Classification Models

June 26, 2022

1 Final Project - Movie Reviews Analysis

- **Team 13: Jimmy Nguyen, Dallin Munger, Tyler Wolff**

2 Packages

[1]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from spacy.lang.en.stop_words import STOP_WORDS as stopwords
from sklearn.metrics import classification_report
from sklearn.pipeline import Pipeline
from sklearn.dummy import DummyClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings('ignore')
import pickle
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
import seaborn as sns
```

```
/shared-libs/python3.7/py/lib/python3.7/site-packages/tqdm/auto.py:22:
TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

3 Step 1: Data Preparation

- Loading Data set for Modeling
- Un-tokenize previous columns that was converted into tokens for descriptive statistics

[2]:
```
# load cleaned data set from previous notebook
df = pd.read_csv("Cleaned Plot Data.csv")
```

```
df.sample(10)
```

[2]:

```
                                                 title  first_genre  \
201   One Piece: 3D2Y - Overcome Ace's Death! Luffy'…    animation
655                                   One Day of Betty        drama
138   Gabriel &quot;Fluffy&quot; Iglesias: One Show …       comedy
49               The Tall Blond Man with One Black Shoe       comedy
1544  Marilyn Waring on Politics, Local & Global, Sh…  documentary
1216                                 This Man Is the One  documentary
395                                      The Quiet One  documentary
643                                  SWAT: Warhead One       action
972        One Bad Cat: The Reverend Albert Wagner Story  documentary
463                                      Just One Look       comedy

                                          cleaned_plot  imdb_rating
201   ['special', 'takes', 'place', 'two', 'year', '…         7.8
655   ['free', 'entry', 'adventurous', 'journey', 'a…         5.7
138   ['gabriel', 'fluffy', 'iglesias', 'discusses',…         7.2
49    ['hapless', 'orchestra', 'player', 'becomes', …         7.2
1544  ['economist', 'marilyn', 'waring', 'uses', 'ex…         NaN
1216  ['retrospective', 'adam', 'adamant', 'lives', …         6.9
395   ['quiet', 'one', 'offers', 'unique', 'never', …         7.1
643   ['los', 'angeles', 'special', 'police', 'force…         2.5
972   ['one', 'bad', 'cat', 'transformative', 'role'…         8.4
463   ['best', 'friends', 'start', 'kung', 'fu', 'le…         6.8
```

[3]:

```
# untokenize plot descriptions
df['cleaned_plot'] = df['cleaned_plot'].str.replace(r'[^\w\s]', '', regex=True).str.strip()
df['cleaned_plot'].sample(10)
```

[3]:

```
7       mentally unstable photo developer targets uppe…
13      struggling recover emotionally brutal assault …
443     brilliant corporate lawyer luther simmonds acc…
1542    twenty years reunification germany still divid…
700     story anakin skywalker central character star …
448     contemporary story set perth western australia…
1570    alan parker lives enviable life one day januar…
1704    origin story worlds greatest heromemoirist sir…
88      making killing fen returns china rich man seek…
973     motley gang characters includes movie star rep…
Name: cleaned_plot, dtype: object
```

