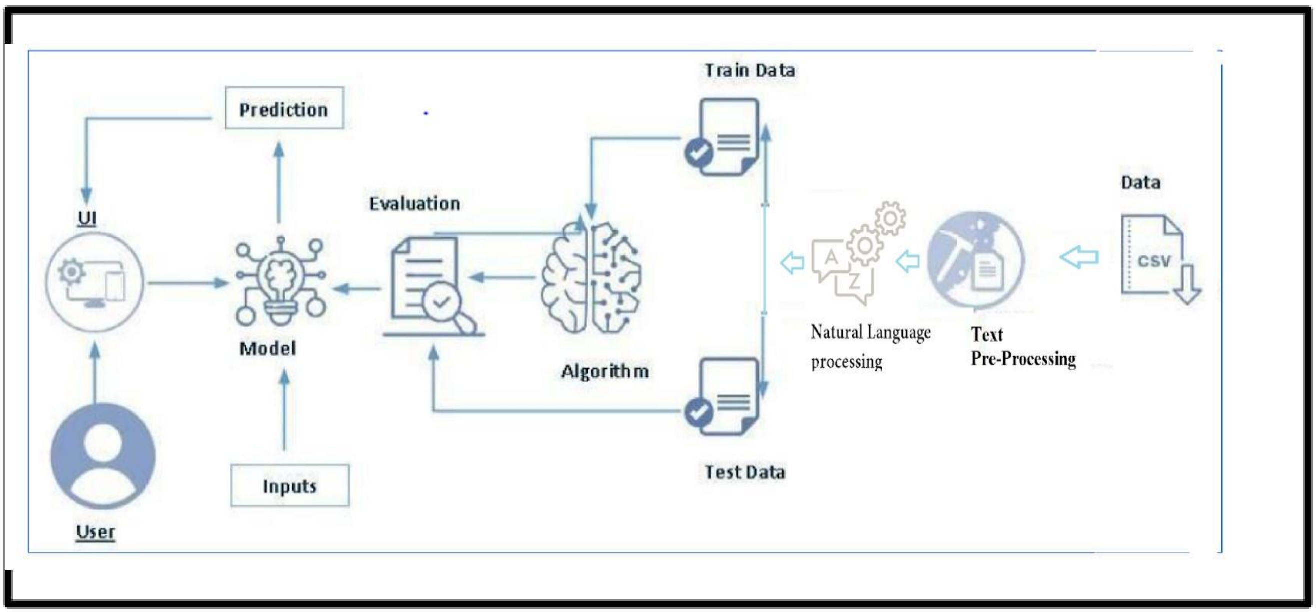
Gilded Emotions: Unearthing Market Sentiments in Gold News

Sentiment Analysis of Commodity News (Gold) is a process of using natural language processing and machine learning techniques to determine the emotional tone of news articles or other text related to the gold commodity market. The goal of sentiment analysis is to understand how people feel about a particular topic, in this case gold, by analyzing the words and phrases used in the text

Sentiment analysis can help to determine whether news about gold is generally positive, negative, or neutral, giving traders and investors an idea of how the market is reacting to the latest developments in the gold industry. For example, positive sentiment in news articles about gold might indicate increasing demand for the precious metal, which could drive up prices. On the other hand, negative sentiment could indicate a decrease in demand or a downturn in the market. By conducting sentiment analysis on a large corpus of news articles about gold, it is possible to gain insights into the overall sentiment of the market and make informed decisions about buying and selling gold.

Technical Architecture:



Project Flow:

● User interacts with the UI to enter the input.
● Entered input is analysed by the model which is integrated.
● Once model analyses the input the prediction is showcased on the UI
To accomplish this, we have to complete all the activities listed below,
● Define Problem / Problem Understanding
o Specify the business problem
o Business requirements
o Literature Survey
o Social or Business Impact.
● Data Collection & Preparation
o Collect the dataset
o Data Preparation
● Exploratory Data Analysis
o Descriptive statistical
o Visual Analysis
● Model Building
o Training the model in multiple algorithms
o Testing the model
● Performance Testing
o Testing model with multiple evaluation metrics
● Model Deployment
o Save the best model
o Integrate with Web Framework

The sentiment analysis of commodity news, especially focusing on gold, plays a pivotal role in understanding market dynamics. This project leverages natural language processing and machine learning techniques to discern the emotional tone in gold-related news articles. The primary objective is to gauge market sentiment and provide insights that can assist traders and investors in making informed decisions. By analyzing text data, we aim to classify news articles as having a positive, negative, or neutral sentiment. This project highlights the signi cance of sentiment analysis in the gold market and its potential to impact nancial decision-making.

Introduction

Sentiment Analysis in Gold News: Introduction

The gold commodity market is known for its unique sensitivity to various factors, including economic events, geopolitical situations, and market sentiment. Understanding how the market perceives gold is crucial for traders and investors. Sentiment analysis, a natural language processing technique, is a valuable tool for deciphering emotional tones in text data. This project introduces the concept of sentiment analysis applied to gold news, emphasizing the relevance of gauging market sentiment. By analyzing the sentiments in news articles, we can infer market reactions and assess potential shifts in demand and pricing. This project explores the use of machine learning to perform sentiment analysis in the context of gold. **bold text**

Literature Review: **Sentiment Analysis in Financial Markets: A Review* *

The literature review section provides an overview of sentiment analysis in nancial markets, focusing on the gold commodity market. Several studies have explored the application of sentiment analysis in predicting market movements and understanding the impact of news sentiment on nancial decisions.

In the eld of supervised learning, algorithms like Decision Trees, Random Forest, and Logistic Regression have been widely employed for sentiment analysis in nancial markets. These models use labeled data to classify sentiment in text data.

Unsupervised learning techniques, such as clustering and topic modeling, have also been explored for sentiment analysis, where patterns and themes in news articles can be identi ed.

Natural Language Processing (NLP) **bold text**concepts are fundamental to sentiment analysis. Techniques like tokenization, stop-word removal, and stemming are essential for text preprocessing.

