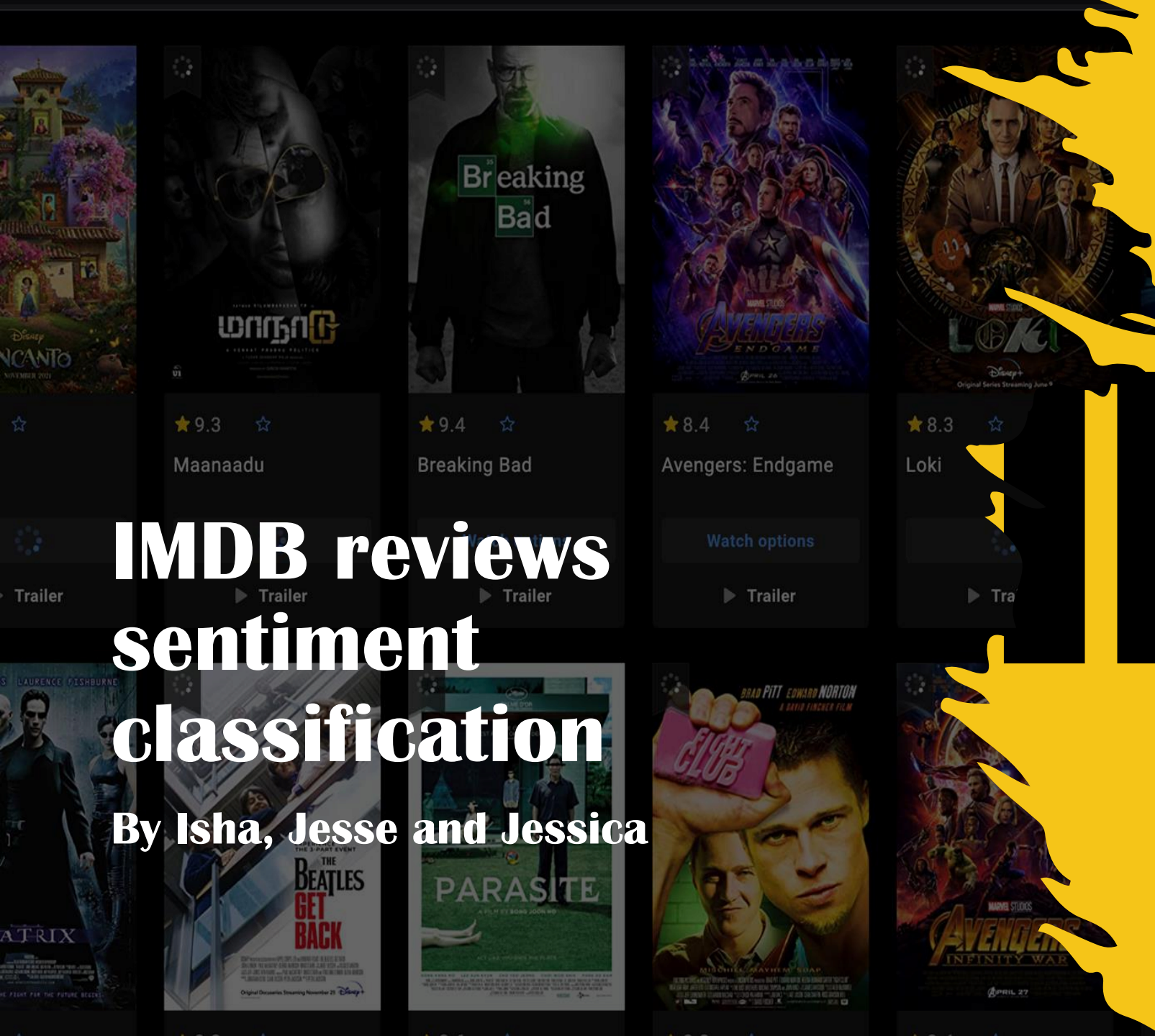
IMDB reviews sentiment classification

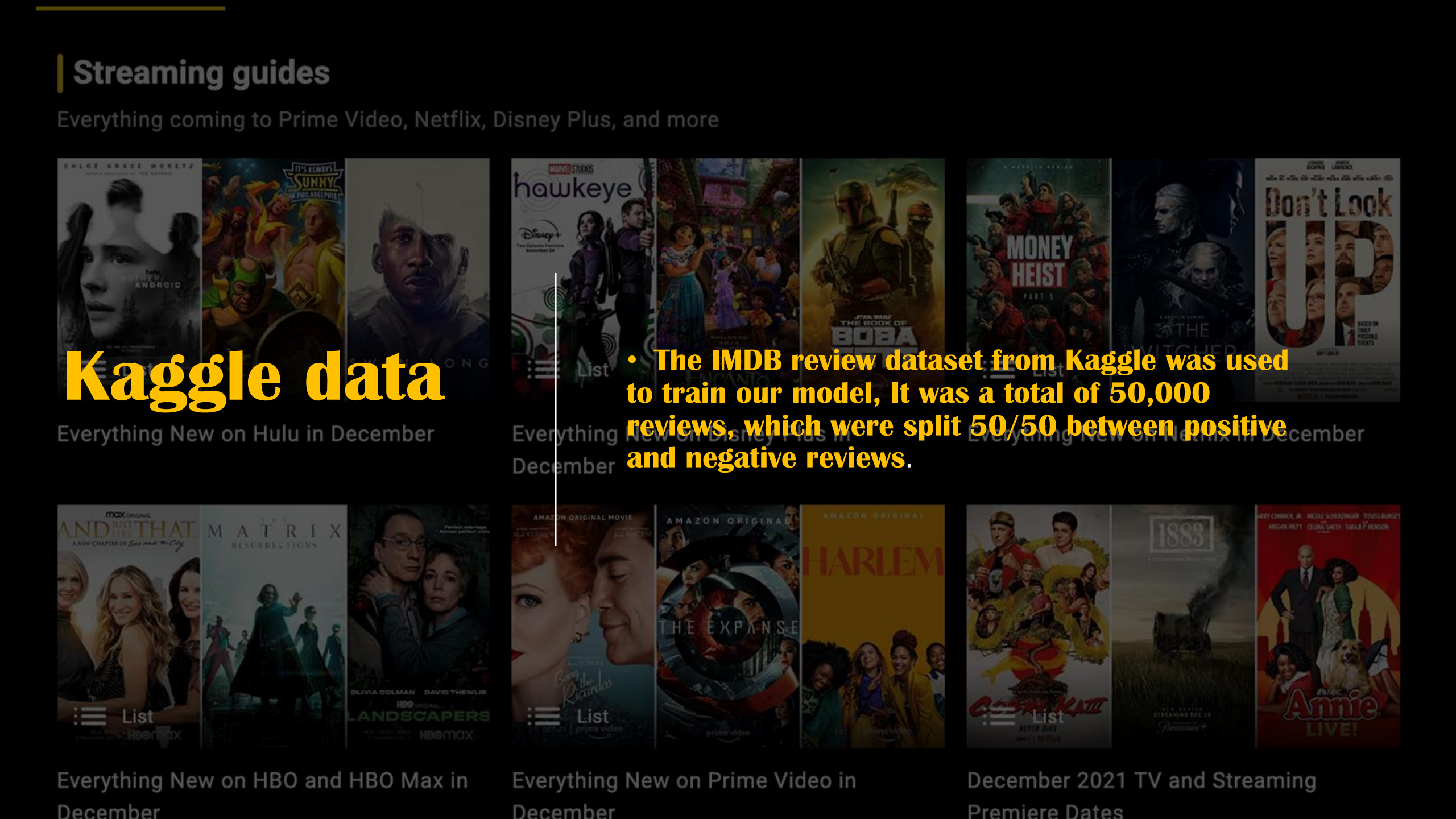
By Isha, Jesse and Jessica



IMDb

Kaggle data

- The IMDB review dataset from Kaggle was used to train our model, It was a total of 50,000 reviews, which were split 50/50 between positive and negative reviews.



IMDB Review Scraping

- We used Splinter and Beautiful Soup to scrape reviews for the latest releases and append to our SQL database

- We extracted the title of the movie and url from the scraped html.

```
# Dependencies
from bs4 import BeautifulSoup
import requests
import pandas as pd
from splinter import Browser
import requests
from webdriver_manager.firefox import GeckoDriverManager
from webdriver_manager.chrome import ChromeDriverManager
```

```
# Chrome Driver
browser = Browser('chrome', executable_path=ChromeDriverManager().install(), headless=Fa
```

```
====== WebDriver manager ======
Current google-chrome version is 96.0.4664
Get LATEST driver version for 96.0.4664
Get LATEST driver version for 96.0.4664
Trying to download new driver from https://chromedriver.storage.googleapis.com/
Driver has been saved in cache [C:\Users\Jtc\.wdm\drivers\chromedriver\win32\96
```

```
# URL of page to be scraped
url = 'https://www.imdb.com/list/ls016522954/?ref_=nv_tvv_dvd'


# Retrieve page with the requests module
response = requests.get(url)
```

```
# Create BeautifulSoup object; parse with 'html.parser'
soup = BeautifulSoup(response.text, 'html.parser')

# Get all div's with the class "caption"
results = soup.find_all('h3',class_="lister-item-header")
```

```
# Loop through returned results
for result in results:
    # Error handling
    try:
        # Identify and return title of listing
        title = result.a.text
        # Identify and return link to listing
        link = result.a['href']

        # Print results only if title, price, and link are
        if (title and link):
            print('-------------')
            print(title)
            print(link)
    except AttributeError as e:
        print(e)

        #?ref_=ttls_li_tt
```

```
-------------
The Survivalist
/title/tt13694706/
-------------
The Addams Family 2
/title/tt11125620/
-------------
Witch Hunt
/title/tt10160974/
-------------
American Night
/title/tt5344054/
-------------
Space Jam: A New Legacy
/title/tt3554046/
-------------
Escape Room 2
/title/tt9844522/
-------------
Six Minutes to Midnight
```

Web Scraping New and Upcoming Releases

- Two URL variables are defined to bookend each side of the scraped URL for each specific release, this will give us the full URL for the required page.
- The FOR LOOP constructs the URL and navigates into that page to scrape the review.
- Film title, URL, and review are added to a dictionary, and then appended to the film reviews list.
- Film reviews list is transformed to Pandas DataFrame and then exported to CSV.