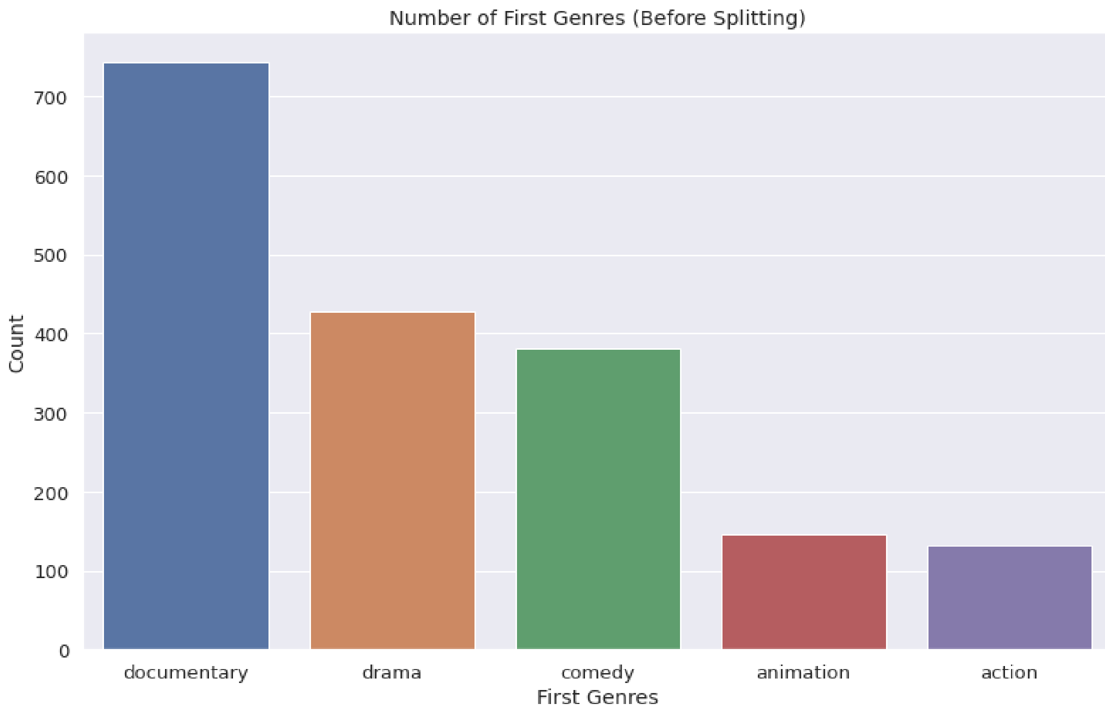
4 Step 2: Train-Test Split

- Checking for Class Imbalance
- Downsampling the Majority class

- Split Data on balanced data set

4.1 Checking for Class Imbalance

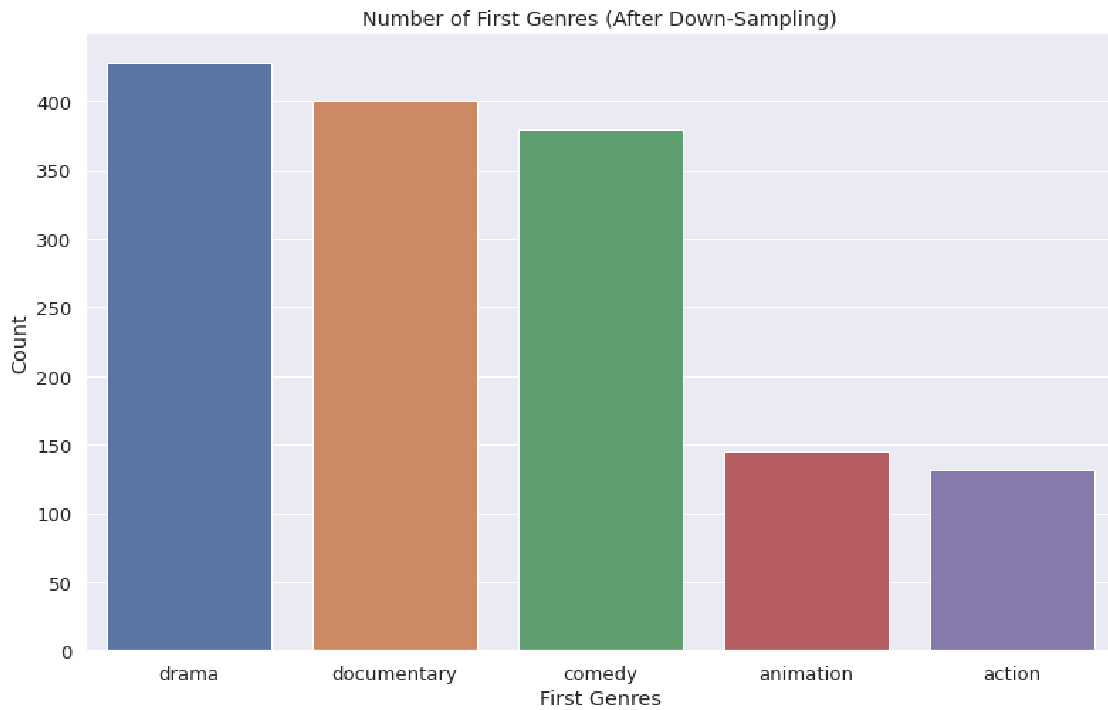
[4]:
```
# plot original target variable classes
plt.figure(figsize=(13,8))
sns.set(font_scale = 1.2)
sns.countplot(x="first_genre", data=df,
order = df['first_genre'].value_counts().index)
plt.title("Number of First Genres (Before Splitting)")
plt.xlabel("First Genres")
plt.ylabel("Count")
plt.show()
```



