TMDB Movies

by

Jiaolu Xie

Ana Maria Torres

Sushma Mylavarapu

Introduction

Introducing the TMDB 10000 Movies Dataset, a comprehensive collection that delves into the fascinating world of cinema. This dataset offers a detailed look at 10,000 movies, making it a valuable resource for movie enthusiasts, analysts, and data scientists. From classic gems to contemporary blockbusters, it's a treasure trove of cinematic information.

Let's watch some movies now!

https://public.tableau.com/app/profile/sushma6917/viz/TMDB-Movie-Database/Story?publish=yes

“If you want a happy ending, that depends, of course, on where you stop your story.”

Orson Welles

Machine Learning - Movies Predict Rating

Cleaning Dataset and columns.

```
genre_list = list(set(df["all_genres"].str.cat(sep=", ").split(",")))
genre_list = list(set([genre.replace(" ","" )for genre in genre_list]))
genre_list
```

✓ 0.0s

```
['Romance',
 'Western',
 'Drama',
 'Music',
 'Adventure',
 'Animation',
 'Mystery',
 'Crime',
 'Horror',
 'ScienceFiction',
 'War',
 'Fantasy',
 'Comedy',
 'History',
 'Family',
 'Thriller',
 'TVMovie',
 'Action']
```

```
df["Is Action"] = genre_action
df["Is Adventure"] = genre_adventure
df["Is Horror"] = genre_horror
df["Is Crime"] = genre_crime
df["Is Fantasy"] = genre_fantasy
df["Is TVMovie"] = genre_tvmovie
df["Is Drama"] = genre_drama
df["Is Thriller"] = genre_thriller
df["Is Romance"] = genre_romance
df["Is Mystery"] = genre_mystery
df["Is Western"] = genre_western
df["Is ScienceFiction"] = genre_sciencefiction
df["Is War"] = genre_war
df["Is Family"] = genre_family
df["Is Comedy"] = genre_comedy
df["Is Music"] = genre_music

df.head()
```

✓ 0.0s

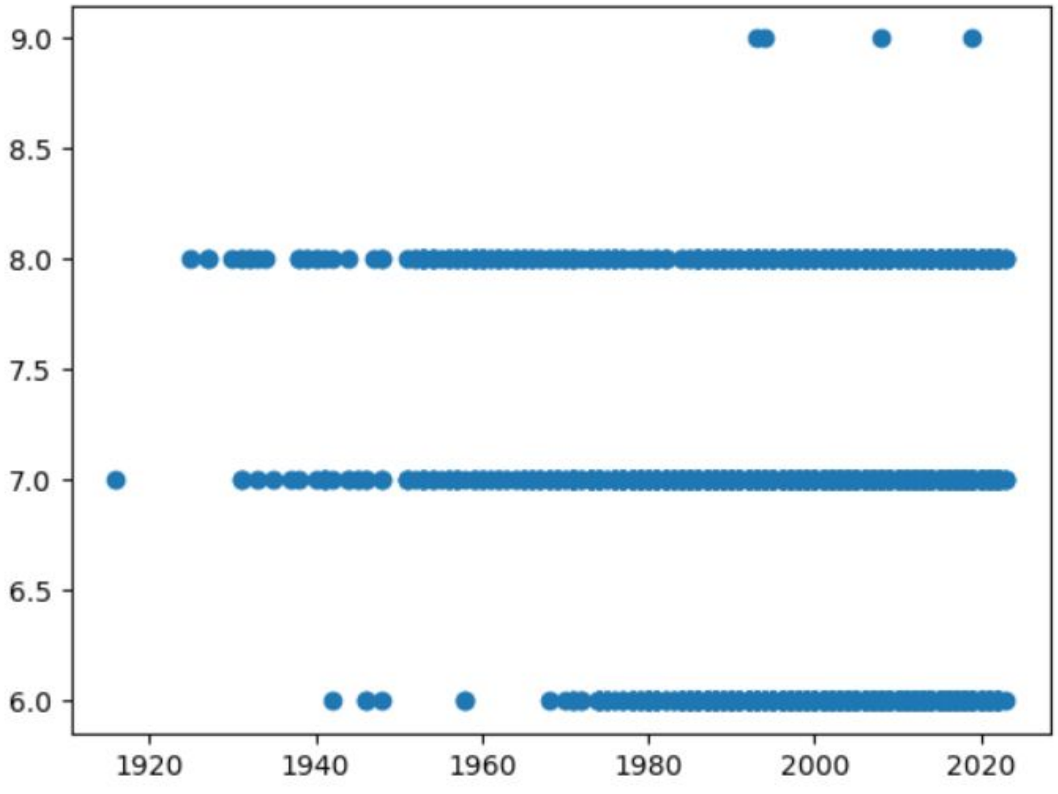
Categories

```python
df = df[["budget","revenue","votes_tmdb", "release_year", "Is Action", "Is Adventure", "Is Horror", "Is Crime", "Is Fantasy", "Is TVMovie",
    "Is Drama", "Is Thriller", "Is Romance", "Is Mystery", "Is Western", "Is ScienceFiction",
    "Is War", "Is Family", "Is Comedy", "Is Music", "rating_tmdb"]]

df.head()
```

