Introduction

IMDb, the Internet Movie Database, is a cornerstone for film enthusiasts and industry professionals alike. Established in 1990, it has evolved into a comprehensive online repository, offering a wealth of information about movies, TV shows, video games, and streaming content. From cast and crew details to release dates, plot summaries, and user ratings, IMDb provides a one-stop destination for exploring the world of entertainment. The platform's multimedia content, including trailers and interviews, enhances the user experience. With a robust social component, allowing users to rate, review, and engage in discussions, IMDb has become an essential tool for navigating the cinematic landscape.

1 import Necessary Library

In [305…

```
import pandas as pd
import numpy as np
```

2 import Dataset

In [306…

```
df = pd.read_csv("/kaggle/input/imdb-movie/IMDB Movie.csv")
```

In [307…

```
df.head()
```

Out[307…

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. <br /><br />The...	positive
2	I thought this was a wonderful way to spend ti...	positive

3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

In [308…

```
df.tail()
```

Out[308…

	review	sentiment
49995	I thought this movie did a down right good job...	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di...	negative
49997	I am a Catholic taught in parochial elementary...	negative
49998	I'm going to have to disagree with the previou...	negative
49999	No one expects the Star Trek movies to be high...	negative

In [309…

```
df.shape
```

Out[309…

```
(50000, 2)
```

In [310…

```
df['sentiment'].value_counts()
```

Out[310…

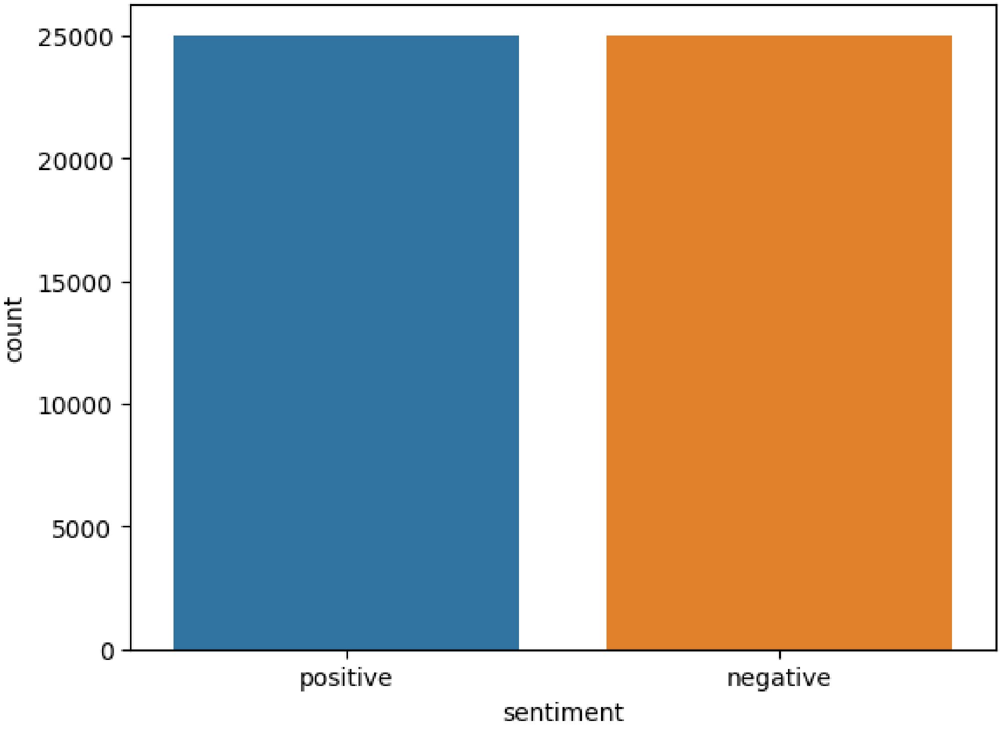
```
sentiment
positive    25000
negative    25000
Name: count, dtype: int64
```