4.2 Downsampling the Majority class

[5]:
```
# Filter for documentaries and sample 400 rows from it
df_sample = df[df['first_genre'] == 'documentary'].sample(n=400)
# Create a separate DataFrame containing all other genres
df_sampleRest = df[df['first_genre'] != 'documentary']
# Concatenate the two DataFrame to create the new balanced bug reports dataset
df_balanced = pd.concat([df_sampleRest, df_sample])
```

```
# plot new downsampled target variable classes
plt.figure(figsize=(13,8))
sns.set(font_scale = 1.2)
sns.countplot(x="first_genre", data=df_balanced,
order = df_balanced['first_genre'].value_counts().index)
plt.title("Number of First Genres (After Down-Sampling)")
plt.xlabel("First Genres")
plt.ylabel("Count")
plt.show()
```



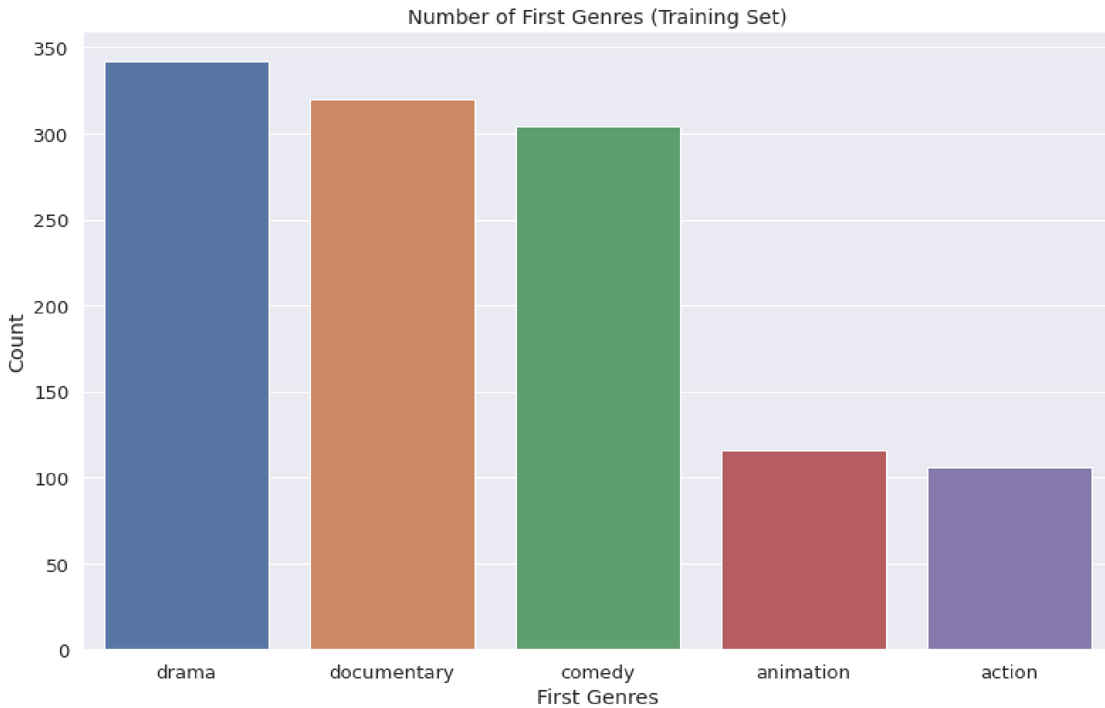
4.3 Split data on balanced data set

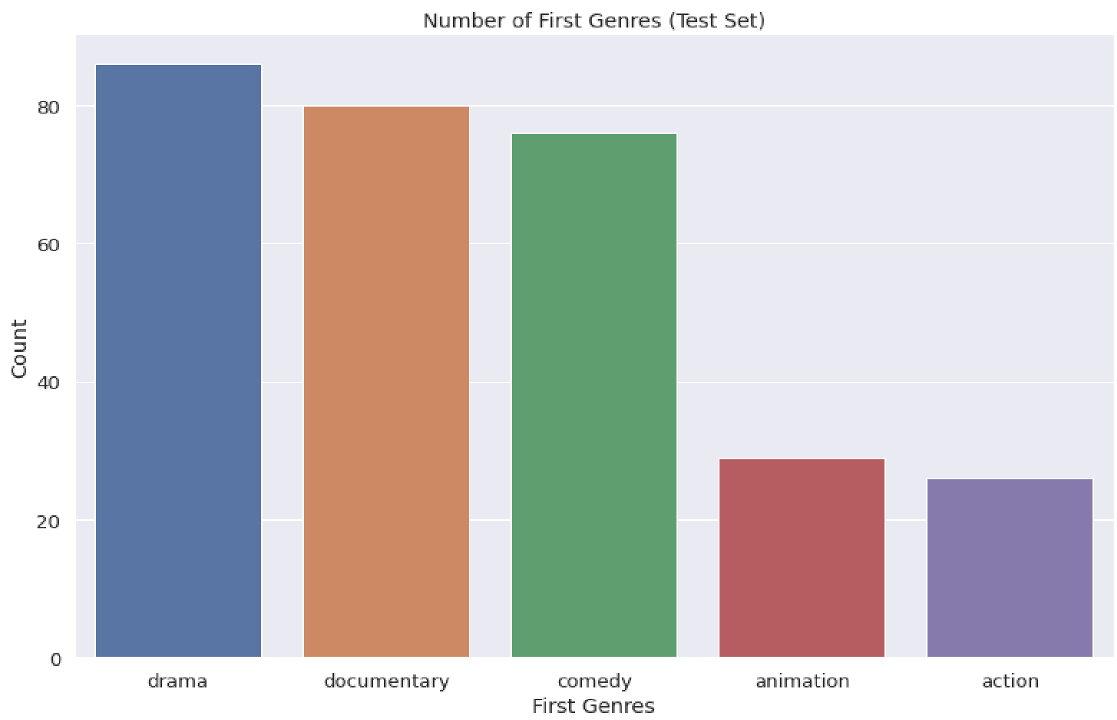
[6]:
```
# Split data set on 80% training and 20% test and keep target variable classes␣
↪balanced
X_train, X_test, Y_train, Y_test =␣
↪train_test_split(df_balanced['cleaned_plot'],\
df_balanced['first_genre'], test_size=0.2, random_state=42,\
stratify=df_balanced['first_genre'])

# Print shapes of training and testing data
print('Size of Training Data ', X_train.shape[0])
print('Size of Test Data ', X_test.shape[0])
```

```
Size of Training Data  1188
```

```
Size of Test Data  297
```

[7]:
```
# plot original target variable classes
plt.figure(figsize=(13,8))
training_y = pd.DataFrame(Y_train)
sns.set(font_scale = 1.2)
sns.countplot(x = "first_genre", data=training_y,