✓ 0.0s Python

votes_tmdb	release_year	Is Action	Is Adventure	Is Horror	Is Crime	Is Fantasy	Is TVMovie	...	Is Thriller	Is Romance	Is Mystery	Is Western	Is ScienceFiction	Is War	Is Family	Is Comedy
24685	1994	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0
14618	1993	0	0	0	0	0	0	...	0	0	0	0	0	1	0	0
4447	2023	1	1	0	0	0	0	...	0	0	0	0	0	0	0	0
16448	2019	0	0	0	0	0	0	...	1	0	0	0	0	0	0	1
30654	2008	1	0	0	1	0	0	...	1	0	0	0	0	0	0	0

Relationship between the votes over the years.



Regressors

```
# Linear regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train_scaled, y_train)
model_predictions = model.predict(X_test_scaled)

print(f"R2: {r2_score(y_test,model_predictions)}")
print(f"MSE: {mean_squared_error(y_test, model_predictions)}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, model_predictions))}")
```

✓ 0.5s

```
R2: 0.22100095750261228
MSE: 0.35933535800407146
RMSE: 0.599445875792028
```

```
# Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
model.fit(X_train_scaled, y_train)
model_predictions = model.predict(X_test_scaled)

print(f"R2: {r2_score(y_test,model_predictions)}")
print(f"MSE: {mean_squared_error(y_test, model_predictions)}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, model_predictions))}")
```

✓ 2.1s

```
R2: 0.31877732484394306
```

```
# Extra Trees Regressor
from sklearn.ensemble import ExtraTreesRegressor
model = ExtraTreesRegressor()
model.fit(X_train_scaled, y_train)
model_predictions = model.predict(X_test_scaled)

print(f"R2: {r2_score(y_test,model_predictions)}")
print(f"MSE: {mean_squared_error(y_test, model_predictions)}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, model_predictions))}")
```

✓ 0.8s

```
R2: 0.2844379086489477
MSE: 0.330073268698061
RMSE: 0.5745200333304844
```

```
# Lasso
from sklearn.linear_model import Lasso
model = Lasso()
model.fit(X_train_scaled, y_train)
model_predictions = model.predict(X_test_scaled)

print(f"R2: {r2_score(y_test,model_predictions)}")
print(f"MSE: {mean_squared_error(y_test, model_predictions)}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, model_predictions))}")
```

✓ 0.0s

```
R2: -0.005772919671866994
MSE: 0.46394122770992036
RMSE: 0.6811323129245304
```

```python
# Ridge
from sklearn.linear_model import Ridge
model = Ridge()
model.fit(X_train_scaled, y_train)
model_predictions = model.predict(X_test_scaled)

print(f"R2: {r2_score(y_test,model_predictions)}")
print(f"MSE: {mean_squared_error(y_test, model_predictions)}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, model_predictions))}")
```

✓ 0.0s

```
R2: 0.2210763580104924
MSE: 0.35930057738559223
RMSE: 0.5994168644487676
```

```python
# Ridge
from sklearn.linear_model import SGDRegressor
model = SGDRegressor()
model.fit(X_train_scaled, y_train)
model_predictions = model.predict(X_test_scaled)

print(f"R2: {r2_score(y_test,model_predictions)}")
print(f"MSE: {mean_squared_error(y_test, model_predictions)}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, model_predictions))}")
```

✓ 0.0s

```
R2: 0.2154708342369689
MSE: 0.3618862838910866
RMSE: 0.60156984955289
```

Random Forest Regressor

```
# Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
model.fit(X_train_scaled, y_train)
model_predictions = model.predict(X_test_scaled)

print(f"R2: {r2_score(y_test,model_predictions)}")
print(f"MSE: {mean_squared_error(y_test, model_predictions)}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, model_predictions))}")
```

