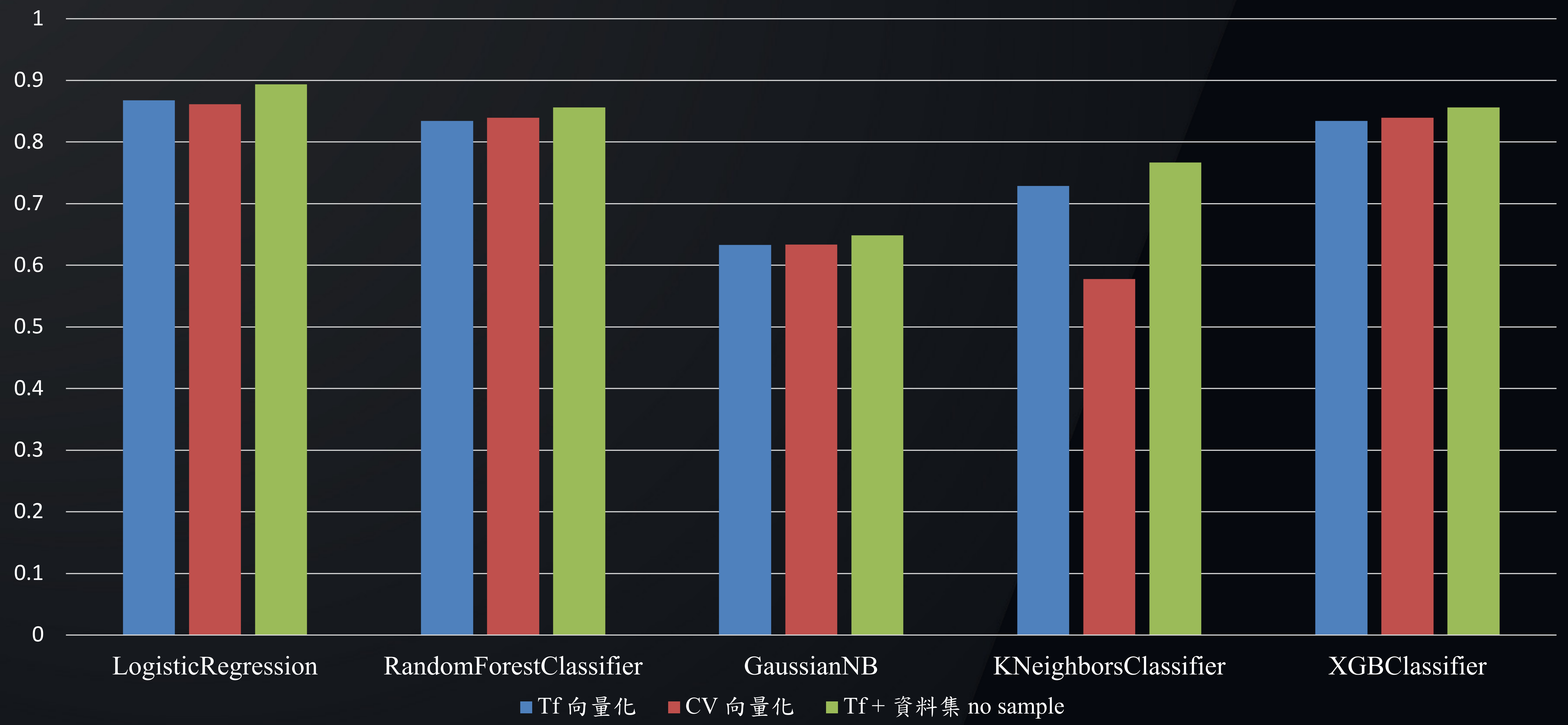
IMDB Dataset

TF-IDF + 資料集不採樣

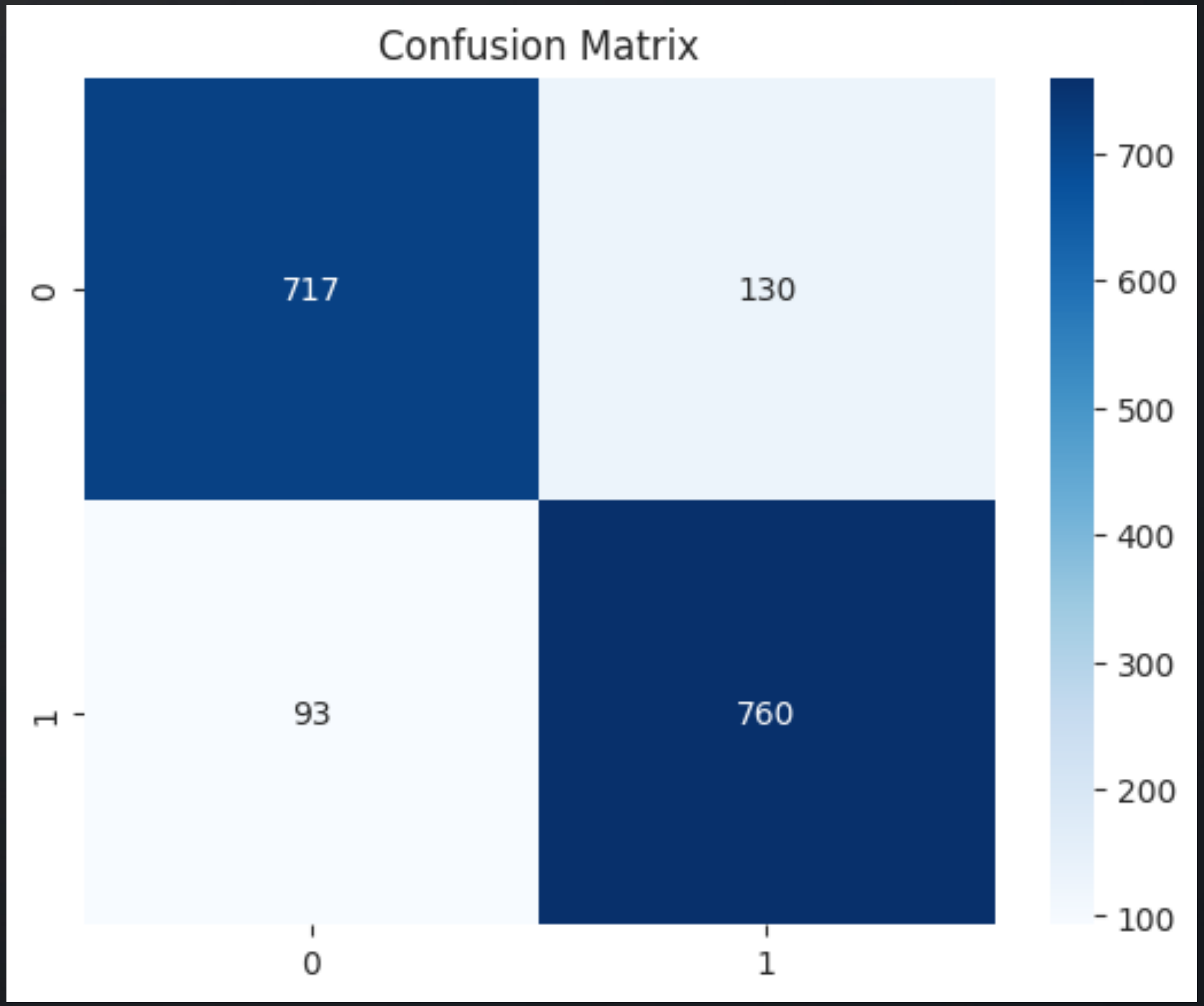


Result

IMDB Dataset



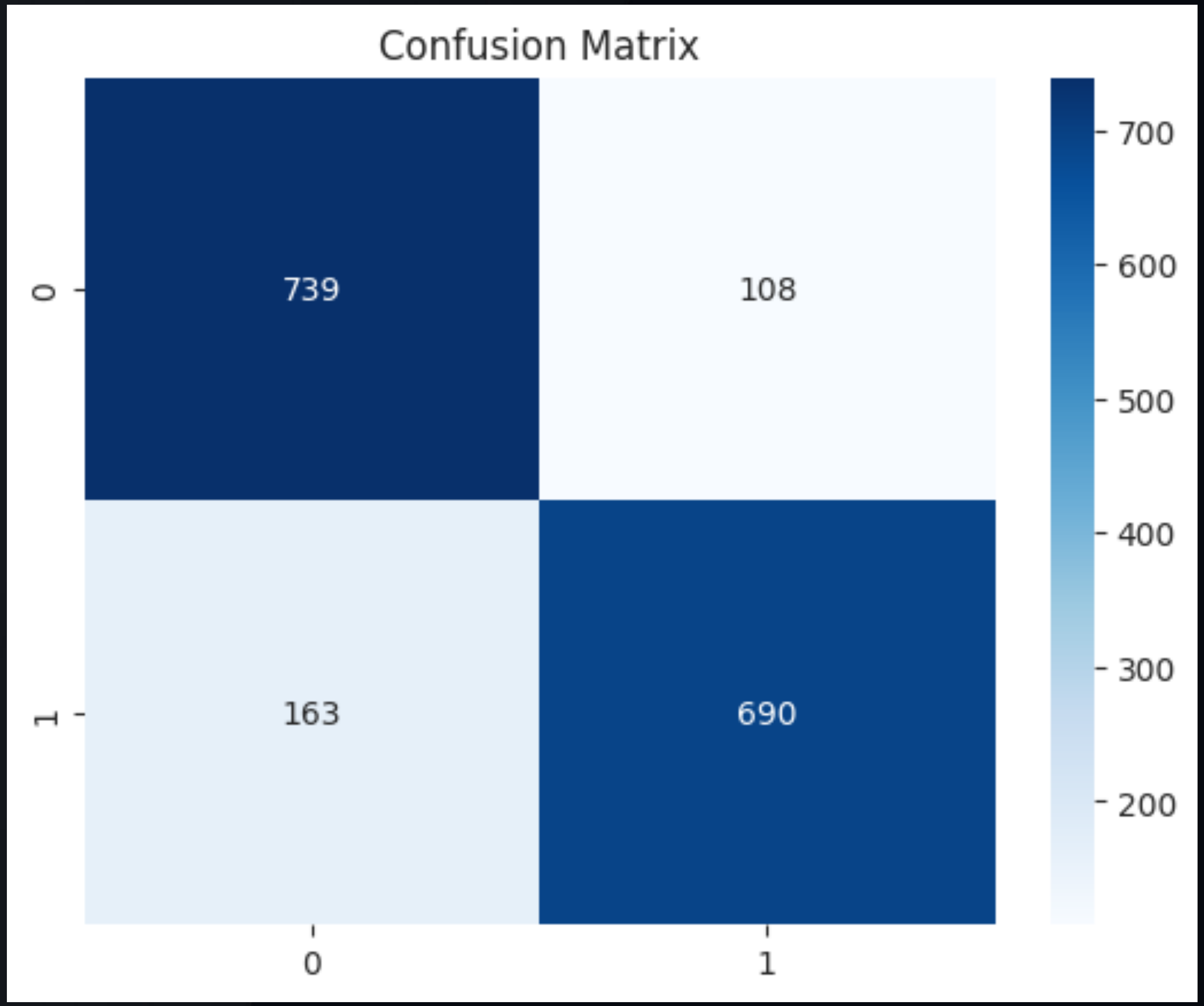
Evaluating Model : LR



Classification Report:

