SVM

kernel = linear

In [14]:

```
import json
import random
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

with open('15152.json', 'r', encoding='utf-8') as f:
    movie_list = json.load(f)

train_data = []
train_labels = []
test_data = []
test_labels = []
for i, movie in enumerate(movie_list):
    if movie['label'] is not None and movie['intro'] is not None:
        if i < 5500: # train
            train_data.append(movie['intro'])
            train_labels.append(movie['label'])
        elif i > 5500 and i < 6001: # test
            test_data.append(movie['intro'])
            test_labels.append(movie['label'])

combined = list(zip(train_data, train_labels)) # 打亂順序
random.shuffle(combined)
train_data[:], train_labels[:] = zip(*combined)

vectorizer = CountVectorizer() # 特徵向量 TD-IDF
tfidf_transformer = TfidfTransformer()
X_train = tfidf_transformer.fit_transform(vectorizer.fit_transform(train_data))
y_train = train_labels

clf = SVC(kernel='linear')
clf.fit(X_train, y_train)

X_test = tfidf_transformer.transform(vectorizer.transform(test_data))
y_pred = clf.predict(X_test)

accuracy = accuracy_score(test_labels, y_pred)
print(f"Prediction Precision: {accuracy:.2%}")
```

```
Prediction Precision: 48.00%
```

kernel = poly

In [15]:

```
import json
import random
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

with open('15152.json', 'r', encoding='utf-8') as f:
    movie_list = json.load(f)

train_data = []
train_labels = []
test_data = []
test_labels = []
for i, movie in enumerate(movie_list):
    if movie['label'] is not None and movie['intro'] is not None:
        if i < 5500: #train
            train_data.append(movie['intro'])
            train_labels.append(movie['label'])
        elif i > 5500 and i < 6001: # test
            test_data.append(movie['intro'])
            test_labels.append(movie['label'])

combined = list(zip(train_data, train_labels)) # 打亂順序
random.shuffle(combined)
train_data[:], train_labels[:] = zip(*combined)

vectorizer = CountVectorizer() # 特徵向量 TD-IDF
tfidf_transformer = TfidfTransformer()
X_train = tfidf_transformer.fit_transform(vectorizer.fit_transform(train_data))
y_train = train_labels

clf = SVC(kernel='poly')
clf.fit(X_train, y_train)

X_test = tfidf_transformer.transform(vectorizer.transform(test_data))
y_pred = clf.predict(X_test)

accuracy = accuracy_score(test_labels, y_pred)
print(f"Prediction Precision: {accuracy:.2%}")
```

```
Prediction Precision: 43.00%
```

kernel = rbf

In [17]:

```
import json
import random
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

with open('15152.json', 'r', encoding='utf-8') as f:
    movie_list = json.load(f)

train_data = []
train_labels = []
test_data = []
test_labels = []
for i, movie in enumerate(movie_list):
    if movie['label'] is not None and movie['intro'] is not None:
        if i < 5500: #train
            train_data.append(movie['intro'])
            train_labels.append(movie['label'])
        elif i > 5500 and i < 6001: # test
            test_data.append(movie['intro'])
            test_labels.append(movie['label'])

combined = list(zip(train_data, train_labels)) # 打亂順序
random.shuffle(combined)
train_data[:], train_labels[:] = zip(*combined)

vectorizer = CountVectorizer() # 特徵向量 TD-IDF
tfidf_transformer = TfidfTransformer()
X_train = tfidf_transformer.fit_transform(vectorizer.fit_transform(train_data))
y_train = train_labels

clf = SVC(kernel='rbf')
clf.fit(X_train, y_train)

X_test = tfidf_transformer.transform(vectorizer.transform(test_data))
y_pred = clf.predict(X_test)

accuracy = accuracy_score(test_labels, y_pred)
print(f"Prediction Precision: {accuracy:.2%}")
```

```
Prediction Precision: 43.60%
```