Evaluation metrics, such as accuracy, precision, recall, and F1 score, help assess the performance of sentiment analysis models.

Resources like nancial news datasets and sentiment lexicons are used to train and validate sentiment analysis models.

While previous research has laid the foundation for sentiment analysis in nancial markets, this project aims to apply these concepts speci cally to the gold commodity market.

OVERVIEW

Milestone 1: De ne Problem / Problem Understanding

Activity 1: Specify the business problem

Activity 2: Business requirements

Activity 3: Literature Survey

Activity 4: Social or Business Impact.

Milestone 2: Data Collection & Preparation Activity 1-Data Collection

Activity 2 : Data Preparation

Activity 2.1: Handling missing values

Activity 2.2: Handling Categorical Values

Activity 2.3: Handling Imbalance Data

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical

Activity 2: Visual analysis

Data Analysis and Visualization

Exploratory Data Analysis

Activity 1: Descriptive statistical

Activity 2: Visual analysis

Activity 2.1: Univariate analysis

Activity 2.2: Bivariate analysis Activity 2.3: Multivariate analysis

Milestone 4: Model Building

Activity 1: Training the model in multiple algorithms Activity 2: Testing the model

Milestone 5: Model Deployment

Activity 1: Save the best model

Activity 2: Integrate with Web Framework

Activity 2.1: Building Html Pages:

Activity 2.2: Build Python code: Activity 2.3: Run the web application

Milestone 1: De ne Problem / Problem Understanding

Activity 1: **Specify the Business Problem**:: Start by clearly de ning the speci c problem or challenge that the project aims to address. Understand the business context and objectives. What is the core issue that the project needs to solve? De ne it in a precise and concise manner.

Activity 2:** Business Requirements**:: Gather and document the business requirements and expectations. Engage with key stakeholders to identify their needs and understand how solving the problem aligns with the organization's goals.

Activity 3: **Literature Survey**:: Conduct a literature survey to review existing research, models, or methods related to the problem at hand. Learn from previous work and understand what approaches have been used successfully and where there might be gaps.

Activity 4: **Social or Business Impact**:: Evaluate the potential impact of solving the problem on both social and business dimensions. Understand how addressing the problem can improve customer satisfaction, revenue, e ciency, or any other relevant key performance indicators.

Milestone 2: Data Collection & Preparation

Milestone 2: Data Collection & Preparation

Activity 1-Data Collection

Activity 2 : Data Preparation

Activity 2.1: Handling missing values

Activity 2.2: Handling Categorical Values

Activity 2.3: Handling Imbalance Data

Activity 1-Data Collection

Double-click (or enter) to edit

import pandas as pd

Read the dataset
df =
pd.read_csv('gold-
dataset-sinha-
khandait.csv')

df.head()

	Dates	URL	News	Price Direction Up	Price Direction Constant	Price Direction Down	Asset Comparision	Past Information	Future Information
0	28-01-2016	http://www.marketwatch.com/story/april-gold-do...	april gold down 20 cents to settle at $1,116.1...	0	0	1	0	1	0
1	13-09-2017	http://www.marketwatch.com/story/gold-prices-s...	gold suffers third straight daily decline	0	0	1	0	1	0

Activity 2 : Data Preparation

Activity 2 : Data Preparation

Gold

```
import pandas as pd

# Read the dataset
df = pd.read_csv('gold-dataset-sinha-khandait.csv')

df.shape
```

(10570, 10)

Activity 2.1: Handling missing values

#Activity 2.1:
Handling missing
values df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex:
10570
entries, 0 to
10569 Data
columns
(total 10
columns):

```
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
0    Dates                     10570 non-null  object
1    URL                       10570 non-null  object
2    News                      10570 non-null  object
3    Price Direction Up        10570 non-null  int64
4    Price Direction Constant  10570 non-null  int64
5    Price Direction Down      10570 non-null  int64
6    Asset Comparision         10570 non-null  int64
7    Past Information          10570 non-null  int64
8    Future Information        10570 non-null  int64  9   Price Sentiment
10570 non-null  object dtypes: int64(6), object(4) memory usage: 825.9+ KB
```

Activity 2.2: Handling Categorical Values

```
#Activity 2.2: Handling Categorical Values

df['Price Sentiment'].value_counts()
```

positive
4412
negative
3814
none
1968
neutral
376
Name:

Price
Sentimen
t, dtype:
int64

df['Price Sentiment'].unique()

array(['negative', 'positive',

'none', 'neutral'],

dtype=object)

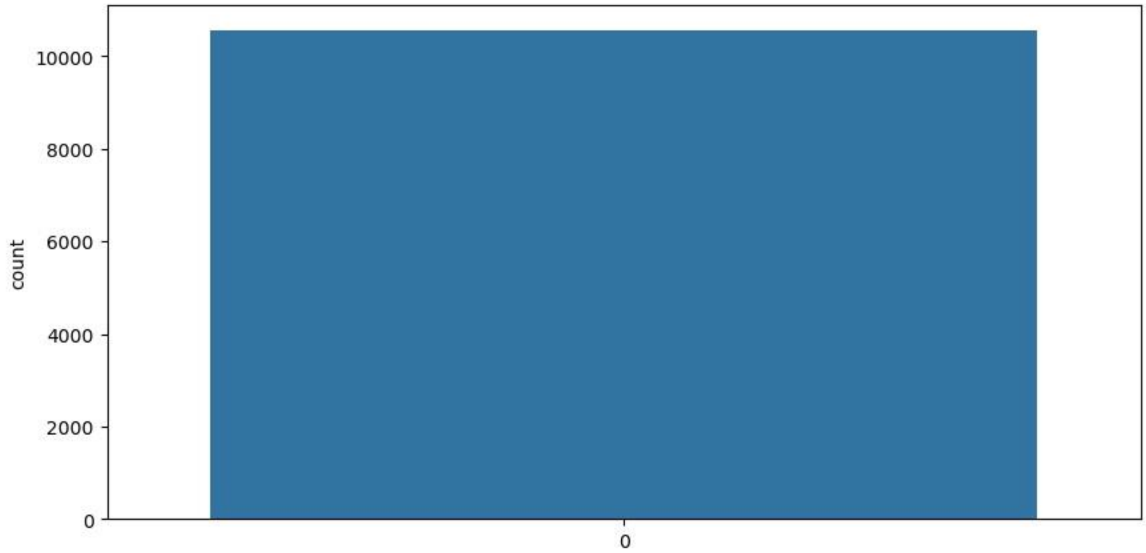
df['Price Sentiment'] = df['Price Sentiment'].map({'negative':1, 'positive':2, 'neutral':3, 'none':2 })