	precision	recall	f1-score	support
0	0.89	0.85	0.87	847
1	0.85	0.89	0.87	853
accuracy			0.87	1700
macro avg	0.87	0.87	0.87	1700
weighted avg	0.87	0.87	0.87	1700

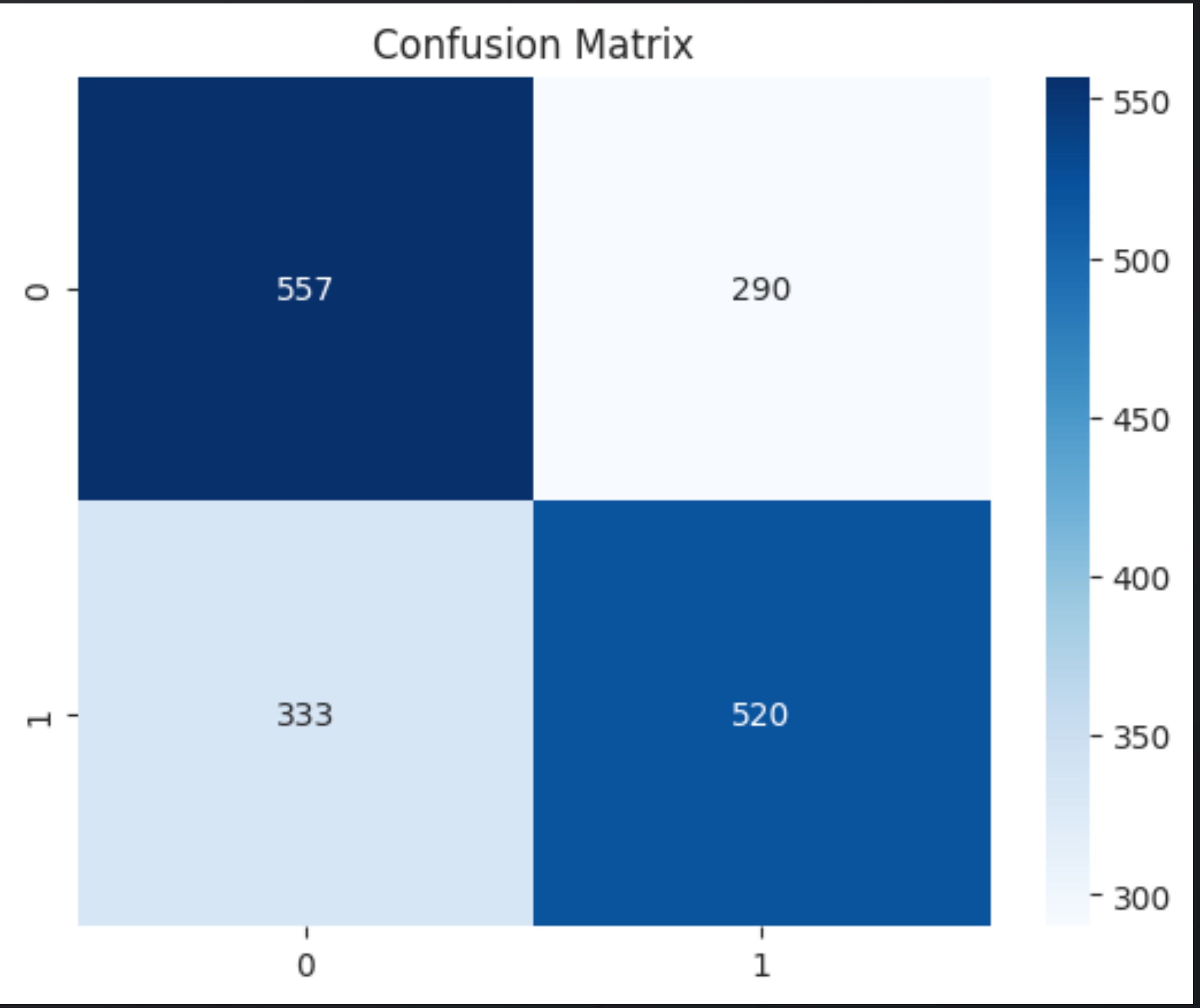
Evaluating Model : RF



Classification Report:

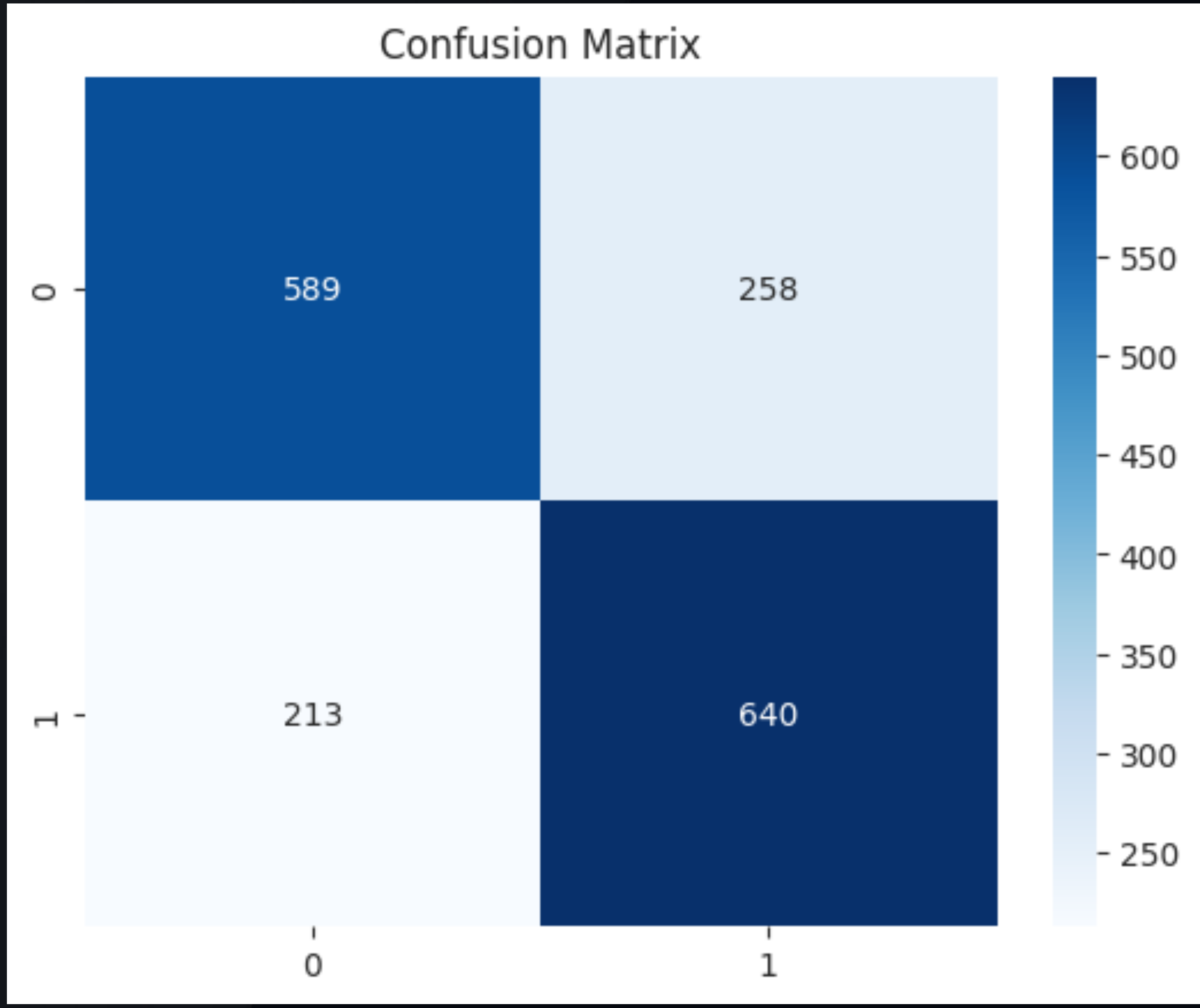
	precision	recall	f1-score	support
0	0.82	0.87	0.85	847
1	0.86	0.81	0.84	853
accuracy			0.84	1700
macro avg	0.84	0.84	0.84	1700
weighted avg	0.84	0.84	0.84	1700

Evaluating Model : GS



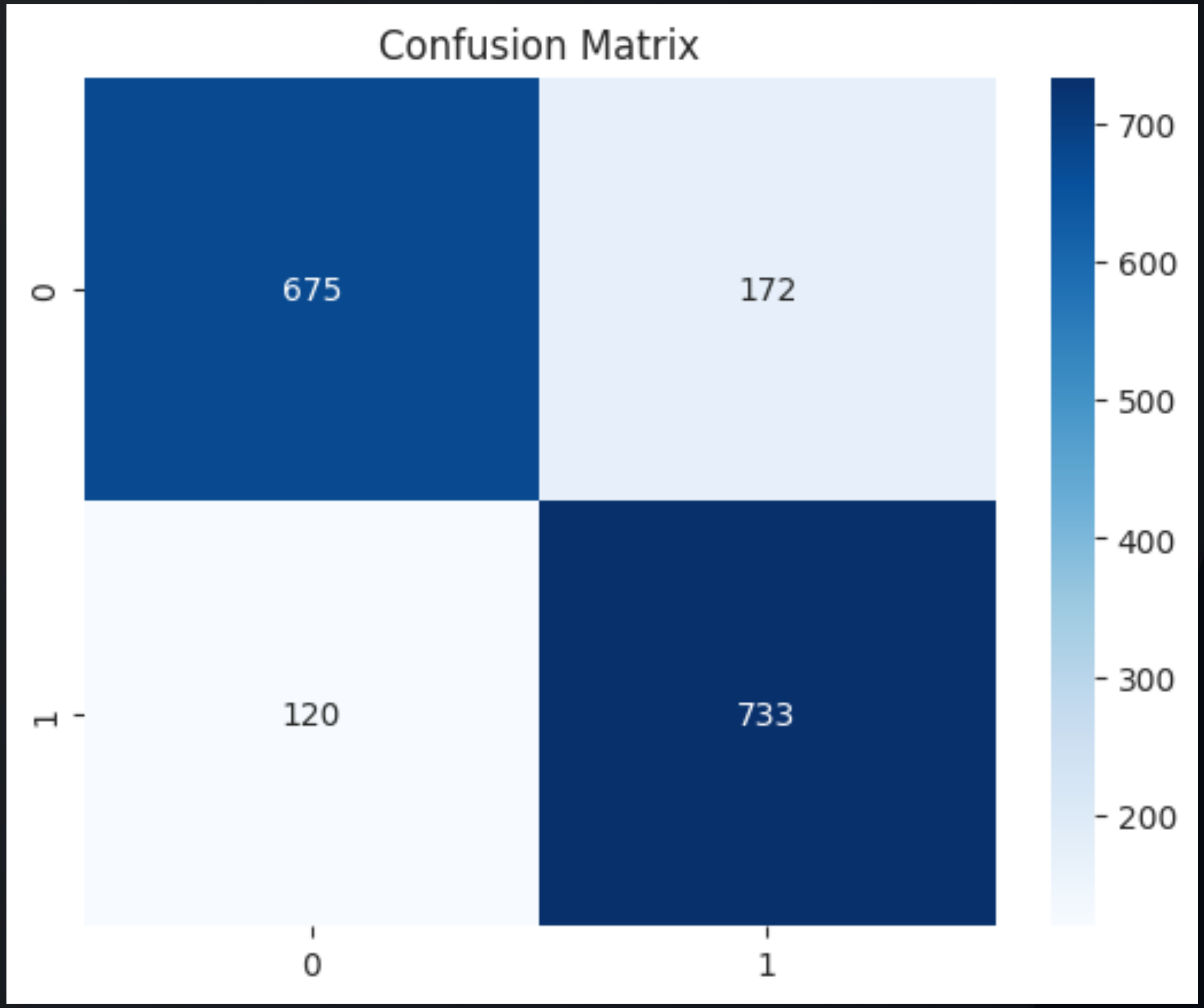

```
Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.66      0.64       847
           1       0.64      0.61      0.63       853

    accuracy                           0.63      1700
   macro avg       0.63      0.63      0.63      1700
weighted avg       0.63      0.63      0.63      1700
```