kernel = sigmoid

In [21]:

```
import json
import random
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

with open('15152.json', 'r', encoding='utf-8') as f:
    movie_list = json.load(f)

train_data = []
train_labels = []
test_data = []
test_labels = []
for i, movie in enumerate(movie_list):
    if movie['label'] is not None and movie['intro'] is not None:
        if i < 5500: #train
            train_data.append(movie['intro'])
            train_labels.append(movie['label'])
        elif i > 5500 and i < 6001: # test
            test_data.append(movie['intro'])
            test_labels.append(movie['label'])

combined = list(zip(train_data, train_labels)) # 打亂順序
random.shuffle(combined)
train_data[:], train_labels[:] = zip(*combined)

vectorizer = CountVectorizer() # 特徵向量 TD-IDF
tfidf_transformer = TfidfTransformer()
X_train = tfidf_transformer.fit_transform(vectorizer.fit_transform(train_data))
y_train = train_labels

clf = SVC(kernel='sigmoid')
clf.fit(X_train, y_train)

X_test = tfidf_transformer.transform(vectorizer.transform(test_data))
y_pred = clf.predict(X_test)

accuracy = accuracy_score(test_labels, y_pred)
print(f"Prediction Precision: {accuracy:.2%}")
```

```
Prediction Precision: 47.60%
```

KNN

In [19]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [20]:

```
import json
import random
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

with open('15152.json', 'r', encoding='utf-8') as f:
    movie_list = json.load(f)

train_data = []
train_labels = []
test_data = []
test_labels = []
for i, movie in enumerate(movie_list):
    if movie['label'] is not None and movie['intro'] is not None:
        if i < 5500: # train
            train_data.append(movie['intro'])
            train_labels.append(movie['label'])
        elif i > 5500 and i < 6001: # test
            test_data.append(movie['intro'])
            test_labels.append(movie['label'])

combined = list(zip(train_data, train_labels)) # 打亂順序
random.shuffle(combined)
train_data[:], train_labels[:] = zip(*combined)

vectorizer = CountVectorizer() # 特徵向量 TD-IDF
tfidf_transformer = TfidfTransformer()
X_train = tfidf_transformer.fit_transform(vectorizer.fit_transform(train_data))
y_train = train_labels

clf = KNeighborsClassifier(n_neighbors=5)
clf.fit(X_train, y_train)

X_test = tfidf_transformer.transform(vectorizer.transform(test_data))
y_pred = clf.predict(X_test)

accuracy = accuracy_score(test_labels, y_pred)
print(f"Prediction Precision: {accuracy:.2%}")
```

```
Prediction Precision: 45.00%
```

決策樹

In [42]:

```
import json
import random
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

with open('15152.json', 'r', encoding='utf-8') as f:
    movie_list = json.load(f)

train_data = []
train_labels = []
test_data = []
test_labels = []
for i, movie in enumerate(movie_list):
    if movie['label'] is not None and movie['intro'] is not None:
        if i < 5500: # train
            train_data.append(movie['intro'])
            train_labels.append(movie['label'])
        elif i > 5500 and i < 6001: # test
            test_data.append(movie['intro'])
            test_labels.append(movie['label'])

combined = list(zip(train_data, train_labels)) # 打亂順序
random.shuffle(combined)
train_data[:], train_labels[:] = zip(*combined)

vectorizer = CountVectorizer() # 特徵向量 TD-IDF
tfidf_transformer = TfidfTransformer()
X_train = tfidf_transformer.fit_transform(vectorizer.fit_transform(train_data))
y_train = train_labels

clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)

X_test = tfidf_transformer.transform(vectorizer.transform(test_data))
y_pred = clf.predict(X_test)

accuracy = accuracy_score(test_labels, y_pred)
print(f"Prediction Precision: {accuracy:.2%}")
```

```
Prediction Precision: 43.60%
```

Discussion

我分別用了SVM kernel = linear, poly, rbf, sigmoid、KNN與決策樹來進行分類，其中SVM kernel = linear的準確率是最高的48.00%，再來是kernel = sigmoid的47.60%，第三個是KNN 45.00%。

In [ ]:

```
```