Data Collection and Preprocessing

Project Title : Amazon Instruments Review

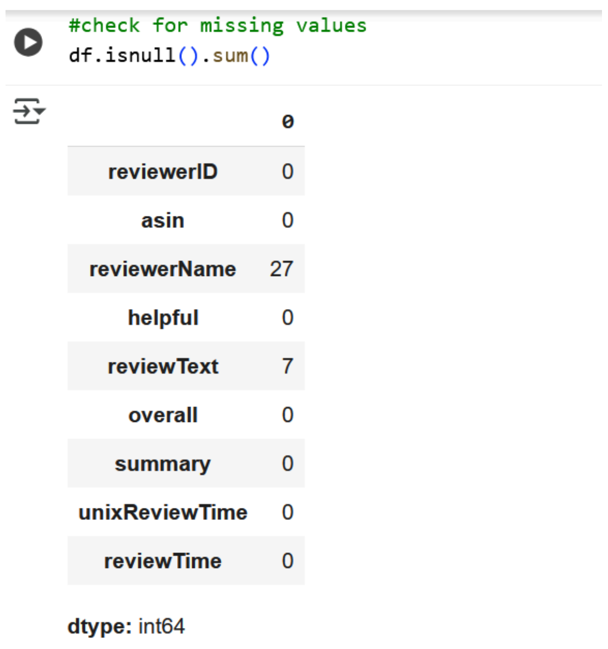
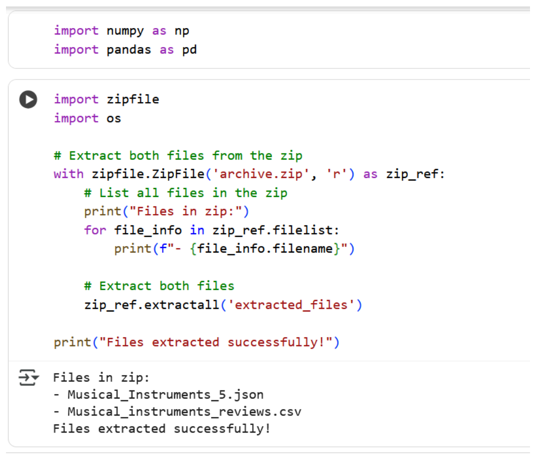
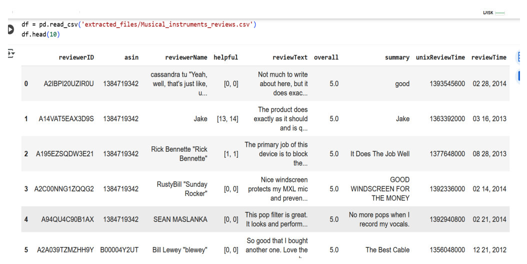
Summarizes the preprocessing steps applied to the Amazon Instrument Reviews dataset to ensure clean and consistent input for machine learning models.

Preprocessing steps :

Section	Description
Data Overview	Loaded CSV dataset, checked missing values, and reviewed class distribution.
Handling Missing Values	Filled missing reviewText with empty strings and reviewerName with “ ".
Concatenation	Combined reviewText and summary into a single column reviews.
Labelling	Assigned labels: Positive (rating > 3), Negative (rating < 3), Neutral (rating = 3).
Text Cleaning	Lowercasing, punctuation removal, number removal, URL removal, newline removal.
Text Preprocessing	Tokenized reviews, removed stopwords, and applied WordNet Lemmatizer.
Polarity Scores	Calculated polarity of reviews using TextBOB for strength
Feature Engineering	Generated features: review length, word count, n-grams and dropped columns.
Encoding	Encoded target sentiment labels using LabelEncoder.
Vectorization	Applied TF-IDF with Unigram,Bigram, Trigram (max 5000 features).

Balancing	Used SMOTE to oversample minority classes.