```
]: url2 = 'https://www.imdb.com'
   url3 = "reviews?ref_=ttls_li_tt"

   film_revie[]

   result in results:
   try:
       title = result.a.text
       print(title)
       link = result.a["href"]
       print(url2+link+url3)
       browser.visit(url2 + link + url3)

       html = browser.html

       soup = BeautifulSoup(html, 'html.parser')

       content = soup.find('div', class_="lister-list")

       review = content.find('div', class_="text")
       print(review.text)

       review_dict = dict()
       review_dict['title'] = title
       review_dict['url'] = url2 + link + url3
       iew_dict['review'] = review.text
       lm_reviews.append(review_dict)

   pt AttributeError as e:
```

```
3]: browser.quit()
```

```
[34]: film_reviews = pd.DataFrame (film_reviews, columns = ['title', 'url','review'])
      print (film_reviews)
```

```
                                                title  \
0                                        The Survivalist
1                                      The Addams Family 2
2                                                Witch Hunt
3                                         American Night
4                                  Space Jam: A New Legacy
                                            Escape Room 2
6                                  Six Minutes to Midnight
                                                 The Stand
                                                   Clarice
9                                        Broken Diamonds
10                                            The Nevers
11                                           Fried Barry
12  Cleanin' Up the Town: Remembering Ghostbusters
13                                             Detention
14                 Aileen Wuornos: American Boogeywoman
15                                      Vengeance Is Mine
16                                               Free Guy
17                                        The Green Knight
```

```
In [35]: film_reviews.head()
```

Out[35]:

	title	url	review
0	The Survivalist	https://www.imdb.com/title/tt13694706/reviews?...	It's day 592 of Covid-19 Delta and the world h...
1	The Addams Family 2	https://www.imdb.com/title/tt11125620/reviews?...	Wednesday uses Uncle Fester in her experiment ...
2	Witch Hunt	https://www.imdb.com/title/tt10160974/reviews?...	Martha (Elizabeth Mitchell) lives in southern ...
3	American Night	https://www.imdb.com/title/tt5344054/reviews?r...	This film start with a man having his package ...
4	Space Jam: A New Legacy	https://www.imdb.com/title/tt3554046/reviews?r...	LeBron James worked hard to be the greatest an...

```
In [37]: film_reviews.to_csv(r'new_upcoming_dvd_reviews.csv')
```

movie_review/postgres@PostgreSQL 13

Query Editor Query History

```sql
CREATE TABLE IMDB_REVIEWS (
    Title varchar(100),
   URL varchar(255),
    Review varchar(255)

);

SELECT * FROM public."IMDB_REVIEWS"
```

