Movies Genre Classifier

The Movies Genre Classifier project aims to automatically predict the genre of a movie based on its plot summary or script using Natural Language Processing (NLP) techniques. Text data from movies often contains diverse vocabulary, expressions, and contextual cues that can indicate the underlying genre, such as action, comedy, drama, or horror.

In this project, the preprocessing pipeline involves cleaning the text by removing special characters, converting text to lowercase, tokenizing into individual words, removing stop words, and applying stemming to reduce words to their root forms. These steps ensure that the textual data is normalized and ready for feature extraction.

A Multinomial Naïve Bayes (MultinomialNB) model is employed for classification, which is particularly well-suited for text classification tasks involving word frequency features. The final model predicts movie genres efficiently, leveraging a well-prepared corpus built from the preprocessed data.

Double-click (or enter) to edit

```
# Importing essential libraries
import numpy as np
import pandas as pd
```

```
# Loading the dataset
df = pd.read_csv('/content/kaggle_movie_train.csv')
```

Exploring the Dataset

```
df.head()
```

	id	text	genre
0	0	eady dead, maybe even wishing he was. INT. 2ND...	thriller
1	2	t, summa cum laude and all. And I'm about to l...	comedy
2	3	up Come, I have a surprise.... She takes him ...	drama
3	4	ded by the two detectives. INT. JEFF'S APARTME...	thriller
4	5	nd dismounts, just as the other children reach...	drama

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.tail()
```

	id	text	genre
22574	28161	n in the world to decide what I'm going to do ...	drama
22575	28162	shards. BOJO LAZ! Laz pushes Deke back through...	drama
22576	28163	OTTIE You've got a thing about Ernie's, haven'...	thriller
22577	28165	....with marked skill and dexterity . LANA wry...	action
22578	28166	rd walks off down the hallway, leaving his pos...	comedy

```
df.shape
```

```
(22579, 3)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22579 entries, 0 to 22578
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      22579 non-null  int64
 1   text    22579 non-null  object
 2   genre   22579 non-null  object
dtypes: int64(1), object(2)
memory usage: 529.3+ KB
```

```
df.describe()
```

	id
count	22579.000000
mean	14134.852651
std	8132.614667
min	0.000000
25%	7096.500000
50%	14168.000000
75%	21159.000000
max	28166.000000

```
df.columns
```

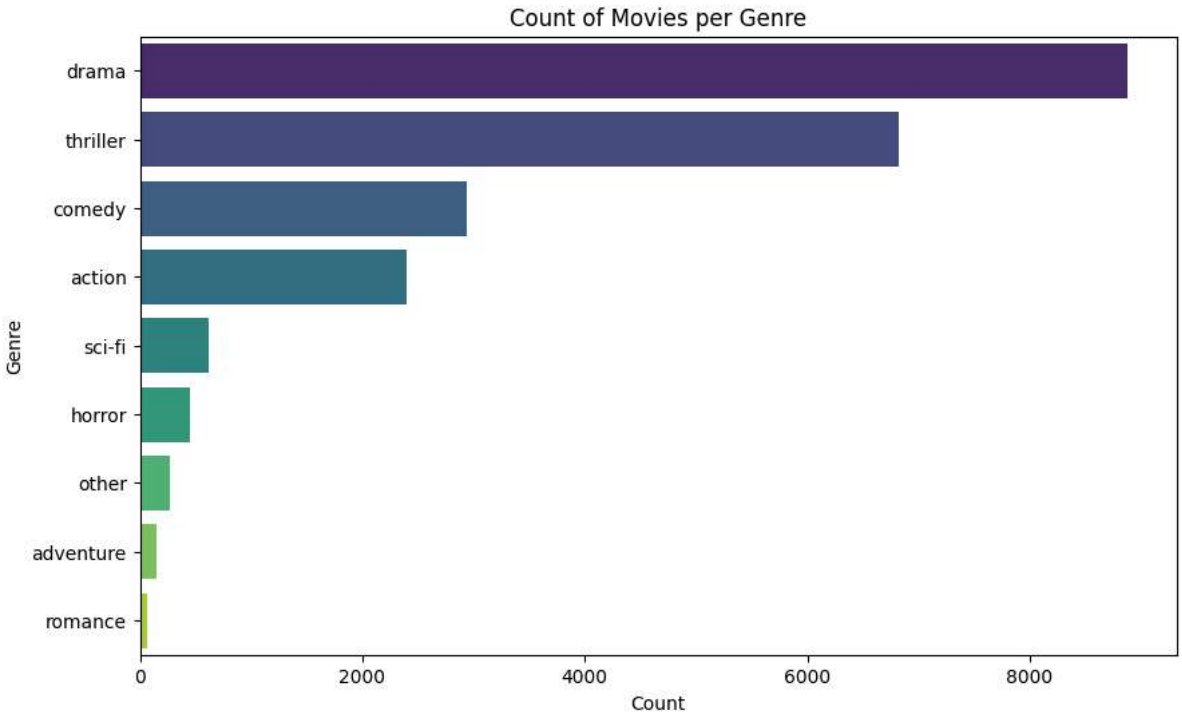
```
Index(['id', 'text', 'genre'], dtype='object')
```

Data Visualization

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
```

```
sns.countplot(y='genre', data=df, order=df['genre'].value_counts().index, palette='viridis')
plt.title('Count of Movies per Genre')
plt.xlabel('Count')
plt.ylabel('Genre')
plt.show()
```

```
/tmp/ipython-input-3443139533.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend

  sns.countplot(y='genre', data=df, order=df['genre'].value_counts().index, palette='viridis')
```