✓ 2.1s

```
R2: 0.3187732484394306
MSE: 0.31423324099722993
RMSE: 0.5605651086156094
```

Classifiers

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train_scaled, y_train)
model_predictions = model.predict(X_test_scaled)

print(confusion_matrix(y_test, model_predictions))
print(classification_report(y_test, model_predictions))
```

1] ✓ 0.5s

```
[181 113   0   0]
[ 85 228  23   0]
[  4  60  27   0]
[  0   0   1   0]]
              precision    recall  f1-score   support

         6.0       0.67      0.62      0.64       294
         7.0       0.57      0.68      0.62       336
         8.0       0.53      0.30      0.38        91
         9.0       0.00      0.00      0.00         1

    accuracy                           0.60       722
   macro avg       0.44      0.40      0.41       722
eighted avg       0.60      0.60      0.60       722
```

```
# SVC
from sklearn.svm import SVC
model = SVC()
model.fit(X_train_scaled, y_train)
model_predictions = model.predict(X_test_scaled)

print(confusion_matrix(y_test, model_predictions))
print(classification_report(y_test, model_predictions))
```

2] ✓ 0.2s

```
172 122   0   0]
 77 242  17   0]
  7  70  14   0]
  0   0   1   0]]
              precision    recall  f1-score   support

         6.0       0.67      0.59      0.63       294
         7.0       0.56      0.72      0.63       336
         8.0       0.44      0.15      0.23        91
         9.0       0.00      0.00      0.00         1

    accuracy                           0.59       722
   macro avg       0.42      0.36      0.37       722
ighted avg       0.59      0.59      0.58       722
```

```python
# KNN
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
model.fit(X_train_scaled, y_train)
model_predictions = model.predict(X_test_scaled)

print(confusion_matrix(y_test, model_predictions))
print(classification_report(y_test, model_predictions))
```

✓ 0.2s

```
[[190 102   2   0]
 [120 198  18   0]
 [ 20  60  11   0]
 [  0   0   1   0]]
              precision    recall  f1-score   support

         6.0       0.58      0.65      0.61       294
         7.0       0.55      0.59      0.57       336
         8.0       0.34      0.12      0.18        91
         9.0       0.00      0.00      0.00         1

    accuracy                           0.55       722
   macro avg       0.37      0.34      0.34       722
weighted avg       0.53      0.55      0.54       722
```

```python
# Logistic Regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train_scaled, y_train)
model_predictions = model.predict(X_test_scaled)

print(confusion_matrix(y_test, model_predictions))
print(classification_report(y_test, model_predictions))
```

✓ 0.2s

```
[[174 120   0   0]
 [ 83 227  26   0]
 [  5  70  16   0]
 [  0   0   1   0]]
              precision    recall  f1-score   support

         6.0       0.66      0.59      0.63       294
         7.0       0.54      0.68      0.60       336
         8.0       0.37      0.18      0.24        91
         9.0       0.00      0.00      0.00         1

    accuracy                           0.58       722
   macro avg       0.40      0.36      0.37       722
weighted avg       0.57      0.58      0.57       722
```

With an accuracy of 0.60 the Random Forest Classifier is the model with the highest accuracy with respect to the data.

Classifiers

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train_scaled, y_train)
model_predictions = model.predict(X_test_scaled)

print(confusion_matrix(y_test, model_predictions))
print(classification_report(y_test, model_predictions))
```

31] ✓ 0.5s

```
[[181 113   0   0]
 [ 85 228  23   0]
 [  4  60  27   0]
 [  0   0   1   0]]
              precision    recall  f1-score   support

         6.0       0.67      0.62      0.64       294
         7.0       0.57      0.68      0.62       336
         8.0       0.53      0.30      0.38        91
         9.0       0.00      0.00      0.00         1

    accuracy                           0.60       722
   macro avg       0.44      0.40      0.41       722
eighted avg       0.60      0.60      0.60       722
```