PostgreSQL Database

Data Output Explain Messages Notifications

	index bigint	title text	url text	review text
1	0	The Survivalist	https://www.imdb.com/title/tt13694706/reviews?ref_=ttls_li_tt	It's day 592 of Covid-19 Delta
2	1	The Addams Family 2	https://www.imdb.com/title/tt11125620/reviews?ref_=ttls_li_tt	Wednesday uses Uncle Fester in her ex
3	2	Witch Hunt	https://www.imdb.com/title/tt10160974/reviews?ref_=ttls_li_tt	Martha (Elizabeth Mitchell) lives in southe
4	3	American Night	https://www.imdb.com/title/tt5344054/reviews?ref_=ttls_li_tt	This film start with a man having his pack
5	4	Space Jam: A New Legacy	https://www.imdb.com/title/tt3554046/reviews?ref_=ttls_li_tt	LeBron James worked hard to be the gr
6	5	Escape Room 2	https://www.imdb.com/title/tt9844522/reviews?ref_=ttls_li_tt	Zoey Davis (Taylor Russell) is on a mission to ta
7	6	Six Minutes to Midnight	https://www.imdb.com/title/tt5114840/reviews?ref_=ttls_li_tt	In 1939, Thomas Miller teaches English at the Augusta

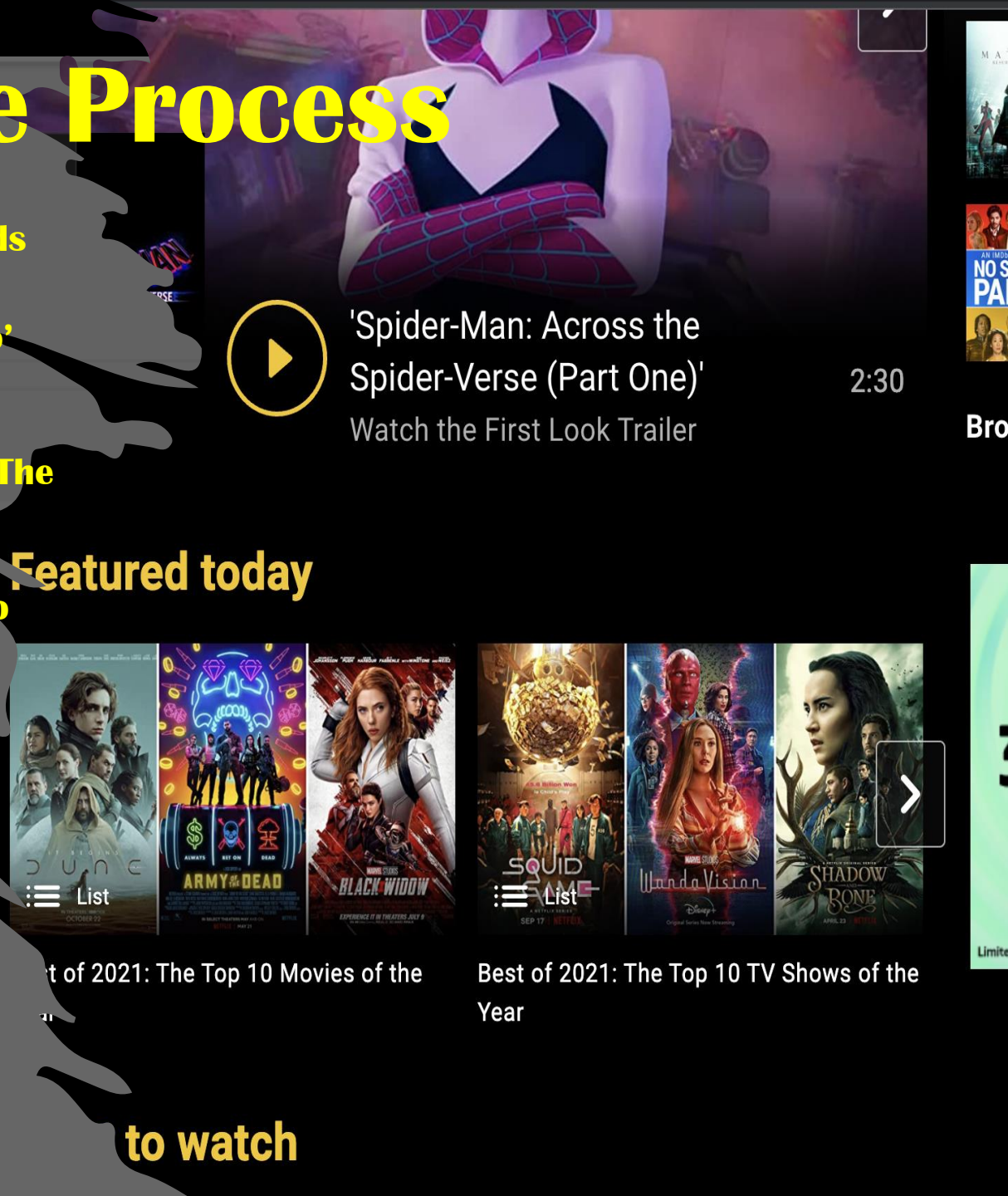
PostgreS

NLP Pipeline Process

```
# Run the hashing term frequency
hashing = HashingTF(inputCol="Wordsfiltered", outputCol="hashedValues")

# Transform into a DF
hashed_df = hashing.transform(removed_frame)
hashed_df.show()

row = hashed_df.count()
col = len(hashed_df.columns)

print(f'Dimension of the Dataframe is: {(row,col)}')
print(f'Number of Rows are: {row}')
print(f'Number of Columns are: {col}')
```

- Tokenizer –sentences were broken up into individual words
- StopWordsRemover – English stop words like ‘a’, ‘and’ ‘so’ ‘what’ were removed
- HashingTF –individual terms were mapped into an index. The same words get assigned the same index.
- Disadvantages of HashingTF is that in larger datasets, two different words may be assigned to the same index.
- This could affect accuracy of our models if a positive and negative word were assigned to the same index.
- Alternatively, we could use CountVectoriser but that also comes with disadvantages like, Its inability in identifying more important and less important words for analysis.

