INTERN PROJECT PHASE – 1

Project - 2

Data Science Project: Sentiment Analysis
Project Title: Sentiment Analysis
Dataset:
https://www.kaggle.com/datasets/abhi8923shriv/sentiment-analysis-dataset/data

Project Overview:
Sentiment Analysis is a data science project that involves the use of machine learning techniques to analyze and classify textual data based on the sentiment expressed. The project aims to build a predictive model capable of determining whether a given text conveys positive, negative, or neutral sentiment.

Project Objectives:
1. Data Exploration:
- Explore the Sentiment Analysis dataset to understand its structure, features, and size.
- Identify key variables such as text content and sentiment labels.

2. Data Preprocessing:
- Perform text preprocessing tasks, including lowercasing, removing stopwords, and handling special characters.
- Tokenize and lemmatize words to prepare the text for sentiment analysis.

3. Exploratory Data Analysis (EDA):
- Conduct exploratory data analysis to gain insights into the distribution of sentiment labels.
- Visualize the distribution using histograms or pie charts to understand the balance of sentiment classes.

4. Text Vectorization:
- Convert the preprocessed text into numerical vectors using techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings.
- Choose an appropriate vectorization method based on the characteristics of the dataset.

5. Model Selection:

- Explore and implement different machine learning models suitable for text classification, such as Naive Bayes, Support Vector Machines, or deep learning models like LSTM (Long Short-Term Memory) networks.
- Evaluate the performance of each model using metrics like accuracy, precision, recall, and F1 score.

6. Hyperparameter Tuning:

- Fine-tune the hyperparameters of the selected model to optimize its performance.
- Utilize techniques like grid search or random search for hyperparameter optimization.

7. Cross-Validation:

- Implement cross-validation techniques to assess the generalization performance of the model and prevent overfitting.

8. Model Interpretability:

- Interpret the model's predictions by analyzing feature importance or using techniques like LIME (Local Interpretable Model-agnostic Explanations).
- Understand which words or features contribute most to sentiment predictions.

9. Evaluation Metrics:

- Evaluate the model's performance using relevant evaluation metrics for sentiment analysis, such as confusion matrix, precision-recall curves, and ROC-AUC.

10. Deployment (Optional):

- Deploy the trained model for real-time sentiment analysis, creating an API or integrating it into a web application.
- Showcase the model's functionality in a user-friendly interface.

11. Documentation:

- Create comprehensive documentation covering data preprocessing steps, model development, and evaluation results.
- Include code snippets, visualizations, and explanations to aid understanding.

1. Introduction

Sentiment Analysis is a natural language processing (NLP) task aimed at determining the sentiment expressed in a piece of text. This project focuses on developing a sentiment analysis model using a dataset containing text and corresponding sentiment labels.

2. Data Preprocessing

2.1. Load Dataset : Load the dataset into a Pandas DataFrame. Ensure correct encoding and file format.

2.2. Explore Dataset : Explore the dataset to understand its structure and characteristics. Print basic information such as column names, data types, and the first few rows.

2.3. Identify Key Variables : Identify and explain key variables like 'selected_text' (text content) and 'sentiment' labels.

2.4. Data Cleaning and Preprocessing : Perform essential data cleaning steps. Lowercase the text, remove stop words, handle special characters, tokenize the text, and lemmatize the words.

3. Exploratory Data Analysis (EDA)

Visualize the distribution of sentiment labels in the dataset using count plots. This step provides insights into the balance of sentiment classes.

4. Text Vectorization

4.1. Using TF-IDF Vectorizer : Vectorize the text data using the Term Frequency-Inverse Document Frequency (TF-IDF) vectorization technique. This step converts text into numerical features while considering the importance of words.

5. Model Development

5.1. Support Vector Machines (SVM) : Train a Support Vector Machines (SVM) model using the TF-IDF vectorized text data. SVM is a supervised machine learning algorithm used for classification tasks.

5.2. Long Short-Term Memory (LSTM) : Train a Long Short-Term Memory (LSTM) neural network model. LSTMs are a type of recurrent neural network (RNN) often used for sequence modeling tasks like sentiment analysis.

6. Model Evaluation

6.1. Support Vector Machines (SVM) : Evaluate the SVM model's performance using standard classification metrics such as accuracy, precision, recall, F1 score, and the classification report.

6.2. Long Short-Term Memory (LSTM) : Evaluate the LSTM model's performance using appropriate metrics. Adjust the evaluation based on the specific requirements of the sentiment analysis task.

7. Hyperparameter Tuning

Tune hyperparameters of the SVM model using grid search. This involves searching through a predefined parameter grid to find the combination that maximizes performance.

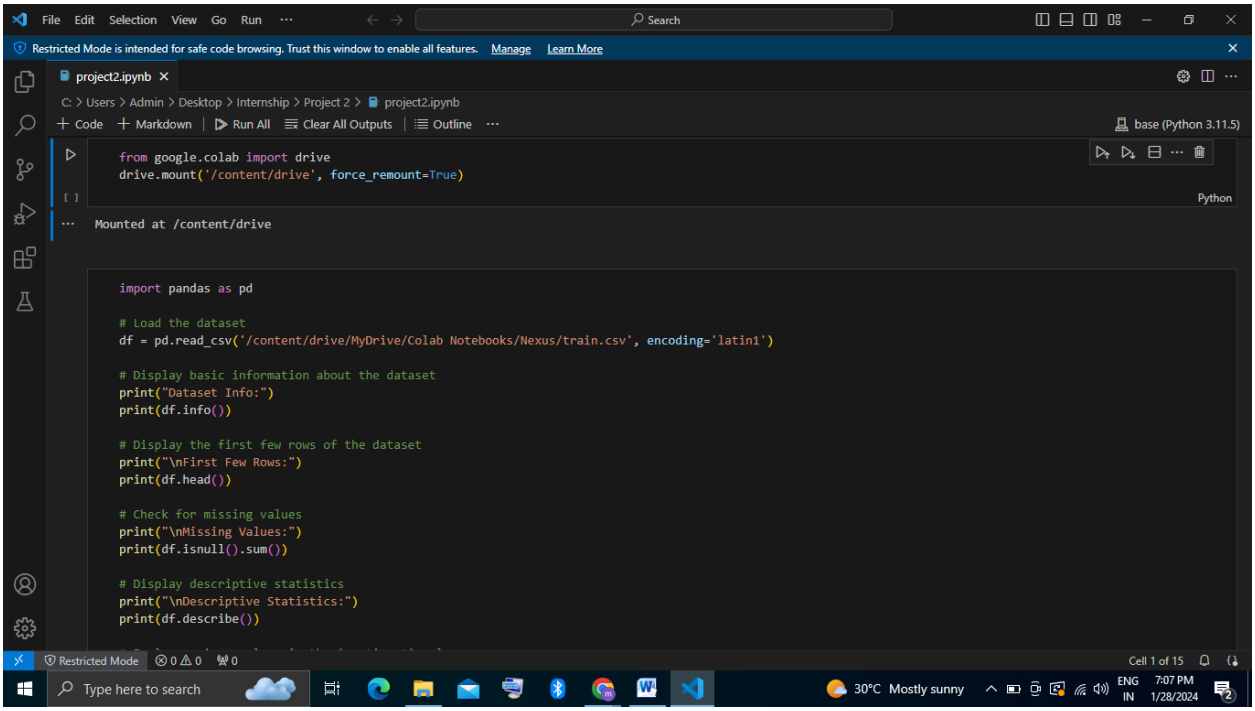
8. Cross-Validation

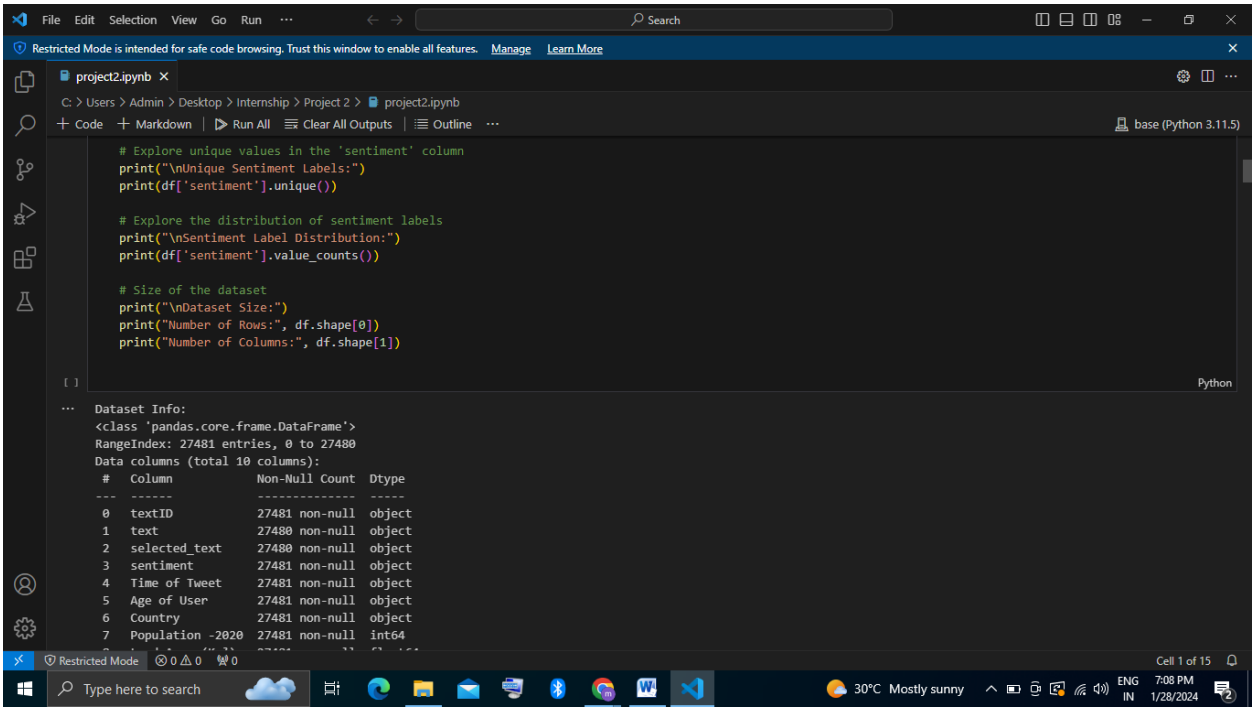
Implement cross-validation techniques, such as Stratified K-Fold, to assess the generalization performance of the LSTM model and prevent overfitting.