Activity 2.3: Handling Imbalance Data

#Activity
2.3:
Handling
Imbalance
Data import
pandas as
pd import
numpy as
np import
matplotlib.p
yplot as plt
import
seaborn as
sns
plt.figure(fig
size=(10,5))
sns.countpl
ot(df['Price
Sentiment'])

df['Price
Sentiment'].
value_count
s()

```
Series([], Name: Price Sentiment, dtype: int64)
```



df.isnull().sum()

Dates	0
URL	0
News	0
Price Direction Up	0
Price Direction Constant	0
Price Direction Down	0
Asset Comparision	0
Past Information	0
Future Information	0
Price Sentiment	0

dtype:
int64

Milestone 3: Exploratory Data Analysis

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical

#Activity 1:
Descriptive
statistical
df.describe()

Price DirectionPrice Direction Price DirectionAsset
Past Future Price
Up Constant DownComparisionInformation
InformationSentiment

count	10570.000000	10570.000000	10570.000000	10570.000000	10570.0000
mean	0.417408	0.042006	0.370104	0.189309	0.969915
std	0.493155	0.200612	0.482855	0.391773	0.170830
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	1.000000
50%	0.000000	0.000000	0.000000	0.000000	1.000000
75%	1.000000	0.000000	1.000000	0.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

Handle missing
values
df.dropna(inplace
=True)

Encode categorical data if needed

Example: df['column_name'] = pd.get_dummies(df['column_name'])

Activity 2: Visual analysis

Data Analysis and Visualization

Activity 2: Visual analysis Data Analysis and Visualization

import matplotlib.pyplot as plt

```
#
Visuali
ze
data
plt.hist
(df['Ne
ws'])
plt.sho
w()
```

Import Libraries

```
import pandas as pd
import numpy as np from
sklearn.model_selection
import train_test_split
from
sklearn.feature_extraction
.text import
TfidfVectorizer from
sklearn.naive_bayes
import MultinomialNB
```

from sklearn.metrics import accuracy_score, classification_report

Load and Prepare the Dataset

Load your dataset
df =
pd.read_csv('gold-
dataset-sinha-
khandait.csv')

Preprocessing (if needed)

Example: Encoding categorical features

Split the
dataset into
features and
target X =
df['News'] y =
df['Price
Sentiment']

Split into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Feature Extraction (TF-IDF)

tfidf_vectorizer = TfidfVectorizer(max_features=5000) # Adjust max_features as needed

X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)

X_test_tfidf = tfidf_vectorizer.transform(X_test)

Build and Train a Machine Learning Model

Build and Train a Machine Learning Model

model =
MultinomialNB()
model.fit(X_train_
tfidf, y_train)

▾MultinomialNB
MultinomialNB()

Make Predictions

y_pred = model.predict(X_test_tfidf)

y_pred

array(['negative', 'positive',

'positive', ..., 'none', 'negative',

'positive'], dtype='<U8') Evaluate

the Model

accuracy =
accuracy_score(y_test,
y_pred) print(f'Accuracy:
{accuracy:.2f}')

classification_rep =
classification_report(y_test, y_pred)
print(classification_rep)

Accuracy: 0.80

precision recall f1-score support

negative 0.85
0.82 0.83 762
neutral 1.00 0.05
0.10 76 none

0.85 0.60 0.70
407 positive 0.75
0.94 0.83 869

accuracy
0.80 2114 macro avg
0.86 0.60 0.62
2114 weighted avg
0.81 0.80 0.78
2114

Exploratory Data Analysis

Activity 1: Descriptive statistical

Activity 2: Visual analysis

Activity 2.1: Univariate analysis

Activity 2.2: Bivariate analysis

Activity 2.3: Multivariate analysis

Exploratory Data Analysis Activity 1: Descriptive statistical Activity 2: Visual analysis Activity 2.1: Univariate analysis Activity 2.2: Bivariate analysis Activity 2.3: Multivariate analysis

Exploratory Data Analysis (EDA)

import
pandas
as pd
import
numpy
as np
import
matplo
tlib.pyp
lot as
plt

import
seabor
n as
sns

df = pd.read_csv('gold-dataset-
sinha-khandait.csv')
print(df.describe())

Price Direction Up Price Direction
Constant Price Direction Down \ count
10570.000000 10570.000000
10570.000000 mean 0.417408
0.042006 0.370104 std
0.493155 0.200612
0.482855 min 0.000000
0.000000 0.000000 25%
0.000000 0.000000
0.000000

50% 0.000000
0.000000 0.000000 75%
1.000000 0.000000
1.000000 max 1.000000
1.000000 1.000000

Asset Comparision Past
Information Future
Information count
10570.000000
10570.000000
10570.00000 mean
0.189309 0.969915
0.03018 std 0.391773
0.170830 0.17109 min
0.000000 0.000000
0.00000 25% 0.000000
1.000000 0.00000