Machine Learning - Movies
Predict Genres by Text Description

Predictor: Text Description

1. Combine all descriptions to text

title	overview	tagline	plot
The Shawshank Redemption	Framed in the 1940s for the double murder of h...	Fear can hold you prisoner. Hope can set you f...	Over the course of several years, two convicts...
The Dark Knight	Batman raises the stakes in his war on crime. ...	Welcome to a world without rules.	When the menace known as the Joker wreaks havo...
Pulp Fiction	A burger-loving hit man, his philosophical par...	Just because you are a character doesn't mean ...	The lives of two mob hitmen, a boxer, a gangst...
City of God	In the slums of Rio, two kids' paths diverge a...	If you run, the beast catches you; if you stay...	In the slums of Rio, two kids' paths diverge a...
The Silence of the Lambs	Clarice Starling is a top student at the FBI's...	To enter the mind of a killer she must challen...	A young F.B.I. cadet must receive the help of ...

```
# A feature transformer that filters out stop words from input.
stopwords_remover = StopWordsRemover(inputCol="words", outputCol="filtered")

df = stopwords_remover.transform(df)

df.select(['label','text', 'words', 'filtered']).show(5)
```

```
+-----+--------------------+--------------------+--------------------+
|label|                text|               words|            filtered|
+-----+--------------------+--------------------+--------------------+
|    2|the shawshank red...|[the, shawshank, ...|[shawshank, redem...|
|    2|the dark knight b...|[the, dark, knigh...|[dark, knight, ba...|
|    2|pulp fiction a bu...|[pulp, fiction, a...|[pulp, fiction, b...|
|    2|city of god in th...|[city, of, god, i...|[city, god, slums...|
|    2|the silence of th...|[the, silence, of...|[silence, lambs, ...|
+-----+--------------------+--------------------+--------------------+
```

2. Split text to list, remove "stopwords"

```
# Calculate term frequency in each article
hashing_tf = HashingTF(inputCol="filtered", outputCol="raw_features", numFeatures=10000)
featurized_data = hashing_tf.transform(df)

# TF-IDF vectorization of articles
idf = IDF(inputCol="raw_features", outputCol="features")
idf_vectorizer = idf.fit(featurized_data)
rescaled_data = idf_vectorizer.transform(featurized_data)

rescaled_data.select("label",'Text', 'words', 'filtered', "features").show()
```

TF-IDF :
Term Frequency -
Inverse Document Frequency

```
+-----+--------------------+--------------------+--------------------+--------------------+
|label|                Text|               words|            filtered|            features|
+-----+--------------------+--------------------+--------------------+--------------------+
|    2|the shawshank red...|[the, shawshank, ...|[shawshank, redem...|(10000,[52,585,79...|
|    2|the dark knight b...|[the, dark, knigh...|[dark, knight, ba...|(10000,[495,579,6...|
|    2|pulp fiction a bu...|[pulp, fiction, a...|[pulp, fiction, b...|(10000,[116,160,4...|
|    2|city of god in th...|[city, of, god, i...|[city, god, slums...|(10000,[234,266,4...|
|    2|the silence of th...|[the, silence, of...|[silence, lambs, ...|(10000,[165,237,2...|
```

3. The calculation of how relevant a word in a series or corpus is to a text. - Predictor

4. Separate training and test set, create model by LogisticRegression

```
# Split Train/Test data
(train, test) = rescaled_data.randomSplit([0.8 , 0.2], seed = 202)
print("Training Dataset Count: " + str(train.count()))
print("Test Dataset Count: " + str(test.count()))
```

```
Training Dataset Count: 7954
Test Dataset Count: 2041
```

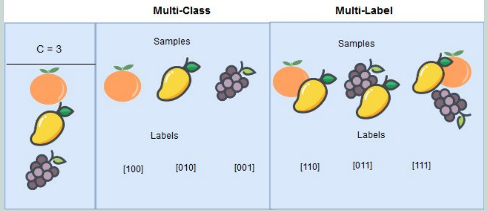
```
lr = LogisticRegression(featuresCol='features',
                        labelCol='label',
                        family="multinomial",
                        regParam=0.3,
                        elasticNetParam=0,
                        maxIter=50)

lrModel = lr.fit(train)
predictions = lrModel.transform(test)
```

Target 1: Multi-Class (Collapse Multi Genres to One)

In 10000 Movies, 4737 are labeled as "Drama".

Collapse Genres:

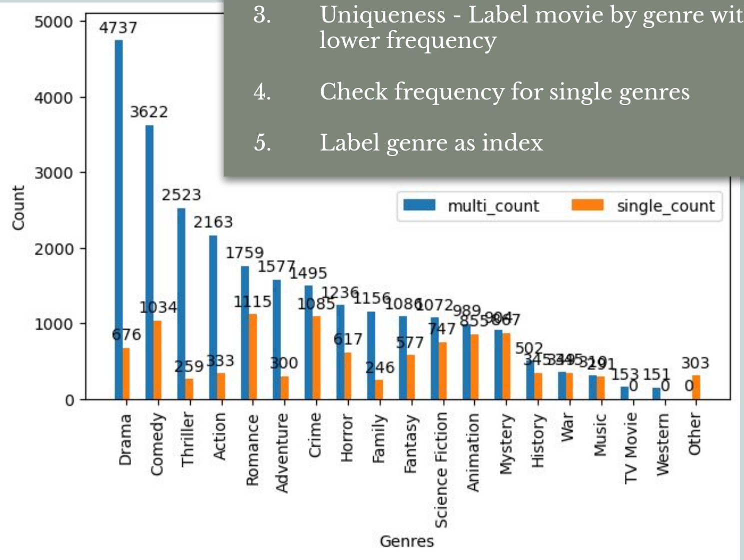
1. Check frequency for over all, multi genres
2. Drop minor genres < 200 to Other
3. Uniqueness - Label movie by genre with lower frequency
4. Check frequency for single genres
5. Label genre as index