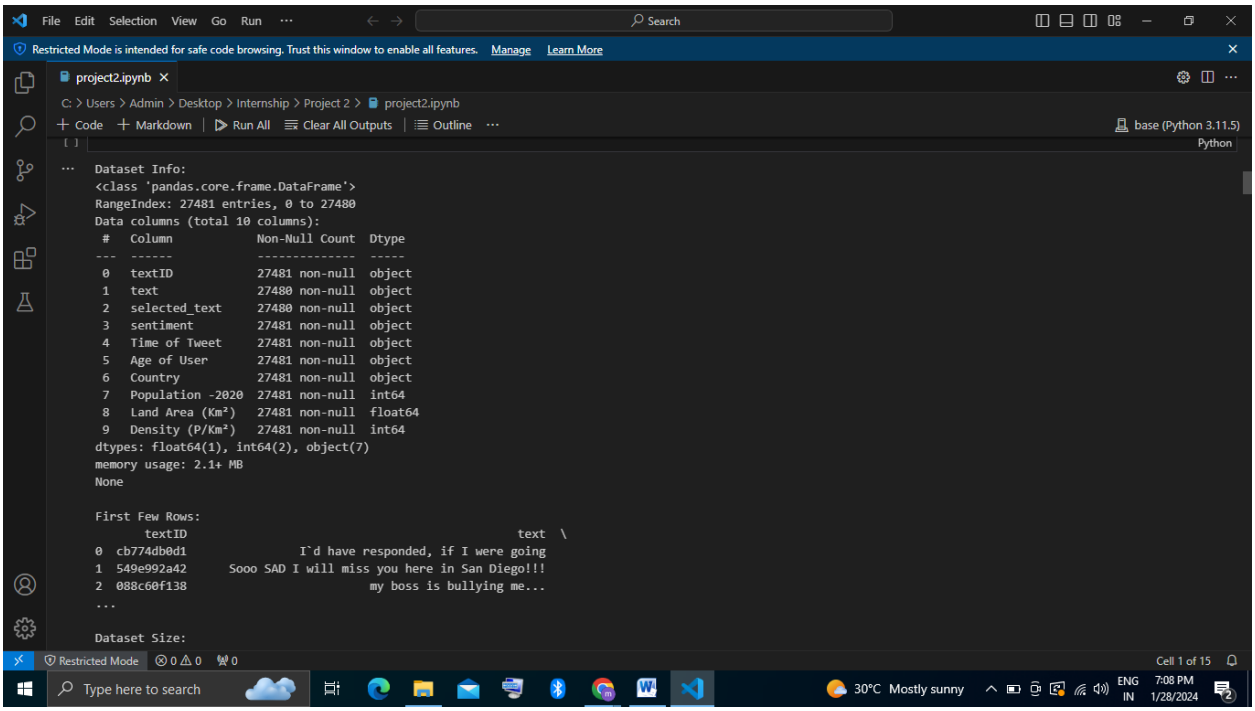
9. Model Interpretability

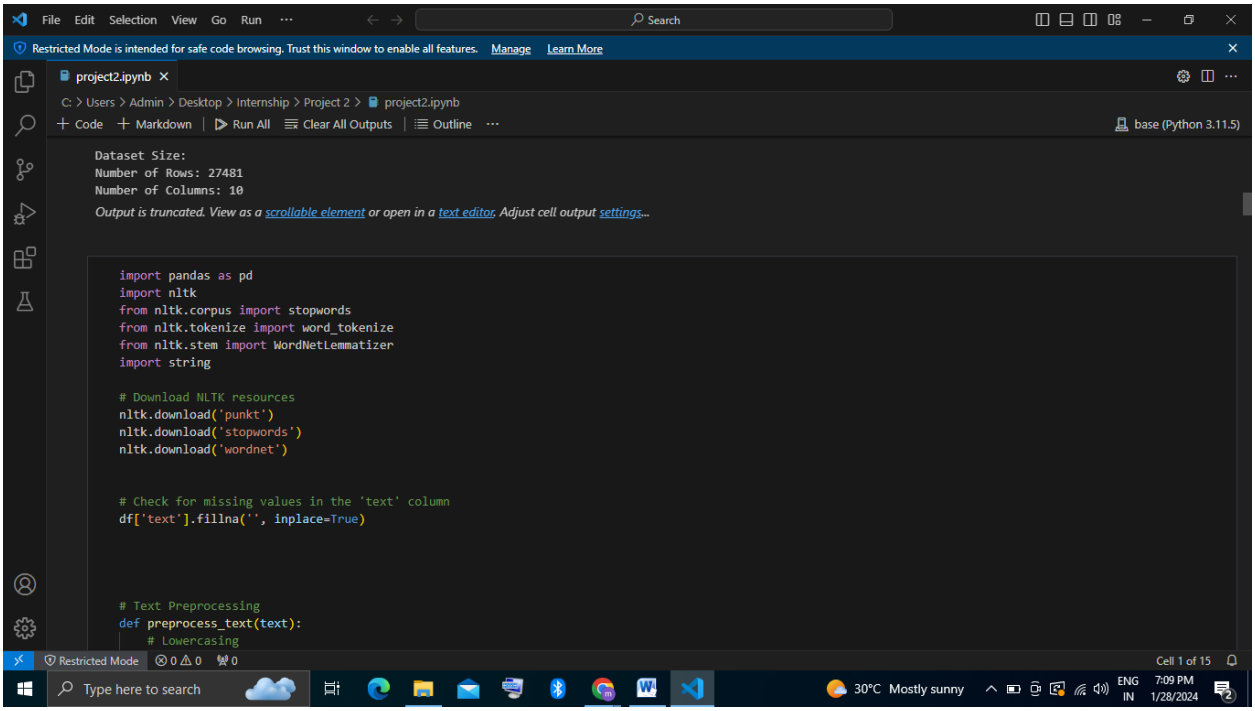
Utilize techniques like LIME (Local Interpretable Model-agnostic Explanations) to interpret the model's predictions. Understand which words or features contribute most to sentiment predictions.

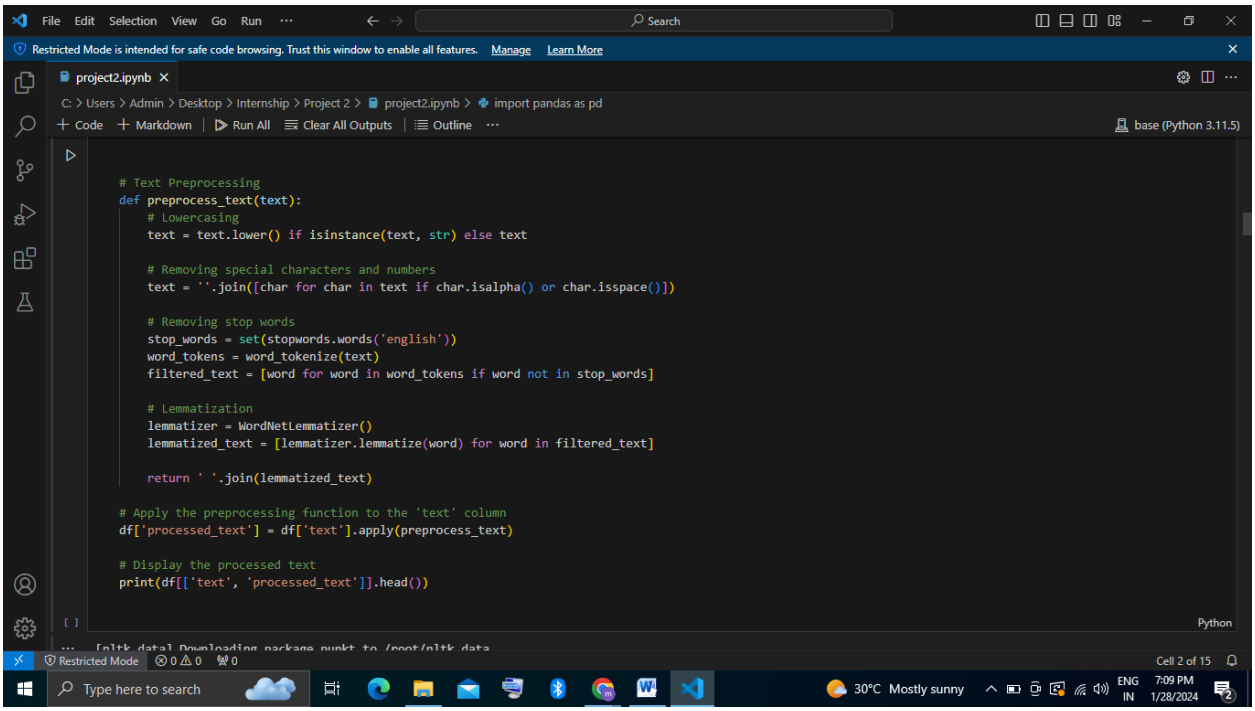
Output :