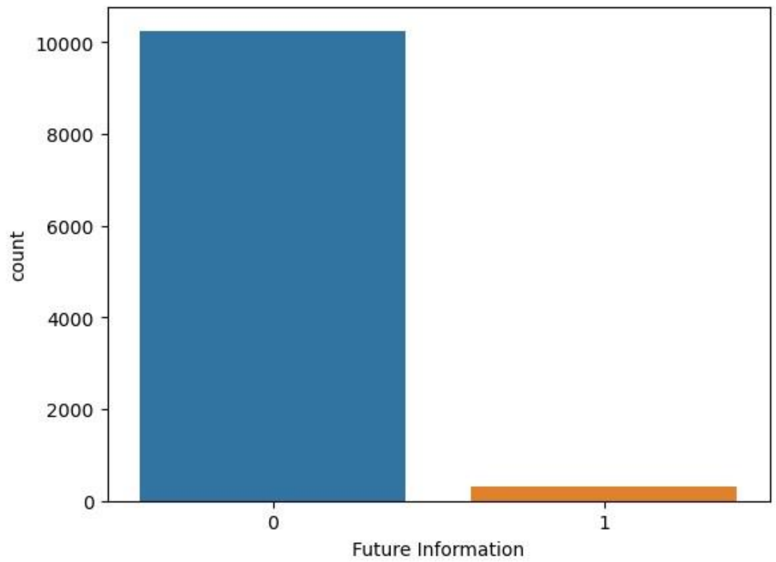
50%	0.000000	1.000000	0.00000

75%	0.000000
1.000000	0.00000 max
1.000000	1.000000
1.00000

Visual Analysis - Univariate

Visual Analysis - Univariate

Countplot
example import
seaborn as sns
sns.countplot(x='
Future
Information',
data=df)
plt.show()

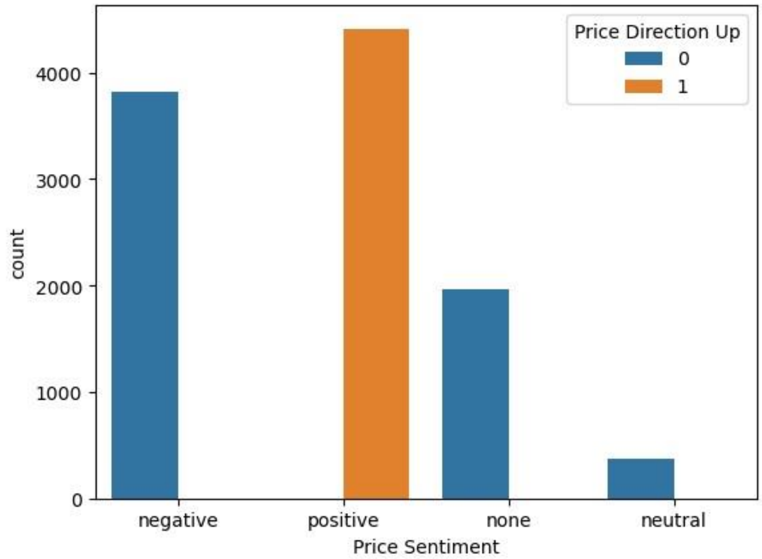


Visual Analysis - Bivariate

Visual Analysis - Bivariate

Countplot with hue
sns.countplot(x='Price

Sentiment', hue='Price Direction
Up', data=df) plt.show()

