In [1]:
```
import numpy as np
import pandas as pd
```

In [3]:
```
df = pd.read_csv(r"C:\Users\Arpan Ghosh\OneDrive\Desktop\intern project cer
df
```

Out[3]:

	Reviewer Name	Review Title	Place of Review	Up Votes	Down Votes	Month	Review text	Ra
0	Kamal Suresh	Nice product	Certified Buyer, Chirakkal	889.0	64.0	Feb 2021	Nice product, good quality, but price is now r...	
1	Flipkart Customer	Don't waste your money	Certified Buyer, Hyderabad	109.0	6.0	Feb 2021	They didn't supplied Yonex Mavis 350. Outside ...	
2	A. S. Raja Srinivasan	Did not meet expectations	Certified Buyer, Dharmapuri	42.0	3.0	Apr 2021	Worst product. Damaged shuttlecocks packed in ...	
3	Suresh Narayanasamy	Fair	Certified Buyer, Chennai	25.0	1.0	NaN	Quite O. K. , but nowadays the quality of the...	
4	ASHIK P A	Over priced	NaN	147.0	24.0	Apr 2016	Over pricedJust â?¹620 ..from retailer.I didn'...	
...	...	...	...	...	...	...	...	
8513	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
8514	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
8515	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
8516	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
8517	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

8518 rows × 8 columns

In [4]:
```
df.shape
```

Out[4]:
```
(8518, 8)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Reviewer Name', 'Review Title', 'Place of Review', 'Up Votes',
       'Down Votes', 'Month', 'Review text', 'Ratings'],
      dtype='object')
```

In [7]:
```
df.describe()
```

Out[7]:

	Up Votes	Down Votes	Ratings
count	8508.000000	8508.000000	8518.000000
mean	0.391396	0.121768	4.181028
std	11.613909	3.248022	1.262200
min	0.000000	0.000000	1.000000
25%	0.000000	0.000000	4.000000
50%	0.000000	0.000000	5.000000
75%	0.000000	0.000000	5.000000
max	889.000000	219.000000	5.000000

In [8]:
```
df.columns = df.columns.str.strip()

df.columns
```

Out[8]:
```
Index(['Reviewer Name', 'Review Title', 'Place of Review', 'Up Votes',
       'Down Votes', 'Month', 'Review text', 'Ratings'],
      dtype='object')
```

In [9]:
```
print(df.isnull().sum())
```

```
Reviewer Name       10
Review Title        10
Place of Review     50
Up Votes            10
Down Votes          10
Month              465
Review text          8
Ratings              0
dtype: int64
```

In [10…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8518 entries, 0 to 8517
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Reviewer Name    8508 non-null   object 
 1   Review Title     8508 non-null   object 
 2   Place of Review  8468 non-null   object 
 3   Up Votes         8508 non-null   float64
 4   Down Votes       8508 non-null   float64
 5   Month            8053 non-null   object 
 6   Review text      8510 non-null   object 
 7   Ratings          8518 non-null   int64  
dtypes: float64(2), int64(1), object(5)
memory usage: 532.5+ KB
```

In [11…

```
df.fillna({'Reviewer Name': 'Anonymous',
           'Review Title': 'No Title',
           'Place of Review': 'Unknown',
           'Up Votes': df['Up Votes'].median(),
           'Down Votes': df['Down Votes'].median(),
           'Month': 'Unknown'},
          inplace=True)
```

In [12…

```
print(df.isnull().sum())
```

```
Reviewer Name      0
Review Title       0
Place of Review    0
Up Votes           0
Down Votes         0
Month              0
Review text        8
Ratings            0
dtype: int64
```

In [13…

```
df.head()
```

Out[13]:

	Reviewer Name	Review Title	Place of Review	Up Votes	Down Votes	Month	Review text	Rat
0	Kamal Suresh	Nice product	Certified Buyer, Chirakkal	889.0	64.0	Feb 2021	Nice product, good quality, but price is now r...	
1	Flipkart Customer	Don't waste your money	Certified Buyer, Hyderabad	109.0	6.0	Feb 2021	They didn't supplied Yonex Mavis 350. Outside ...	
2	A. S. Raja Srinivasan	Did not meet expectations	Certified Buyer, Dharmapuri	42.0	3.0	Apr 2021	Worst product. Damaged shuttlecocks packed in ...	
3	Suresh Narayanasamy	Fair	Certified Buyer, Chennai	25.0	1.0	Unknown	Quite O. K. , but nowadays the quality of the...	
4	ASHIK P A	Over priced	Unknown	147.0	24.0	Apr 2016	Over pricedJust â?¹620 ..from retailer.I didn'...	

In [14...
```
df['Month'] = pd.to_datetime(df['Month'], errors='coerce')
```

In [15...
```
df['Up Votes'] = df['Up Votes'].astype(int)
df['Down Votes'] = df['Down Votes'].astype(int)
```

In [16...
```
df.columns = df.columns.str.replace(' ', '_')