Data Preprocessing Code Screenshots :

Section	CODE
Data Overvie  W	 

```
#check for missing values
df.isnull().sum()
```

```
                    0
    reviewerID      0
          asin      0
  reviewerName     27
       helpful      0
    reviewText      7
       overall      0
       summary      0
unixReviewTime      0
    reviewTime      0

dtype: int64
```

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

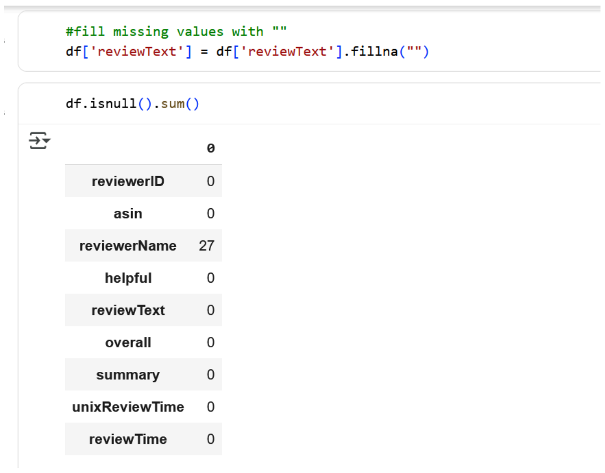
```
import zipfile
import os

# Extract both files from the zip
with zipfile.ZipFile('archive.zip', 'r') as zip_ref:
    # List all files in the zip
    print("Files in zip:")
    for file_info in zip_ref.filelist:
        print(f"- {file_info.filename}")

    # Extract both files
    zip_ref.extractall('extracted_files')

print("Files extracted successfully!")
```

```
Files in zip:
- Musical_Instruments_5.json
- Musical_instruments_reviews.csv
Files extracted successfully!
```

```
df = pd.read_csv('extracted_files/Musical_instruments_reviews.csv')
df.head(10)
```

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	A2IBPI20UZIR0U	1384719342	cassandra tu "Yeah, well, that's just like, u...	[0, 0]	Not much to write about here, but it does exac...	5.0	good	1393545600	02 28, 2014
1	A14VAT5EAX3D9S	1384719342	Jake	[13, 14]	The product does exactly as it should and is q...	5.0	Jake	1363392000	03 16, 2013
2	A195EZSQDW3E21	1384719342	Rick Bennette "Rick Bennette"	[1, 1]	The primary job of this device is to block the...	5.0	It Does The Job Well	1377648000	08 28, 2013
3	A2C00NNG1ZQQG2	1384719342	RustyBill "Sunday Rocker"	[0, 0]	Nice windscreen protects my MXL mic and preven...	5.0	GOOD WINDSCREEN FOR THE MONEY	1392336000	02 14, 2014
4	A94QU4C90B1AX	1384719342	SEAN MASLANKA	[0, 0]	This pop filter is great. It looks and perform...	5.0	No more pops when I record my vocals.	1392940800	02 21, 2014
5	A2A039TZMZHH9Y	B00004Y2UT	Bill Lewey "blewey"	[0, 0]	So good that I bought another one. Love the	5.0	The Best Cable	1356048000	12 21, 2012

Handling missing values

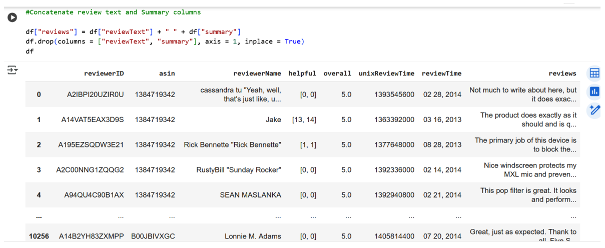


```
#fill missing values with ""
df['reviewText'] = df['reviewText'].fillna("")
```

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

```
                 0
    reviewerID   0
          asin   0
  reviewerName  27
       helpful   0
    reviewText   0
       overall   0
       summary   0
unixReviewTime   0
    reviewTime   0
```

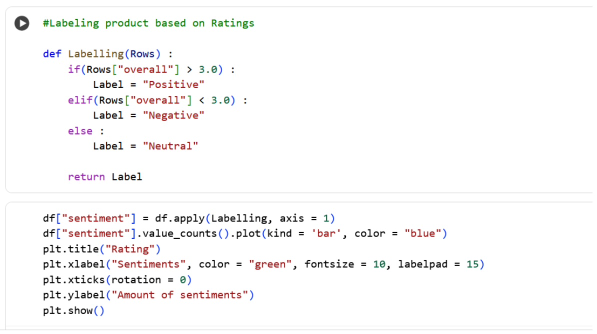
Concatenation



```
#Concatenate review text and Summary columns

df["reviews"] = df["reviewText"] + " " + df["summary"]
df.drop(columns = ["reviewText", "summary"], axis = 1, inplace = True)
df
```