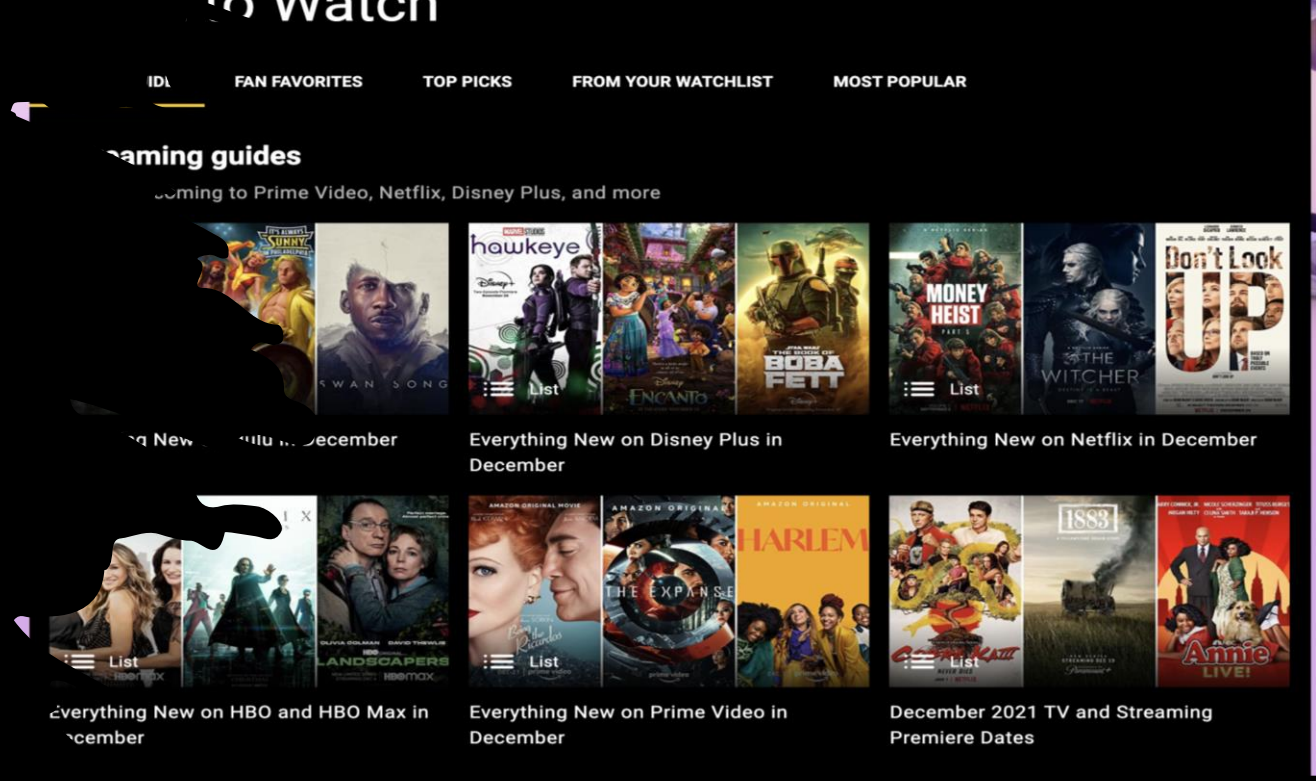
review	sentiment	words	Wordsfiltered	hashedValues
One of th...	[illegible]	[illegible]	[illegible]	[illegible]
A wonderf...	[illegible]	[illegible]	[illegible]	(262144,[521,...
I thought this wa...	positive	[i, thought, this...	[thought, wonderf...	(262144,[1043,139...
Basically there's...	negative	[basically, there...	[basically, famil...	(262144,[6512,853...
Petter Mattei's "...	positive	[petter, mattei's...	[petter, mattei's...	(262144,[2751,392...
Probably my all-t...	positive	[probably, my, al...	[probably, all-ti...	(262144,[5381,158...
I sure would like...	positive	[i, sure, would, ...	[sure, like, see,...	(262144,[1889,543...
This show was an ...	negative	[this, show, was,...	[show, amazing,, ...	(262144,[2437,8...
Encouraged by the...	negative	[encouraged, by, ...	[encouraged, posi...	(262144,[8538,149...
If you like original gut wrenching laugh...	positive	[if, you, like, o...	[like, original, ...	(262144,[5549,2...
Phil the Alien is...	negative	[phil, the, alien...	[phil, alien, one...	(262144,[1335,12...
I saw this movie ...	negative	[i, saw, this, mo...	[saw, movie, 12,...	(262144,[5451,102...
So im not a big f...	negative	[so, im, not, a, ...	[im, big, fan, bu...	(262144,[5[illegible],127...
The cast played S...	negative	[the, cast, playe...	[cast, played, sh...	(262144,[5670,761...
This a fantastic ...	positive	[this, a, fantast...	[fantastic, movie...	(262144,[6558,215...
Kind of drawn in ...	negative	[kind, of, drawn,...	[kind, drawn, ero...	(262144,[2101,243...
Some films just s...	positive	[some, films, jus...	[films, simply, r...	(262144,[2701,161...
This movie made i...	negative	[this, movie, mad...	[movie, made, one...	(262144,[329,3535...
I remember this f...	positive	[i, remember, thi...	[remember, film,i...	(262144,[1009,378...
An awful film! It...	negative	[an, awful, film!...	[awful, film!, mu...	(262144,[3924,156...