# Check the updated data types and column names
print(df.dtypes)
print(df.columns)
```

```
Reviewer_Name              object
Review_Title               object
Place_of_Review            object
Up_Votes                    int32
Down_Votes                  int32
Month              datetime64[ns]
Review_text                object
Ratings                     int64
dtype: object
Index(['Reviewer_Name', 'Review_Title', 'Place_of_Review', 'Up_Votes',
       'Down_Votes', 'Month', 'Review_text', 'Ratings'],
      dtype='object')
```

In [17…

```
print(df.describe())
```

```
          Up_Votes   Down_Votes      Ratings
count  8518.000000  8518.000000  8518.000000
mean      0.390937     0.121625     4.181028
std      11.607097     3.246117     1.262200
min       0.000000     0.000000     1.000000
25%       0.000000     0.000000     4.000000
50%       0.000000     0.000000     5.000000
75%       0.000000     0.000000     5.000000
max     889.000000   219.000000     5.000000
```

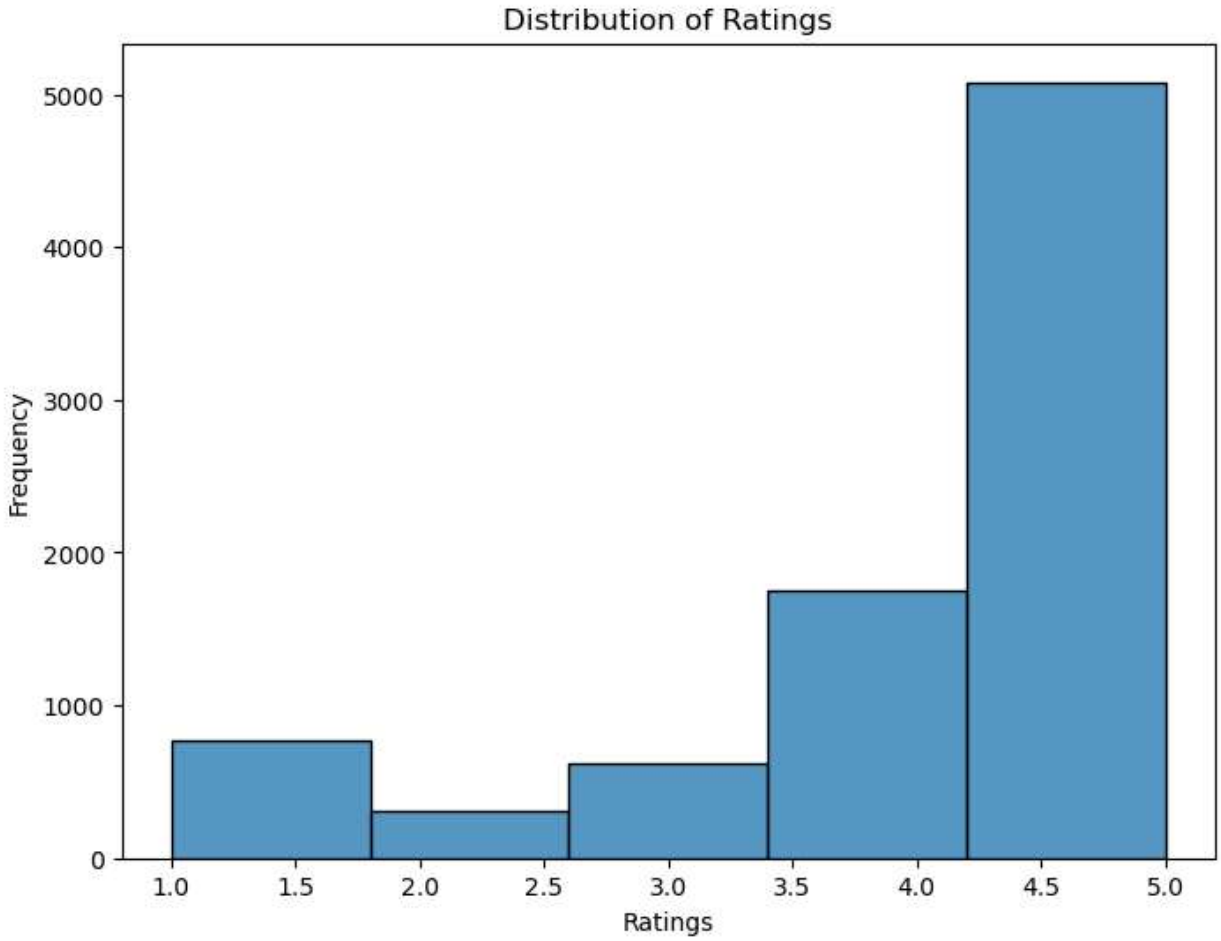
In [18…

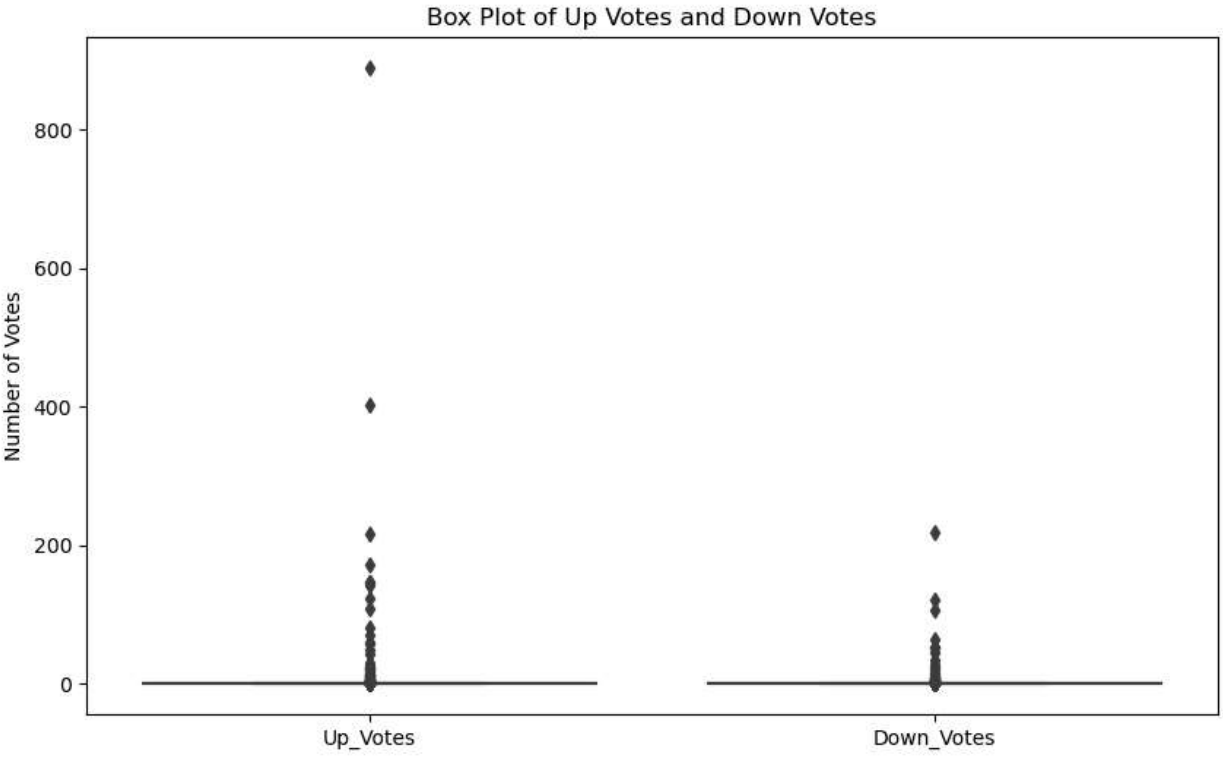
```
## Data Vizualisation
##distribution analysis

import matplotlib.pyplot as plt
import seaborn as sns

# Histogram of Ratings
plt.figure(figsize=(8, 6))
sns.histplot(df['Ratings'], bins=5, kde=False)
plt.title('Distribution of Ratings')
plt.xlabel('Ratings')
plt.ylabel('Frequency')
plt.show()