In [311…

```
import seaborn as sns
sns.countplot(x='sentiment',data=df)
```

Out[311…

```
<Axes: xlabel='sentiment', ylabel='count'>
```



In [312

In [312…

```
positive_review=list(df[df['sentiment']=='positive']['review'])[:100]
negative_review=list(df[df['sentiment']=='negative']['review'])[:100]
```

In [313…

```
from wordcloud import WordCloud, STOPWORDS
from matplotlib import pyplot as plt
stopwords = set(STOPWORDS)
stopwords
```

Out[313…

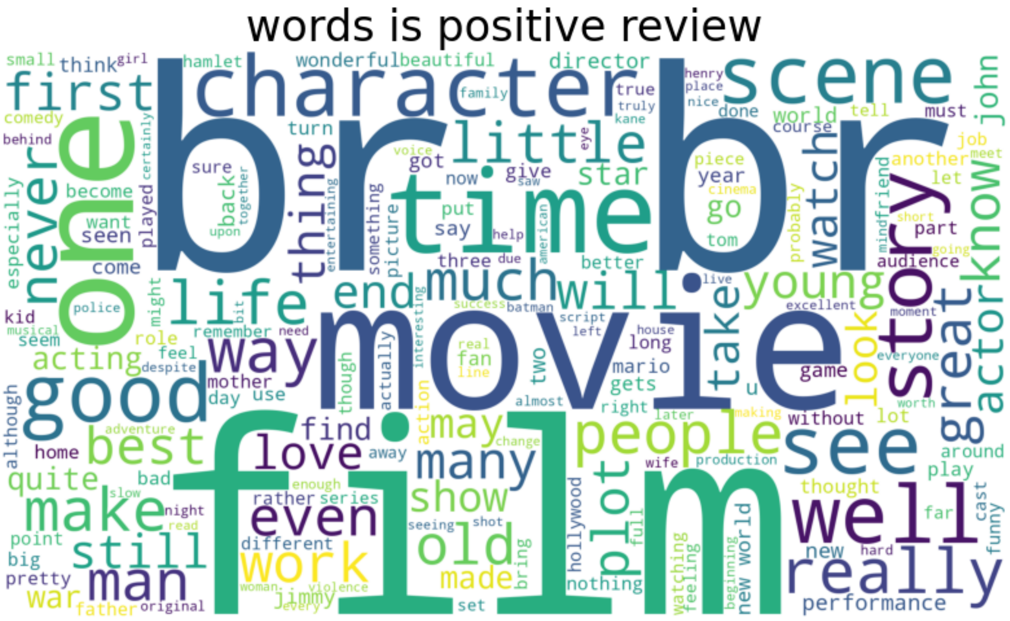
```
{'a',
 'about',
 'above',
 'after',
 'again',
 'against',
 'all',
 'also',
 'am',
 'an',
 'and',
 'any',
 'are',
 "aren't",
 'as',
 'at',
 'be',
 'because',
 'been',
 'before',
 'being',
 'below',
 'between',
 'both',
 'but',
 'by',
 'can',
 "can't",
 'cannot',
 'com',
 'could',
 "couldn't",
 'did',
 "didn't",
 'do',
 'does',
 "doesn't",
 'doing',
 "don't",
 'down',
 'during',
 'each',
 'else',
 'ever',
 'few',
 'for',
 'from',
 'further',
 'get',
 'had',
 "hadn't",
 'has',
 "hasn't",
 'have',
 "haven't",
 'having',
 'he',
 "he'd",
 "he'd",
 "he'll",
 "he's",
 'hence',
 'her',
 'here',
 "here's",
 'hers',
 'herself',
 'him',
 'himself',
 'his',
 'how',
 "how's",
 'however',
 'http',
 'i',
 "i'd",
 "i'll",
 "i'm",
 "i've",
 'if',
 'in',
 'into',
 'is',
 "isn't",
 'it',
 "it's",
 'its',
 'itself',
 'just',
 'k',
 "let's",
 'like',
 'me',
 'more',
 'most',
 "mustn't",
 'my',
 'myself',
 'no',
 'nor',
 'not',
 'of',
 'off',
 'on',
 'once',
 'only',
 'or',
 'other',
 'otherwise',
 'ought',
 'our',
 'ours',
 'ourselves',
 'out',
 'over',
 'own',
 'r',
 'same',
 'shall',
 "shan't",
 'she',
 "she'd",
 "she'll",
 "she's",
 'should',
 "shouldn't",
 "shouldn't",
 'since',
 'so',
 'some',
 'such',
 'than',
 'that',
 "that's",
 'the',
 'their',
 'theirs',
 'them',
 'themselves',
 'then',
 'there',
 "there's",
 'therefore',
 'these',
 'they',
 "they'd",
 "they'll",
 "they're",
 "they've",
 'this',
 'those',
 'through',
 'to',
 'too',
 'under',
 'until',
 'up',
 'very',
 'was',
 "wasn't",
 'we',
 "we'd",
 "we'll",
 "we're",
 "we've",
 'were',
 "weren't",
 'what',
 "what's",
 'when',
 "when's",
 'where',
 "where's",
 'which',
 'while',
 'who',
 "who's",
 'whom',
 'why',
 "why's",
 'with',
 "won't",
 'would',
 "wouldn't",
 'www',
 'you',
 "you'd",
 "you'll",
 "you're",
 "you've",
 'your',
 'yours',
 'yourself',
 'yourselves'}
```