Evaluating Model : KNN




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

           0       0.73      0.70      0.71       847
           1       0.71      0.75      0.73       853

    accuracy                           0.72      1700
   macro avg       0.72      0.72      0.72      1700
weighted avg       0.72      0.72      0.72      1700
```

Evaluating Model : XGB



Classification Report:

	precision	recall	f1-score	support
0	0.85	0.80	0.82	847
1	0.81	0.86	0.83	853
accuracy			0.83	1700
macro avg	0.83	0.83	0.83	1700
weighted avg	0.83	0.83	0.83	1700

THE

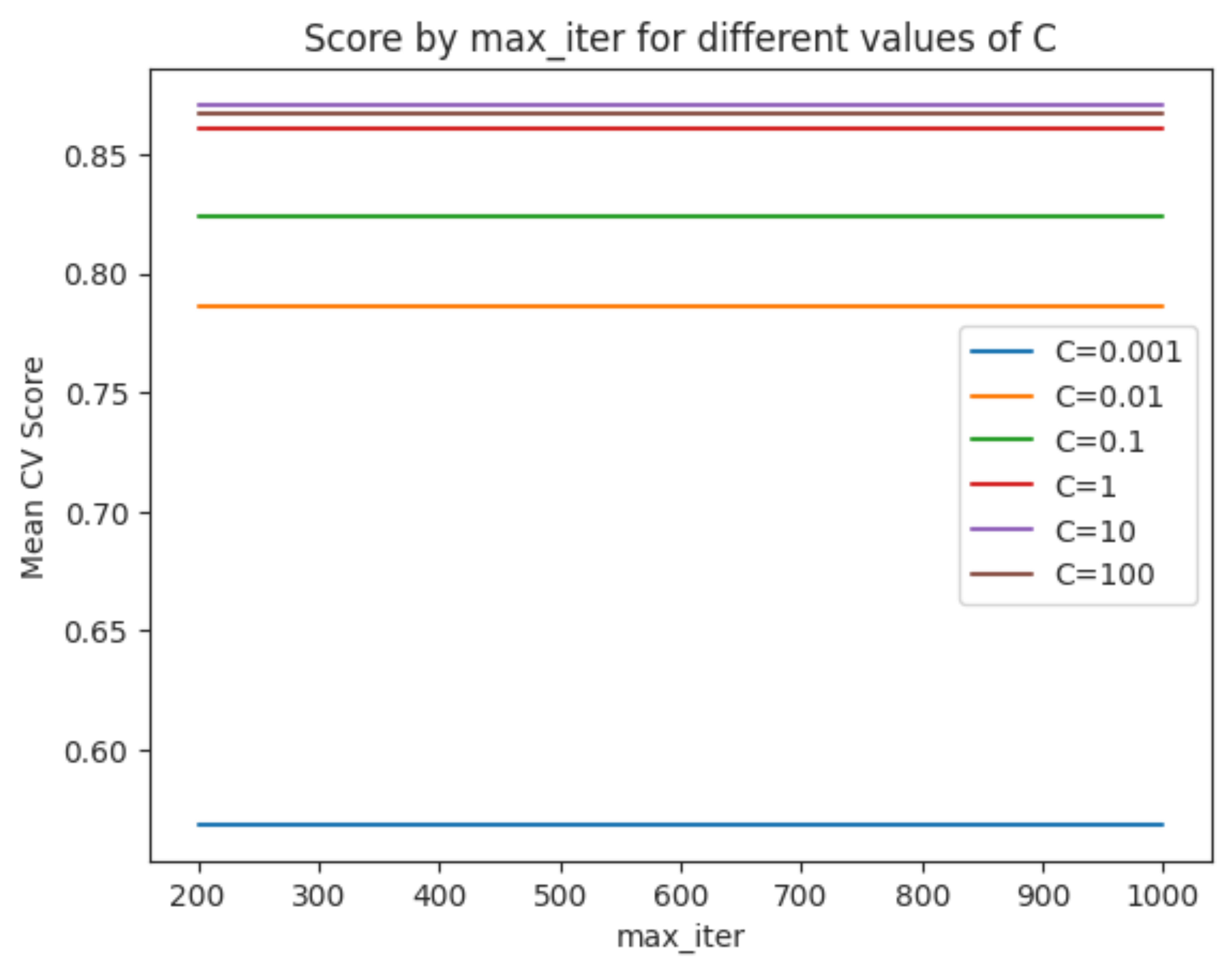
Grid Search

Logistic Regression

Grid Search

IMDB Dataset

Logistic Regression