	reviewerID	asin	reviewerName	helpful	overall	unixReviewTime	reviewTime	reviews
0	A2IBPI20UZIR0U	1384719342	cassandra tu "Yeah, well, that's just like, u...	[0, 0]	5.0	1393545600	02 28, 2014	Not much to write about here, but it does exac...
1	A14VAT5EAX3D9S	1384719342	Jake	[13, 14]	5.0	1363392000	03 16, 2013	The product does exactly as it should and is q...
2	A195EZSQDW3E21	1384719342	Rick Bennette "Rick Bennette"	[1, 1]	5.0	1377648000	08 28, 2013	The primary job of this device is to block the...
3	A2C00NNG1ZQQG2	1384719342	RustyBill "Sunday Rocker"	[0, 0]	5.0	1392336000	02 14, 2014	Nice windscreen protects my MXL mic and preven...
4	A94QU4C90B1AX	1384719342	SEAN MASLANKA	[0, 0]	5.0	1392940800	02 21, 2014	This pop filter is great. It looks and perform...
...	...	...	...	...	...	...	...	...
10256	A14B2YH83ZXMPP	B00JBIVXGC	Lonnie M. Adams	[0, 0]	5.0	1405814400	07 20, 2014	Great, just as expected. Thank to

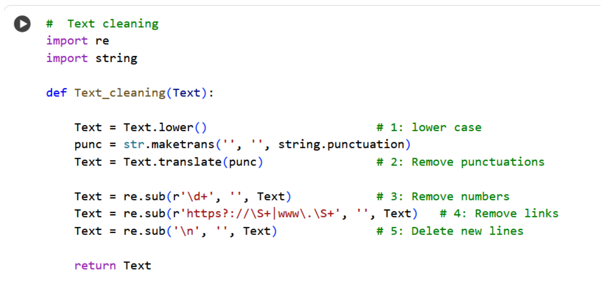
Labelling



```
#Labeling product based on Ratings

def Labelling(Rows) :
    if(Rows["overall"] > 3.0) :
        Label = "Positive"
    elif(Rows["overall"] < 3.0) :
        Label = "Negative"
    else :
        Label = "Neutral"

    return Label
```

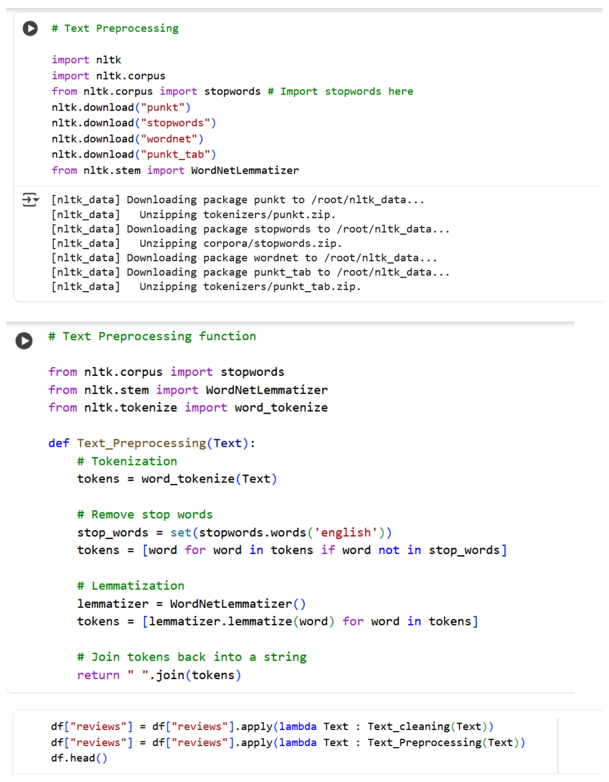
```
df["sentiment"] = df.apply(Labelling, axis = 1)
df["sentiment"].value_counts().plot(kind = 'bar', color = "blue")
plt.title("Rating")
plt.xlabel("Sentiments", color = "green", fontsize = 10, labelpad = 15)
plt.xticks(rotation = 0)
plt.ylabel("Amount of sentiments")
plt.show()
```

Text Cleaning



```
#  Text cleaning
import re
import string

def Text_cleaning(Text):

    Text = Text.lower()                                 # 1: lower case
    punc = str.maketrans('', '', string.punctuation)
    Text = Text.translate(punc)                         # 2: Remove punctuations

    Text = re.sub(r'\d+', '', Text)                     # 3: Remove numbers
    Text = re.sub(r'https?://\S+|www\.\S+', '', Text)   # 4: Remove links
    Text = re.sub('\n', '', Text)                       # 5: Delete new lines

    return Text
```