imdb_id	all_genres	key_genre	spoken_language	origin	release_year	runtime	title	overview
tt0111161	Drama, Crime	Crime	English	United States of America	1994	142	The Shawshank Redemption	Framed in the 1940s for the double murder of h...
tt0108052	Drama, History, War	War	German	United States of America	1993	195	Schindler's List	The true story of how businessman Oskar Schind...
tt9362722	Animation, Action, Adventure	Animation	English	United States of America	2023	140	Spider-Man: Across the Spider-Verse	After reuniting with Gwen Stacy, Brooklyn's fu...
tt0245429	Animation, Family, Fantasy	Animation	Japanese	Japan	2001	125	Spirited Away	A young girl, Chihiro, becomes trapped in a st...
tt6751668	Comedy, Thriller, Drama	Thriller	English	South Korea	2019	133	Parasite	All unemployed, Ki-taek's family takes peculia...

genres	count	label_genre
Drama	4737	Drama
Comedy	3622	Comedy
Thriller	2523	Thriller
Action	2163	Action
Romance	1759	Romance
Adventure	1577	Adventure
Crime	1495	Crime
Horror	1236	Horror
Family	1156	Family
Fantasy	1086	Fantasy
Science Fiction	1072	Science Fiction
Animation	989	Animation
Mystery	904	Mystery
History	502	History
War	349	War
Music	310	Music
TV Movie	153	Other
Western	151	Other

index	label_genre	count
1	Romance	1115
2	Crime	1085
3	Comedy	1034
4	Mystery	867
5	Animation	855
6	Science Fiction	747
7	Drama	676
8	Horror	617
9	Fantasy	577
10	History	345
11	War	345
12	Action	333
13	Other	303
14	Adventure	300
15	Music	291
16	Thriller	259
17	Family	246



Test-set

Accuracy is :

0.38

```python
# predictions.show()
predictions.select( "imdb_id", "Text", 'probability','prediction', 'label').show()
```

```
+---------+--------------------+--------------------+----------+-----+
|  imdb_id|                Text|         probability|prediction|label|
+---------+--------------------+--------------------+----------+-----+
|tt0016332|seven chances str...|[3.84512027177039...|       1.0|    1|
|tt0022958|grand hotel guest...|[3.41526233711686...|       3.0|    1|
|tt0031725|ninotchka a stern...|[3.07564056937368...|      13.0|    1|
|tt0032599|his girl friday w...|[2.01644521545172...|       7.0|    1|
|tt0033804|the lady eve it s...|[2.66588376462300...|       1.0|    1|
|tt0034583|casablanca in cas...|[3.84054501682187...|       4.0|    1|
|tt0037558|brief encounter r...|[2.97976692726575...|       1.0|    1|
|tt0046345|summer with monik...|[3.36936738959307...|       1.0|    1|
|tt0048356|marty marty  a bu...|[3.00703629480749...|       1.0|    1|
|tt0049866|toto  peppino  an...|[2.74161111286411...|       3.0|    1|
|tt0053604|the apartment bud...|[2.95857641399855...|       1.0|    1|
|tt0055093|lola a bored youn...|[3.56315756535316...|       1.0|    1|
|tt0055471|splendor in the g...|[2.14844118787898...|       1.0|    1|
|tt0057187|irma la douce nes...|[2.74466268360173...|       2.0|    1|
|tt0062873|the young girls o...|[2.11298065911643...|       1.0|    1|
|tt0063518|romeo and juliet ...|[3.13681740517222...|       1.0|    1|
|tt0065651|bed and board par...|[1.40272415799966...|       1.0|    1|
|tt0065772|claire s knee on ...|[2.33275678625280...|       3.0|    1|
|tt0067549|the panic in need...|[2.80064249854042...|       9.0|    1|
|tt0069097|play it again  sa...|[3.15341947804211...|       1.0|    1|
+---------+--------------------+--------------------+----------+-----+
only showing top 20 rows
```

```python
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
accuracy_rate = evaluator.evaluate(predictions)
print("Test-set Accuracy is : ", evaluator.evaluate(predictions))
```

```
Test-set Accuracy is :  0.3792171568161955
```