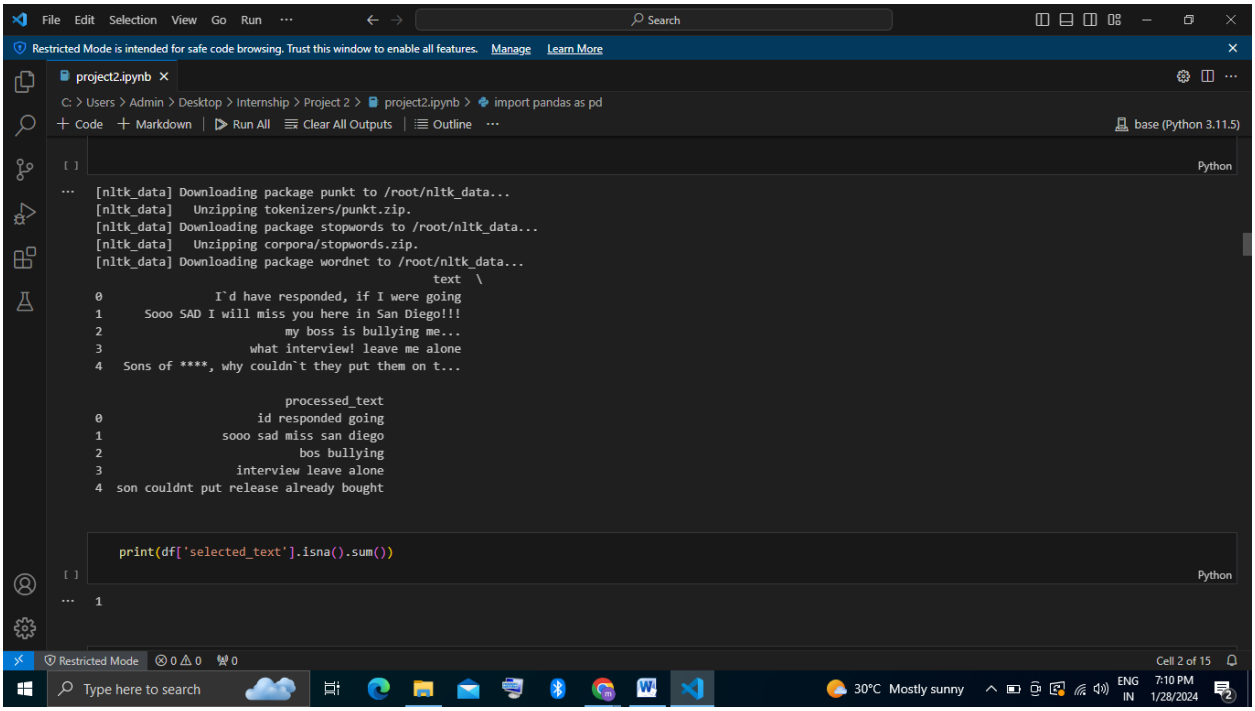


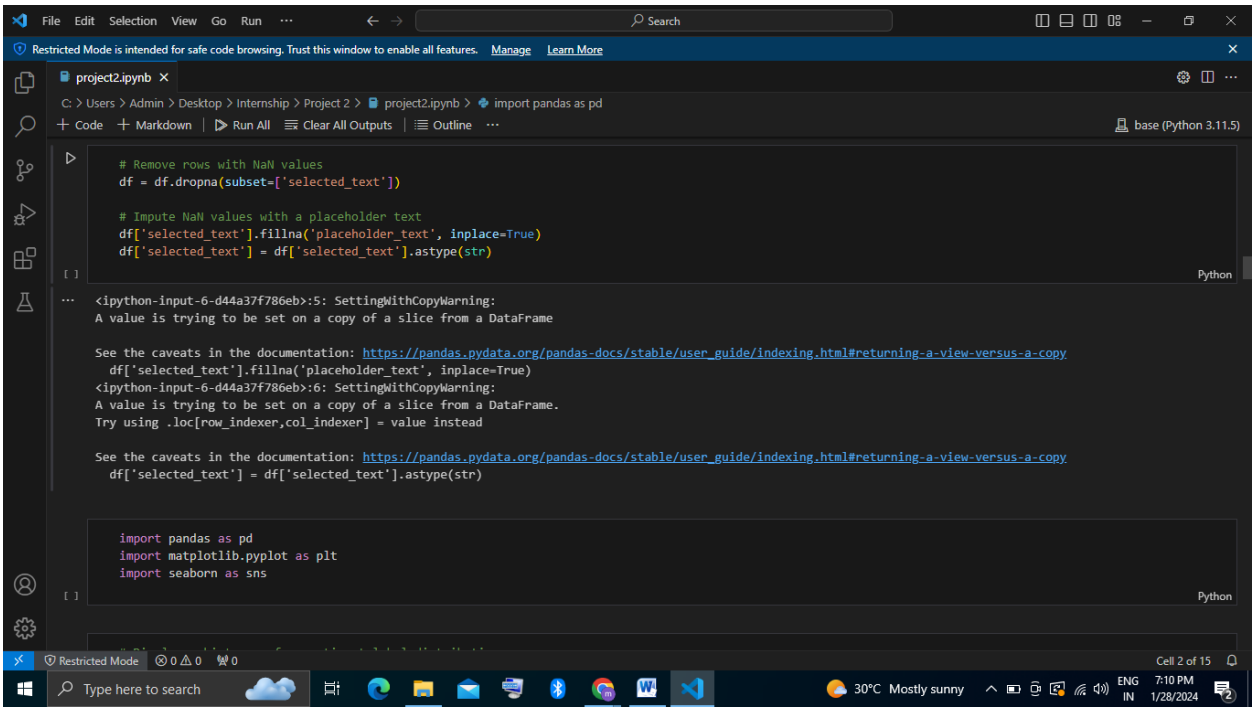

File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
project2.ipynb
C: > Users > Admin > Desktop > Internship > Project 2 > project2.ipynb
+ Code + Markdown Run All Clear All Outputs Outline
base (Python 3.11.5)
# Explore unique values in the 'sentiment' column
print("\nUnique Sentiment Labels:")
print(df['sentiment'].unique())
# Explore the distribution of sentiment labels
print("\nSentiment Label Distribution:")
print(df['sentiment'].value_counts())
# Size of the dataset
print("\nDataset Size:")
print("Number of Rows:", df.shape[0])
print("Number of Columns:", df.shape[1])
Python
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27481 entries, 0 to 27480
Data columns (total 10 columns):
Column Non-Null Count Dtype
--- ------ -------------- -----
0 textID 27481 non-null object
1 text 27480 non-null object
2 selected_text 27480 non-null object
3 sentiment 27481 non-null object
4 Time of Tweet 27481 non-null object
5 Age of User 27481 non-null object
6 Country 27481 non-null object
7 Population -2020 27481 non-null int64
Restricted Mode
Cell 1 of 15
Type here to search
30°C Mostly sunny
ENG IN
7:08 PM 1/28/2024

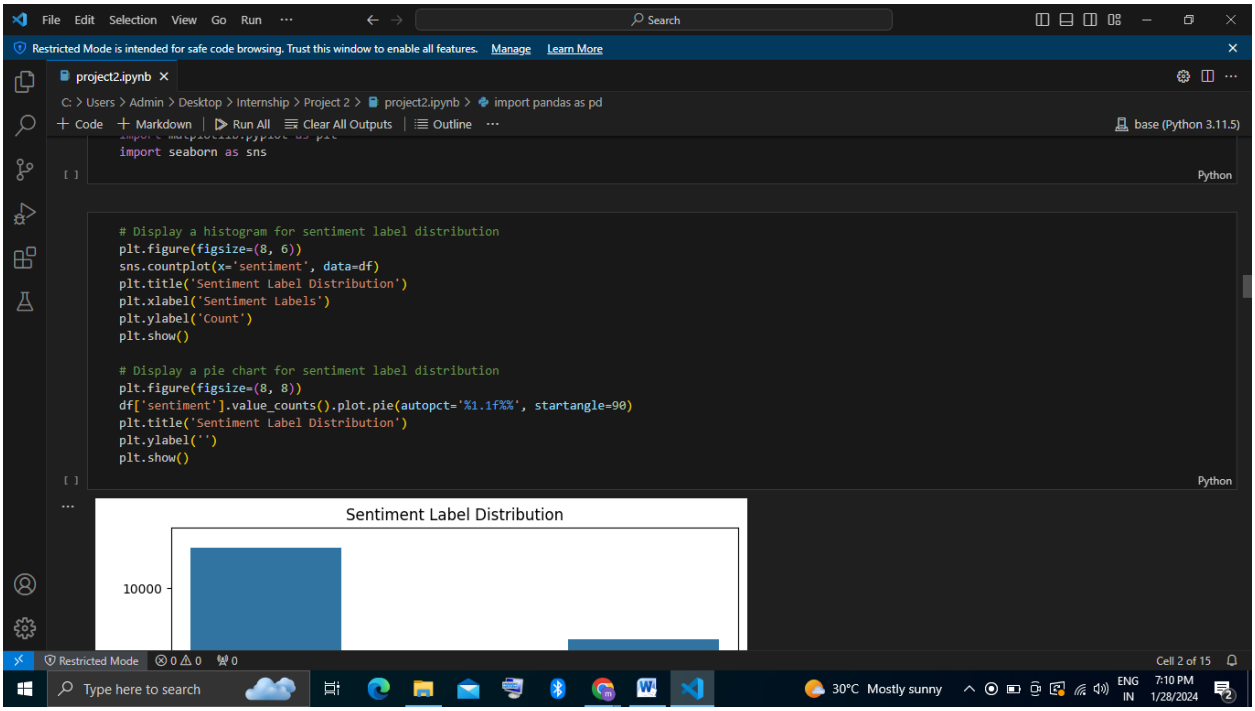

File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
project2.ipynb
C: > Users > Admin > Desktop > Internship > Project 2 > project2.ipynb
+ Code + Markdown Run All Clear All Outputs Outline
base (Python 3.11.5)
Python
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27481 entries, 0 to 27480
Data columns (total 10 columns):
Column Non-Null Count Dtype
--- ------ -------------- -----
0 textID 27481 non-null object
1 text 27480 non-null object
2 selected_text 27480 non-null object
3 sentiment 27481 non-null object
4 Time of Tweet 27481 non-null object
5 Age of User 27481 non-null object
6 Country 27481 non-null object
7 Population -2020 27481 non-null int64
8 Land Area (Km²) 27481 non-null float64
9 Density (P/Km²) 27481 non-null int64
dtypes: float64(1), int64(2), object(7)
memory usage: 2.1+ MB
None
First Few Rows:
textID text \
0 cb774db0d1 I`d have responded, if I were going
1 549e992a42 Sooo SAD I will miss you here in San Diego!!!
2 088c60f138 my boss is bullying me...
...
Dataset Size:
Restricted Mode
Cell 1 of 15
Type here to search
30°C Mostly sunny
ENG IN
7:08 PM 1/28/2024