Text Preprocessing



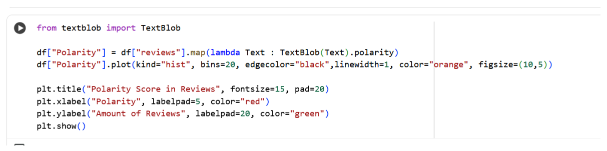
```
# Text Preprocessing

import nltk
import nltk.corpus
from nltk.corpus import stopwords # Import stopwords here
nltk.download("punkt")
nltk.download("stopwords")
nltk.download("wordnet")
nltk.download("punkt_tab")
from nltk.stem import WordNetLemmatizer
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
```

```
# Text Preprocessing function

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

def Text_Preprocessing(Text):
    # Tokenization
    tokens = word_tokenize(Text)

    # Remove stop words
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    # Join tokens back into a string
    return " ".join(tokens)
```

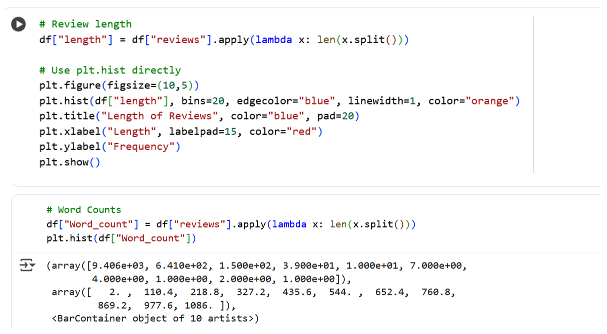
```
df["reviews"] = df["reviews"].apply(lambda Text : Text_cleaning(Text))
df["reviews"] = df["reviews"].apply(lambda Text : Text_Preprocessing(Text))
df.head()
```

Polarity Scores



```
from textblob import TextBlob

df["Polarity"] = df["reviews"].map(lambda Text : TextBlob(Text).polarity)
df["Polarity"].plot(kind="hist", bins=20, edgecolor="black",linewidth=1, color="orange", figsize=(10,5))

plt.title("Polarity Score in Reviews", fontsize=15, pad=20)
plt.xlabel("Polarity", labelpad=5, color="red")
plt.ylabel("Amount of Reviews", labelpad=20, color="green")
plt.show()
```

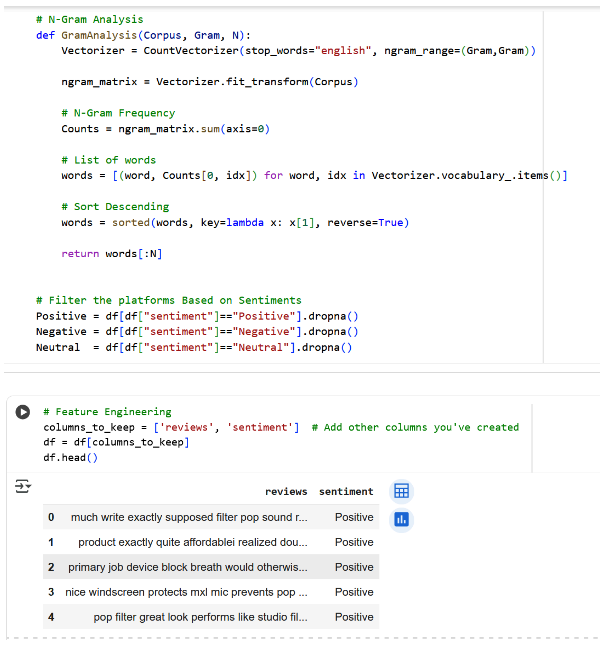
Feature Engineering



```
# Review length
df["length"] = df["reviews"].apply(lambda x: len(x.split()))

# Use plt.hist directly
plt.figure(figsize=(10,5))
plt.hist(df["length"], bins=20, edgecolor="blue", linewidth=1, color="orange")
plt.title("Length of Reviews", color="blue", pad=20)
plt.xlabel("Length", labelpad=15, color="red")
plt.ylabel("Frequency")
plt.show()
```

```
# Word Counts
df["Word_count"] = df["reviews"].apply(lambda x: len(x.split()))
plt.hist(df["Word_count"])
```

```
(array([9.406e+03, 6.410e+02, 1.500e+02, 3.900e+01, 1.000e+01, 7.000e+00,
        4.000e+00, 1.000e+00, 2.000e+00, 1.000e+00]),
 array([   2. ,  110.4,  218.8,  327.2,  435.6,  544. ,  652.4,  760.8,
         869.2,  977.6, 1086. ]),
 <BarContainer object of 10 artists>)
```