Data Cleaning and Preprocessing

```
# Finding unique genres
movie_genre = list(df['genre'].unique())
movie_genre.sort()
movie_genre
```

```
['action',
 'adventure',
 'comedy',
 'drama',
 'horror',
 'other',
 'romance',
 'sci-fi',
 'thriller']
```

```
# Mapping the genres to values
genre_mapper = {'other': 0, 'action': 1, 'adventure': 2, 'comedy':3, 'drama':4, 'horror':5, 'romance':6, 'sci-fi':7, 'thriller': 8}
df['genre'] = df['genre'].map(genre_mapper)
df.head()
```

	id	text	genre
0	0	eady dead, maybe even wishing he was. INT. 2ND...	8
1	2	t, summa cum laude and all. And I'm about to l...	3
2	3	up Come, I have a surprise.... She takes him ...	4
3	4	ded by the two detectives. INT. JEFF'S APARTME...	8
4	5	nd dismounts, just as the other children reach...	4

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Finding any NaN values
df.isna().any()
```

	0
id	False
text	False
genre	False

dtype: bool

```
# Removing the 'id' column
df.drop('id', axis=1, inplace=True)
df.columns
df.head()
```

	text	genre
0	eady dead, maybe even wishing he was. INT. 2ND...	8
1	t, summa cum laude and all. And I'm about to l...	3
2	up Come, I have a surprise.... She takes him ...	4
3	ded by the two detectives. INT. JEFF'S APARTME...	8
4	nd dismounts, just as the other children reach...	4

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.2.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
```

```
# Importing essential libraries for performing Natural Language Processing on given dataset
import nltk
import re
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
df.shape
```

```
(22579, 2)
```

```
# Cleaning the text
corpus = []
ps = PorterStemmer()

for i in range(0, df.shape[0]):

  # Cleaning special character from the dialog/script
  dialog = re.sub(pattern='[^a-zA-Z]', repl=' ', string=df['text'][i])

  # Converting the entire dialog/script into lower case
  dialog = dialog.lower()

  # Tokenizing the dialog/script by words
  words = dialog.split()

  # Removing the stop words
  dialog_words = [word for word in words if word not in set(stopwords.words('english'))]
```

```
  # Stemming the words
  words = [ps.stem(word) for word in dialog_words]

  # Joining the stemmed words
  dialog = ' '.join(words)

  # Creating a corpus
  corpus.append(dialog)
```

```
corpus[0:10]
```

```
['eadi dead mayb even wish int nd floor hallway three night orderli lead liza door orderli white guy open door step room three white guy mid look wild straight jacket jerri liza reach end rope shake head int decrepit hospit room night ball fetal realli head press cement tri sing jerri blue moon blue moon int nd floor hallway three night liza stand lean rail wall orderli sure go know bad orderli okay liza start hall orderli follow orderli got new patient last week want see liza wave hopeless stop chicken wire window end hall look light break jerri somewher orderli look gotta get back work',
 'summa cum laud launch brand new magazin call expos homag miss juli conroy xenia ohio juli grin juli know find excel editor chief ted yellow page juli let finger walk suddenli music chang peopl ted grin ted play song extend hand dare ask danc juli take hand better ted juli begin danc kiss b g charli jimmi feign tear charli sucker happi end hug jimmi hold start rise nelson hous cloud xenia ted v guess everybodi pretti much live happili ever parent give groceri store descend cloud quickli find ext london buckingham palac day mom dad take pictur smooch front palac ted v manag sneak away second honeymoon',
 'come surpris take hand lead hallway salvator look feel pang seem smaller age wither bodi slightli stoop hair gather knot back head must tire want rest time funer salvator interrupt mamma take hour air know maria smile iron tell year salvator get messag feel guilti think seem incred never come maria open door step asid let son whisper put thing go go salvator lake step flabbergast sight old room perfectli reconstruct preserv look like museum museum past despit bed cloth cupboard book shelv perfectli clear one ever live',