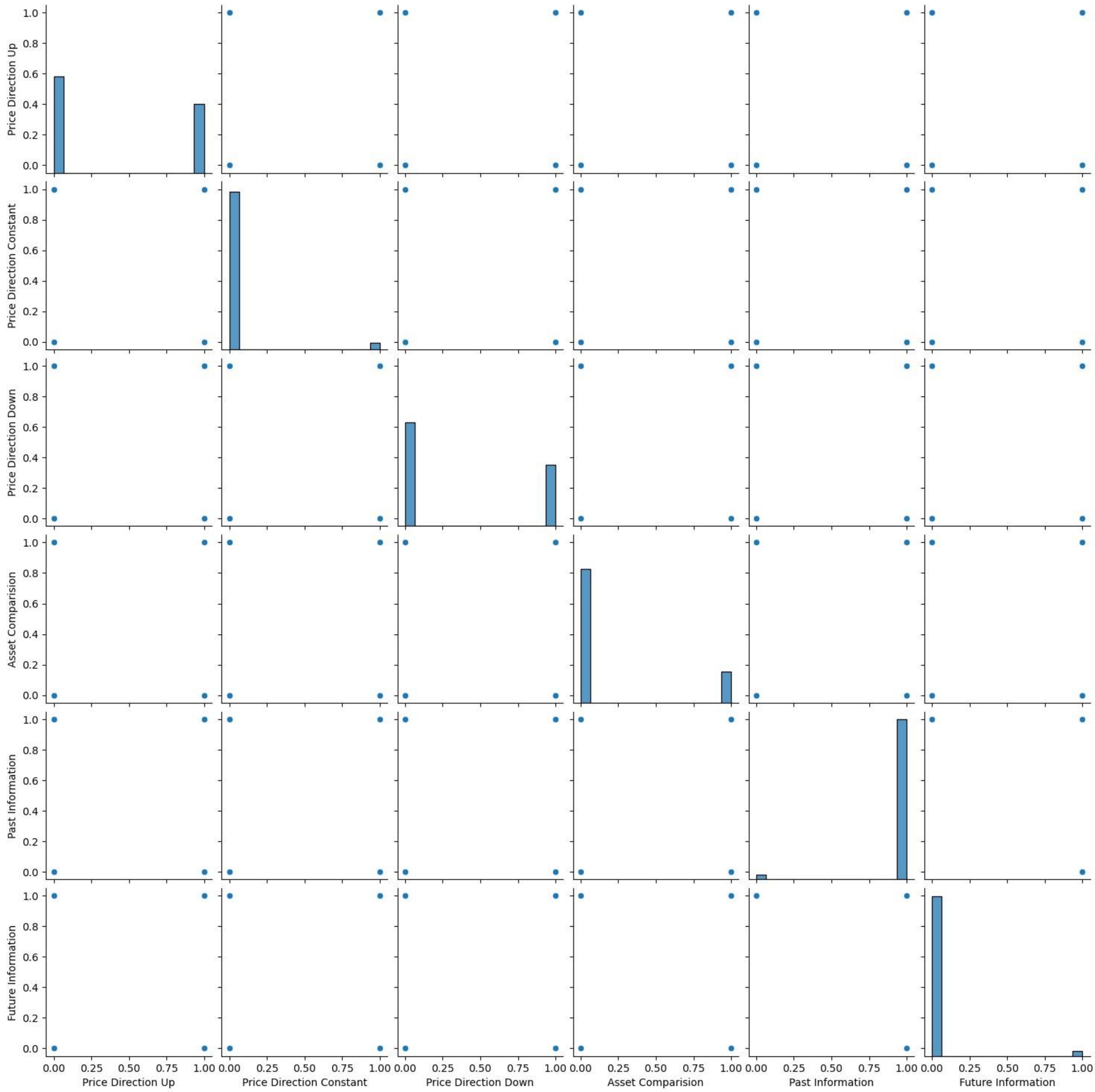


Visual Analysis - Multivariate

Visual Analysis - Multivariate

Multivariate analysis
using pairplot
sns.pairplot(df)

plt.show()


Price Direction Up
Price Direction Constant
Price Direction Down
Asset Comparision
Past Information
Future Information
0.00
0.25
0.50
0.75
1.00
0.0
0.2
0.4
0.6
0.8
1.0

```
import nltk
import re
from
nltk.corpus
import
stopwords
from
nltk.stem
import
PorterStem
mer from
nltk.stem
import
WordNetLe
mmatizer
```

```
# Sample text data
```

```
text_data = "Text pre-processing is a crucial step in NLP. It converts raw text
into a more meaningful representation for analysis."
```

```
# 1.
Tokenizatio
n words =
nltk.word_t
okenize(tex
t_data)
```

```
# 2. Stop-word removal stop_words =
set(stopwords.words("english"))
filtered_words = [word for word in
words if word.lower() not in
stop_words]
```

```
# 3. Stemming (using Porter
Stemmer) stemmer =
PorterStemmer()
stemmed_words =
[stemmer.stem(word) for
word in filtered_words]
```

```
# 4. Lemmatization (using WordNet
Lemmatizer) lemmatizer =
WordNetLemmatizer()
lemmatized_words =
[lemmatizer.lemmatize(word) for
word in stemmed_words]

# 5. Regular expression-based cleaning
(removing non-alphanumeric characters)
cleaned_text = re.sub(r'[^a-zA-Z0-9]', ' ', '
'.join(lemmatized_words))

print("Origi
nal Text:")
print(text_d
ata)
print("\nPre
-processed
Text:")
print(cleane
d_text)
```

```
logistic_regression_model = LogisticRegression()
logistic_regression_model.fit(X_train_tfidf, y_train)
logistic_regression_predictions =
logistic_regression_model.predict(X_test_tfidf)
```

```
C:\Users\mailt\anaconda3_2\Lib\site-
packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning:
lbfgs failed to converge (st STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations
(max_iter) or scale the data as shown in:
https://scikit-
learn.org/stable/modules/preprocessing.
html Please also refer to the
documentation for alternative solver
options:    https://scikit-
learn.org/stable/modules/linear_model.
html#logistic-regression  n_iter_i =
_check_optimize_result(
```

```
logistic_regression_report =
classification_report(y_test,
logistic_regression_predictions)
logistic_regression_confusion =
confusion_matrix(y_test,
logistic_regression_predictions) print("Logistic
Regression Classification Report:\n",
logistic_regression_report) print("Logistic
Regression Confusion Matrix:\n",
logistic_regression_confusion)
```

```
Logistic Regression
Classification Report:
precision    recall  f1-
score   support
```

negative 0.91
0.90 0.91 762
neutral 0.85 0.51
0.64 76 none
0.80 0.84 0.82
407 positive 0.90
0.92 0.91 869

accuracy
0.88 2114 macro avg
0.87 0.79 0.82
2114 weighted avg
0.88 0.88 0.88
2114

Logistic Regression Confusion Matrix:

[[689 3 33 37]

[13 39 10 14]

[25 2 342 38]

[30 2 40 797]]

Train and Testing the SVM model (Support Vector Machine):

Train and Testing the SVM model (Support Vector Machine):

```
svm_model = SVC()
svm_model.fit(X_tr
ain_tfidf, y_train)
svm_predictions =
svm_model.predict
(X_test_tfidf)
svm_report =
classification_repor
t(y_test,
svm_predictions)
svm_confusion =
confusion_matrix(y
```

_test,
svm_predictions)
print("SVM
Classification
Report:\n",
svm_report)
print("SVM
Confusion
Matrix:\n",
svm_confusion)

SVM Classification
Report:
precision recall f1-
score support

negative 0.92
0.90 0.91 762
neutral 0.76 0.58
0.66 76 none
0.79 0.87 0.83
407 positive 0.91
0.91 0.91 869

accuracy
0.89 2114 macro avg
0.85 0.81 0.83
2114 weighted avg
0.89 0.89 0.88
2114

SVM Confusion Matrix:

[[685 5 38 34]

[11 44 10 11]

[24 1 353 29]

[26 8 46 789]]

Activity 1.2: SVM (Support Vector machine)

Activity 1.2: SVM (Support Vector machine)

from sklearn.feature_extraction.text import
TfidfVectorizer from sklearn.model_selection import
train_test_split from sklearn.metrics import
accuracy_score, f1_score, recall_score,
precision_score, confusion_matrix from sklearn.svm
import SVC import re

df = pd.read_csv("gold-dataset-sinha-khandait.csv")

#Let us ignore the news headlines that do not have any price movement

information in it, i.e. drop rows with "Price Sentiment" as 'none' df =

df[df["Price Sentiment"] != 'none']

print("Commodity News
Headlines")
display(df[["News","Price
Sentiment"]])