In [314…

```
def create_cloud(string, title=None):
    cloud = WordCloud(height=1080,
                      width=1920,
                      background_color='white',
                      min_font_size=10,
                      stopwords=STOPWORDS).generate(string)
    plt.figure(figsize=(10, 20))
    plt.imshow(cloud)
    plt.axis("off")
    if title:
        plt.title(title, fontdict={'fontsize':24})
    plt.show()
```

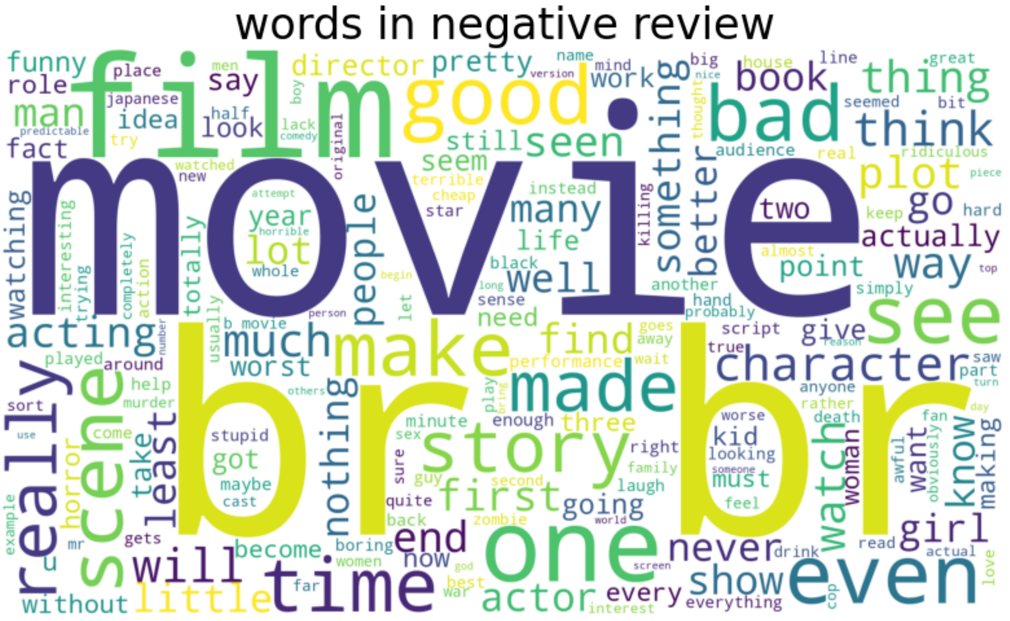
In [315…

```
create_cloud(' '.join(positive_review).lower(), 'words is positive review')
```