Train Test Split

- Data was split randomly into training and testing as a 70/30 split

- More data was needed in the training data set to give a better accuracy calculated on the test set



Copy of IMDB_data_cleaning_.ipynb

File Edit View Insert Runtime Tools Help Last saved at 17:49

```
# Train test split
training, testing = df.randomSplit([0.7, 0.3],1)
```

```
print("Training Dataset Count: " + str(df.count()))
print("Training Dataset Count: " + str(training.count()))
print("Test Dataset Count: " + str(testing.count()))
```

```
Training Dataset Count: 50000
Training Dataset Count: 35094
Test Dataset Count: 14906
```

```
# Show training data
training.show()
```

```
+--------------------+---------+
|              review|sentiment|
+--------------------+---------+
|                Turkish...| positive|
|!!!! SPOILER...| negative|
|! MILD SPOILER...| negative|
|POSSIBLE MIL...| negative|
|sporadica...| negative|
|beat is ...| positive|
|ace Od...| positive|
|...| positive|
|...| positive|
```

Logistics regression model

- For the first model, we used logistic regression, where our output column is 0 or 1
- In our case 0 was a positive review and 1 was a negative review

- Logistic Regression F1 Score: 0.860
- Logistic Regression Accuracy: 0.860

```python
# LOGISTIC REGRESSION MODEL

# Create all the steps for the pipeline
label_indexer = StringIndexer(inputCol='sentiment',outputCol='label')
# code to change positive sentiment to 1 values - stringOrderType="frequencyAsc"
tokenizer = Tokenizer(inputCol="review", outputCol="Wordsfiltered")
stopremove = StopWordsRemover(inputCol='Wordsfiltered',outputCol='hashedValues')
hashingTF = HashingTF(inputCol="hashedValues", outputCol='features')
lr = LogisticRegression(maxIter=20, regParam=0.001)

# Define pipeline
pipeline = Pipeline(stages=[label_indexer, tokenizer, stopremove, hashingTF, lr])

# Fit the pipeline to training reviews.
lrmodel = pipeline.fit(training)

# Tranform the model with the testing data
predictions_lr = lrmodel.transform(testing)

predictions_lr.filter(predictions_lr['label'] == 0) \
    .select("review","Wordsfiltered","features","probability","label","prediction") \
    .orderBy("probability", ascending=False) \
    .show(n = 10, truncate = 30)

# Evaluate Logistic Regression model
f1_eval = MulticlassClassificationEvaluator(metricName='f1',predictionCol="prediction")
print("Logistic Regression F1 Score: ", f1_eval.evaluate(predictions_lr))
accuracy_score = MulticlassClassificationEvaluator(metricName='accuracy',predictionCol="prediction")
print("Logistic Regression Accuracy: ", accuracy_score.evaluate(predictions_lr))
```

Star Wars: The Rise of Skywalker (2019)

★ 1/10

Satanic
18 March 2020

It is this type of film that is promoted by the agents of devils (demons) on earth. Well, I got a text (it is words sent to your phone by the phone company called Target) that this is free and we atched it. Of course it is free. it is pushed by the Devil and his minions. Be warned it has no recognition of Jesus or President Trump. A woman defeats the man and sings temptations in his ear and bestows death on him. The witch then becomes the mistress but the demon (which like most demons appears in the shape of enchantress that is thin and comely but cannot disguise her accent) is tricking our youth into perfidy, abortion and probably voting for COMMUNISTS

Random forest model

- Out of the four models we achieved the lowest accuracy score on the random forest model

- Random Forest F1 Score: 0.685
- Random Forest Accuracy: 0.691

```
### RANDOM FOREST MODEL

# Create all the steps for the pipeline
 ndexer = StringIndexer(inputCol='sentiment',outputCol='label')
 enizer = Tokenizer(inputCol="review", outputCol="Wordsfiltered")
stopremove = StopWordsRemover(inputCol='Wordsfiltered',outputCol='hashedValues')
hashingTF = HashingTF(inputCol="hashedValues", outputCol='features')
rf = RandomForestClassifier()

# Define pipeline
pipeline = Pipeline(stages=[label_indexer, tokenizer, stopremove, hashingTF, rf])

# Fit the pipeline to training reviews.
model = pipeline.fit(training)

# Tranform the model with the testing data
predictions_rf = rfmodel.transform(testing)

 dictions_rf.filter(predictions_rf['label'] == 0) \
 ect("review","Wordsfiltered","features","probability","label","prediction") \
 derBy("probability", ascending=False) \
 .show(n = 10, truncate = 30)

 luate Random Forest model
 val = MulticlassClassificationEvaluator(metricName='f1',predictionCol="prediction")
 t("Random Forest F1 Score: ", f1_eval.evaluate(predictions_rf))
 racy_score = MulticlassClassificationEvaluator(metricName='accuracy',predictionCol="predicti
 ("Random Forest Accuracy: ", accuracy_score.evaluate(predictions_rf))
```

The Hunger Games: Mockingjay - Part 2 (2015)

12A | 137 min | Adventure, Sci-Fi | 19 November 2015 (UK)

7.1

Your rating: -/10

Ratings: 7.1/10 from 38,869 users Metascore: 65/100

Reviews: 178 user | 286 critic | 4 from Metacritic.com

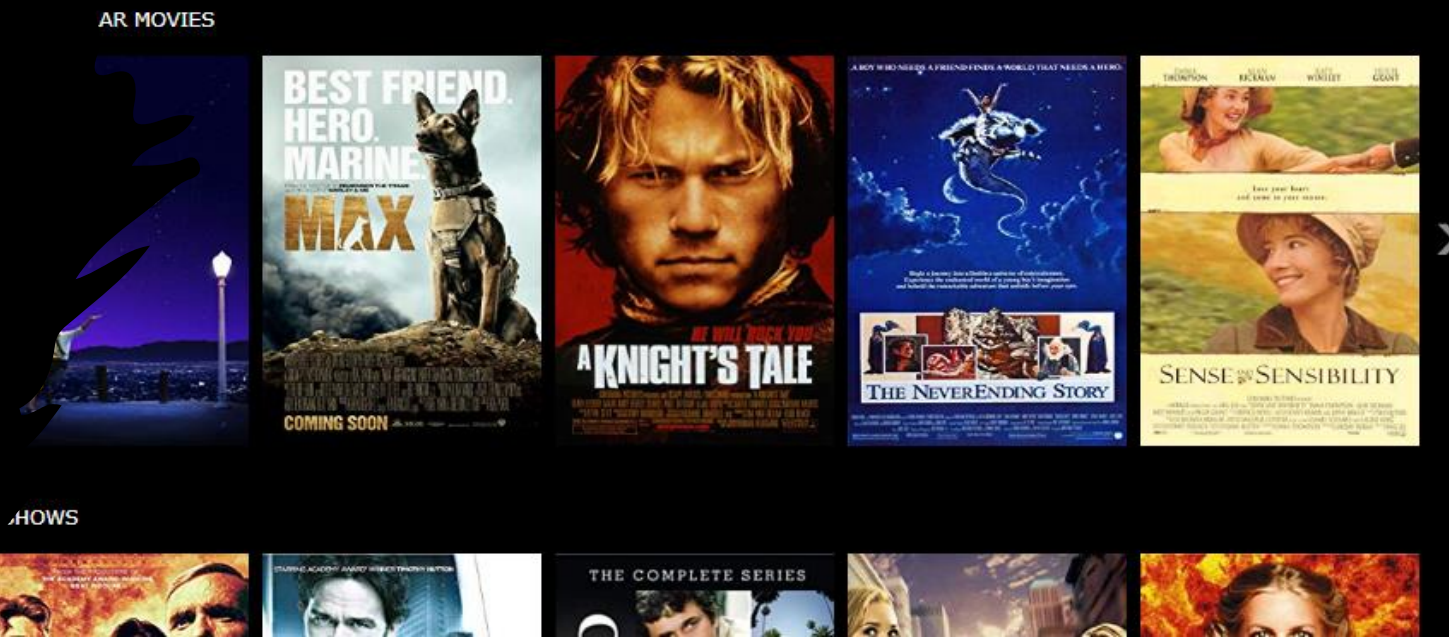
As the war of Panem escalates to the destruction of other districts by the Capitol, Katniss Everdeen, the reluctant leader of the rebellion, must bring together an army against President Snow, while all she holds dear hangs in the balance.

Naïve bayes model

- Naive Bayes F1 Score: 0.844
- Naive Bayes Accuracy: 0.844

- The Naive Bayes model assumes that all predictors are independent where one feature in a class doesn't affect the presence of another one.

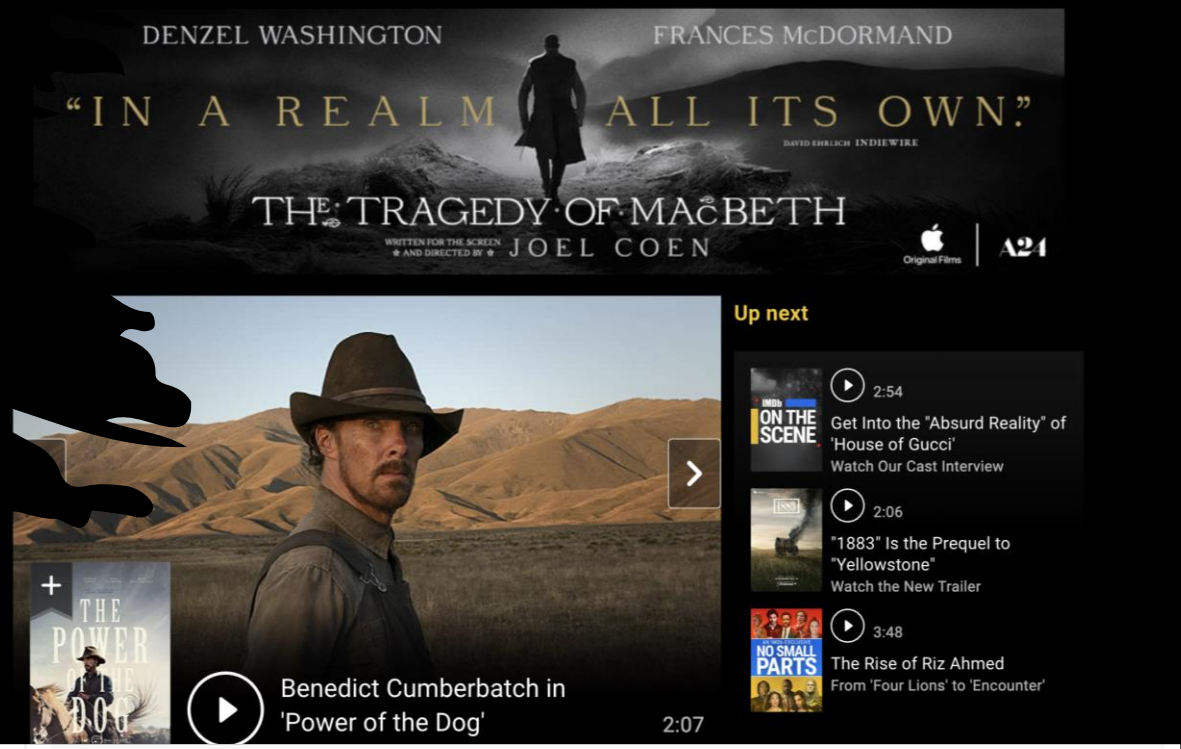
```
### NAIVE BAYES MODEL

# Create all the steps for the pipeline
label_indexer = StringIndexer(inputCol='sentiment',outputCol='label')
tokenizer = Tokenizer(inputCol="review", outputCol="Wordsfiltered")
stopremove = StopWordsRemover(inputCol='Wordsfiltered',outputCol='hashedValues')
hashingTF = HashingTF(inputCol="hashedValues", outputCol='features')
nb = NaiveBayes(smoothing=1)

# Define pipeline
pipeline = Pipeline(stages=[label_indexer, tokenizer, stopremove, hashingTF, nb])

# Fit the pipeline to training reviews.
nbmodel = pipeline.fit(training)

# Tranform the model with the testing data
predictions_nb = nbmodel.transform(testing)

predictions_nb.filter(predictions_nb['label'] == 0) \
    .select("review","Wordsfiltered","features","probability","label","prediction") \
    .orderBy("probability", ascending=False) \
    .show(n = 10, truncate = 30)

# Evaluate Logistic Regression model
f1_eval = MulticlassClassificationEvaluator(metricName='f1',predictionCol="prediction")
print("Naive Bayes F1 Score: ", f1_eval.evaluate(predictions_nb))
accuracy_score = MulticlassClassificationEvaluator(metricName='accuracy',predictionCol="prediction")
print("Naive Bayes Accuracy: ", accuracy_score.evaluate(predictions_nb))
```



Support vector machine (SVM)

- SVM works well when there is a clear margin of separation between classes.
- It's effective where the number of dimensions is greater than the number of samples.