#The following piece of code
is used to clean the
headlines def
cleaner(impure_data):

temp
_list
= []
for
item
in

```
impu
re_d
ata:

    #finding
words which
start with @
item =
re.sub('@\S+', '',
item)

    #finding
words which start
with http     item
=
re.sub('http\S+\s*'
, '', item)

    #finding special characters, but not "emoji"
item = re.sub('[%s]' %
re.escape("""!"#$%&'()*+,-
./:;<=>?@[\]^_`{|}~"""), '', item)
temp_list.append(item)    return temp_list

#Let us create a simple
SVM model with tfidf
vectorizer def
headline_sentiment(df)
:    headlines =
df["News"]    polarity =
df["Price
Sentiment"].tolist()

  #cleaning headlines i.e. removing @mentions, http(s) links and
special characters such as punctuations    clean_headline =
cleaner(headlines)

  #initializing tf-idf vectorizer
tf_idfvectorizer =
```

TfidfVectorizer(sublinear_tf=True,
use_idf=True)

#splitting the data into train and test dataset in 70 : 30 ratio at random

X_train, X_test, y_train, y_test = train_test_split(clean_headline, polarity, test_size = 0.3)

train_corpus_tf_idf =
tf_idfvectorizer.fit_transform(X_train)
test_corpus_tf_idf =
tf_idfvectorizer.transform(X_test)

#using SVC package to initialize a classifier with Linear kernel and other default parameters

SVM_L = SVC(kernel= 'linear')

#fitting the sparse matrix in the classifier with their respective sentiments

SVM_L.fit(train_corpus_tf_idf, y_train)

#predicting the
sentiments for the test
dataset y_pred =
SVM_L.predict(test_corp
us_tf_idf)

#this prints accuracy score
for the test dataset
print("Testing
Accuracy:",accuracy_score(y_t
est,y_pred))

#this prints
confusion matrix for
the test dataset
labels =
np.unique(y_test) m
=
confusion_matrix(y_te
st,y_pred,

```
labels=labels)
print("\nConfusion
matrix on test data")
cm = pd.DataFrame(m,
index=labels,
columns=labels)
cm.index = "Actual: " +
cm.index
cm.columns =
"Predicted: " +
cm.columns
display(cm)

   #saving the data into a
csv file in the current
folder     temp_df =
pd.DataFrame()
temp_df["News"] =
X_test     temp_df["Actual
Price Sentiment"] =
y_test
temp_df["Predicted
Sentiment"] = y_pred
temp_df.to_csv("predicte
d.csv")

   print('Predictions

on Test Data are as

follows:')

display(temp_df)

return(tf_idfvectorizer

,SVM_L)


vectorizer,model = headline_sentiment(df)
```

Looking at the confusion matrix, it is clear that the performance on neutral will be poor.

Positive and negative headlines are likely to be identified correctly

```
# Testing the model vector =
vectorizer.transform(["Gold
expected to beat expectations."])
sentiment =
model.predict(vector)
print(sentiment)


#Trying sample headlines vector =
vectorizer.transform(["The price of
gold continues declining."])
sentiment = model.predict(vector)
print(sentiment)

#Trying sample headlines
vector =
vectorizer.transform(["Gold
price continues to improve."])
sentiment =
model.predict(vector)
print(sentiment)

#Trying sample headlines vector =
vectorizer.transform(["Gold price
expected to remain steady."])
sentiment = model.predict(vector)
print(sentiment)


#The following piece of code is used to clean the headlines
```

```
Commodity News Headlines
```

News	Price	Sentiment

0	april gold down 20 cents to settle at $1,116.1...	negative
1	gold suffers third straight daily decline	negative
2	Gold futures edge up after two-session decline	positive
4	Gold snaps three-day rally as Trump, lawmakers...	negative
5	Dec. gold climbs $9.40, or 0.7%, to settle at ...	positive
...	...	...
10565	gold seen falling from 3-week high this week	negative
10566	dominic frisby : now looks like a good time to...	positive
10567	Gold heading for worst week since November on ...	negative
10568	august gold up $7.60 at $878.80 an ounce on nymex	positive
10569	december gold down $1 at $749 an ounce on nymex	negative

8602 rows × 2 columns

```
Testing Accuracy: 0.9217357613328168

Confusion matrix on test data
```

	Predicted: negative	**Predicted: neutral**	**Predicted: positive**
Actual: negative	1070	8	76

Actual: neutral	17	75	24
Actual: positive	65	12	1234

```
Predictions on Test Data are as follows:
```

	News	**Actual Price Sentiment**	**Predicted Sentiment**
0	commodity futures sink gold down 52oz	negative	negative
1	december gold up 620 at 131610 an ounce	positive	positive
2	gold weakens on lack of support silver up	positive	positive
3	Dec gold settle at 134170oz down 3 or 02	negative	negative
4	Gold gains in Asia as Trump probe widens with ...	positive	positive
...	...	...	...
2576	gold reverses early gains drops more than 1	positive	negative
2577	gold futures close lower but gain almost 11 ye...	negative	positive
2578	April gold settles at 124970oz up 320 or 03	positive	positive

from sklearn.model_selection **2579** gold futures up 17 to 168850 an ounceimport train_test_split positive positive

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2, random_state=0)

2580 gold loses rs 250 on us rate hike chances now ... negative negative

print("X_train:", len(X_train)) 2581 rows × 3 columns

print(["X_test:"'positive',]len(X_test))

```
['negative']
['positive']
```

print("y
_train:",
len(y_tr
ain))
['neut
ral']
print("y
_test:",
len(y_te
st))