order = training_y['first_genre'].value_counts().index)
plt.title("Number of First Genres (Training Set)")
plt.xlabel("First Genres")
plt.ylabel("Count")
plt.show()

# plot original target variable classes
plt.figure(figsize=(13,8))
test_y = pd.DataFrame(Y_test)
sns.set(font_scale = 1.2)
sns.countplot(x = "first_genre", data=test_y,
order = test_y['first_genre'].value_counts().index)
plt.title("Number of First Genres (Test Set)")
plt.xlabel("First Genres")
plt.ylabel("Count")
plt.show()
```





5 Step 3: Training the Machine Learning Model

- Use TFIDF transformer for words to features
- Model 1: Linear SVC

5.1 Use TFIDF transformer for words to features

[8]:
```
tfidf = TfidfVectorizer(min_df = 10, stop_words=stopwords)
X_train_tf = tfidf.fit_transform(X_train)
```

5.2 Model 1: Linear SVC

[9]:
```
model1 = LinearSVC(random_state=0, tol=1e-5)
model1.fit(X_train_tf, Y_train)
```

[9]:
```
LinearSVC(random_state=0, tol=1e-05)
```

6 Step 4: Model Evaluation

- Model 1 Evaluation
- Baseline model evaluation
- Using Cross-Validation to Estimate Accuracy

6.1 Model 1 Evaluation

```
[10]: # Model 1 evaluation
X_test_tf = tfidf.transform(X_test)
Y_pred = model1.predict(X_test_tf)
print ('Accuracy Score - ', accuracy_score(Y_test, Y_pred))
```

```
Accuracy Score -  0.5353535353535354
```

6.2 Baseline Model Evaluation

```
[11]: clf = DummyClassifier(strategy='most_frequent')
clf.fit(X_train, Y_train)
Y_pred_baseline = clf.predict(X_test)
print ('Accuracy Score - ', accuracy_score(Y_test, Y_pred_baseline))
```

```
Accuracy Score -  0.2895622895622896
```