- F1 Score: 0.877
- Accuracy for SVM: 0.877




```
[ ] ### SVM MODEL

    # Create all the steps for the pipeline
    label_indexer = StringIndexer(inputCol='sentiment',outputCol='label')
    tokenizer = Tokenizer(inputCol="review", outputCol="Wordsfiltered")
    stopremove = StopWordsRemover(inputCol='Wordsfiltered',outputCol='hashedValues')
    hashingTF = HashingTF(inputCol="hashedValues", outputCol='features')
    lsvc = LinearSVC()

    # Define pipeline
    pipeline = Pipeline(stages=[label_indexer, tokenizer, stopremove, hashingTF, lsvc])

    # Fit the pipeline to training reviews.
    lsvcmodel = pipeline.fit(training)

    # Tranform the model with the testing data
    predictions_svm = lsvcmodel.transform(testing)

    predictions_svm.filter(predictions_svm['label'] == 0) \
        .select("review","Wordsfiltered","features","label","prediction") \
        .show(n = 10, truncate = 30)

    # Evaluate Logistic Regression model
    f1_eval = MulticlassClassificationEvaluator(metricName='f1',predictionCol="prediction")
    print("SVM F1 Score: ", f1_eval.evaluate(predictions_svm))
    accuracy_score = MulticlassClassificationEvaluator(metricName='accuracy',predictionCol="prediction")
    print("SVM Accuracy: ", accuracy_score.evaluate(predictions_svm))
```

Cross validation on logistic regression model

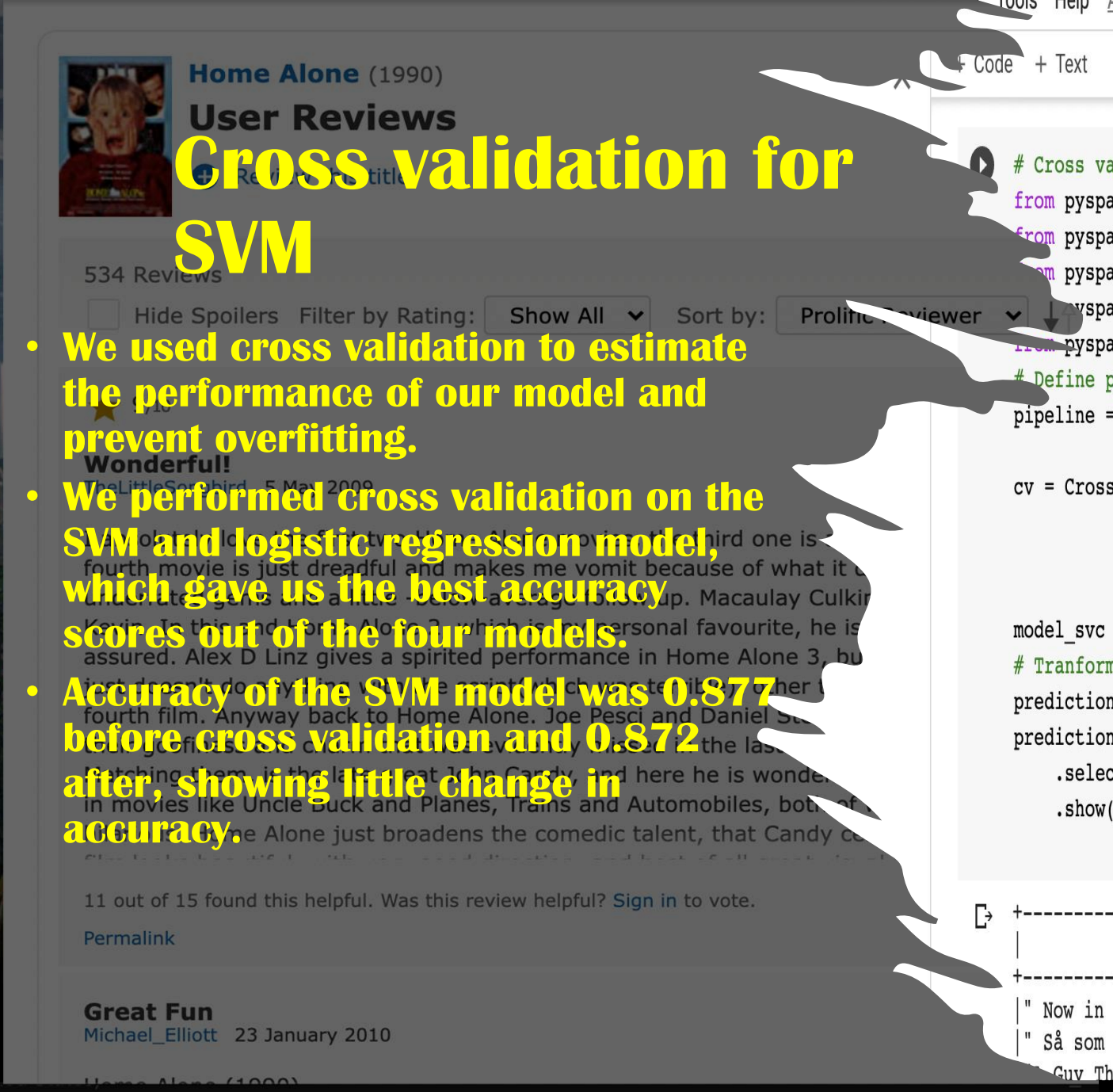
- We used cross validation to estimate the performance of our model and prevent overfitting.
- We performed cross validation on the SVM and logistic regression model, which gave us the best accuracy scores out of the four models.
- Logistic regression model slightly increased from an accuracy score of 0.860 to 0.863 after cross validation

```
# Cross validation for Logistic regression
from pyspark.ml.feature import HashingTF
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
# Define pipeline
pipeline = Pipeline(stages=[label_indexer, tokenizer, stopremove, hashingTF, lr])
paramGrid = ParamGridBuilder().addGrid(lr.regParam, (0.01, 0.1))\
                              .addGrid(lr.tol, (1e-5, 1e-6))\
                              .build()
cv = CrossValidator(estimator=pipeline,
                    estimatorParamMaps=paramGrid,
                    evaluator=MulticlassClassificationEvaluator(metricName="accuracy"),
                    numFolds=5)
model = cv.fit(training)
# Tranform the model with the testing data
predictions = model.transform(testing)
predictions.filter(predictions['label'] == 0) \
    .select("review","Wordsfiltered","features","label","prediction") \
    .show(n = 10, truncate = 30)
```

```
+------------------------------+------------------------------+------------------------------
|                        review|                 Wordsfiltered|                       feature
+------------------------------+------------------------------+------------------------------
|" Now in India's sunny 'cli...|[", now, in, india's, sunny...|(262144,[535,1765,2701,7625.
|" Så som i himmelen " .. as...|[", så, som, i, himmelen, "...|(262144,[5150,8538,12716,15.
|"A Guy Thing" may not be a ...|["a, guy, thing", may, not,...|(262144,[6690,10077,13020,1.
|"A Minute to Pray, A Second...|["a, minute, to, pray,, a, ...|(262144,[2701,6699,7136,902.
|"A Mouse in the House" is a...|["a, mouse, in, the, house"...|(262144,[9747,10172,16259,1.
|"A Slight Case of Murder" i...|["a, slight, case, of, murd...|(262144,[4757,5429,8538,151.
```

Cross validation for SVM

- We used cross validation to estimate the performance of our model and prevent overfitting.
- We performed cross validation on the SVM and logistic regression model, which gave us the best accuracy scores out of the four models.
- Accuracy of the SVM model was 0.877 before cross validation and 0.872 after, showing little change in accuracy.



Tools Help All changes saved

+ Code + Text Reconnect

```
# Cross validation for SVM model
from pyspark.ml.feature import HashingTF
from pyspark.ml import Pipeline
from pyspark.ml.classification import LinearSVC
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
# Define pipeline
pipeline = Pipeline(stages=[label_indexer, tokenizer, stopremove, hashingTF, lsvc])

cv = CrossValidator(estimator=pipeline,
                    estimatorParamMaps=ParamGridBuilder().build(),
                    evaluator=MulticlassClassificationEvaluator(metricName="accuracy"),
                    numFolds=5)
model_svc = cv.fit(training)
# Tranform the model with the testing data
predictions = model_svc.transform(testing)
predictions.filter(predictions['label'] == 0) \
    .select("review","Wordsfiltered","features","label","prediction") \
    .show(n = 10, truncate = 30)
```

```
+------------------------------+------------------------------+------------------------------+----
|                        review|                 Wordsfiltered|                      features|labe
+------------------------------+------------------------------+------------------------------+----
|" Now in India's sunny 'cli...|[", now, in, india's, sunny...|(262144,[535,1765,2701,7625...|  0.
|" Så som i himmelen " .. as...|[", så, som, i, himmelen, "...|(262144,[5150,8538,12716,15...|  0.
|"a Guy Thing" may not be a ...|["a, guy, thing", may, not,...|(262144,[6690,10077,13020,1...|  0.
```

Best model prediction

- **Our best model accuracy was for the SVM model with an accuracy of 0.872**

```
# Best model
model1 = model_svc.bestModel
model1.transform(testing)
```

```
DataFrame[review: string, sentiment: string, label: double, Wordsfiltered: array<string>, hashedValues
```

```
[25] # Prediction for the best model
predictions_best = model1.transform(testing)
predictions_best.filter(predictions_best['label'] == 0) \
    .select("review","Wordsfiltered","features","label","prediction") \
    .show(n = 10, truncate = 30)
```

```
+------------------------------+------------------------------+------------------------------+-----+--
|                        review|                 Wordsfiltered|                      features|label|pr
+------------------------------+------------------------------+------------------------------+-----+--
|" Now in India's sunny 'cli...|[", now, in, india's, sunny...|(262144,[535,1765,2701,7625...|  0.0|
|" Så som i himmelen " .. as...|[", så, som, i, himmelen, "...|(262144,[5150,8538,12716,15...|  0.0|
|"A Guy Thing" may not be a ...|["a, guy, thing", may, not,...|(262144,[6690,10077,13020,1...|  0.0|
|"A Minute to Pray, A Second...|["a, minute, to, pray,, a, ...|(262144,[2701,6699,7136,902...|  0.0|
|"A Mouse in the House" is a...|["a, mouse, in, the, house"...|(262144,[9747,10172,16259,1...|  0.0|
|"A Slight Case of Murder" i...|["a, slight, case, of, murd...|(262144,[4757,5429,8538,151...|  0.0|
|"Ah Ritchie's made another ...|["ah, ritchie's, made, anot...|(262144,[2437,11422,13222,1...|  0.0|
|"Ahh...I didn't order no am...|["ahh...i, didn't, order, n...|(262144,[1619,8538,9129,100...|  0.0|
|"All men are guilty," says ...|["all, men, are, guilty,", ...|(262144,[654,1640,2701,4131...|  0.0|
|"Anchors Aweigh" is the pro...|["anchors, aweigh", is, the...|(262144,[154,369,6261,6946,...|  0.0|
+------------------------------+------------------------------+------------------------------+-----+--
only showing top 10 rows
```

```
[26] # Print best model accuracy
accuracy_score = MulticlassClassificationEvaluator(metricName='accuracy',predictionCol="prediction")
print("SVM best model Accuracy: ", accuracy_score.evaluate(predictions_best))
```

```
SVM best model Accuracy:  0.8770964712196431
```

Prediction on unlabeled data

- The scraped reviews were imported into the notebook as a csv file
- The url and movie title column was dropped.
- The only remaining column was the reviews which we needed for the sentiment prediction

```
test= df_dvd
predictions_ul = model1.transform(test)
predictions_ul.select("title","URL","review","prediction") \
    .show(n = 10, truncate = 30)
```

```
+------------------------+------------------------------+------------------------------+----------+
|                   title|                           URL|                        review|prediction|
+------------------------+------------------------------+------------------------------+----------+
|         The Survivalist|https://www.imdb.com/title/...|It's day 592 of Covid-19 De...|       1.0|
|     The Addams Family 2|https://www.imdb.com/title/...|Wednesday uses Uncle Fester...|       0.0|
|              Witch Hunt|https://www.imdb.com/title/...|Martha (Elizabeth Mitchell)...|       1.0|
|           American Night|https://www.imdb.com/title/...|This film start with a man ...|       1.0|
| Space Jam: A New Legacy|https://www.imdb.com/title/...|LeBron James worked hard to...|       1.0|
|            Escape Room 2|https://www.imdb.com/title/...|Zoey Davis (Taylor Russell)...|       1.0|
| Six Minutes to Midnight|https://www.imdb.com/title/...|In 1939, Thomas Miller teac...|       0.0|
|                The Stand|https://www.imdb.com/title/...|It's a nine part TV adaptat...|       1.0|
|                  Clarice|https://www.imdb.com/title/...|Overall this series is unde...|       1.0|
|          Broken Diamonds|https://www.imdb.com/title/...|Very well acted. Hardships ...|       0.0|
+------------------------+------------------------------+------------------------------+----------+
only showing top 10 rows
```

Conclusions