In [316…

```
create_cloud(' '.join(negative_review).lower(),'words in negative review')
```



In [317

In [317…

```
def text_processing(data):
    from bs4 import BeautifulSoup
    import re
    def decontracted(phrase):
        # specific
        phrase= re.sub(r'<br /><br />',' ',phrase)
        phrase = re.sub(r"won't", "will not", phrase)
        phrase = re.sub(r"can\'t", "can not", phrase)

        # general
        phrase = re.sub(r"n\'t", " not", phrase)
        phrase = re.sub(r"\'re", " are", phrase)
        phrase = re.sub(r"\'s", " is", phrase)
        phrase = re.sub(r"\'d", " would", phrase)
        phrase = re.sub(r"\'ll", " will", phrase)
        phrase = re.sub(r"\'t", " not", phrase)
        phrase = re.sub(r"\'ve", " have", phrase)
        phrase = re.sub(r"\'m", " am", phrase)
        phrase = re.sub(r'"', " ", phrase)
        return phrase
    stopwords=set(STOPWORDS)

    # Combining all the above sentence
    from tqdm import tqdm
    preprocessed_reviews = []
    # tqdm is for printing the status bar
    for sentance in tqdm(data['review'].values):
        sentance = re.sub(r"http\S+", "", sentance)
        sentance = BeautifulSoup(sentance, 'lxml').get_text()
        sentance = decontracted(sentance)
        sentance = re.sub("\S*\d\S*", "", sentance).strip()
        # https://gist.github.com/sebleier/554280
        sentance = ' '.join(e.lower() for e in sentance.split() if e not in s
        preprocessed_reviews.append(sentance.strip())

    from nltk.stem import PorterStemmer

    porter = PorterStemmer()
    list_of_sentence=[]
    for  sentence in preprocessed_reviews:
        words_in_sentence=[]
        for words in sentence.split():
            words_in_sentence.append(porter.stem(words))

        list_of_sentence.append(' '.join(words_in_sentence))
    return(list_of_sentence)
```

In [318…

```
x=text_processing(df[:1000])
```

```
 82%|████████▏ | 818/1000 [00:00<00:00, 2073.64it/s]/tmp/ipykernel_43/22288752
65.py:29: MarkupResemblesLocatorWarning: The input looks more like a filename t
han markup. You may want to open this file and pass the filehandle into Beautif
ul Soup.
  sentance = BeautifulSoup(sentance, 'lxml').get_text()
100%|██████████| 1000/1000 [00:00<00:00, 2049.99it/s]
```

In [319…

```
df = df[:1000]
```

In [320…

```
df.head()
```

Out[320…

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive

1	A wonderful little production. <br /><br />The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

In [321…

```
df['cleaned_review']=x
```

In [322…

```
df.head()
```

Out[322…

	review	sentiment	cleaned_review
0	One of the other reviewers has mentioned that ...	positive	one review mention watch oz episod will hooked...
1	A wonderful little production. <br /><br />The...	positive	a wonder littl production. the film techniqu u...
2	I thought this was a wonderful way to spend ti...	positive	i thought wonder way spend time hot summer wee...
3	Basically there's a family where a little boy ...	negative	basic famili littl boy (jake) think zombi clos...
4	Petter Mattei's "Love in the Time of Money" is...	positive	petter mattei love time money visual stun film...