Compare Machine Learning & Educated Random Guess

1. Generate random number in range 1 to 17 by frequency

2. Repeat guessing for 100 times

index	label_genre	count	probability
1	Romance	1115	0.111556
2	Crime	1085	0.108554
3	Comedy	1034	0.103452
4	Mystery	867	0.086743
5	Animation	855	0.085543
6	Science Fiction	747	0.074737
7	Drama	676	0.067634
8	Horror	617	0.061731
9	Fantasy	577	0.057729
10	History	345	0.034517
11	War	345	0.034517
12	Action	333	0.033317
13	Other	303	0.030315
14	Adventure	300	0.030015
15	Music	291	0.029115
16	Thriller	259	0.025913
17	Family	246	0.024612

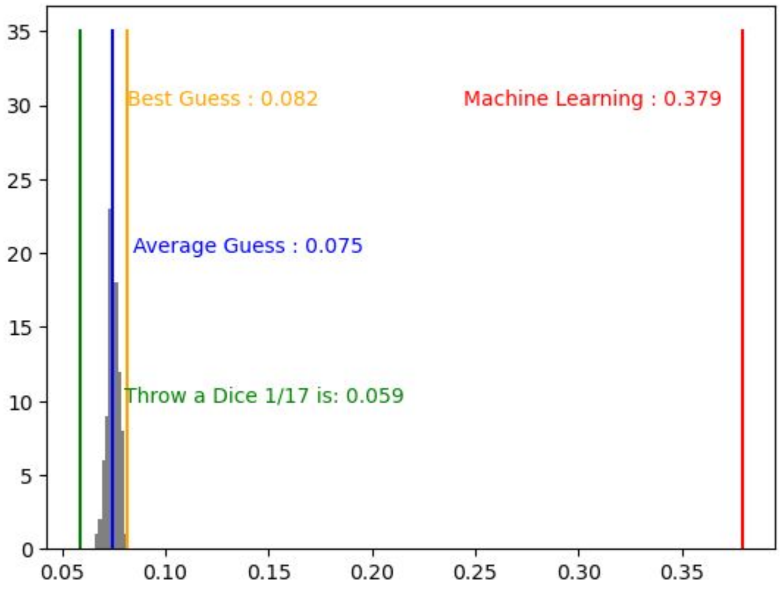
Throw a Even Dice 1/17 is: 0.059

Best Accuracy of Educated Random Guess (100 Repeats) is: 0.082

Average Accuracy of Educated Random Guess (100 Repeats) is: 0.075

Machine Learning Test-set Accuracy is : 0.379

Machine Learning is 4.7 times better than Educated Random Guess



Target 2: Multi-Label

1. Convert genres to binary columns

```
# convert multi labels to target variable
multilabel_binarizer = MultiLabelBinarizer()
multilabel_binarizer.fit(movie_generes)
y = multilabel_binarizer.transform(movie_generes)
```

```
# check labels in selection. avoide error in reading list format
label_ls = multilabel_binarizer.classes_
label_ls = [i for i in label_ls]
label_ls
```

```
['Action',
 'Adventure',
 'Animation',
 'Comedy',
 'Crime',
 'Drama',
 'Family',
 'Fantasy',
 'History',
 'Horror',
 'Music',
 'Mystery',
 'Romance',
 'Science Fiction',
 'TV Movie',
 'Thriller',
 'War',
 'Western']
```

```
# sample target in y
y[0]
```

```
array([0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
# use sk-learn's OneVsRestClassifier class to solve this problem as a Binary Relevance or one-vs-all problem:
lr = LogisticRegression()
clf = OneVsRestClassifier(lr)

# fit model on train data
clf.fit(xtrain_tfidf, ytrain)

# make predictions for validation set
y_pred = clf.predict(xval_tfidf)
```

```
# sample result
multilabel_binarizer.inverse_transform(y_pred)[0]
```

```
('Drama', 'Romance')
```

2. Predict multiple result

3. Select results
   with probability > 0.35
   or
   with max probability

Hence, keep 1 to 3 genres as prediction.

```
# evaluate performance
performance_score = f1_score(yval, y_pred, average="micro")
print("Test-set Performance Score is : ", performance_score)
```

```
Test-set Performance Score is :  0.5003933910306845
```

```
# predict probabilities
y_pred_prob = clf.predict_proba(xval_tfidf)

# threshold value lest then threshold is 0, > is 1
threshold_value = 0.35

# pick the predict result, use threshold value and max probability if threshold value doesn't get any result
y_pred_new = []

for i in range(len( y_pred_prob )) :
    temp_pred = y_pred_prob[i]
    temp_max = max(temp_pred)
    one_pred = ( temp_pred > threshold_value ) | (temp_pred == temp_max ).astype(int)
    y_pred_new.append(one_pred)

accuracy_predict = f1_score(yval, y_pred_new, average="micro")
print(f"Machine Learning Test-set with Threshold {threshold_value} Accuracy is : {accuracy_predict}" )
```

```
Machine Learning Test-set with Threshold 0.35 Accuracy is : 0.6170123611984076
```

Performance Score is : 0.617

Prediction Samples

```
Movie Title:  Being 17
Movie Description:  Damien lives with his mother Marianne, a doctor, while his father,
of duty abroad with the French military. At school, Damien is bullied by Thomas,  who
mmunity up in the mountains. The boys find themselves living together when Marianne in
d stay with them while his mother is ill in hospital. Damien must learn to live with t
im.
Predicted Genre:  [('Drama',)]
Actual Genre:  ['Drama']

Movie Title:  Sudden Impact
Movie Description:  When a young rape victim takes justice into her own hands and beco
t's up to Dirty Harry Callahan, on suspension from the SFPD, to bring her to justice.
Predicted Genre:  [('Action', 'Crime', 'Drama', 'Thriller')]
Actual Genre:  ['Thriller', 'Crime', 'Action']

Movie Title:  Pitch Black
Movie Description:  When their ship crash-lands on a remote planet, the marooned passe
scaped convict Riddick isn't the only thing they have to fear. Deadly creatures lurk i
to attack in the dark, and the planet is rapidly plunging into the utter blackness of
he body count rising, the doomed survivors are forced to turn to Riddick with his eeri
rough the darkness to safety. With time running out, there's only one rule: Stay in th
Predicted Genre:  [('Action', 'Horror', 'Science Fiction', 'Thriller')]
Actual Genre:  ['Thriller', 'Science Fiction', 'Action']

Movie Title:  Saints and Soldiers
Movie Description:  Five American soldiers fighting in Europe during World War II stru
d territory after being separated from U.S. forces during the historic Malmedy Massacr
Predicted Genre:  [('Action', 'Drama', 'War')]
Actual Genre:  ['War', 'Drama', 'Action', 'Adventure', 'History']

Movie Title:  Explorers
Movie Description:  The visionary dreams of three curious and adventuresome young boys
lity in Explorers, the action-fantasy from director Joe Dante, who combines keen humor,
th unexpected twists. In their makeshift laboratory, the boys use an amazing discovery and their ingenuity to
build their own spaceship and launch themselves on a fantastic interplanetary journey.
Predicted Genre:  [('Adventure', 'Comedy', 'Drama', 'Family')]
Actual Genre:  ['Family', 'Science Fiction', 'Fantasy']
```

```
Movie Title:  Explorers
Movie Description:  The visionary dreams of three curious and adventuresome young boys become an exciting rea
lity in Explorers, the action-fantasy from director Joe Dante, who combines keen humor, warmth and fantasy wi
th unexpected twists. In their makeshift laboratory, the boys use an amazing discovery and their ingenuity to
build their own spaceship and launch themselves on a fantastic interplanetary journey.
Predicted Genre:  [('Adventure', 'Comedy', 'Drama', 'Family')]
Actual Genre:  ['Family', 'Science Fiction', 'Fantasy']

Movie Title:  Cleaner
Movie Description:  Single father and former cop Tom Cutler has an unusual occupation: he cleans up death sce
nes. But when he's called in to sterilize a wealthy suburban residence after a brutal shooting, Cutler is sho
cked to learn he may have unknowingly erased crucial evidence, entangling himself in a dirty criminal cover-u
p.
Predicted Genre:  [('Action', 'Crime', 'Drama', 'Thriller')]
Actual Genre:  ['Crime', 'Thriller', 'Mystery']

Movie Title:  Ariel
Movie Description:  After the coal mine he works at closes and his father commits suicide, a Finnish man leav
es for the city to make a living but there, he is framed and imprisoned for various crimes.
Predicted Genre:  [('Drama',)]
Actual Genre:  ['Drama', 'Comedy', 'Romance']

Movie Title:  Macbeth
Movie Description:  Scotland, 11th century. Driven by the twisted prophecy of three witches and the ruthless
ambition of his wife, warlord Macbeth, bold and brave, but also weak and hesitant, betrays his good king and
his brothers in arms and sinks into the bloody mud of a path with no return, sown with crime and suspicion.
Predicted Genre:  [('Drama',)]
Actual Genre:  ['War', 'Drama']

Movie Title:  '71
Movie Description:  A young British soldier must find his way back to safety after his unit accidentally aban
dons him during a riot in the streets of Belfast.
Predicted Genre:  [('Drama',)]
Actual Genre:  ['Thriller', 'Action', 'Drama', 'War']

Movie Title:  Cold Prey
Movie Description:  When one of them breaks a leg, 5 friends snowboarding in the Norwegian mountains take she
lter in an abandoned ski lodge and soon realize they're not alone.
Predicted Genre:  [('Horror', 'Thriller')]
Actual Genre:  ['Horror', 'Mystery', 'Thriller']
```

Implement to book? - Winnie-the-Pooh

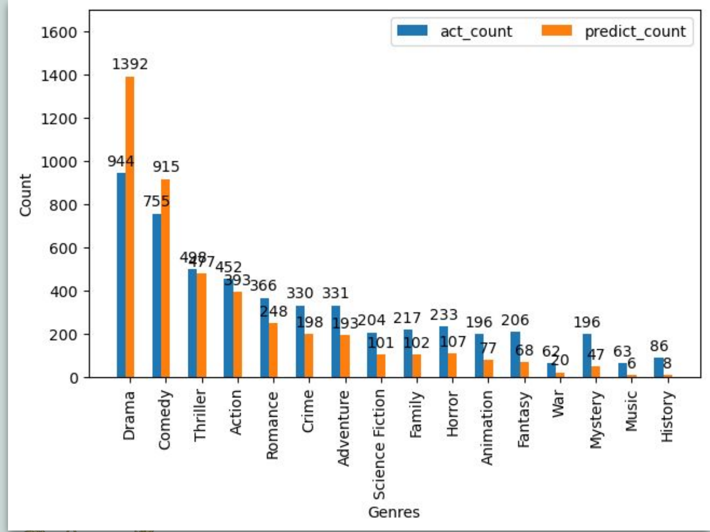
```
print('Book Description: ' , text ,"\nPredicted Genre: ", infer_tags(text) )
```

```
Book Description:

    But his arms were so stiff ... they stayed up straight in the air for more than a week, and whenever a fly c
ame and settled on his nose he had to blow it off. And I think - but I am not sure - that that is why he is alwa
ys called Pooh.


Predicted Genre:  [('Comedy',)]
```

```
Book Description:

    A. A. Milne named the character Winnie-the-Pooh after a teddy bear owned by his son, Christopher Robin Miln
e, on whom the character Christopher Robin was based. Shepard in turn based his illustrations of Pooh on his own
son's teddy bear named Growler, instead of Christopher Robin's bear.[4] The rest of Christopher Milne's toys - P
iglet, Eeyore, Kanga, Roo, and Tigger - were incorporated into Milne's stories.[5][6] Two more characters, Owl a
nd Rabbit, were created by Milne's imagination, while Gopher was added to the Disney version. Christopher Robi
n's toy bear is on display at the Main Branch of the New York Public Library in New York City.

Predicted Genre:  [('Animation', 'Comedy', 'Family')]
```

```
Book Description:

    Winnie-the-Pooh (also known as Edward Bear, Pooh Bear or simply Pooh) is a fictional anthropomorphic teddy b
ear created by English author A. A. Milne and English illustrator E. H. Shepard. Winnie-the-Pooh first appeared
by name in a children's story commissioned by London's Evening News for Christmas Eve 1925.

Predicted Genre:  [('Drama', 'Family')]
```

Compare Frequency in Actual Genres & Predict Genres



In Testing-set (20% of 10000 Movies):

1. Actual genres match distribution of population.

2. Prediction shows more skewed distribution.

Distribution in Target also impact prediction, not only features in Predictor which prefer to be captured.

5000 Movies to 10000 Movies. Accuracy Improved by 0.02.

> "Garbage in, garbage out!
> - George Fuechsel, 1957"

If we put bad information into our computer models, we will get bad information out of them. Expect controversy, bad insights, poor decisions, and bad policy to follow.

Code Reference:

https://www.kaggle.com/code/ashokkumarpalivela/multiclass-classification-using-pyspark

https://www.analyticsvidhya.com/blog/2019/04/predicting-movie-genres-nlp-multi-label-classification/

Thank you

TMDB Movies

Team 6

Jiaolu Xie
Ana Maria Torres
Sushma Mylavarapu