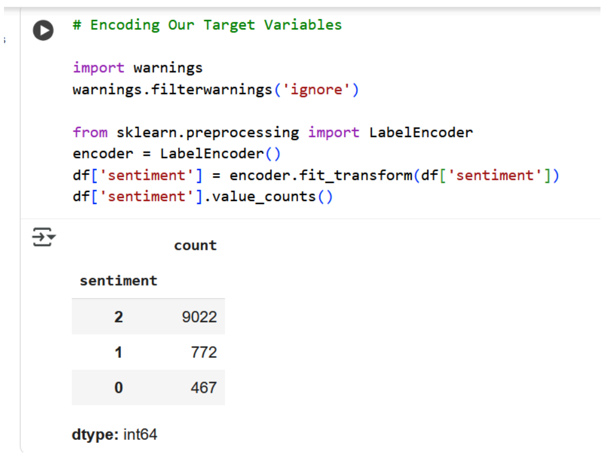


```
# N-Gram Analysis
def GramAnalysis(Corpus, Gram, N):
    Vectorizer = CountVectorizer(stop_words="english", ngram_range=(Gram,Gram))

    ngram_matrix = Vectorizer.fit_transform(Corpus)

    # N-Gram Frequency
    Counts = ngram_matrix.sum(axis=0)

    # List of words
    words = [(word, Counts[0, idx]) for word, idx in Vectorizer.vocabulary_.items()]

    # Sort Descending
    words = sorted(words, key=lambda x: x[1], reverse=True)

    return words[:N]


# Filter the platforms Based on Sentiments
Positive = df[df["sentiment"]=="Positive"].dropna()
Negative = df[df["sentiment"]=="Negative"].dropna()
Neutral  = df[df["sentiment"]=="Neutral"].dropna()
```

```
# Feature Engineering
columns_to_keep = ['reviews', 'sentiment']  # Add other columns you've created
df = df[columns_to_keep]
df.head()
```

	reviews	sentiment
0	much write exactly supposed filter pop sound r...	Positive
1	product exactly quite affordablei realized dou...	Positive
2	primary job device block breath would otherwis...	Positive
3	nice windscreen protects mxl mic prevents pop ...	Positive
4	pop filter great look performs like studio fil...	Positive

Encoding

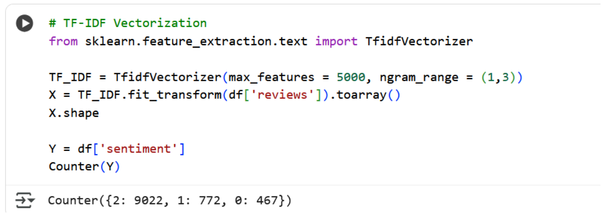


```
# Encoding Our Target Variables

import warnings
warnings.filterwarnings('ignore')

from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df['sentiment'] = encoder.fit_transform(df['sentiment'])
df['sentiment'].value_counts()
```

	count
sentiment	
2	9022
1	772
0	467

dtype: int64

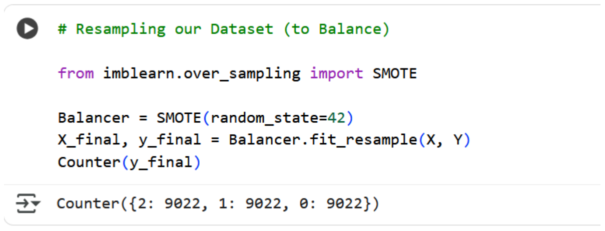
Vectorization



```
# TF-IDF Vectorization
from sklearn.feature_extraction.text import TfidfVectorizer

TF_IDF = TfidfVectorizer(max_features = 5000, ngram_range = (1,3))
X = TF_IDF.fit_transform(df['reviews']).toarray()
X.shape

Y = df['sentiment']
Counter(Y)
```

```
Counter({2: 9022, 1: 772, 0: 467})
```

Balancing



```
# Resampling our Dataset (to Balance)

from imblearn.over_sampling import SMOTE

Balancer = SMOTE(random_state=42)
X_final, y_final = Balancer.fit_resample(X, Y)
Counter(y_final)
```

```
Counter({2: 9022, 1: 9022, 0: 9022})
```