In [323…

```
X = df['cleaned_review']
Y = df['sentiment']
```

In [324…

```
X
```

Out[324…

```
0      one review mention watch oz episod will hooked...
1      a wonder littl production. the film techniqu u...
2      i thought wonder way spend time hot summer wee...
3      basic famili littl boy (jake) think zombi clos...
4      petter mattei love time money visual stun film...
                             ...
995    noth sacred. just ask erni fosselius. these da...
996    i hate it. i hate self-awar pretenti inan masq...
997    i usual tri profession construct i critic movi...
998    if go see film histori class someth school, tr...
999    thi zoolog textbook, given depict anim accurat...
Name: cleaned_review, Length: 1000, dtype: object
```

In [325…

```
Y
```

Out[325…

```
0      positive
1      positive
2      positive
3      negative
4      positive
         ...
995    positive
996    negative
997    negative
998    negative
```

```
999    negative
Name: sentiment, Length: 1000, dtype: object
```

In [326…

```
Y = list(Y)
for i in range(len(Y)):
    if Y[i]=='positive':
        Y[i]=1
    else:
        Y[i]=0

df['sentiment_score']=Y

Y=df['sentiment_score']
```

In [327…

```
df
```

Out[327…

	review	sentiment	cleaned_review	sentiment_score
0	One of the other reviewers has mentioned that ...	positive	one review mention watch oz episod will hooked...	1
1	A wonderful little production. <br /><br />The...	positive	a wonder littl production. the film techniqu u...	1
2	I thought this was a wonderful way to spend ti...	positive	i thought wonder way spend time hot summer wee...	1
3	Basically there's a family where a little boy ...	negative	basic famili littl boy (jake) think zombi clos...	0
4	Petter Mattei's "Love in the Time of Money" is...	positive	petter mattei love time money visual stun film...	1
...	...	...	...	...
995	Nothing is sacred. Just ask Ernie Fosselius. T...	positive	noth sacred. just ask erni fosselius. these da...	1
996	I hated it. I hate self-aware pretentious inan...	negative	i hate it. i hate self-awar pretenti inan masq...	0
997	I usually try to be professional and construct...	negative	i usual tri profession construct i critic movi...	0
998	If you like me is going to see this in a film ...	negative	if go see film histori class someth school, tr...	0
999	This is like a zoology textbook, given that it...	negative	thi zoolog textbook, given depict anim accurat...	0

1000 rows × 4 columns

In [328…

```
from sklearn.model_selection import train_test_split
```

In [329…

```
X_train, X_test, y_train, y_test = train_test_split(X[:1000], Y[:1000], test_
```

In [330…

```
X_train
```

Out[330…

```
105    marion davi star remark comedi show peopl rele...
68     i sure produc need trade name somewhat success...
479    joseph bradi clarenc doolittl two sailors, fou...
399    thi movi fairli entertain comedi murphi law ap...
434    yes, indeed, good movie. a love biangle, (sorr...
                             ...
835    the stori told before. a deadli diseas spread ...
192    nifti littl episod play mainli laughs, clever ...
629    let keep simple: my two kid glu movie. it flaw...
559    so i rent netflix somebodi gave roger ebert bo...
684    the perfect murder foil wife(play mari ellen t...
Name: cleaned_review, Length: 700, dtype: object
```

In [331…

```
y_train
```

Out[331…

```
105    1
68     0
479    1
399    1
434    0
      ..
835    1
192    1
629    1
559    0
684    0
Name: sentiment_score, Length: 700, dtype: int64
```

In [332…

```
list(y_test).count(0)
```

Out[332…

```
162
```

In [333…

```
list(y_test).count(1)
```

Out[333…

```
138
```

In [334…

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
X_train_bow = vectorizer.fit_transform(X_train)
X_test_bow = vectorizer.transform(X_test)
```

In [335…

```
X_train_bow.shape, X_test_bow.shape
```

Out[335…

```
((700, 12936), (300, 12936))
```

In [336…

```
X_test_bow
```

Out[336…

```
<300x12936 sparse matrix of type '<class 'numpy.int64'>'
	with 28221 stored elements in Compressed Sparse Row format>
```

In [337…