X_train: 8456

Milestone 4: Model Building

Milestone 4: Model Building

Activity 1: Training the model in multiple algorithms

Activity 1: Training the model in multiple algorithms

#Activity 1.1: Logistic
Regression model import
numpy as np import
pandas as pd from
sklearn.feature_extraction
.text import
TfidfVectorizer from
sklearn.model_selection
import train_test_split

from sklearn.metrics
import accuracy_score,
confusion_matrix from
sklearn.linear_model
import LogisticRegression
import re

df =
pd.read_csv("gold-

```
dataset-sinha-
khandait.csv") df =
df[df["Price
Sentiment"] != 'none']

#Data Preprocessing

def cleaner(impure_data):


temp
_list
= []
for
item
in
impu
re_d
ata:

        # Apply cleaning steps (remove mentions,
links, special characters, etc.)        item =
re.sub('@\S+', '', item)        item =
re.sub('http\S+\s*', '', item)        item =
re.sub('[%s]' % re.escape(""""!"#$%&'()*+,-
./:;<=>?@[\]^_`{|}~""""), '', item)
temp_list.append(item)     return temp_list

#Create a
Logistic
Regression
Model def
headline_se
ntiment_log
istic(df):

    headlines =
df["News"]
polarity =
df["Price
```

Sentiment"].tolis
t()

Clean
headlines
clean_headli
ne =
cleaner(hea
dlines)

Initialize tf-idf vectorizer
tf_idfvectorizer =
TfidfVectorizer(sublinear_tf=True,
use_idf=True)

Split data into train and test

X_train, X_test, y_train, y_test = train_test_split(clean_headline, polarity,
test_size=0.3)

Transform headlines using
tf-idf train_corpus_tf_idf =
tf_idfvectorizer.fit_transform(
X_train) test_corpus_tf_idf
=
tf_idfvectorizer.transform(X_t
est)

Initialize
Logistic Regression
model log_reg =
LogisticRegression()

Fit the model with training
data
log_reg.fit(train_corpus_tf_idf,
y_train)

Predict
sentiments for the
test dataset

```
y_pred =
log_reg.predict(test_
corpus_tf_idf)

   # Print accuracy score for
the test dataset
print("Testing Accuracy:",
accuracy_score(y_test,
y_pred))

   # Print confusion
matrix for the test
dataset     labels =
np.unique(y_test)     m
=
confusion_matrix(y_tes
t, y_pred, labels=labels)
print("\nConfusion
matrix on test data")
cm = pd.DataFrame(m,
index=labels,
columns=labels)
cm.index = "Actual: " +
cm.index     cm.columns
= "Predicted: " +
cm.columns
display(cm)

   # Save the data into a
CSV file in the current
folder     temp_df =
pd.DataFrame()
temp_df["News"] =
X_test
temp_df["Actual Price
Sentiment"] = y_test
temp_df["Predicted
Sentiment"] = y_pred
```

temp_df.to_csv("predict
ed.csv")

print('Predictions on Test Data

are as follows:')

display(temp_df) return

tf_idfvectorizer, log_reg

vectorizer_logistic,

model_logistic =

headline_sentiment_logistic(df)

#Making Predictions with the Logistic
Regression Model vector =
vectorizer_logistic.transform(["Gold
expected to beat expectations."])
sentiment =
model_logistic.predict(vector)
print(sentiment)

```
Testing Accuracy: 0.9259976753196435

Confusion matrix on test data
```

	Predicted: negative	Predicted: neutral	Predicted: positive
Actual: negative	1094	4	66
Actual: neutral	28	61	38
Actual: positive	53	2	1235

```
Predictions on Test Data are as follows:
```

News	Actual Price Sentiment	Predicted Sentiment

0	charts gold and silver still weak qrtly mthl...	negative	negative
1	gold up over 050 per cent in morning trade	positive	positive
2	Gold logs worst daily loss in 10 weeks after s...	negative	negative
3	Gold extends gains as Fed minutes reveal weak ...	positive	positive
4	Gold silver gain on more bets the Fed will tak...	positive	positive
...	...	...	...
2576	gold prices close out week 53 lower	negative	negative
2577	gold breaches new record above 1084 an ounce	positive	positive
2578	gold ends 05 higher at 115960 an ounce	positive	positive
2579	crude oil gold stay up after income data	positive	positive
2580	gold futures slide 022 on weak global pointers	negative	negative

2581 rows × 3 columns

```
['negative']
```

import pandas as pd from

sklearn.model_selection

import train_test_split

from

sklearn.feature_extraction

.text import

TfidfVectorizer from

sklearn.linear_model

import LogisticRegression

import joblib df =

pd.read_csv("gold-

```
dataset-sinha-
khandait.csv")

# Assuming you have 'X' as
text data and 'y' as sentiment
labels X = df['News']

y = df['Price Sentiment']

# Load your gold news dataset with labeled sentiments (positive, negative,
neutral)

# Replace with your actual dataset loading code

# Assuming you have 'X' as text data and 'y' as sentiment labels

# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create a TfidfVectorizer
for text preprocessing
vectorizer =
TfidfVectorizer()

X_train_tfidf = vectorizer.fit_transform(X_train)

# Train a Logistic
Regression classifier
logistic_model =
LogisticRegression(max
_iter=1000)
logistic_model.fit(X_tr
ain_tfidf, y_train)

# Save the trained model and
vectorizer to files
joblib.dump(logistic_model,
'logistic_model.pkl')
joblib.dump(vectorizer,
'tfidf_vectorizer.pkl')
```

Milestone 5: Model Deployment

Activity 1: Save the best model

Activity 2: Integrate with Web Framework

Activity 2.1: Building Html Pages:

Activity 2.2: Build Python code: Activity 2.3: Run the web application

Milestone 6: Model Deployment

Activity 1: Save the best model

Saving the best model after comparing its performance using different evaluation metrics meansselecting the model with the highest performance This can be useful in avoiding the need to retrainthe model every time it is needed and also to be able to use it in the future.

```
In [ ]:
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
import joblib

df = pd.read_csv("gold-dataset-sinha-khandait.csv")

# Assuming you have 'X' as text data and 'y' as sentiment labels
X = df['News']
y = df['Price Sentiment']
# Load your gold news dataset with labeled sentiments (positive, negative, neutral)
# Replace with your actual dataset loading code

# Assuming you have 'X' as text data and 'y' as sentiment labels

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a TfidfVectorizer for text preprocessing
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)

# Train a Logistic Regression classifier
logistic_model = LogisticRegression(max_iter=1000)
logistic_model.fit(X_train_tfidf, y_train)

# Save the trained model and vectorizer to files
joblib.dump(logistic_model, 'logistic_model.pkl')
joblib.dump(vectorizer, 'tfidf_vectorizer.pkl')
```

```
Out[4]: ['tfidf_vectorizer.pkl']
```