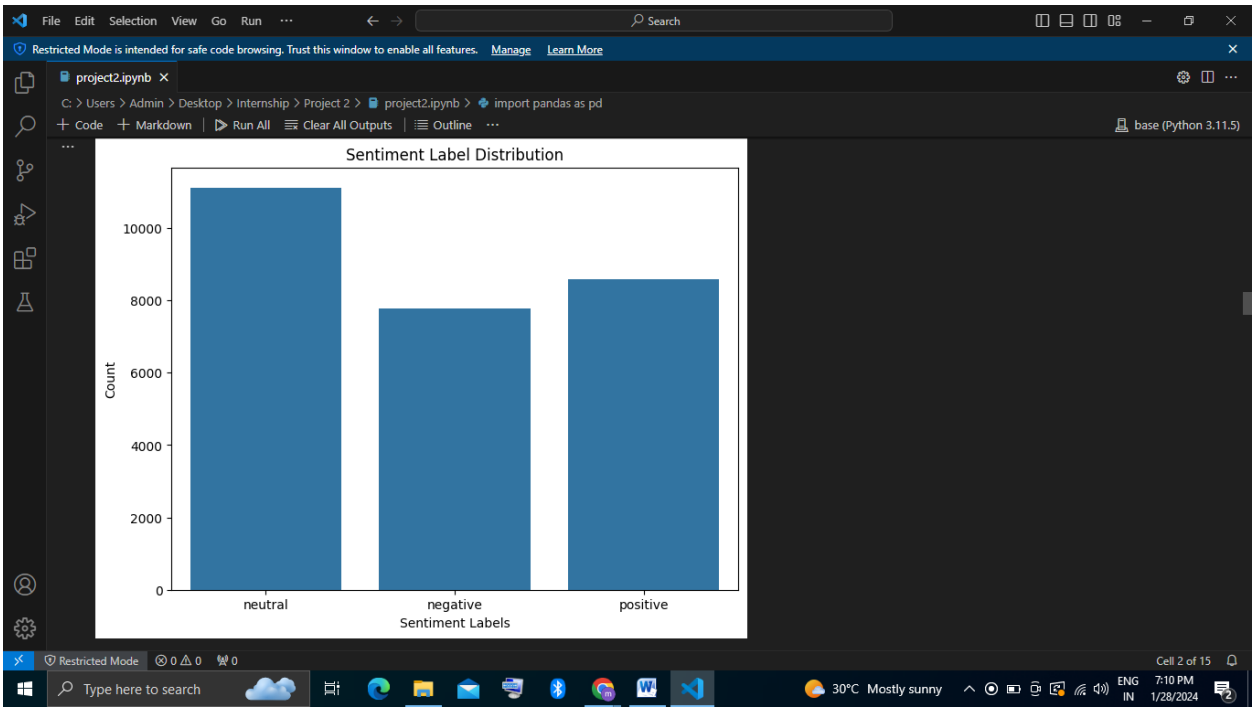

File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
project2.ipynb
C: > Users > Admin > Desktop > Internship > Project 2 > project2.ipynb
Code Markdown Run All Clear All Outputs Outline
base (Python 3.11.5)
Dataset Size:
Number of Rows: 27481
Number of Columns: 10
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import string
# Download NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
# Check for missing values in the 'text' column
df['text'].fillna('', inplace=True)
# Text Preprocessing
def preprocess_text(text):
# Lowercasing
Restricted Mode
Cell 1 of 15
Type here to search
30°C Mostly sunny
ENG IN
7:09 PM 1/28/2024

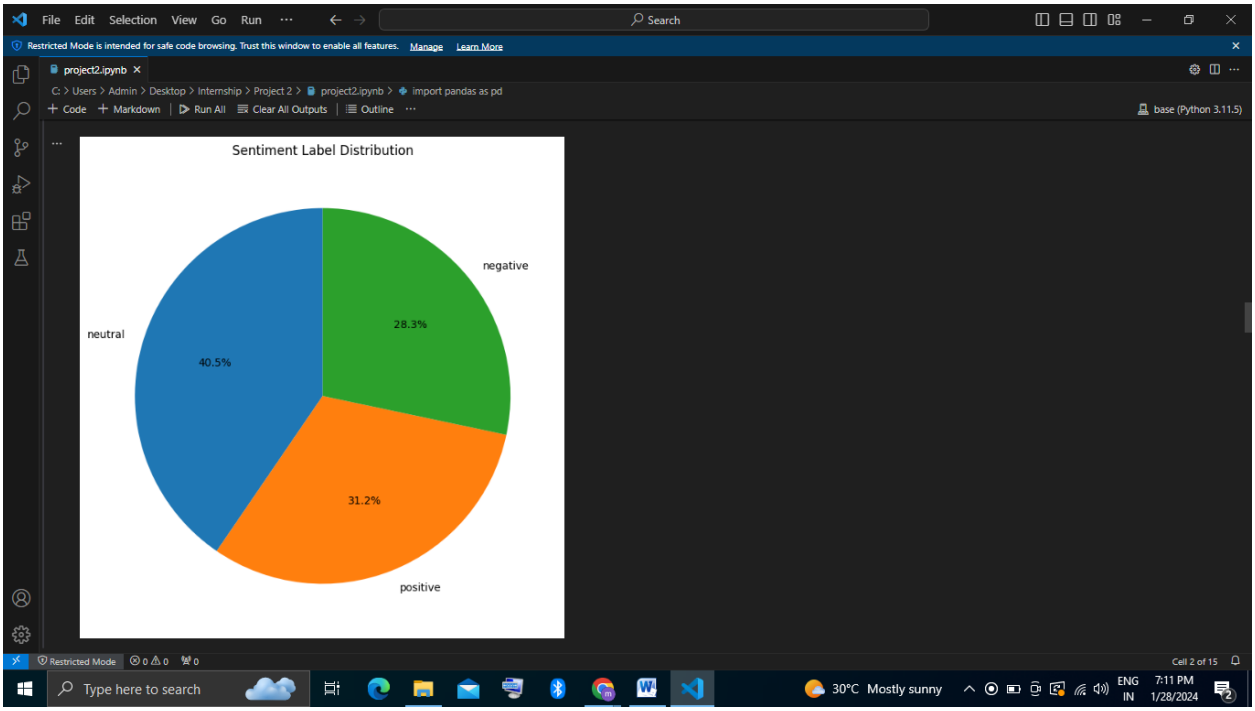

File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
project2.ipynb
C: > Users > Admin > Desktop > Internship > Project 2 > project2.ipynb > import pandas as pd
Code Markdown Run All Clear All Outputs Outline
base (Python 3.11.5)
# Text Preprocessing
def preprocess_text(text):
# Lowercasing
text = text.lower() if isinstance(text, str) else text
# Removing special characters and numbers
text = ''.join([char for char in text if char.isalpha() or char.isspace()])
# Removing stop words
stop_words = set(stopwords.words('english'))
word_tokens = word_tokenize(text)
filtered_text = [word for word in word_tokens if word not in stop_words]
# Lemmatization
lemmatizer = WordNetLemmatizer()
lemmatized_text = [lemmatizer.lemmatize(word) for word in filtered_text]
return ' '.join(lemmatized_text)
Apply the preprocessing function to the 'text' column
df['processed_text'] = df['text'].apply(preprocess_text)
# Display the processed text
print(df[['text', 'processed_text']].head())
[ ]
Python
[nltk_data] Downloading package punkt to /root/nltk_data
Restricted Mode
Cell 2 of 15
Type here to search
30°C Mostly sunny
ENG IN
7:09 PM 1/28/2024

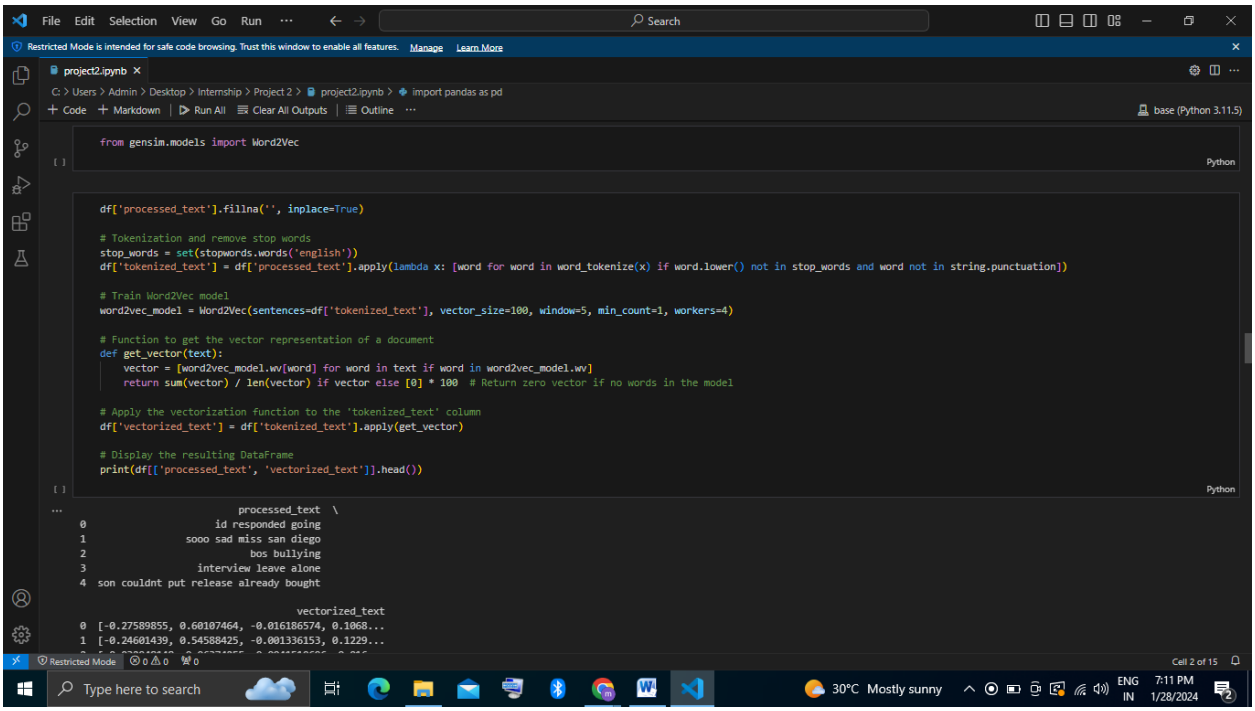

File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
project2.ipynb
C: > Users > Admin > Desktop > Internship > Project 2 > project2.ipynb > import pandas as pd
Code Markdown Run All Clear All Outputs Outline
base (Python 3.11.5)
Python
[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...
text \
0 I`d have responded, if I were going
1 Sooo SAD I will miss you here in San Diego!!!
2 my boss is bullying me...
3 what interview! leave me alone
4 Sons of ****, why couldn`t they put them on t...
processed_text
0 id responded going
1 sooo sad miss san diego
2 bos bullying
3 interview leave alone
4 son couldnt put release already bought
print(df['selected_text'].isna().sum())
Python
1
Restricted Mode
Cell 2 of 15
Type here to search
30°C Mostly sunny
ENG IN
7:10 PM 1/28/2024

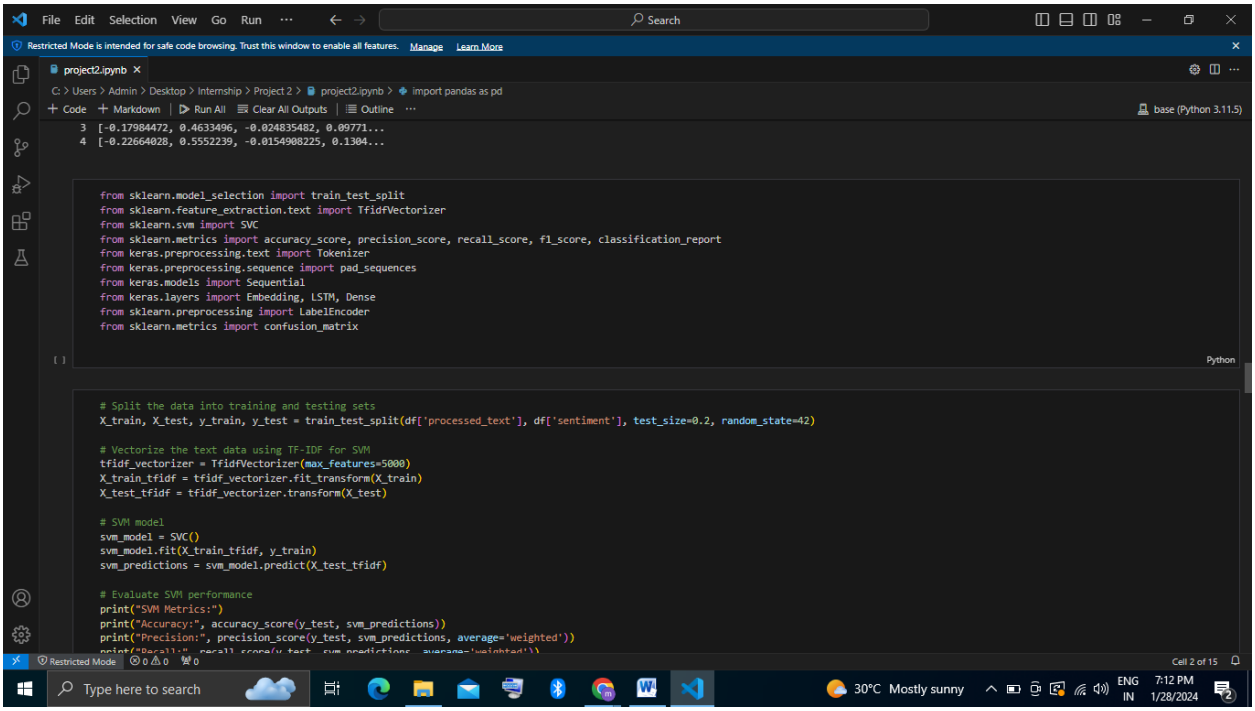

File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
project2.ipynb
C: > Users > Admin > Desktop > Internship > Project 2 > project2.ipynb > import pandas as pd
Code Markdown Run All Clear All Outputs Outline
base (Python 3.11.5)
# Remove rows with NaN values
df = df.dropna(subset=['selected_text'])
# Impute NaN values with a placeholder text
df['selected_text'].fillna('placeholder_text', inplace=True)
df['selected_text'] = df['selected_text'].astype(str)
Python
<ipython-input-6-d44a37f786eb>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df['selected_text'].fillna('placeholder_text', inplace=True)
<ipython-input-6-d44a37f786eb>:6: SettingWithCopyWarning:
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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df['selected_text'] = df['selected_text'].astype(str)
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
Python
Restricted Mode
Cell 2 of 15
Type here to search
30°C Mostly sunny
ENG IN
7:10 PM 1/28/2024

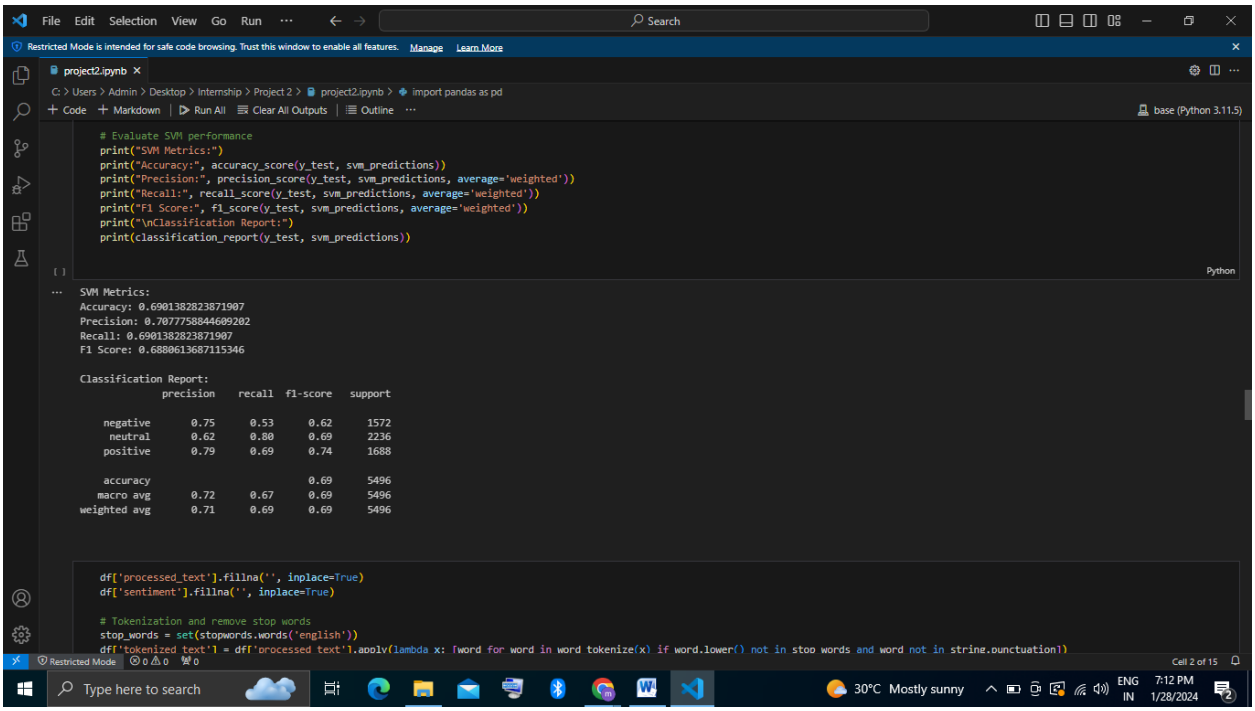

File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
project2.ipynb
C: > Users > Admin > Desktop > Internship > Project 2 > project2.ipynb > import pandas as pd
Code Markdown Run All Clear All Outputs Outline
base (Python 3.11.5)
import seaborn as sns
Python
# Display a histogram for sentiment label distribution
plt.figure(figsize=(8, 6))
sns.countplot(x='sentiment', data=df)
plt.title('Sentiment Label Distribution')
plt.xlabel('Sentiment Labels')
plt.ylabel('Count')
plt.show()
# Display a pie chart for sentiment label distribution
plt.figure(figsize=(8, 8))
df['sentiment'].value_counts().plot.pie(autopct='%1.1f%%', startangle=90)
plt.title('Sentiment Label Distribution')
plt.ylabel('')
plt.show()
Python
Sentiment Label Distribution
10000
Restricted Mode
Cell 2 of 15
Type here to search
30°C Mostly sunny
ENG IN
7:10 PM 1/28/2024

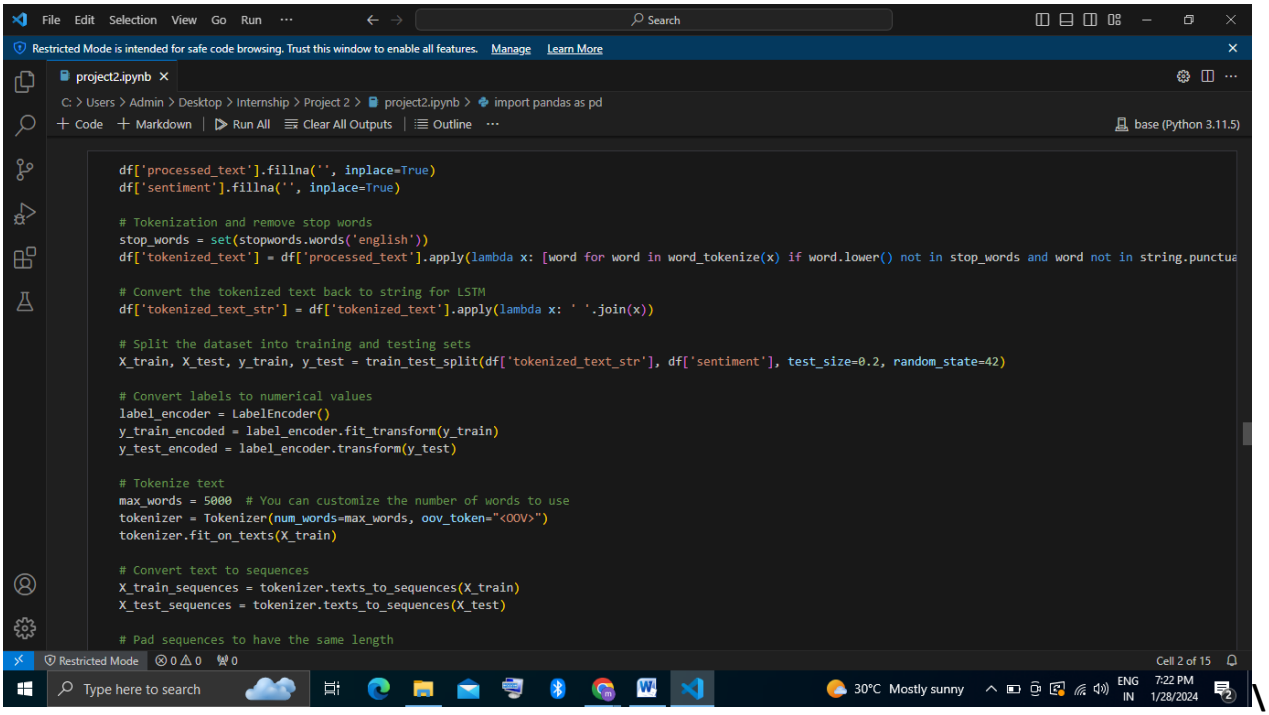

File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
project2.ipynb
C: > Users > Admin > Desktop > Internship > Project 2 > project2.ipynb > import pandas as pd
Code Markdown Run All Clear All Outputs Outline
base (Python 3.11.5)
Sentiment Label Distribution
10000
8000
6000
4000
2000
0
Count
neutral
negative
positive
Sentiment Labels
Restricted Mode
Cell 2 of 15
Type here to search
30°C Mostly sunny
ENG IN
7:10 PM 1/28/2024


File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
project2.ipynb
C: > Users > Admin > Desktop > Internship > Project 2 > project2.ipynb > import pandas as pd
+ Code + Markdown Run All Clear All Outputs Outline
base (Python 3.11.5)
Sentiment Label Distribution
negative
neutral
28.3%
40.5%
31.2%
positive
Restricted Mode
Cell 2 of 15
Type here to search
30°C Mostly sunny
ENG IN
7:11 PM 1/28/2024

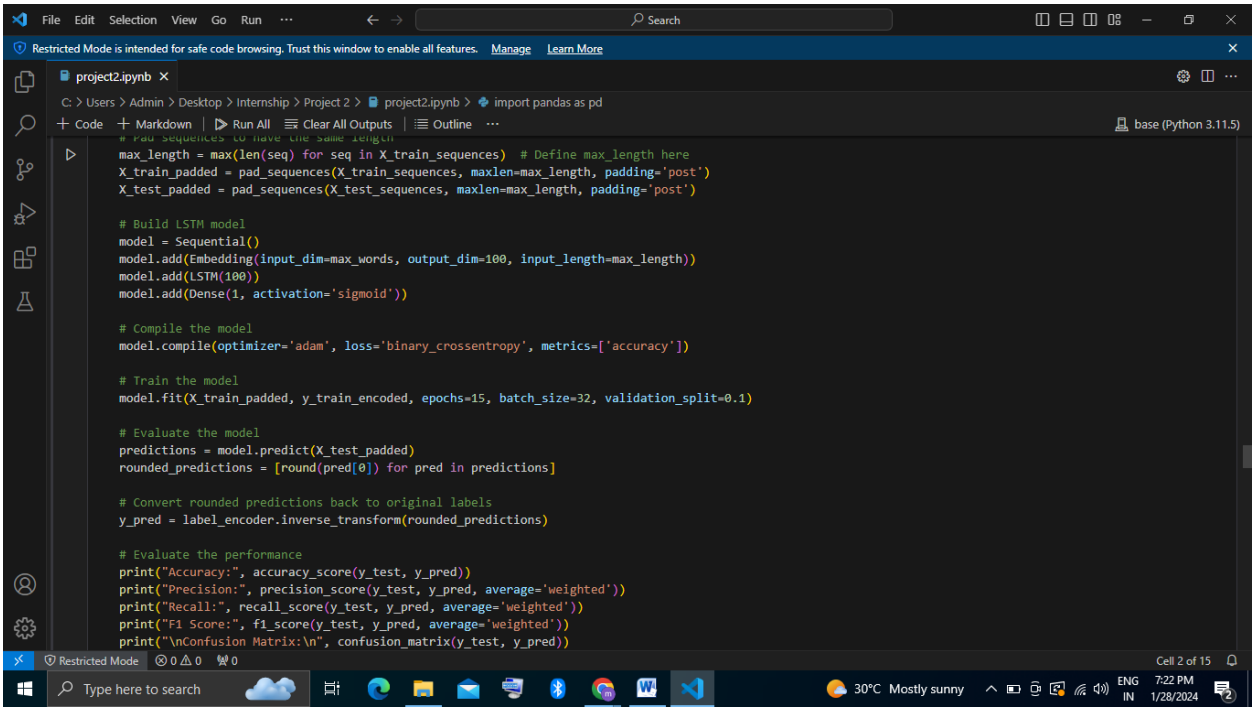

File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
project2.ipynb
C: > Users > Admin > Desktop > Internship > Project 2 > project2.ipynb > import pandas as pd
+ Code + Markdown Run All Clear All Outputs Outline
base (Python 3.11.5)
from gensim.models import Word2Vec
Python
df['processed_text'].fillna('', inplace=True)
# Tokenization and remove stop words
stop_words = set(stopwords.words('english'))
df['tokenized_text'] = df['processed_text'].apply(lambda x: [word for word in word_tokenize(x) if word.lower() not in stop_words and word not in string.punctuation])
# Train Word2Vec model
word2vec_model = Word2Vec(sentences=df['tokenized_text'], vector_size=100, window=5, min_count=1, workers=4)
# Function to get the vector representation of a document
def get_vector(text):
vector = [word2vec_model.wv[word] for word in text if word in word2vec_model.wv]
return sum(vector) / len(vector) if vector else [0] * 100 # Return zero vector if no words in the model
Apply the vectorization function to the 'tokenized_text' column
df['vectorized_text'] = df['tokenized_text'].apply(get_vector)
# Display the resulting DataFrame
print(df[['processed_text', 'vectorized_text']].head())
Python
processed_text \
0 id responded going
1 sooo sad miss san diego
2 bos bullying
3 interview leave alone
4 son couldnt put release already bought
vectorized_text
0 [-0.27589855, 0.60107464, -0.016186574, 0.1068...
1 [-0.24601439, 0.54588425, -0.001336153, 0.1229...
Restricted Mode
Cell 2 of 15
Type here to search
30°C Mostly sunny
ENG IN
7:11 PM 1/28/2024


File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
project2.ipynb
C: > Users > Admin > Desktop > Internship > Project 2 > project2.ipynb > import pandas as pd
+ Code + Markdown Run All Clear All Outputs Outline
base (Python 3.11.5)
3 [-0.17984472, 0.4633496, -0.024835482, 0.09771...
4 [-0.22664028, 0.5552239, -0.0154908225, 0.1304...
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
Python
Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df['processed_text'], df['sentiment'], test_size=0.2, random_state=42)
# Vectorize the text data using TF-IDF for SVM
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
# SVM model
svm_model = SVC()
svm_model.fit(X_train_tfidf, y_train)
svm_predictions = svm_model.predict(X_test_tfidf)
# Evaluate SVM performance
print("SVM Metrics:")
print("Accuracy:", accuracy_score(y_test, svm_predictions))
print("Precision:", precision_score(y_test, svm_predictions, average='weighted'))
Restricted Mode
Cell 2 of 15
Type here to search
30°C Mostly sunny
ENG IN
7:12 PM 1/28/2024

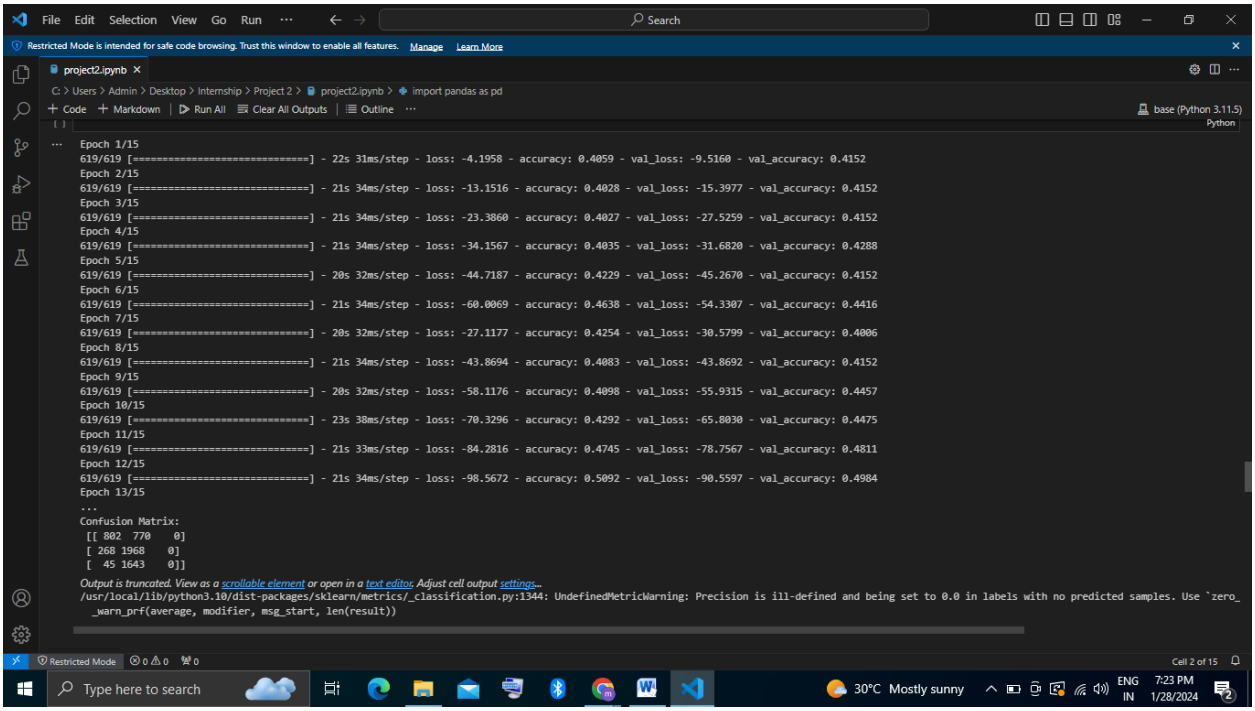

File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
project2.ipynb
C: > Users > Admin > Desktop > Internship > Project 2 > project2.ipynb > import pandas as pd
+ Code + Markdown Run All Clear All Outputs Outline
base (Python 3.11.5)
# Evaluate SVM performance
print("SVM Metrics:")
print("Accuracy:", accuracy_score(y_test, svm_predictions))
print("Precision:", precision_score(y_test, svm_predictions, average='weighted'))
print("Recall:", recall_score(y_test, svm_predictions, average='weighted'))
print("F1 Score:", f1_score(y_test, svm_predictions, average='weighted'))
print("\nClassification Report:")
print(classification_report(y_test, svm_predictions))
Python
SVM Metrics:
Accuracy: 0.6901382823871907
Precision: 0.7077758844609202
Recall: 0.6901382823871907
F1 Score: 0.6880613687115346
Classification Report:
precision recall f1-score support
negative 0.75 0.53 0.62 1572
neutral 0.62 0.80 0.69 2236
positive 0.79 0.69 0.74 1688
accuracy 0.69 5496
macro avg 0.72 0.67 0.69 5496
weighted avg 0.71 0.69 0.69 5496
df['processed_text'].fillna('', inplace=True)
df['sentiment'].fillna('', inplace=True)
# Tokenization and remove stop words
stop_words = set(stopwords.words('english'))
df['tokenized_text'] = df['processed_text'].apply(lambda x: [word for word in word_tokenize(x) if word.lower() not in stop_words and word not in string.punctuation])
Restricted Mode
Cell 2 of 15
Type here to search
30°C Mostly sunny
ENG IN
7:12 PM 1/28/2024

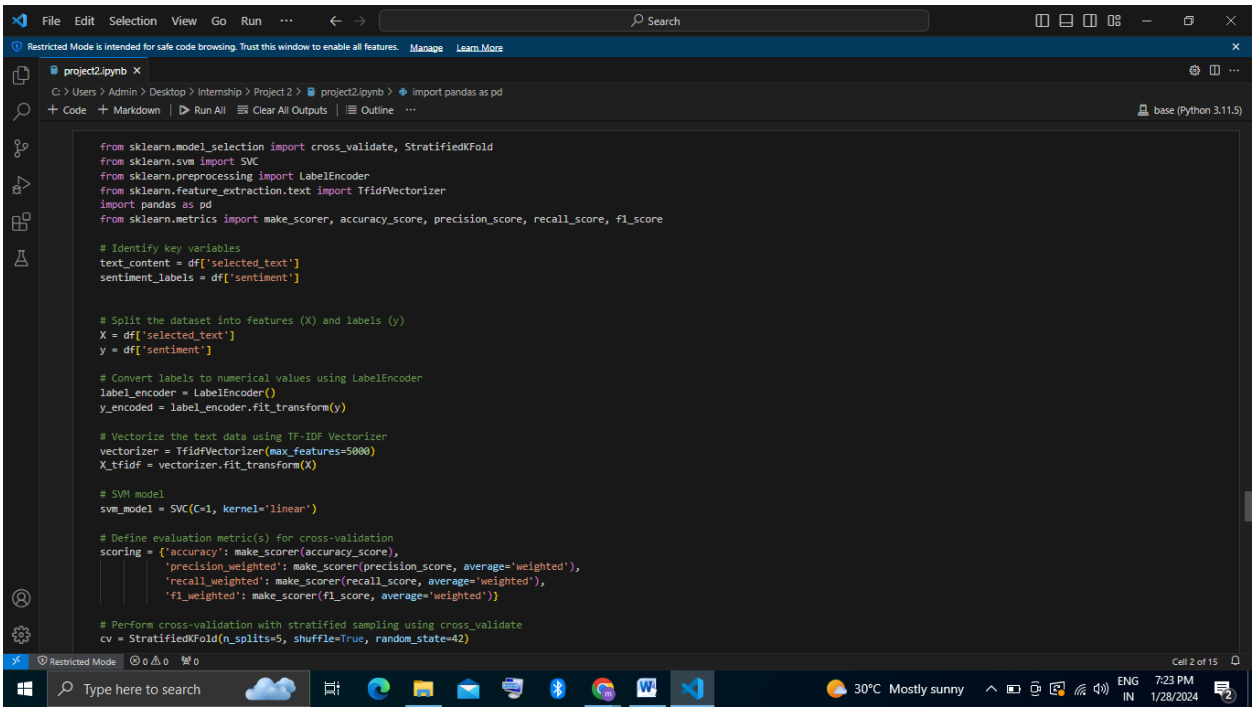


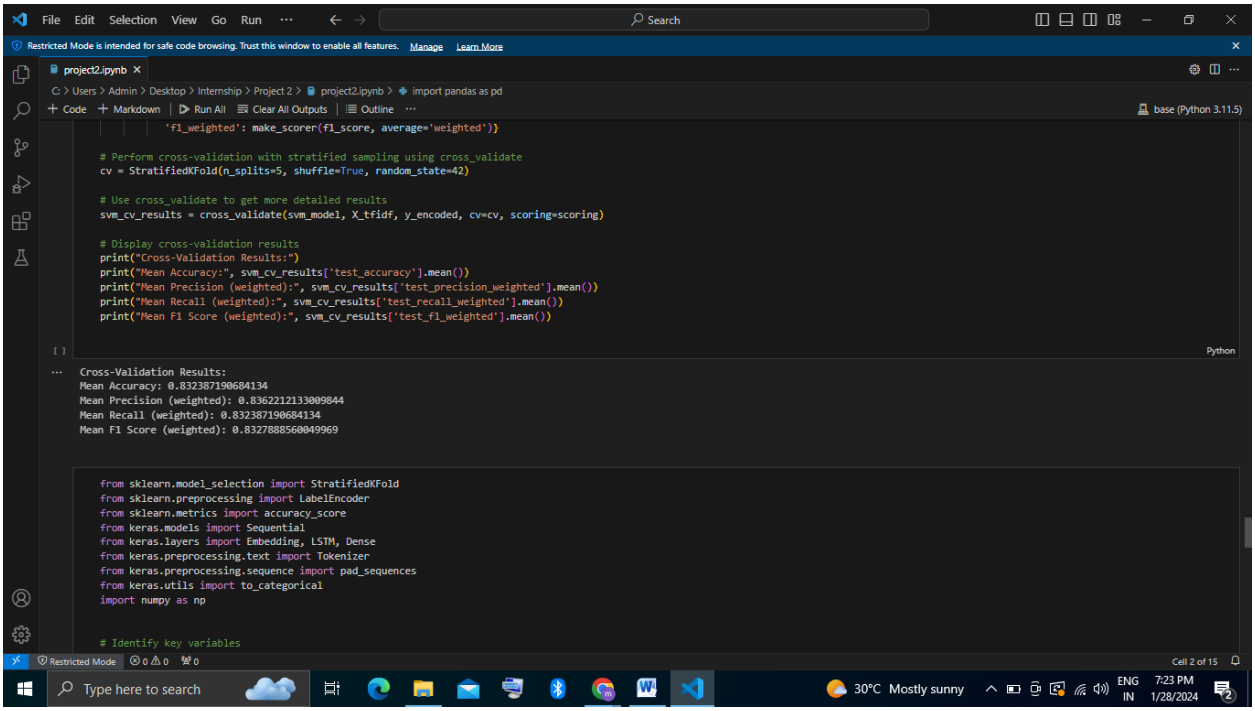
```
df['processed_text'].fillna('', inplace=True)
df['sentiment'].fillna('', inplace=True)

# Tokenization and remove stop words
stop_words = set(stopwords.words('english'))
df['tokenized_text'] = df['processed_text'].apply(lambda x: [word for word in word_tokenize(x) if word.lower() not in stop_words and word not in string.punctua

# Convert the tokenized text back to string for LSTM
df['tokenized_text_str'] = df['tokenized_text'].apply(lambda x: ' '.join(x))

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df['tokenized_text_str'], df['sentiment'], test_size=0.2, random_state=42)

# Convert labels to numerical values
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Tokenize text
max_words = 5000  # You can customize the number of words to use
tokenizer = Tokenizer(num_words=max_words, oov_token="<OOV>")
tokenizer.fit_on_texts(X_train)

# Convert text to sequences
X_train_sequences = tokenizer.texts_to_sequences(X_train)
X_test_sequences = tokenizer.texts_to_sequences(X_test)

# Pad sequences to have the same length
```



```
# Pad sequences to have the same length
max_length = max(len(seq) for seq in X_train_sequences)  # Define max_length here
X_train_padded = pad_sequences(X_train_sequences, maxlen=max_length, padding='post')
X_test_padded = pad_sequences(X_test_sequences, maxlen=max_length, padding='post')

# Build LSTM model
model = Sequential()
model.add(Embedding(input_dim=max_words, output_dim=100, input_length=max_length))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train_padded, y_train_encoded, epochs=15, batch_size=32, validation_split=0.1)

# Evaluate the model
predictions = model.predict(X_test_padded)
rounded_predictions = [round(pred[0]) for pred in predictions]

# Convert rounded predictions back to original labels
y_pred = label_encoder.inverse_transform(rounded_predictions)

# Evaluate the performance
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred, average='weighted'))
print("Recall:", recall_score(y_test, y_pred, average='weighted'))
print("F1 Score:", f1_score(y_test, y_pred, average='weighted'))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```


File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
project2.ipynb
C: > Users > Admin > Desktop > Internship > Project 2 > project2.ipynb > import pandas as pd
+ Code + Markdown Run All Clear All Outputs Outline
base (Python 3.11.5)
Python
Epoch 1/15
619/619 [==============================] - 22s 31ms/step - loss: -4.1958 - accuracy: 0.4059 - val_loss: -9.5160 - val_accuracy: 0.4152
Epoch 2/15
619/619 [==============================] - 21s 34ms/step - loss: -13.1516 - accuracy: 0.4028 - val_loss: -15.3977 - val_accuracy: 0.4152
Epoch 3/15
619/619 [==============================] - 21s 34ms/step - loss: -23.3860 - accuracy: 0.4027 - val_loss: -27.5259 - val_accuracy: 0.4152
Epoch 4/15
619/619 [==============================] - 21s 34ms/step - loss: -34.1567 - accuracy: 0.4035 - val_loss: -31.6820 - val_accuracy: 0.4288
Epoch 5/15
619/619 [==============================] - 20s 32ms/step - loss: -44.7187 - accuracy: 0.4229 - val_loss: -45.2670 - val_accuracy: 0.4152
Epoch 6/15
619/619 [==============================] - 21s 34ms/step - loss: -60.0069 - accuracy: 0.4638 - val_loss: -54.3307 - val_accuracy: 0.4416
Epoch 7/15
619/619 [==============================] - 20s 32ms/step - loss: -27.1177 - accuracy: 0.4254 - val_loss: -30.5799 - val_accuracy: 0.4006
Epoch 8/15
619/619 [==============================] - 21s 34ms/step - loss: -43.8694 - accuracy: 0.4083 - val_loss: -43.8692 - val_accuracy: 0.4152
Epoch 9/15
619/619 [==============================] - 20s 32ms/step - loss: -58.1176 - accuracy: 0.4098 - val_loss: -55.9315 - val_accuracy: 0.4457
Epoch 10/15
619/619 [==============================] - 23s 38ms/step - loss: -70.3296 - accuracy: 0.4292 - val_loss: -65.8030 - val_accuracy: 0.4475
Epoch 11/15
619/619 [==============================] - 21s 33ms/step - loss: -84.2816 - accuracy: 0.4745 - val_loss: -78.7567 - val_accuracy: 0.4811
Epoch 12/15
619/619 [==============================] - 21s 34ms/step - loss: -98.5672 - accuracy: 0.5092 - val_loss: -90.5597 - val_accuracy: 0.4984
Epoch 13/15
...
Confusion Matrix:
[[802 770 0]
[268 1968 0]
[45 1643 0]]
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_
_warn_prf(average, modifier, msg_start, len(result))
Restricted Mode
Cell 2 of 15
Type here to search
30°C Mostly sunny
ENG IN
7:23 PM 1/28/2024

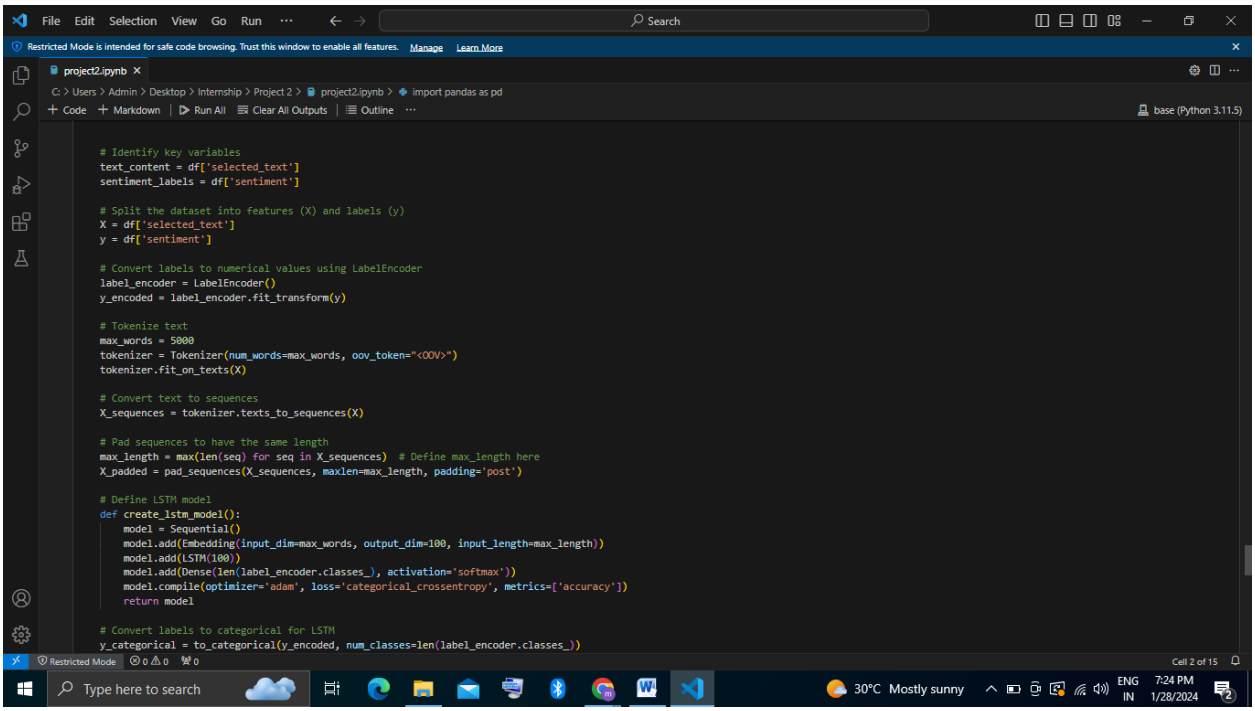

File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
project2.ipynb
C: > Users > Admin > Desktop > Internship > Project 2 > project2.ipynb > import pandas as pd
+ Code + Markdown Run All Clear All Outputs Outline
base (Python 3.11.5)
from sklearn.model_selection import cross_validate, StratifiedKFold
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
from sklearn.metrics import make_scorer, accuracy_score, precision_score, recall_score, f1_score
# Identify key variables
text_content = df['selected_text']
sentiment_labels = df['sentiment']
# Split the dataset into features (X) and labels (y)
X = df['selected_text']
y = df['sentiment']
# Convert labels to numerical values using LabelEncoder
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
# Vectorize the text data using TF-IDF Vectorizer
vectorizer = TfidfVectorizer(max_features=5000)
X_tfidf = vectorizer.fit_transform(X)
# SVM model
svm_model = SVC(C=1, kernel='linear')
# Define evaluation metric(s) for cross-validation
scoring = {'accuracy': make_scorer(accuracy_score),
'precision_weighted': make_scorer(precision_score, average='weighted'),
'recall_weighted': make_scorer(recall_score, average='weighted'),
'f1_weighted': make_scorer(f1_score, average='weighted')}
# Perform cross-validation with stratified sampling using cross_validate
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
Restricted Mode
Cell 2 of 15
Type here to search
30°C Mostly sunny
ENG IN
7:23 PM 1/28/2024



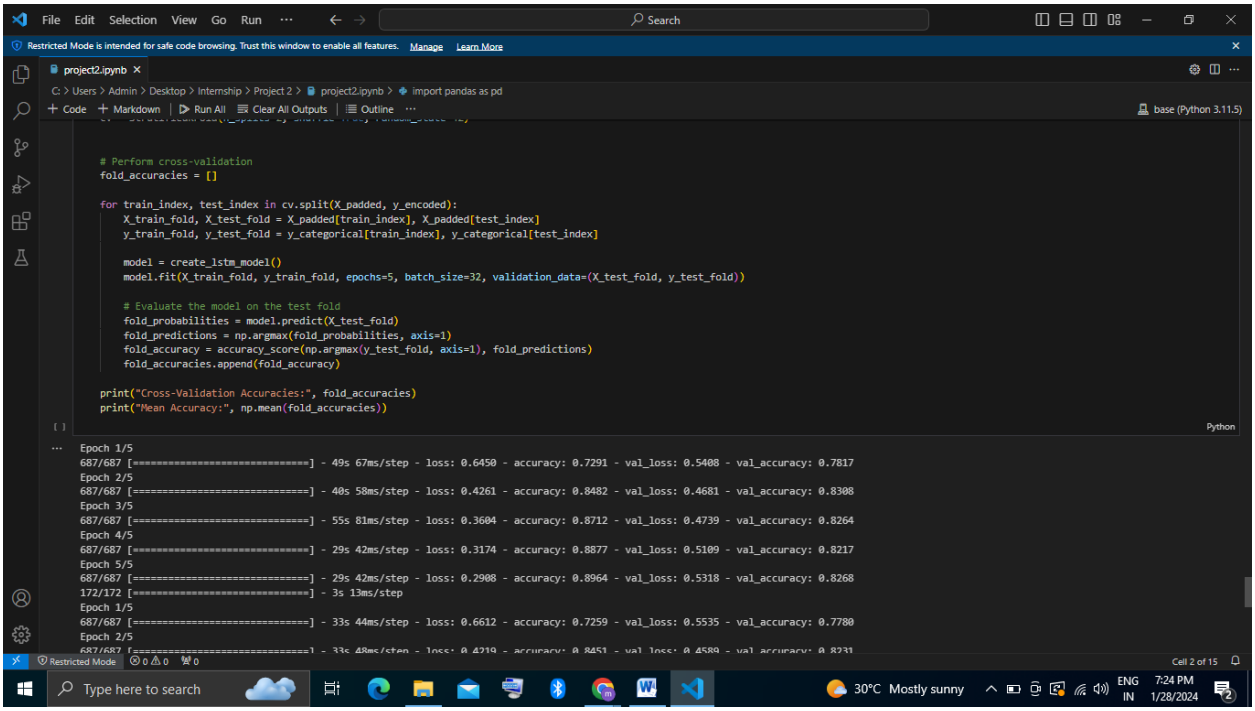
```
                'f1_weighted': make_scorer(f1_score, average='weighted')}