```
X_train.shape, X_test.shape
```

Out[337…

```
((700,), (300,))
```

In [338…

```
y_train.shape, y_test.shape
```

Out[338…

```
((700,), (300,))
```

(1) KNN Algorithm

In [339…

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
for i in range(10,30):

    print('K',i)

    # initialization
    neigh = KNeighborsClassifier(n_neighbors=i)

    # Training
    neigh.fit(X_train_bow, y_train)

    # Test the training data
    y_pred_train = neigh.predict(X_train_bow)
    accuracy_train = accuracy_score(y_pred_train,y_train)
    f1_train = f1_score(y_pred_train,y_train)

    # Test the test data
    y_pred_test = neigh.predict(X_test_bow)
    accuracy_test = accuracy_score(y_pred_test,y_test)
    f1_test = f1_score(y_pred_test,y_test)

    print(accuracy_train,accuracy_test)
    print(f1_train,f1_test)
    print()
```

```
K 10
0.69 0.5533333333333333
0.7290886392009986 0.5838509316770186

K 11
0.66 0.5333333333333333
0.7337807606263983 0.6089385474860335

K 12
0.6757142857142857 0.55
0.7241798298906439 0.5896656534954408

K 13
0.6614285714285715 0.5066666666666667
0.7363737486095662 0.6

K 14
0.6914285714285714 0.53
0.7422434367541767 0.5936599423631124

K 15
0.6528571428571428 0.5166666666666667
0.7338444687842279 0.6214099216710183

K 16
0.6742857142857143 0.5366666666666666
0.733644859813084 0.6084507042253522

K 17
0.6414285714285715 0.5233333333333333
```

```
0.7238723872387239 0.6285714285714286

K 18
0.6571428571428571 0.56
0.725400457665904 0.641304347826087

K 19
0.6257142857142857 0.5133333333333333
0.7164502164502164 0.6256410256410257

K 20
0.6428571428571429 0.5533333333333333
0.7203579418344519 0.6417112299465241

K 21
0.6071428571428571 0.5266666666666666
0.7089947089947091 0.6377551020408163

K 22
0.6271428571428571 0.5433333333333333
0.7159956474428727 0.6422976501305483

K 23
0.6 0.53
0.7089397089397088 0.6430379746835443

K 24
0.6171428571428571 0.5433333333333333
0.7136752136752136 0.6422976501305483

K 25
0.6028571428571429 0.52
0.7104166666666667 0.6363636363636364

K 26
0.6057142857142858 0.5333333333333333
0.7044967880085653 0.6391752577319587

K 27
0.5957142857142858 0.51
0.7055150884495316 0.631578947368421

K 28
0.6071428571428571 0.5166666666666667
0.7083775185577942 0.6272493573264781

K 29
0.59 0.5166666666666667
0.7044284243048403 0.6401985111662531
```

In [340…

```python
X_train = X_train_bow.toarray()
X_test = X_test_bow.toarray()
```

In [341…

```python
X_train[0]
```

Out[341…

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

In [342…

```python
X_test[0]
```

Out[342…

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

⚙ Machine Learning Algorithm

(1) KNN 🔄

In [343…

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
```

In [344…

```
neigh = KNeighborsClassifier(n_neighbors=20)
```

In [345…

```
neigh.fit(X_train_bow, y_train)
```

Out[345…

```
KNeighborsClassifier(n_neighbors=20)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [346…

```
# Test the training data
y_pred_train = neigh.predict(X_train_bow)
accuracy_train = accuracy_score(y_pred_train,y_train)
f1_train = f1_score(y_pred_train,y_train)


# Test the test data
y_pred_test = neigh.predict(X_test_bow)
accuracy_test = accuracy_score(y_pred_test,y_test)
f1_test = f1_score(y_pred_test,y_test)


print(accuracy_train,accuracy_test)
print("f1_train : ",f1_train)
print("f1_test : ",f1_test)
```

```
0.6428571428571429 0.5533333333333333
f1_train :  0.7203579418344519
f1_test :  0.6417112299465241
```

(2) Naive Bayes classifier

In [347…

```
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import MultinomialNB

from sklearn import metrics
```

In [348…

```
# GaussianNB
```

In [349…

```
G_classifier = GaussianNB()
```

In [350…

```
G_classifier.fit(X_train, y_train)
```

Out[350…

```
GaussianNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [351…

```
train_predictions = G_classifier.predict(X_train)

train_accuracy21 = accuracy_score(y_train, train_predictions)
```

In [352…

```
test_predictions = G_classifier.predict(X_test)

test_accuracy21 = accuracy_score(y_test, test_predictions)
```

In [353…

```
print(f"Training Accuracy: {train_accuracy21}")
print(f"Testing Accuracy: {test_accuracy21}")
```

```
Training Accuracy: 0.9985714285714286
Testing Accuracy: 0.6233333333333333
```

In [354…

```
# BernoulliNB
```

In [355…

```
B_classifier = BernoulliNB()
```

In [356…

```
B_classifier.fit(X_train, y_train)
```

Out[356…

```
BernoulliNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [357…

```
train_predictions = B_classifier.predict(X_train)

train_accuracy22 = accuracy_score(y_train, train_predictions)
```

In [358…

```
test_predictions = G_classifier.predict(X_test)

test_accuracy22 = accuracy_score(y_test, test_predictions)
```

In [359…

```
print(f"Training Accuracy: {train_accuracy22}")
print(f"Testing Accuracy: {test_accuracy22}")
```

```
Training Accuracy: 0.9928571428571429
Testing Accuracy: 0.6233333333333333
```

In [360…

```
# MultinomialNB
```

In [361…

```
M_classifier = MultinomialNB()
```

In [362…

```
M_classifier.fit(X_train, y_train)
```

Out[362…

```
MultinomialNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [363…

```
train_predictions = M_classifier.predict(X_train)

train_accuracy23 = accuracy_score(y_train, train_predictions)
```

In [364…

```
test_predictions = M_classifier.predict(X_test)

test_accuracy23 = accuracy_score(y_test, test_predictions)
```

In [365…

```
print(f"Training Accuracy: {train_accuracy23}")
print(f"Testing Accuracy: {test_accuracy23}")
```

```
Training Accuracy: 0.9914285714285714
Testing Accuracy: 0.7866666666666666
```

In [366…

```
# GaussianNB
# BernoulliNB
# MultinomialNB

# Being the best of them | BernoulliNB |
```

(3) Decision Tree

In [367…

```
from sklearn.tree import DecisionTreeClassifier
```

In [368…

```
clf = DecisionTreeClassifier()
```

In [369…

```
clf.fit(X_train, y_train)
```

Out[369…

```
DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [370…

```
train_predictions = clf.predict(X_train)

train_accuracy3 = accuracy_score(y_train, train_predictions)
```

In [371…

```
test_predictions = clf.predict(X_test)

test_accuracy3 = accuracy_score(y_test, test_predictions)
```

In [372…

```
print(f"Training Accuracy: {train_accuracy3}")
print(f"Testing Accuracy: {test_accuracy3}")
```

```
Training Accuracy: 1.0
Testing Accuracy: 0.6866666666666666
```

(4) Random Forest 🔄

In [373…

```
from sklearn.ensemble import RandomForestClassifier
```

In [374…

```
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [375…

```
rf_classifier.fit(X_train, y_train)
```

Out[375…

```
RandomForestClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [376…

```
train_predictions = rf_classifier.predict(X_train)

train_accuracy4 = accuracy_score(y_train, train_predictions)
```

In [377…

```
test_predictions = rf_classifier.predict(X_test)

test_accuracy4 = accuracy_score(y_test, test_predictions)
```

In [378…

```
print(f"Training Accuracy: {train_accuracy4}")
print(f"Testing Accuracy: {test_accuracy4}")
```

```
Training Accuracy: 1.0
Testing Accuracy: 0.8166666666666667
```

(5) Boosting Algorithm 🔄

In [379…

```
from sklearn.ensemble import AdaBoostClassifier
```

In [380…

```
base_classifier = DecisionTreeClassifier(max_depth=1)
```

In [381…

```
adaboost_classifier = AdaBoostClassifier(base_classifier, n_estimators=50, ra
```

In [382…

```
adaboost_classifier.fit(X_train, y_train)
```

Out[382…

```
AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=1),
                   random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation

or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [383…

```
train_predictions = adaboost_classifier.predict(X_train)

train_accuracy5 = accuracy_score(y_train, train_predictions)
```

In [384…

```
test_predictions = adaboost_classifier.predict(X_test)

test_accuracy5 = accuracy_score(y_test, test_predictions)
```

In [385…

```
print(f"Training Accuracy: {train_accuracy5}")
print(f"Testing Accuracy: {test_accuracy5}")
```

```
Training Accuracy: 0.8871428571428571
Testing Accuracy: 0.7566666666666667
```

(6).SVM 🔄

In [386…

```
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
```

In [387…

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [388…

```
svm_classifier = SVC(kernel='linear', C=1.0)
```

In [389…

```
svm_classifier.fit(X_train, y_train)
```

Out[389…

```
SVC(kernel='linear')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [390…

```
train_predictions = svm_classifier.predict(X_train)

train_accuracy6 = accuracy_score(y_train, train_predictions)
```

In [391…

```
test_predictions = svm_classifier.predict(X_test)

test_accuracy6 = accuracy_score(y_test, test_predictions)
```

In [392…

```
print(f"Training Accuracy: {train_accuracy6}")
print(f"Testing Accuracy: {test_accuracy6}")
```

```
Training Accuracy: 1.0
Testing Accuracy: 0.7566666666666667
```

(7). Logistic Regression

In [393…

```
from sklearn import linear_model
```

In [394…

```
lrg = linear_model.LogisticRegression()
```

In [395…

```
lrg.fit(X_train, y_train)
```

Out[395…

```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [396…

```
train_predictions = lrg.predict(X_train)

train_accuracy7 = accuracy_score(y_train, train_predictions)
```

In [397…

```
test_predictions = lrg.predict(X_test)

test_accuracy7 = accuracy_score(y_test, test_predictions)
```

In [398…

```
print(f"Training Accuracy: {train_accuracy7}")
print(f"Testing Accuracy: {test_accuracy7}")
```

```
Training Accuracy: 1.0
Testing Accuracy: 0.77
```

(8).Linear Regression

In [399…

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

In [400…

```
model = LinearRegression()
```

In [401…

```
model.fit(X_train, y_train)
```

Out[401…

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [402…

```
train_predictions = clf.predict(X_train)

train_accuracy8 = accuracy_score(y_train, train_predictions)
```

In [403…

```
test_predictions = clf.predict(X_test)

test_accuracy8 = accuracy_score(y_test, test_predictions)
```

In [404…

```
print(f"Training Accuracy: {train_accuracy8}")
print(f"Testing Accuracy: {test_accuracy8}")
```

```
Training Accuracy: 0.9514285714285714
Testing Accuracy: 0.7033333333333334
```

(9).Gradient Boosting Machines (GBM)

In [405…

```
from sklearn.ensemble import GradientBoostingClassifier
```

In [406…

```
model = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_c
```

In [407…

```
model.fit(X_train, y_train)
```

Out[407…

```
GradientBoostingClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [408…

```
train_predictions = model.predict(X_train)

train_accuracy9 = accuracy_score(y_train, train_predictions)
```

In [409…

```
test_predictions = model.predict(X_test)

test_accuracy9 = accuracy_score(y_test, test_predictions)
```

In [410…

```
print(f"Training Accuracy: {train_accuracy9}")
print(f"Testing Accuracy: {test_accuracy9}")
```

```
Training Accuracy: 0.9742857142857143
Testing Accuracy: 0.8033333333333333
```

Random Forest, Decision Tree, Gradient Boosting Machines (GBM), Algorithm is the best accuracy

(GradientBoostingClassifier)

Training Accuracy mean: 1.0

Testing Accuracy mean: 0.95