# Box plot of Up Votes and Down Votes
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[['Up_Votes', 'Down_Votes']])
plt.title('Box Plot of Up Votes and Down Votes')
plt.ylabel('Number of Votes')
plt.show()
```





In [19…

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Download NLTK resources
nltk.download('vader_lexicon')
```

```
# Initialize the Sentiment Intensity Analyzer
sid = SentimentIntensityAnalyzer()
```

```
[nltk_data] Downloading package vader_lexicon to C:\Users\Arpan
[nltk_data]     Ghosh\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

In [20…

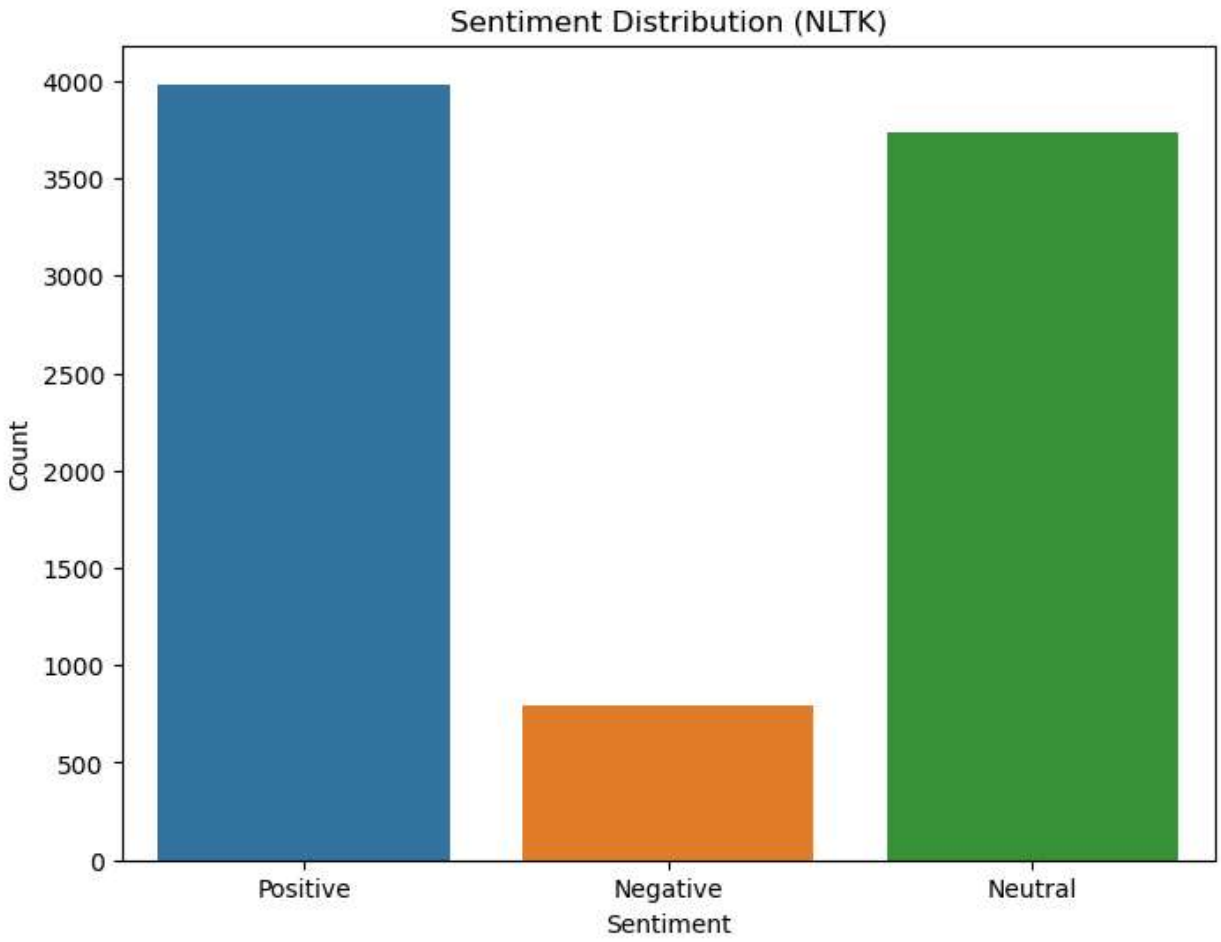
```
def get_sentiment_nltk(text):
    # Check if the text is not NaN
    if isinstance(text, str):
        scores = sid.polarity_scores(text)
        if scores['compound'] > 0:
            return 'Positive'
        elif scores['compound'] < 0:
            return 'Negative'
        else:
            return 'Neutral'
    else:
        return 'Neutral'  # Return 'Neutral' for NaN values
```

In [21…

```
df['Sentiment_NLTK'] = df['Review_text'].apply(get_sentiment_nltk)