```
[ ]  from sklearn.model_selection import GridSearchCV

     lr = LogisticRegression(penalty='l2', solver='newton-cg', random_state=42)
     params = {'C': [0.001, 0.01, 0.1, 1, 10, 100], 'max_iter': [200, 400, 600, 800, 1000]}
     grid_search = GridSearchCV(lr, params, verbose=3, cv=5)
     grid_search.fit(X_train, y_train)
```

Grid Search

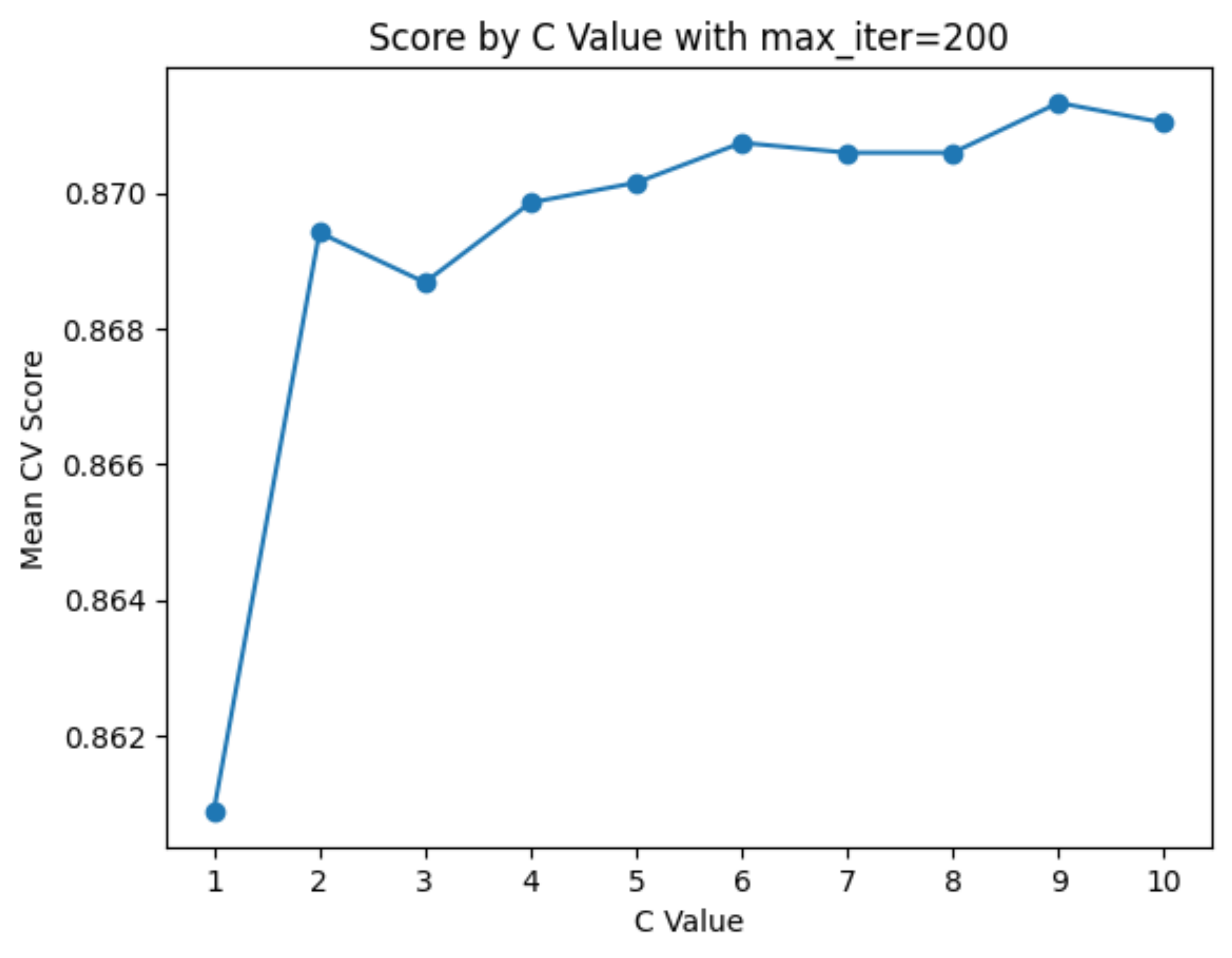
IMDB Dataset

Logistic Regression

```
[11] from sklearn.model_selection import GridSearchCV
     lr = LogisticRegression(penalty='l2', solver='newton-cg', random_state=42)
     params = {'C': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 'max_iter': [200]}
     grid_search = GridSearchCV(lr, params, verbose=3, cv=5)
     grid_search.fit(X_train, y_train)
```

```
[12] grid_search.best_score_

     0.8713235294117647
```

```
[13] grid_search.best_estimator_

     LogisticRegression
     LogisticRegression(C=9, max_iter=200, random_state=42, solver='newton-cg')
```



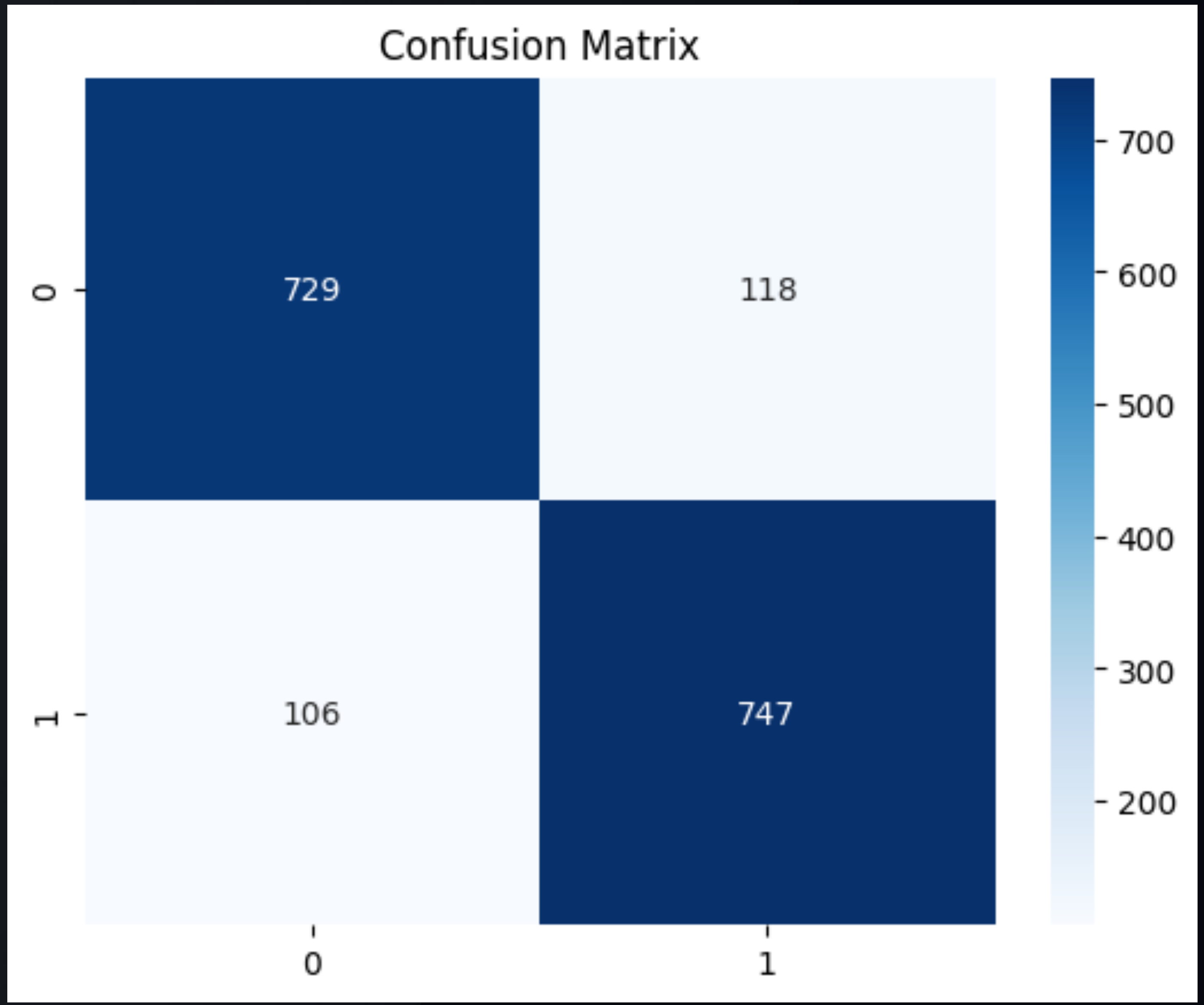
Grid Search

IMDB Dataset

Logistic Regression

最佳

C = 9
max_iter = 200



Classification Report:

	precision	recall	f1-score	support
0	0.87	0.86	0.87	847
1	0.86	0.88	0.87	853
accuracy			0.87	1700
macro avg	0.87	0.87	0.87	1700
weighted avg	0.87	0.87	0.87	1700

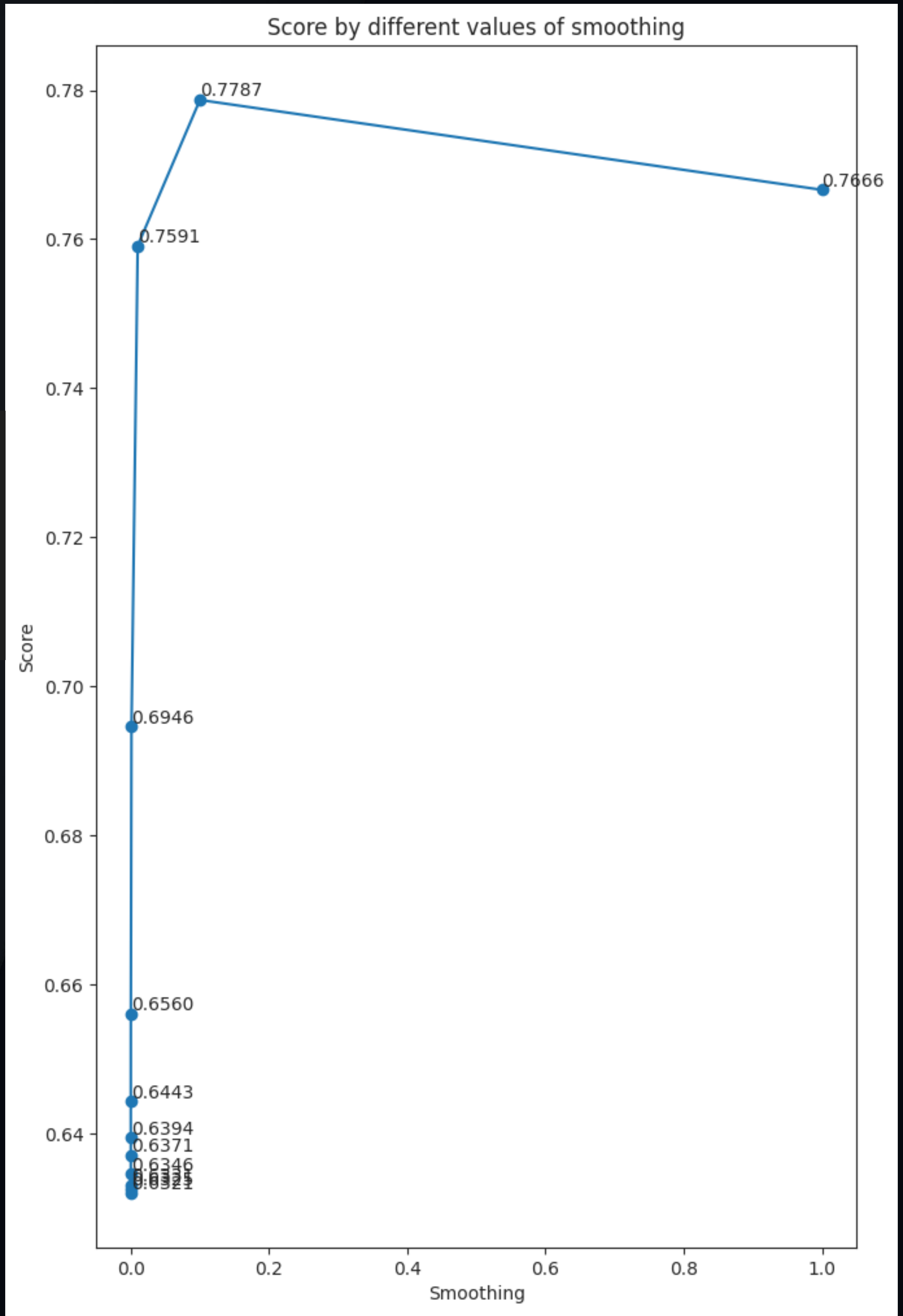
Naive Baise

Grid Search

IMDB Dataset

Naive Baise