Activity 2: Integrate with Web Framework

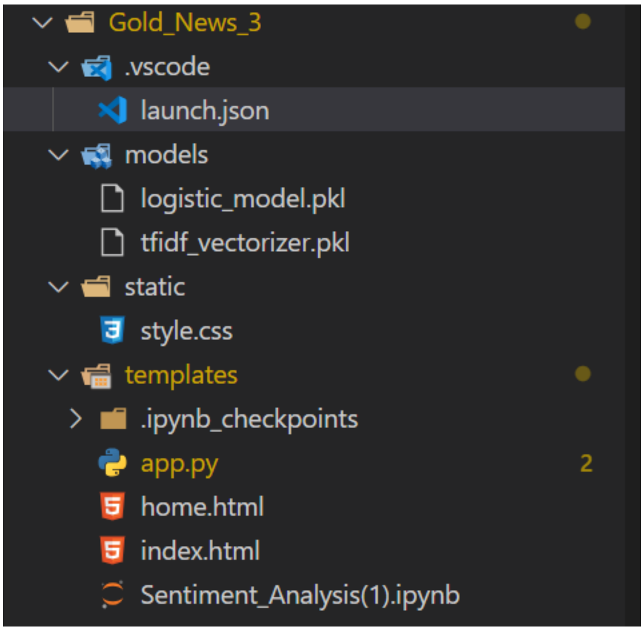
In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

- **Activity 2.1: Building Html Pages:**
- For this project create two HTML files namely

- Index.html
- and save them in the templates folder. Refer this for templates, static and python file



launch.json × | home.html ● | index.html Gold_News_3\... ● | Sentiment_

Gold_News_3 > .vscode > launch.json > [ ] configurations > { } 0

```json
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Python: Flask",
            "type": "python",
            "request": "launch",
            "program": "${Gold_News_3}/app.py",
            "console": "integratedTerminal",
            "env": {
                "FLASK_APP": "app.py",
                "FLASK_ENV": "development"
            },
            "cwd": "${Gold_News_3}",
            "stopOnEntry": false,
            "jinja": true
        }
    ]
}
```

style.css ● | logistic_model.pkl | home.html

Gold_News_3 > static > style.css > ...

```css
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    text-align: center;
}

.container {
    background-color: #ffffff;
    max-width: 500px;
    margin: 0 auto;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.
}

.heading {
    font-size: 24px;
    margin-top: 0;
    color: #333;
}

.subheading {
    font-size: 16px;
    margin-top: 10px;
    color: #666;
```

style.css ● logistic_model.pkl home.html ● index.html Gold_News_3\... ● app.py Gold

Gold_News_3 > templates > app.py

```python
from flask import Flask, render_template, request
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
import joblib
import re


import os  # Import the 'os' module

# ...

# Load your trained Logistic Regression model (logistic_model.pkl)
model_path = os.path.join("models", "logistic_model.pkl")  # Specify the correct path
model = joblib.load(model_path)
#model = jo                                   tic_model.pkl')


# Load the TfidfVectorizer for text preprocessing
vectorizer_path = os.path.join("models", "tfidf_vectorizer.pkl")  # Specify the corre
vectorizer = joblib.load(vectorizer_path)

# ...


app = Flask(__name__)
```

"Tfidf": Unknown word. cSpell

View Problem   No quick fixes

style.css ● logistic_model.pkl home.html ● index.html Gold_News_3\... ● app.py Gold_News_3\... 2

Gold_News_3 > templates > home.html > html > body > div.container > a.analyze-link

```html
<!DOCTYPE html>
<html>
<head>
    <title>Gold News Sentiment Analysis</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <h1 class="heading">Gold News Sentiment Analysis</h1>
        <p class="subheading">Analyze the sentiment of gold news articles.</p>
        <a class="analyze-link" href="{{ url_for('index') }}">Start Analysis</a>
    </div>
</body>
</html>
```

style.css ● logistic_model.pkl home.html ● index.html Gold_News_3\... ● app.py Gold_News_3\... 2

Gold_News_3 > templates > index.html > ...

```html
<!DOCTYPE html>
<html>
<head>
    <title>Gold News Analysis</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <h1 class="heading">Gold News Analysis</h1>
        <p class="subheading">Enter a gold news article to analyze its sentiment.</p>
        <form method="POST" action="/analyze">
            <textarea class="input-text" name="news_text" rows="4" cols="50" placeholder="Enter gol
            <br>
            <input class="analyze-button" type="submit" value="Analyze">
        </form>
    </div>
</body>
</html>
```

Conclusion: Analyzing Market Sentiment in Gold News: Concluding Remarks
bold text

In conclusion, the sentiment analysis project focusing on gold news provides valuable insights for traders and investors in the gold market. By applying machine learning techniques and NLP concepts, we have been able to gauge the emotional tone of news articles, classifying them as positive, negative, or neutral. The results of our analysis can be instrumental in decision-making within the gold market. Positive sentiment may indicate increased demand for the precious metal, potentially driving up prices. Conversely, negative sentiment could signify a decline in demand or a downturn in the market.

This project underscores the importance of sentiment analysis as a tool to unearth market sentiments and facilitate data-driven nancial decisions. While we have applied basic models like Logistic Regression, there is room for further enhancement using advanced techniques and models, such as deep learning and ensemble methods. The project contributes to a deeper understanding of the gold market's sentiment