# Visualize sentiment distribution

plt.figure(figsize=(8, 6))
sns.countplot(x='Sentiment_NLTK', data=df)
plt.title('Sentiment Distribution (NLTK)')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
```



Text Data

In [22…
```
df['Sentiment_NLTK'].value_counts(normalize=True)
```

Out[22]:
```
Positive    0.467598
Neutral     0.438835
Negative    0.093567
Name: Sentiment_NLTK, dtype: float64
```

In [23…
```
df = df[df.Sentiment_NLTK != "Neutral"]

df.shape
```

Out[23]:
```
(4780, 9)
```

In [24…
```
df['Sentiment_NLTK'].value_counts(normalize=True)
```

Out[24]:
```
Positive    0.833264
Negative    0.166736
Name: Sentiment_NLTK, dtype: float64
```

In [25…
```
# Define a dictionary to map sentiment categories to numerical values
sentiment_mapping = {'Positive': 1, 'Negative': 0}

# Map sentiment categories to numerical values in the 'Sentiment_NLTK' colu
```

```
df['Sentiment_NLTK_numeric'] = df['Sentiment_NLTK'].map(sentiment_mapping)

# Display the updated DataFrame
print(df[['Sentiment_NLTK', 'Sentiment_NLTK_numeric']].head())
```

```
  Sentiment_NLTK  Sentiment_NLTK_numeric
0       Positive                       1
1       Negative                       0
2       Negative                       0
3       Positive                       1
4       Positive                       1
```

```
C:\Users\Arpan Ghosh\AppData\Local\Temp\ipykernel_13876\3472611559.py:5: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Sentiment_NLTK_numeric'] = df['Sentiment_NLTK'].map(sentiment_mappin
g)
```

In [26…
```
df.loc[0, 'Review_text']
```

Out[26]:
```
'Nice product, good quality, but price is now rising which is a bad sign.
800-850 was an affordable price, especially when we play everyday. So kind
ly help us out in terms of the price. Thank You.READ MORE'
```

In [27…
```
df = df[["Review_text", "Sentiment_NLTK_numeric"]]
df.shape
```

Out[27]:
```
(4780, 2)
```

In [28…
```
X = df.Review_text # the column text contains textual data to extract featu
y = df.Sentiment_NLTK_numeric # this is the column we are learning to pred
print(X.shape, y.shape)
```

```
(4780,) (4780,)
```

In [29…
```
from sklearn.model_selection import train_test_split

# split X and y into training and testing sets.
# By default, it splits 75% training and 25% test
# random_state=1 for reproducibility

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)

print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(3585,) (3585,)
(1195,) (1195,)
```

Data cleaning

In [30…
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

def display_wordcloud(data):
    wc = WordCloud(background_color='black',
                   width=1600,
                   height=800).generate(' '.join(data))
    plt.figure(1,figsize=(30,20))
    plt.imshow(wc)
    plt.axis('off')
    plt.show()
```

In [31…
```
import string

import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

# Initialize WordNet lemmatizer
lemmatizer = WordNetLemmatizer()
```

In [32…
```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to C:\Users\Arpan
[nltk_data]     Ghosh\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[32]:
```
True
```

In [33…
```
def clean(doc): # doc is a string of text
    # This text contains a lot of <br/> tags.
    doc = doc.replace("</br>", " ")

    # Remove punctuation and numbers.
    doc = "".join([char for char in doc if char not in string.punctuation

    # Converting to lower case
    doc = doc.lower()

    # Tokenization
    tokens = nltk.word_tokenize(doc)

    # Lemmatize
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]

    # Stop word removal
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [word for word in lemmatized_tokens if word.lower()
```

```
    # Join and return
    return " ".join(filtered_tokens)
```

In [34…
```
from sklearn.feature_extraction.text import CountVectorizer

# instantiate a vectorizer
vect = CountVectorizer(preprocessor=clean)

# use it to extract features from training data
%time X_train_dtm = vect.fit_transform(X_train)

print(X_train_dtm.shape)
```

```
CPU times: total: 1.22 s
Wall time: 2.3 s
(3585, 2547)
```

In [35…
```
X_test_dtm = vect.transform(X_test)

print(X_test_dtm.shape)
```

```
(1195, 2547)
```

In [36…
```
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB() # instantiate a Multinomial Naive Bayes model
%time nb.fit(X_train_dtm, y_train)
```

```
CPU times: total: 15.6 ms
Wall time: 8.55 ms
```

Out[36]:
```
MultinomialNB
MultinomialNB()
```

In [37…
```
import mlflow
mlflow.set_experiment("review_sentiment_prediction")
```