# Perform cross-validation with stratified sampling using cross_validate
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Use cross_validate to get more detailed results
svm_cv_results = cross_validate(svm_model, X_tfidf, y_encoded, cv=cv, scoring=scoring)

# Display cross-validation results
print("Cross-Validation Results:")
print("Mean Accuracy:", svm_cv_results['test_accuracy'].mean())
print("Mean Precision (weighted):", svm_cv_results['test_precision_weighted'].mean())
print("Mean Recall (weighted):", svm_cv_results['test_recall_weighted'].mean())
print("Mean F1 Score (weighted):", svm_cv_results['test_f1_weighted'].mean())
```

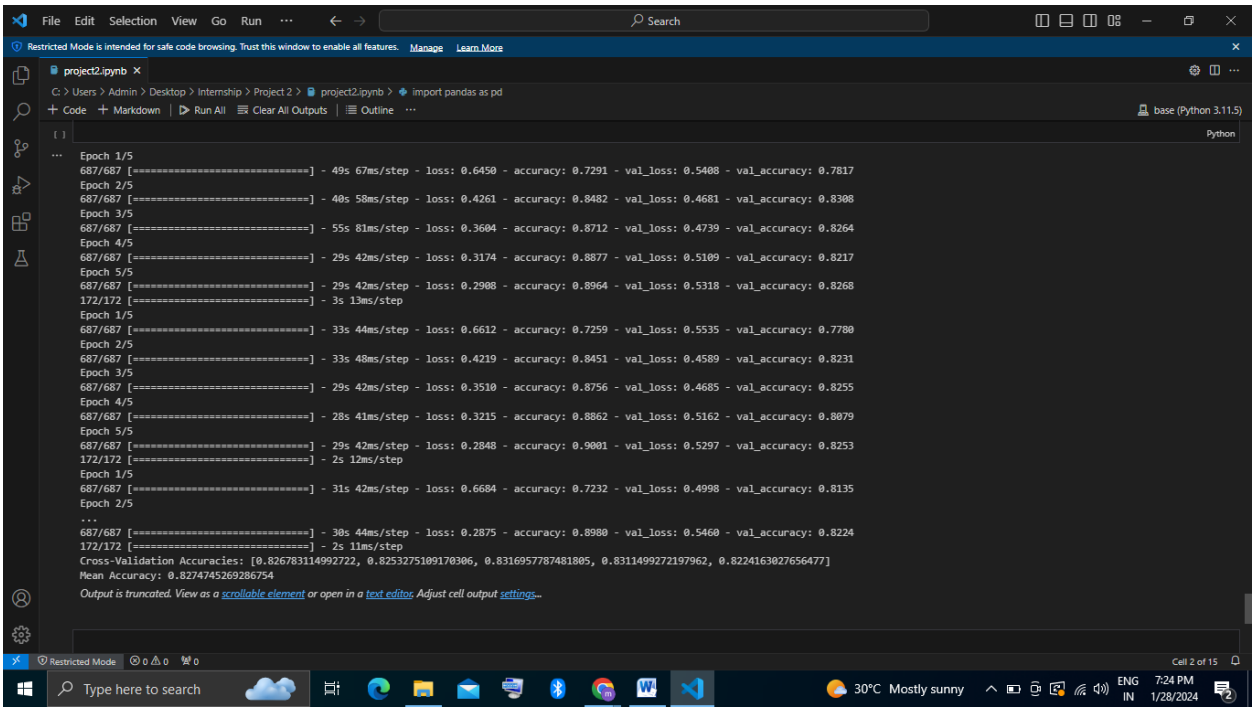
```
Cross-Validation Results:
Mean Accuracy: 0.832387190684134
Mean Precision (weighted): 0.8362212133009844
Mean Recall (weighted): 0.832387190684134
Mean F1 Score (weighted): 0.8327888560049969
```

```
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
import numpy as np


# Identify key variables
```



```
# Identify key variables
text_content = df['selected_text']
sentiment_labels = df['sentiment']

# Split the dataset into features (X) and labels (y)
X = df['selected_text']
y = df['sentiment']

# Convert labels to numerical values using LabelEncoder
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Tokenize text
max_words = 5000
tokenizer = Tokenizer(num_words=max_words, oov_token="<OOV>")
tokenizer.fit_on_texts(X)

# Convert text to sequences
X_sequences = tokenizer.texts_to_sequences(X)

# Pad sequences to have the same length
max_length = max(len(seq) for seq in X_sequences)  # Define max_length here
X_padded = pad_sequences(X_sequences, maxlen=max_length, padding='post')

# Define LSTM model
def create_lstm_model():
    model = Sequential()
    model.add(Embedding(input_dim=max_words, output_dim=100, input_length=max_length))
    model.add(LSTM(100))
    model.add(Dense(len(label_encoder.classes_), activation='softmax'))
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model

# Convert labels to categorical for LSTM
y_categorical = to_categorical(y_encoded, num_classes=len(label_encoder.classes_))
```





10. Conclusion

Summarized key findings, discussed model performance. Highlighted the significance of the sentiment analysis model in understanding textual sentiment.