```python
from sklearn.model_selection import GridSearchCV
gnb = GaussianNB()
params = {'var_smoothing': [1e-12,1e-11,1e-10,1e-9, 1e-8, 1e-7, 1e-6,1e-5,1e-4,1e-3,1e-2,1e-1,1e-0,1e1]}

grid_search = GridSearchCV(gnb, params, verbose=3, cv=5)
grid_search.fit(X_train, y_train)
```



Grid Search

IMDB Dataset

Naive Baise

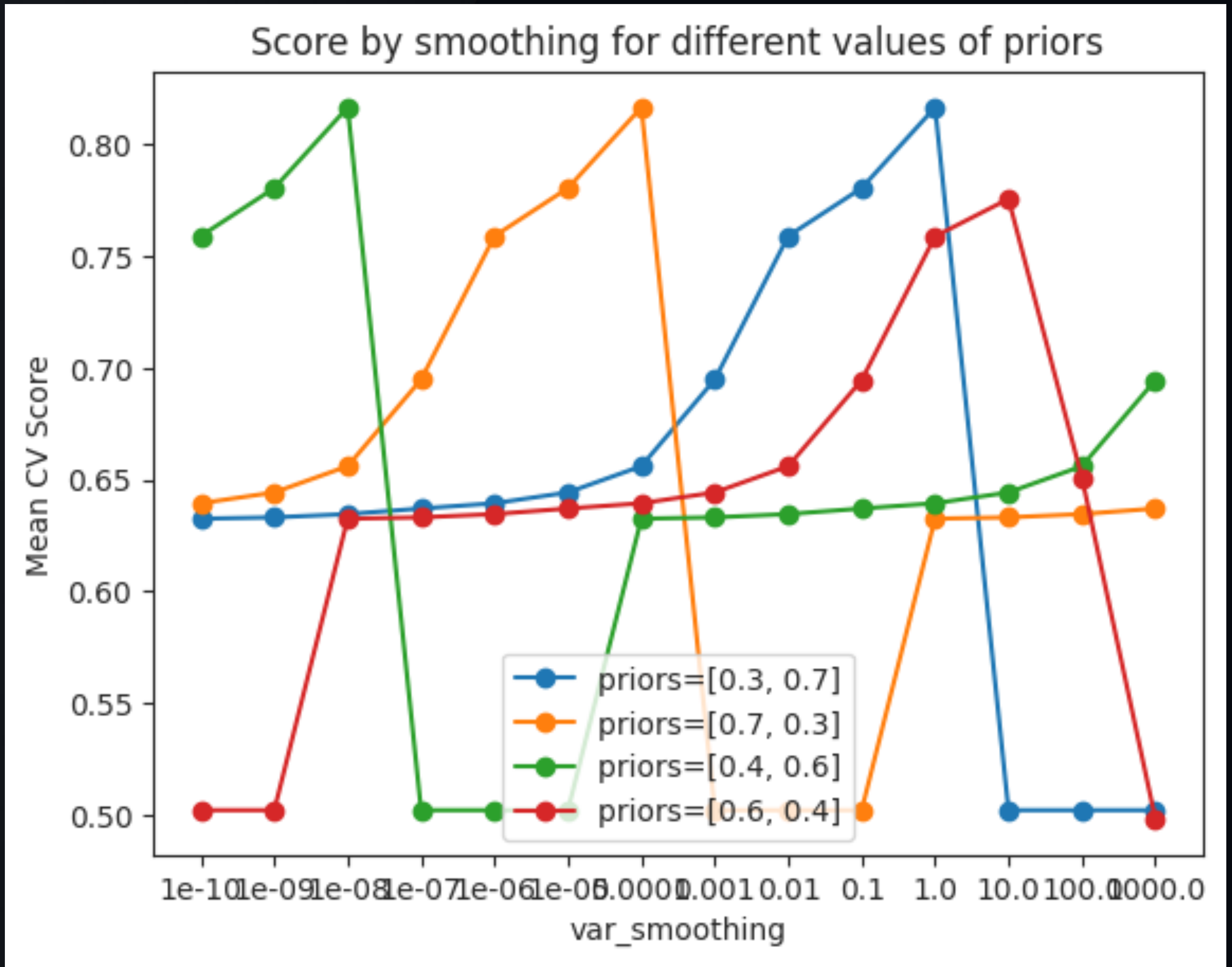
```
from sklearn.model_selection import GridSearchCV
gnb = GaussianNB()
params = {'var_smoothing': [1e-10,1e-9, 1e-8, 1e-7, 1e-6,1e-5,1e-4,1e-3,1e-2,1e-1,1e-0,1e1,1e2,1e3],
          'priors':[[0.3,0.7],[0.7,0.3],[0.4,0.6],[0.6,0.4]]}

grid_search = GridSearchCV(gnb, params, verbose=3, cv=5)
grid_search.fit(X_train, y_train)
```

```
1 grid_search.best_estimator_
```

```
GaussianNB
GaussianNB(priors=[0.3, 0.7], var_smoothing=1.0)
```

```
1 print("Min mean test score:", min(results['mean_test_score']))
2 print("Max mean test score:", max(results['mean_test_score']))
```

```
Min mean test score: 0.4980882352941176
Max mean test score: 0.816764705882353
```



Random Forest

Grid Search

IMDB Dataset

Random Forest

```
[14] from sklearn.model_selection import GridSearchCV

     rf_model = RandomForestClassifier()

     param_grid = {
         'max_depth': [None, 10, 20],
         'min_samples_split': [2, 5, 10],
         'min_samples_leaf': [1, 2, 4],
         'n_estimators': [50, 100, 200]
     }

     grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=2, scoring='accuracy', verbose=3)
```

```
[15] grid_search.fit(X_train, y_train)

     Fitting 2 folds for each of 81 candidates, totalling 162 fits
```

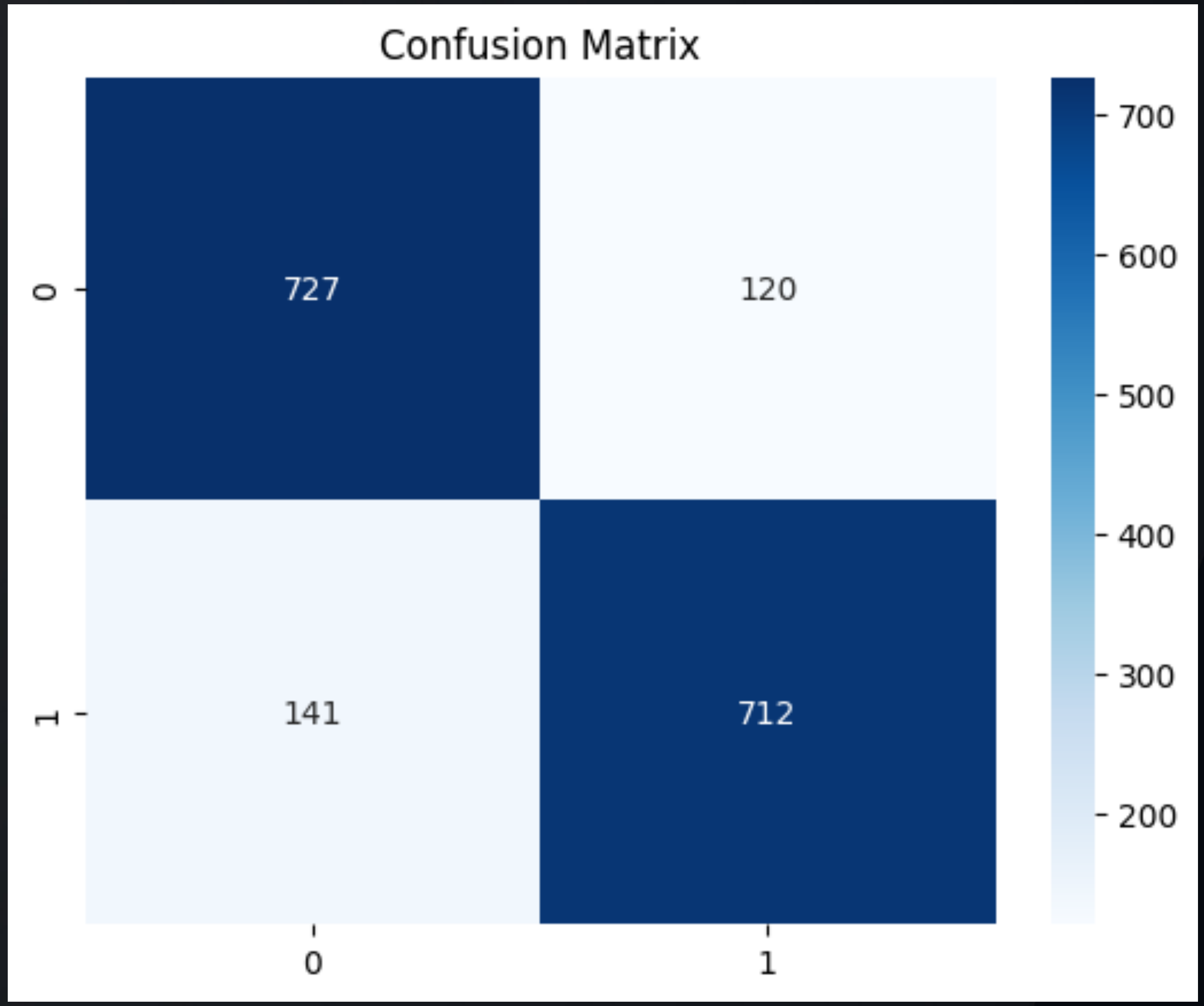
Grid Search

IMDB Dataset

Random Forest

```
[35] print(grid_search.best_params_)

     {'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 5, 'n_estimators': 200}

[36] grid_search.best_estimator_

     ▾ RandomForestClassifier
     RandomForestClassifier(min_samples_leaf=4, min_samples_split=5,
                            n_estimators=200)
```

Random Forest Model (GridSearchCV):



Classification Report:

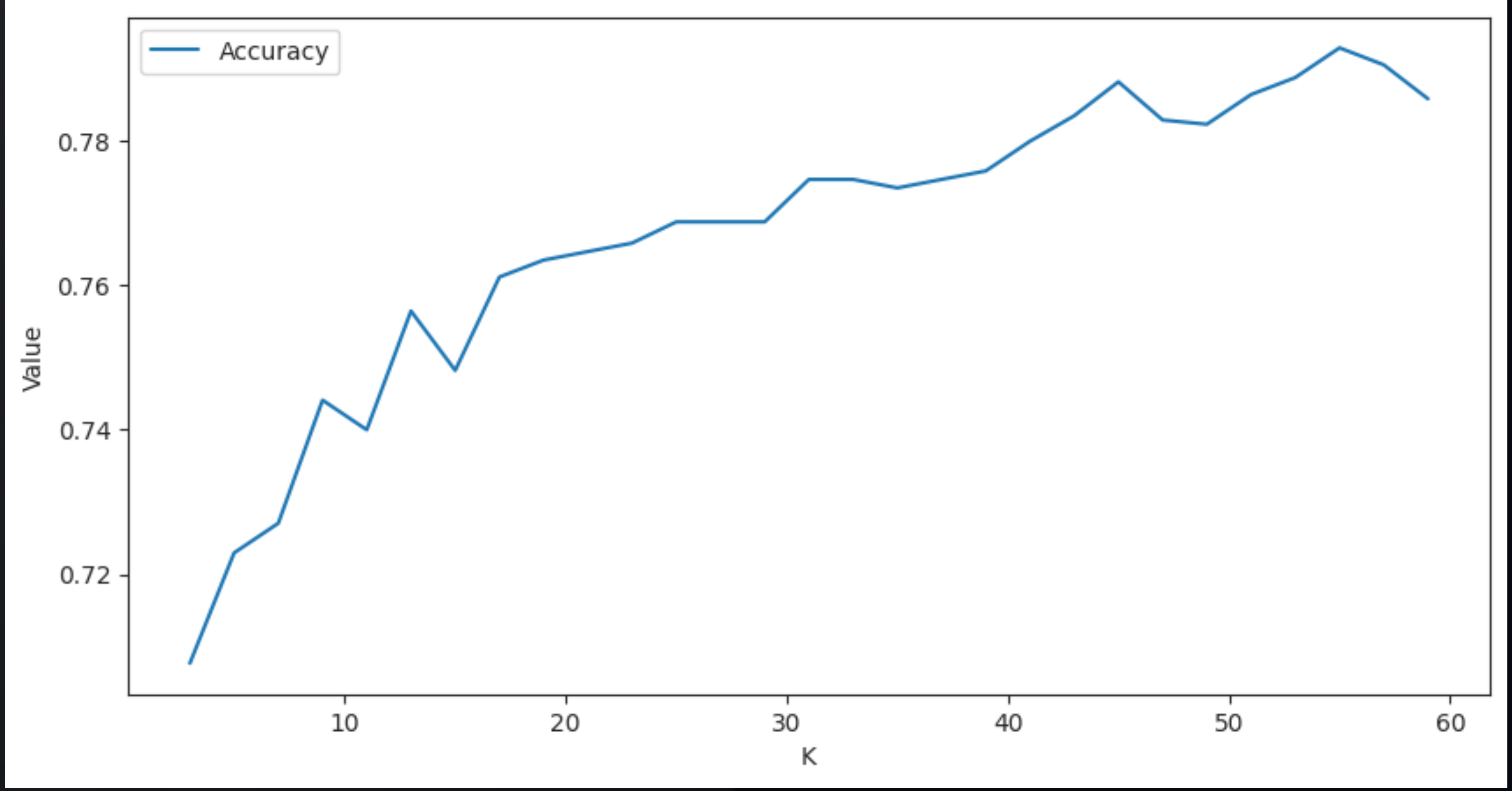
	precision	recall	f1-score	support
0	0.84	0.86	0.85	847
1	0.86	0.83	0.85	853
accuracy			0.85	1700
macro avg	0.85	0.85	0.85	1700
weighted avg	0.85	0.85	0.85	1700

KNN

Grid Search

IMDB Dataset

KNN



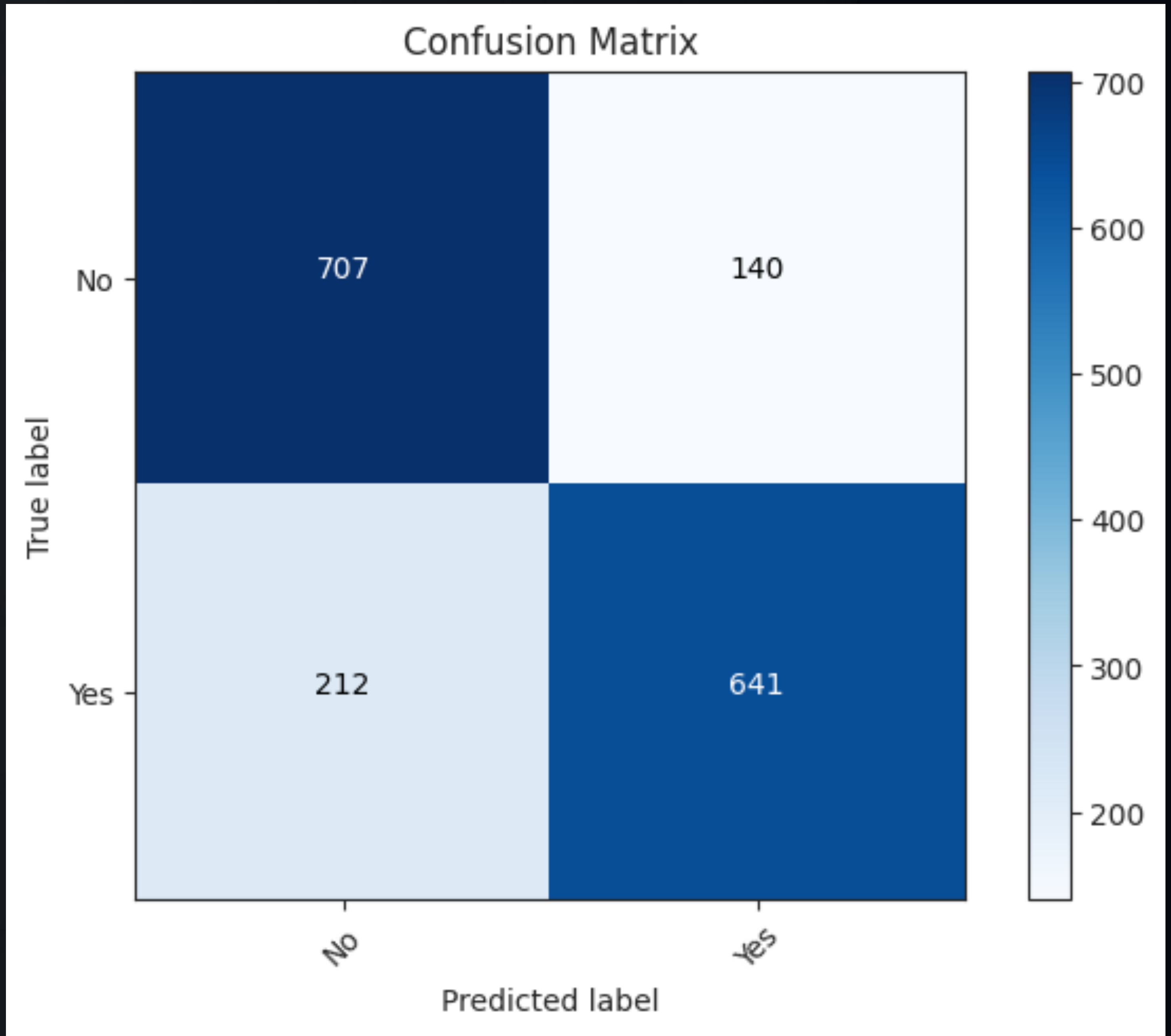
Grid Search

IMDB Dataset

KNN

最佳

$\boxed{K = 55}$



	precision	recall	f1-score	support
0	0.77	0.83	0.80	847
1	0.82	0.75	0.78	853
accuracy			0.79	1700
macro avg	0.80	0.79	0.79	1700
weighted avg	0.80	0.79	0.79	1700

THANK YOU