```
2024/04/24 12:10:43 INFO mlflow.tracking.fluent: Experiment with name 'rev
iew_sentiment_prediction' does not exist. Creating a new experiment.
```

Out[37]:
```
<Experiment: artifact_location='file:///C:/Users/Arpan%20Ghosh/mlruns/1384
65458553181901', creation_time=1713940843015, experiment_id='1384654585531
81901', last_update_time=1713940843015, lifecycle_stage='active', name='re
view_sentiment_prediction', tags={}>
```

Running the Experiment

In [38…
```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectoriz
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
```

In [39…

```
import warnings

warnings.filterwarnings('ignore')
```

In [40…

```
pipe = Pipeline(
    [
        ('vectorization', CountVectorizer()),
        ('nb', MultinomialNB())
    ]
)

MAX_FEATURES = [1000, 1500, 2000]
ALPHA = [1, 10]

# Observe the Key Value Pair format
parameter_grid = [{'vectorization__preprocessor' : [clean],
                   'vectorization__max_features' : MAX_FEATURES,
                   'nb__alpha' : ALPHA}]

clf = GridSearchCV(
    estimator=pipe,
    param_grid=parameter_grid,
    scoring='f1',
    cv=5,
    return_train_score=True,
    verbose=1
)

%time clf.fit(X_train, y_train)

print("Best estimator found on train set")
print(clf.best_estimator_)
print()

print('Score on Test Data: ', clf.score(X_test, y_test))
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
CPU times: total: 32.1 s
Wall time: 58.3 s
Best estimator found on train set
Pipeline(steps=[('vectorization',
                 CountVectorizer(max_features=1000,
                                 preprocessor=<function clean at 0x000001F
7B6B41260>)),
                ('nb', MultinomialNB(alpha=1))])

Score on Test Data:  0.9642857142857142
```

Auto Logging All Experiment Runs using MLFlow

In [41…
```
import joblib
from joblib import Memory

import os
import mlflow
```

In [42…
```
# Define a memory object to cache intermediate results
cachedir = '.cache'
memory = Memory(location=cachedir, verbose=0)

# Define the pipeline with caching
pipe = Pipeline(
    [
        ('vectorization', CountVectorizer()),
        ('nb', MultinomialNB())
    ],
    memory=memory
)

MAX_FEATURES = [1000, 1500, 2000]
ALPHA = [1, 10]

# Observe the Key Value Pair format
parameter_grid = [
    {
        'vectorization__preprocessor': [clean],
        'vectorization__max_features': MAX_FEATURES,
        'nb__alpha': ALPHA
    }
]

clf = GridSearchCV(
    estimator=pipe,
    param_grid=parameter_grid,
    scoring='f1',
    cv=5,
    return_train_score=True,
    verbose=1
)

%time clf.fit(X_train, y_train)

print("Best estimator found on train set")
print(clf.best_estimator_)
print()

print('Score on Test Data: ', clf.score(X_test, y_test))
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
CPU times: total: 16.1 s
Wall time: 33.6 s
Best estimator found on train set
Pipeline(memory=Memory(location=.cache\joblib),
         steps=[('vectorization',
                 CountVectorizer(max_features=1000,
                                 preprocessor=<function clean at 0x000001F
7B6B41260>)),
                ('nb', MultinomialNB(alpha=1))])

Score on Test Data:  0.9642857142857142
```

In [43…

```python
import mlflow
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV

pipe = Pipeline(
    [
        ('vectorization', CountVectorizer()),
        ('nb', MultinomialNB())
    ]
)

MAX_FEATURES = [1000, 1500, 2000]
ALPHA = [1, 10]

# Observe the Key Value Pair format
parameter_grid = [{'vectorization__preprocessor' : [clean],
                   'vectorization__max_features' : MAX_FEATURES,
                   'nb__alpha' : ALPHA}]

clf = GridSearchCV(
    estimator=pipe,
    param_grid=parameter_grid,
    scoring='f1',
    cv=5,
    return_train_score=True,
    verbose=1
)

%time clf.fit(X_train, y_train)

mlflow.sklearn.autolog(max_tuning_runs=None)

with mlflow.start_run() as run:
    %time clf.fit(X_train, y_train)  # Here you should use clf instead of g

print("Best estimator found on train set")
print(clf.best_estimator_)
```

```
print()

print('Score on Test Data: ', clf.score(X_test, y_test))
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
CPU times: total: 33.2 s
Wall time: 59.1 s
```

```
2024/04/24 12:15:15 WARNING mlflow.utils.git_utils: Failed to import Git
(the Git executable is probably not on your PATH), so Git SHA is not avail
able. Error: Failed to initialize: Bad git executable.
The git executable must be specified in one of the following ways:
    - be included in your $PATH
    - be set via $GIT_PYTHON_GIT_EXECUTABLE
    - explicitly set via git.refresh(<full-path-to-git-executable>)

All git commands will error until this is rectified.