6.3 Cross-validation to Estimate Accuracy

```
[12]: tfidf = TfidfVectorizer(min_df = 10, stop_words=stopwords)
df_tf = tfidf.fit_transform(df_balanced['cleaned_plot']).toarray()

# Cross Validation with 5 folds
scores = cross_val_score(estimator=model1, X=df_tf,␣
 ↪y=df_balanced['first_genre'], cv=5)
print ("Validation scores from each iteration of the cross validation ", scores)
print ("Mean value across of validation scores ", scores.mean())
print ("Standard deviation of validation scores ", scores.std())
```

```
Validation scores from each iteration of the cross validation  [0.45791246
0.48821549 0.44781145 0.40740741 0.48821549]
Mean value across of validation scores  0.45791245791245794
Standard deviation of validation scores  0.029964438331699653
```

7 Performing Hyperparameter Tuning with Grid Search

- Set up parameters pipeline
- Select best hyperparameters
- Model evaluation after best parameters selected

7.1 Set up hyper-parameters pipeline and train

```
[13]: # Add parameters in pipeline
training_pipeline = Pipeline(
steps=[('tfidf', TfidfVectorizer(stop_words=stopwords)),
('model', LinearSVC(random_state=42, tol=1e-5))])
grid_param = [{
```

```
'tfidf__min_df': [5, 10],
'tfidf__ngram_range': [(1, 3), (1, 6)],
'model__penalty': ['l2'],
'model__loss': ['hinge'],
'model__max_iter': [10000]
}, {
'tfidf__min_df': [5, 10],
'tfidf__ngram_range': [(1, 3), (1, 6)],
'model__C': [1, 10],
'model__tol': [1e-2, 1e-3]
}]

# grid search to find best parameters
gridSearchProcessor = GridSearchCV(estimator=training_pipeline,
param_grid=grid_param,
cv=5)
gridSearchProcessor.fit(df_balanced['cleaned_plot'],
df_balanced['first_genre'])

# best parameters
best_params = gridSearchProcessor.best_params_

# best model
best_model = gridSearchProcessor.best_estimator_
```

7.2 Select best hyper-parameters

[14]:
```
# print out best parameters
print("Best alpha parameter identified by grid search ", best_params)
best_result = gridSearchProcessor.best_score_
print("Best result identified by grid search ", best_result)
```

```
Best alpha parameter identified by grid search  {'model__C': 1, 'model__tol':
0.001, 'tfidf__min_df': 5, 'tfidf__ngram_range': (1, 3)}
Best result identified by grid search  0.503030303030303
```

[15]:
```
# see other parameter results
gridsearch_results = pd.DataFrame(gridSearchProcessor.cv_results_)
gridsearch_results[['rank_test_score', 'mean_test_score',
'params']].sort_values(by=['rank_test_score'])[:5]
```

[15]:
```
   rank_test_score  mean_test_score  \
9                1         0.503030
8                1         0.503030
5                3         0.502357
4                3         0.502357
0                5         0.501010
```

```
                                              params
9  {'model__C': 1, 'model__tol': 0.001, 'tfidf__m…
8  {'model__C': 1, 'model__tol': 0.001, 'tfidf__m…
5  {'model__C': 1, 'model__tol': 0.01, 'tfidf__mi…
4  {'model__C': 1, 'model__tol': 0.01, 'tfidf__mi…
0  {'model__loss': 'hinge', 'model__max_iter': 10…
```

7.3 Model Evaluation After Hyper-parameter Tuning

[16]:

```
# Step 4 - Model Evaluation
Y_pred = best_model.predict(X_test)
print('Accuracy Score - ', accuracy_score(Y_test, Y_pred))
print(classification_report(Y_test, Y_pred))
```

```
Accuracy Score -  0.9494949494949495
              precision    recall  f1-score   support

      action       1.00      0.96      0.98        26
   animation       0.96      0.93      0.95        29
      comedy       0.96      0.89      0.93        76
 documentary       0.96      0.99      0.98        80
       drama       0.91      0.97      0.94        86

    accuracy                           0.95       297
   macro avg       0.96      0.95      0.95       297
weighted avg       0.95      0.95      0.95       297
```

8 Export Model for Deployment

[17]:

```
# save the model to disk
filename = '/work/Movies_Reviews_Analysis/models/final_classification_model.pkl'
pickle.dump(best_model, open(filename, 'wb'))

# some time later...

# load the model from disk
#final_model = pickle.load(open(filename, 'rb'))
```

Created in Deepnote