 'ded two detect int jeff apart night medium shot thorwald fight dislodg jeff grip ext jeff apart night close shot look jeff face show strain pain thorwald attack brick floor patio seem hundr feet int jeff apart night medium shot thorwald jeff struggl ext neighborhood night semi close shot doyl pull top wall lisa stella two men look lisa white face frighten int jeff apart night medium shot thorwald smash jeff arm hand jeff grip begin slip ext neighborhood night semi close shot doyl reach top wall look jeff ext neighborhood night medium long shot jeff seen doyl angl hang somehow weather thorwald insan attack ext neighborhood night semi close shot doyl reach servic revolv look call one dete',
 'nd dismount children reach throw arm embrac charlott hurri behind martin lock eye envelop hug children ext fresh water plantat even summer oak tree cover leav martin hous partial rebuilt habit workshop alreadi complet martin children nathan samuel margaret william play tall grass front hous two great dane charlott sit front porch nurs infant martin walk workshop trail susan carri complet rock chair chair work art thin light spider web perfectli turn wood nail glue step onto porch next charlott place rock chair next martin two pound fourteen ounc charlott love smile make minut adjust chair posit sit settl back',
 'breadth bluff gabe pull ancient binocular scan crack gabe pov crack pictur mine shaft design madman crack move upward errat side straight width crack uneven rang six inch six feet look outsid gabe turn binocular insid crack look crack goe way bluff rout gabe tunnel mountain instead go side gabe get side jessi gone far right think better shape gabe simpl ye would done jessi want lead gabe cute ext top bluff day vista point see everyth els mountain rang thing taller tower two mountain lie drop mere four thousand feet qualen said way across h',
 'uild man pajama run rain cabbi lose grip bumper terrenc jerk closer sewer man grab cabbi hand pull resid gather sidewalk polic car siren approach someth give man pajama fall backward puddl small crowd look see terrenc pull free hole moan semi conscious move bodi past bleed stump leg use follow blood swirl eddi rain water flow black storm drain cut fierc bull charg matador red muletta snort blood crowd goe wild bull fight arena blaze spanish sun camera dolli past cheer spaniard find small group american student earli twenti gord man believ paid good money watch guy tight pant kill cow sherri disgust make',
 'ell rita hayworth disgustingli rich well make money make quick start littl war think slick smother sabl like betti grabl disgustingli rich build castl cost passel resid pan presid aspir higher higher get marri buy girl darn pretti head swirl rita hayworth swim highbal stew eyebal well rita hayworth disgustingli rich well rita hayworth chorin nifti soft shoe turn schaefer turn mank schaefer serious truli care ever work mank yeah swell well rita hayworth conclud littl danc break well resum song well ev ry summer sail sea littl yacht normandi pet littl dachshund friend kiss louella big rear end disgustingli rich louella storm eat salmon play ba',
 'memphi goe back garag budgi cackl cut ext rancho palo verd busi district ford escort drive upscal street palo verd three kid insid driver freb littl dim back mirror man black alway wear mirror shade passeng seat kip memphi younger brother car pull stop fanci store close line affluent busi district freb consult piec paper freb corner hawthorn granvia tumbler mess said lotu would corner hawthorn granvia kip mess point corner build exot motor ltd twenti foot high glass window surround showroom exot dream car porsch ferrari lamborghini berton lotu esprit v gleam night showroom light freb mirror man startl freb mirror man shittin',
 'e reel world spin sweat pour pressur build insid skull brain put centrifug neo believ believ cypher go pop vomit violent neo pitch forward black int neo room blink regain conscious room dark neo stretch bed neo go back morpheu sit like shadow chair far corner morpheu could would realli want deep neo know answer morpheu feel owe apolog rule free mind reach certain age danger troubl let go mind turn seen happen broke rule stare dark confess much neo morpheu matrix first built man born insid abil chang want remak mat']
```

```
df[df['genre']==4].index
```

```
Index([    2,     4,     7,    10,    11,    12,    13,    14,    15,    16,
           ...
           22553, 22560, 22561, 22563, 22564, 22567, 22568, 22571, 22574, 22575],
          dtype='int64', length=8873)
```

```
len(corpus)
```

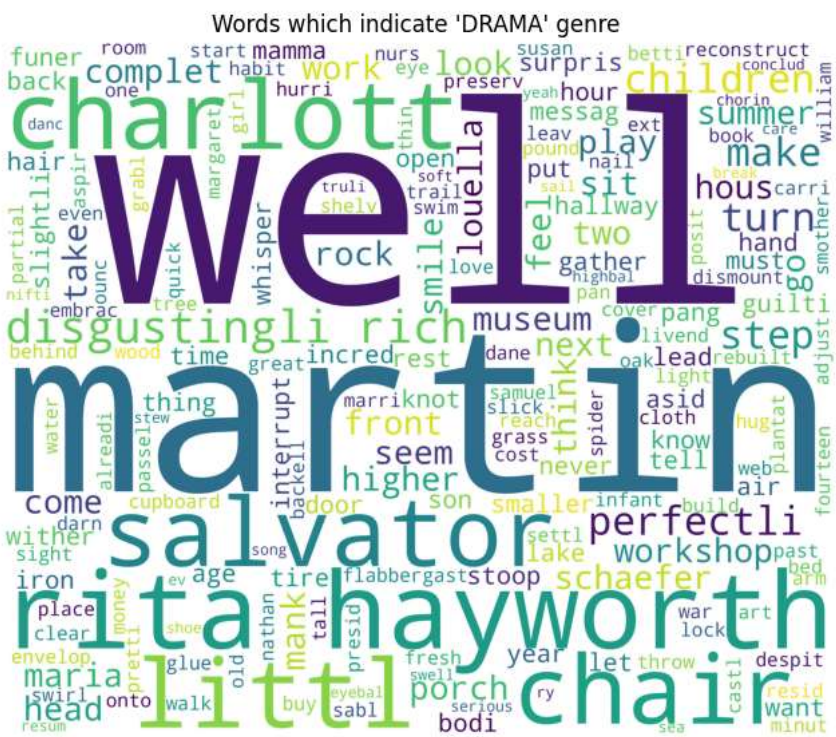
```
22579
```

```
drama_words = []
for i in list(df[df['genre']==4].index):
  drama_words.append(corpus[i])

action_words = []
for i in list(df[df['genre']==1].index):
  action_words.append(corpus[i])
```

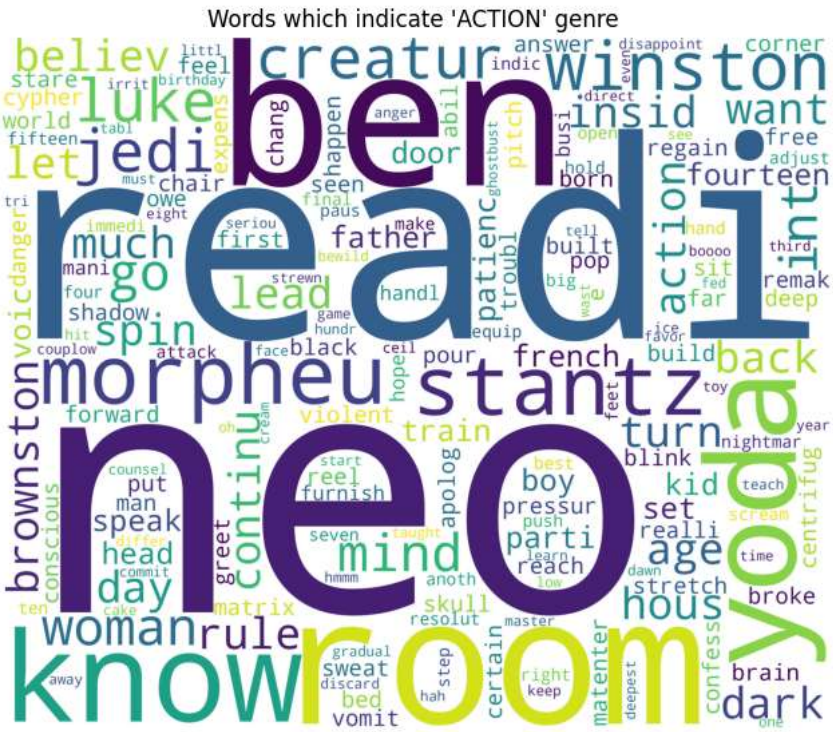
```
comedy_words = []
for i in list(df[df['genre']==3].index):
  comedy_words.append(corpus[i])

drama = ''
action = ''
comedy = ''
for i in range(0, 3):
  drama += drama_words[i]
  action += action_words[i]
  comedy += comedy_words[i]
```

WordCloud

```
# Creating wordcloud for drama genre
from wordcloud import WordCloud
wordcloud1 = WordCloud(background_color='white', width=3000, height=2500).generate(drama)
plt.figure(figsize=(8,8))
plt.imshow(wordcloud1)
plt.axis('off')
plt.title("Words which indicate 'DRAMA' genre ")
plt.show()
```




```
# Creating wordcloud for action genre
wordcloud2 = WordCloud(background_color='white', width=3000, height=2500).generate(action)
plt.figure(figsize=(8,8))
plt.imshow(wordcloud2)
plt.axis('off')
plt.title("Words which indicate 'ACTION' genre ")
plt.show()
```




```
# Creating wordcloud for comedy genre
wordcloud3 = WordCloud(background_color='white', width=3000, height=2500).generate(comedy)
plt.figure(figsize=(8,8))
plt.imshow(wordcloud3)
plt.axis('off')
plt.title("Words which indicate 'COMEDY' genre ")
plt.show()
```

MODEL BUILDING

```
# Creating the Bag of Words model
from sklearn.feature_extraction.text import CountVectorizer
```

```
cv = CountVectorizer(max_features=10000, ngram_range=(1,2))
X = cv.fit_transform(corpus).toarray()
```

```
y = df['genre'].values
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
print('X_train size: {}, X_test size: {}'.format(X_train.shape, X_test.shape))
```

```
X_train size: (18063, 10000), X_test size: (4516, 10000)
```

```
# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import MultinomialNB
nb_classifier = MultinomialNB()
nb_classifier.fit(X_train, y_train)
```

```
MultinomialNB
MultinomialNB()
```

```
# Predicting the Test set results
nb_y_pred = nb_classifier.predict(X_test)
```

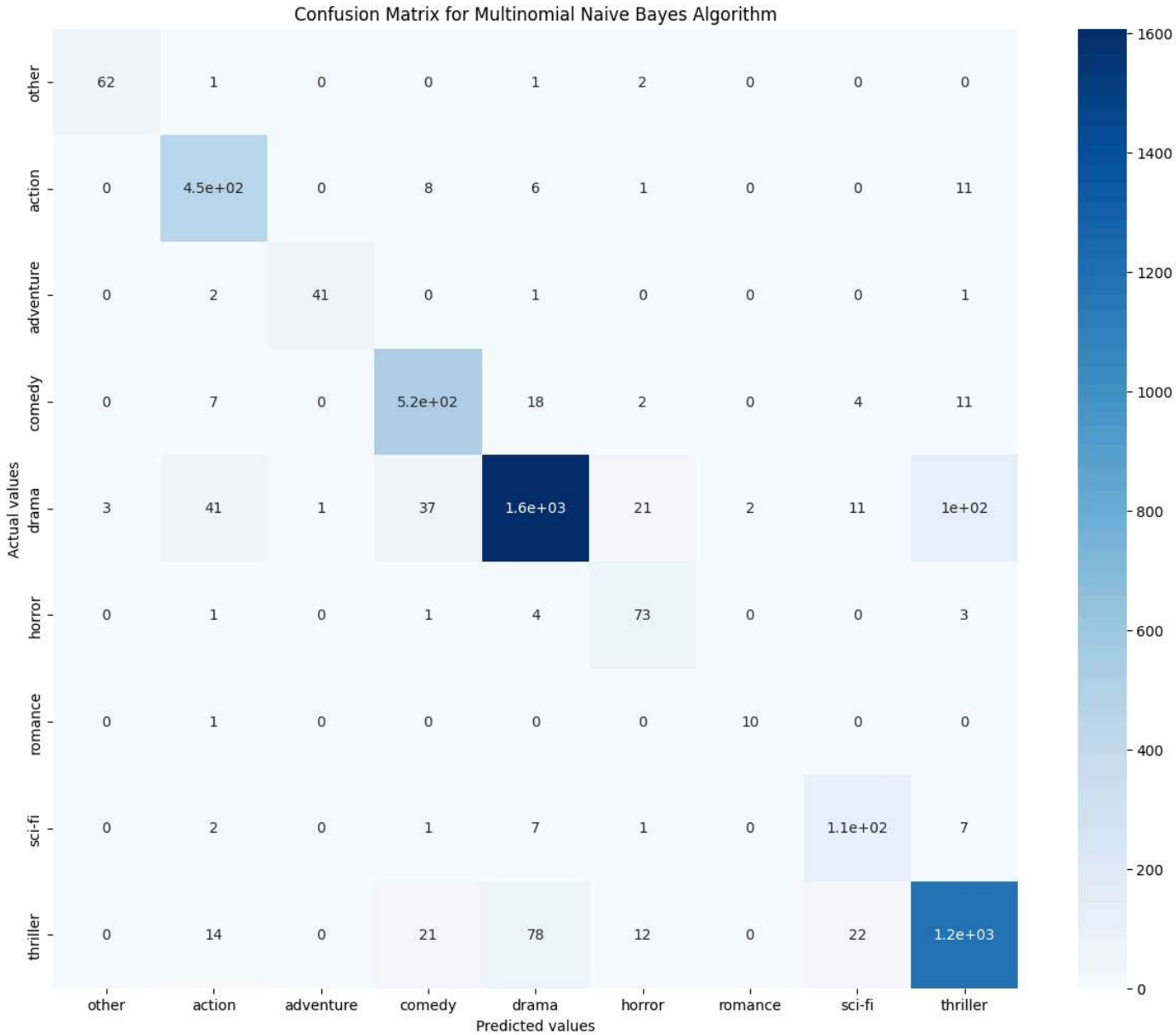
```
# Calculating Accuracy
from sklearn.metrics import accuracy_score
score1 = accuracy_score(y_test, nb_y_pred)
print("---- Score ----")
print("Accuracy score is: {}%".format(round(score1*100,2)))
```

```
---- Score ----
Accuracy score is: 89.57%
```

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
nb_cm = confusion_matrix(y_test, nb_y_pred)
nb_cm
```

```
array([[  62,    1,    0,    0,    1,    2,    0,    0,    0],
       [   0,  450,    0,    8,    6,    1,    0,    0,   11],
       [   0,    2,   41,    0,    1,    0,    0,    0,    1],
       [   0,    7,    0,  517,   18,    2,    0,    4,   11],
       [   3,   41,    1,   37, 1607,   21,    2,   11,  104],
       [   0,    1,    0,    1,    4,   73,    0,    0,    3],
       [   0,    1,    0,    0,    0,    0,   10,    0,    0],
       [   0,    2,    0,    1,    7,    1,    0,  114,    7],
       [   0,   14,    0,   21,   78,   12,    0,   22, 1171]])
```

```
# Plotting the confusion matrix
plt.figure(figsize=(15,12))
axis_labels = ['other', 'action', 'adventure', 'comedy', 'drama', 'horror', 'romance', 'sci-fi', 'thriller']
sns.heatmap(data=nb_cm, annot=True, cmap="Blues", xticklabels=axis_labels, yticklabels=axis_labels)
plt.xlabel('Predicted values')
plt.ylabel('Actual values')
plt.title('Confusion Matrix for Multinomial Naive Bayes Algorithm')
plt.show()
```




```
# Hyperparameter tuning the Naive Bayes Classifier
best_accuracy = 0.0
alpha_val = 0.0
for i in np.arange(0.1,1.1,0.1):
  temp_classifier = MultinomialNB(alpha=i)
  temp_classifier.fit(X_train, y_train)
  temp_y_pred = temp_classifier.predict(X_test)
  score = accuracy_score(y_test, temp_y_pred)
  print("Accuracy score for alpha={} is: {}%".format(round(i,1), round(score*100,2)))
  if score>best_accuracy:
    best_accuracy = score
    alpha_val = i
print('--------------------------------------------')
print('The best accuracy is {}% with alpha value as {}'.format(round(best_accuracy*100, 2), round(alpha_val,1)))
```

```
Accuracy score for alpha=0.1 is: 91.41%
Accuracy score for alpha=0.2 is: 91.14%
Accuracy score for alpha=0.3 is: 90.88%
Accuracy score for alpha=0.4 is: 90.66%
Accuracy score for alpha=0.5 is: 90.39%
Accuracy score for alpha=0.6 is: 90.17%
Accuracy score for alpha=0.7 is: 90.08%
Accuracy score for alpha=0.8 is: 89.99%
```

```
Accuracy score for alpha=0.9 is: 89.79%
Accuracy score for alpha=1.0 is: 89.57%
-----------------------------------------
The best accuracy is 91.41% with alpha value as 0.1
```

```
classifier = MultinomialNB(alpha=0.1)
classifier.fit(X_train, y_train)
```

```
MultinomialNB
MultinomialNB(alpha=0.1)
```

```
def genre_prediction(sample_script):
  sample_script = re.sub(pattern='[^a-zA-Z]',repl=' ', string=sample_script)
  sample_script = sample_script.lower()
  sample_script_words = sample_script.split()
  sample_script_words = [word for word in sample_script_words if not word in set(stopwords.words('english'))]
  ps = PorterStemmer()
  final_script = [ps.stem(word) for word in sample_script_words]
  final_script = ' '.join(final_script)

  temp = cv.transform([final_script]).toarray()
  return classifier.predict(temp)[0]
```

```
# For generating random integer
from random import randint
# Loading test dataset
test = pd.read_csv('/content/kaggle_movie_test.csv')
test.columns
```

```
Index(['id', 'text'], dtype='object')
```

```
test.shape
```

```
(5589, 2)
```

```
test.drop('id', axis=1, inplace=True)
test.head(10)
```

	text
0	glances at her. BOOK Maybe I ought to learn t...
1	hout breaking stride. Tatiana sees her and can...
2	dead bodies. GEORDI Mitchell... DePaul... LANG...
3	take myself. BRANDON How bad is the other thi...
4	her body to shield his own. KAY Freeze it, Bug...
5	im from ear to ear. Ya want me to make a state...
6	BEN We need to help Reed Sue shakes her head,...
7	slowly. At the entrance to the alley stands a ...
8	edge of the field. Neil steps closer. THE TOMB...
9	special, take ya in the kitchen and suck your ...

Next steps: Generate code with test | View recommended plots | New interactive sheet

```
# Predicting values
row = randint(0,test.shape[0]-1)
sample_script = test.text[row]

print('Script: {}'.format(sample_script))
value = genre_prediction(sample_script)
print('Prediction: {}'.format(list(genre_mapper.keys())[value]))
```

```
Script: ib? M.J. Yessir. I have reflected on that, sir. Which explains my gushing deference to you, sir. QUINN is somehow cheered by thi
Prediction: thriller
```

```
# Predicting values
row = randint(0,test.shape[0]-1)
sample_script = test.text[row]

print('Script: {}'.format(sample_script))
value = genre_prediction(sample_script)
print('Prediction: {}'.format(list(genre_mapper.keys())[value]))
```

```
r back, and the whole evening's gone. JOE What's playing on Fordham Road? I think there's a good picture in the Loew's Paradise. GEORGE
```

Start coding or generate with AI.

Task

Create a Gradio interface to predict the genre of a movie script using the trained Multinomial Naive Bayes model and CountVectorizer.

Save the trained model

Subtask:

Save the trained `MultinomialNB` model and the `CountVectorizer` to disk so they can be loaded later for the Gradio interface.

Reasoning: Save the trained Naive Bayes model and the CountVectorizer object to disk using joblib.

```
import joblib

joblib.dump(classifier, 'nb_model.joblib')
joblib.dump(cv, 'vectorizer.joblib')
```

```
['vectorizer.joblib']
```

Load the model and vectorizer

Subtask:

Create a new code cell to load the saved model and vectorizer.

Reasoning: Load the saved model and vectorizer using joblib.

```
loaded_model = joblib.load('nb_model.joblib')
loaded_vectorizer = joblib.load('vectorizer.joblib')
```

Define the prediction function

Subtask:

Create a Python function that takes raw text as input, preprocesses it using the loaded vectorizer, and then uses the loaded model to predict the genre. This function will be used by the Gradio interface.

Reasoning: Define a function to preprocess the input text, vectorize it, and predict the genre using the loaded model and vectorizer.

```
def predict_genre(sample_script):
  """Predicts the genre of a movie script.

  Args:
    sample_script: The raw text of the movie script.

  Returns:
    The predicted genre name.
  """
  # Preprocessing the input script
```

```
  sample_script = re.sub(pattern='[^a-zA-Z]', repl=' ', string=sample_script)
  sample_script = sample_script.lower()
  sample_script_words = sample_script.split()
  sample_script_words = [word for word in sample_script_words if word not in set(stopwords.words('english'))]
  ps = PorterStemmer()
  final_script = [ps.stem(word) for word in sample_script_words]
  final_script = ' '.join(final_script)

  # Vectorizing the preprocessed script
  temp = loaded_vectorizer.transform([final_script]).toarray()

  # Predicting the genre
  prediction = loaded_model.predict(temp)[0]

  # Mapping the predicted value back to genre name
  predicted_genre = None
  for genre_name, genre_value in genre_mapper.items():
    if genre_value == prediction:
      predicted_genre = genre_name
      break

  return predicted_genre
```

Create the gradio interface

Subtask:

Use the `gradio` library to create a simple web interface. This interface will take a text input (for the movie script) and output the predicted genre.

Reasoning: Import the gradio library and define the Gradio interface.

```
import gradio as gr

iface = gr.Interface(
    fn=predict_genre,
    inputs=gr.Textbox(label="Enter Movie Script"),
    outputs=gr.Label(label="Predicted Genre")
)
```

Reasoning: Launch the Gradio interface.

```
iface.launch(debug=True)
```

```
It looks like you are running Gradio on a hosted Jupyter notebook, which requires `share=True`. Automatically setting `share=True` (you

Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=False in launch().
* Running on public URL: https://4b4dd28c8a386f1ec5.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working dir
```