This initial message can be silenced or aggravated in the future by settin
g the
$GIT_PYTHON_REFRESH environment variable. Use one of the following values:
    - quiet|q|silence|s|silent|none|n|0: for no message or exception
    - warn|w|warning|log|l|1: for a warning message (logging level CRITICA
L, displayed by default)
    - error|e|exception|raise|r|2: for a raised exception

Example:
    export GIT_PYTHON_REFRESH=quiet

2024/04/24 12:15:15 WARNING mlflow.sklearn: Unrecognized dataset type <cla
ss 'pandas.core.series.Series'>. Dataset logging skipped.
Fitting 5 folds for each of 6 candidates, totalling 30 fits
CPU times: total: 39.1 s
Wall time: 1min 17s
Best estimator found on train set
Pipeline(steps=[('vectorization',
                 CountVectorizer(max_features=1000,
                                 preprocessor=<function clean at 0x000001F
7B6B41260>)),
                ('nb', MultinomialNB(alpha=1))])

Score on Test Data:  0.9642857142857142
```

In [44…

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectoriz
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import f1_score
from joblib import Memory  # Import Memory from joblib

# Create a memory object for caching intermediate results
```

```
cachedir = '.cache'
memory = Memory(location=cachedir, verbose=0)

# Define the pipelines for different algorithms
pipelines = {
    'naive_bayes': Pipeline([
        ('vectorization', CountVectorizer()),
        ('classifier', MultinomialNB())
    ], memory=memory),
    'decision_tree': Pipeline([
        ('vectorization', CountVectorizer()),
        ('classifier', DecisionTreeClassifier())
    ], memory=memory),
    'logistic_regression': Pipeline([
        ('vectorization', CountVectorizer()),
        ('classifier', LogisticRegression())
    ], memory=memory)
}

# Define parameter grid for each algorithm
param_grids = {
    'naive_bayes': [
        {
            'vectorization__max_features': [1000, 1500, 2000, 5000],
            'classifier__alpha': [1, 10]
        }
    ],
    'decision_tree': [
        {
            'vectorization__max_features': [1000, 1500, 2000, 5000],
            'classifier__max_depth': [None, 5, 10]
        }
    ],
    'logistic_regression': [
        {
            'vectorization__max_features': [1000, 1500, 2000, 5000],
            'classifier__C': [0.1, 1, 10],
            'classifier__penalty': ['elasticnet'],
            'classifier__l1_ratio': [0.4, 0.5, 0.6],
            'classifier__solver': ['saga'],
            'classifier__class_weight': ['balanced']
        }
    ]
}

# Perform GridSearchCV for each algorithm
best_models = {}

for algo in pipelines.keys():
    print("*"*10, algo, "*"*10)
    grid_search = GridSearchCV(estimator=pipelines[algo],
                               param_grid=param_grids[algo],
```

```
                                cv=5,
                                scoring='f1',
                                return_train_score=True,
                                verbose=1
                               )

    grid_search.fit(X_train, y_train)

    # Log the best model and parameters
    with mlflow.start_run() as run:
        mlflow.sklearn.autolog()
        mlflow.log_params(grid_search.best_params_)
        mlflow.log_metric('f1_score', f1_score(y_test, grid_search.predict

    best_models[algo] = grid_search.best_estimator_

    print('Best parameters found:', grid_search.best_params_)
    print('Score on Test Data:', f1_score(y_test, grid_search.predict(X_te
```

```
2024/04/24 12:18:01 INFO mlflow.utils.autologging_utils: Created MLflow au
tologging run with ID '202112b890c2464aa174afadc2110c27', which will track
hyperparameters, performance metrics, model artifacts, and lineage informa
tion for the current sklearn workflow
2024/04/24 12:18:01 WARNING mlflow.sklearn: Unrecognized dataset type <cla
ss 'pandas.core.series.Series'>. Dataset logging skipped.
********** naive_bayes **********
Fitting 5 folds for each of 8 candidates, totalling 40 fits
2024/04/24 12:18:12 WARNING mlflow.sklearn: Unrecognized dataset type <cla
ss 'pandas.core.series.Series'>. Dataset logging skipped.
2024/04/24 12:18:12 WARNING mlflow.sklearn: Unrecognized dataset type <cla
ss 'pandas.core.series.Series'>. Dataset logging skipped.
2024/04/24 12:18:12 INFO mlflow.utils.autologging_utils: Created MLflow au
tologging run with ID '3fb8bdf3a3344d4083663d3eecd8f942', which will track
hyperparameters, performance metrics, model artifacts, and lineage informa
tion for the current sklearn workflow
2024/04/24 12:18:12 WARNING mlflow.sklearn: Unrecognized dataset type <cla
ss 'pandas.core.series.Series'>. Dataset logging skipped.
Best parameters found: {'classifier__alpha': 1, 'vectorization__max_featur
es': 2000}
Score on Test Data: 0.960513326752221
********** decision_tree **********
Fitting 5 folds for each of 12 candidates, totalling 60 fits
```

```
2024/04/24 12:18:28 INFO mlflow.sklearn.utils: Logging the 5 best runs, 7
runs will be omitted.
2024/04/24 12:18:28 WARNING mlflow.sklearn: Unrecognized dataset type <cla
ss 'pandas.core.series.Series'>. Dataset logging skipped.
2024/04/24 12:18:28 WARNING mlflow.sklearn: Unrecognized dataset type <cla
ss 'pandas.core.series.Series'>. Dataset logging skipped.
2024/04/24 12:18:29 INFO mlflow.utils.autologging_utils: Created MLflow au
tologging run with ID 'aee86ea1f6fa473d856d3dfd227ad3be', which will track
hyperparameters, performance metrics, model artifacts, and lineage informa
tion for the current sklearn workflow
2024/04/24 12:18:29 WARNING mlflow.sklearn: Unrecognized dataset type <cla
ss 'pandas.core.series.Series'>. Dataset logging skipped.
Best parameters found: {'classifier__max_depth': None, 'vectorization__max
_features': 2000}
Score on Test Data: 0.9674511767651478
********** logistic_regression **********
Fitting 5 folds for each of 36 candidates, totalling 180 fits
2024/04/24 12:20:38 INFO mlflow.sklearn.utils: Logging the 5 best runs, 31
runs will be omitted.
2024/04/24 12:20:38 WARNING mlflow.sklearn: Unrecognized dataset type <cla
ss 'pandas.core.series.Series'>. Dataset logging skipped.
2024/04/24 12:20:38 WARNING mlflow.sklearn: Unrecognized dataset type <cla
ss 'pandas.core.series.Series'>. Dataset logging skipped.
Best parameters found: {'classifier__C': 10, 'classifier__class_weight':
'balanced', 'classifier__l1_ratio': 0.5, 'classifier__penalty': 'elasticne
t', 'classifier__solver': 'saga', 'vectorization__max_features': 2000}
Score on Test Data: 0.9718875502008033
```

In [45…

```python
import mlflow
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.datasets import fetch_20newsgroups

# Load data
categories = ['alt.atheism', 'soc.religion.christian', 'comp.graphics', 's
data = fetch_20newsgroups(subset='all', categories=categories, shuffle=Tru

X = data.data
y = data.target

# Create a pipeline with TfidfVectorizer and MultinomialNB
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('clf', MultinomialNB())
])

# Define the parameters for grid search
parameters = {
    'tfidf__max_features': [1000, 2000, 3000],
```

```
    'tfidf__ngram_range': [(1, 1), (1, 2)],
    'clf__alpha': [0.5, 1.0, 1.5]
}

# Perform grid search with cross-validation
grid_search = GridSearchCV(pipeline, parameters, cv=5, verbose=1, n_jobs=-
grid_search.fit(X, y)

# Log parameters and metrics to MLflow
with mlflow.start_run():
    # Log parameters
    for key, value in grid_search.best_params_.items():
        mlflow.log_param(key, value)

    # Log score
    mlflow.log_metric("score", grid_search.best_score_)

    # Log model
    mlflow.sklearn.log_model(grid_search.best_estimator_, "model")
```

```
2024/04/24 12:23:33 INFO mlflow.utils.autologging_utils: Created MLflow au
tologging run with ID 'e4e99e1c8b984ac1b30e79a93982054c', which will track
hyperparameters, performance metrics, model artifacts, and lineage informa
tion for the current sklearn workflow
2024/04/24 12:23:33 WARNING mlflow.sklearn: Unrecognized dataset type <cla
ss 'list'>. Dataset logging skipped.
Fitting 5 folds for each of 18 candidates, totalling 90 fits
2024/04/24 12:25:19 INFO mlflow.sklearn.utils: Logging the 5 best runs, 13
runs will be omitted.
```

In [ ]:

In [ ]: