SENTIMENTAL ANALYSIS OF COMMODITY NEWS(GOLD)

MINIPROJECT

SUBMITTED BY:
ABHISHAKE GOTTAM(21UK1A6616)
SANGEETHA VENISHETTI(21UK1A6633)
MOHAMMAD REHANA TABASSUM(21UK1A6633)
RAGHUPATHI BHUKYA(21UK1A6656)

Under the guidance of Ms. K.Navya

DEPARTMENT :
COMPUTER SCIENCE OF ENGINEERING (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

VAAGDEVI ENGINEERING COLLEGE

BOLLIKUNTA, WARANGAL (T. S) – 506005

2021-2025

VAAGDEVI ENGINEERING COLLEGE

WARANGAL

CERTIFICATE OF COMPLETION

INDUSTRY ORIENTED MINI PROJECT

MINI PROJECT

This is to certify that the mini project report entitled by "SENTIMENTAL ANALYSIS OF COMMODITY NEWS(GOLD)" is being submitted by G.ABHISHAKE(21UK1A6616), V.SANGEETHA(21UK1A6637),MD.REHANA TABASSUM(21UK1A6633),B.RAGHUPATHI

(21UK1A6656) in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in computer science and engineering(AI&ML).

Project guide:

Ms. K.Navya

Head of the Department

Dr. SHARMILA REDDY

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. PRASAD RAO**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-1 in the institute.

We extend our heartfelt thanks to **Dr. K. SHARMILA**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-1.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, For their constant supervision as well as for providing necessary information regarding the UG Project Phase-1and for their support in completing the UG Project Phase-1.

We express heartfelt thanks to the guide, **K.Navya** , Assistant professor, Department of CSE for his constant support and giving necessary guidance For completion of this UG Project Phase-1.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

Gottam Abhishake **(21UK1A6616)**

Venishetti Sangeetha **(21UK1A6637)**

MD.Rehana Tabassum **(21UK1A6633)**

Bhukya Raghupathi **(21UK1A6656)**

ABSTRACT

Gold is one of the valuable materials that is used for funding trading purchases. Nowadays, more investors are interested in gold investments due to the sudden increase in gold prices. However, transactions involving gold are risky, the price of gold fluctuates wildly due to the unpredictability of the gold market. Hence, there is a need for the development of gold price prediction scheme to assist and support investors, marketers, and financial institutions in making effective economic and monetary decisions. This project analyzes the correlation between gold price movements and sentiments of Arabic tweets in Egypt. After performing sentiment analysis on these tweets, three supervised machine learning algorithms were used for predicting the gold price. The algorithms include Multiple linear regression, Ridge regression, and Lasso regression. The result of this work shows that the Lasso regression model performs better than the other two models. However, it is concluded that there is a weak correlation between gold prices and Twitter data. Therefore, gold prices cannot be accurately predicted using Twitter data alone.

TABLE OF CONTENTS:-

1.INTRODUCTION

1.1.OVERVIEW

This study explores the application of sentiment analysis to understand the public perception and market trends related to gold. Given gold's historical significance as a financial asset and its role as a hedge against economic instability, analyzing sentiment around it can provide valuable insights for investors and policymakers. We employ natural language processing (NLP) techniques to analyze large volumes of textual data from social media, financial news, and forums to gauge the sentiment toward gold. Our methodology includes data collection, preprocessing, sentiment scoring, and trend analysis. The results indicate a strong correlation between sentiment shifts and gold price fluctuations, highlighting the potential of sentiment analysis as a predictive tool for gold market movements. This research contributes to the financial literature by demonstrating the viability of sentiment analysis in enhancing investment strategies and understanding market dynamics.

1.2.OBJECTIVES:

Sentiment Analysis: Analyze public sentiment towards gold from multiple data sources.

Trend Identification: Identify trends and patterns in sentiments that may influence gold prices.

Market Insights: Provide actionable insights for investors and market participants.

Data Correlation: Explore the relationship between sentiment data and gold price movements.

1.3.METHODOLOGY:

Data Collection: Gather textual data related to gold from social media, news articles, and financial reports.

Data Preprocessing: Clean and preprocess the data, including tokenization, stopword removal, and normalization.

Sentiment Analysis: Apply natural language processing (NLP) techniques to classify the sentiment as positive, negative, or neutral.

Machine Learning Models: Use models such as logistic regression, SVM, and LSTM for sentiment classification.

Correlation Analysis: Analyze the relationship between sentiment trends and historical gold price data.

Aim Of The Project:

The aim of a project focused on sentiment analysis of gold would generally be to understand and quantify the sentiments and opinions expressed about gold in various text sources. This can help in making more informed decisions related to gold investments, trading, and market predictions. The project would typically involve the following objectives:

1. Data Collection:

- Gather text data from diverse sources such as news articles, social media, financial reports, forums, and blogs where gold is discussed.

2. Preprocessing:

- Clean and preprocess the collected data to remove noise and irrelevant information.
- Tokenize the text and perform tasks like stemming, lemmatization, and stop-word removal.

3. Sentiment Analysis:

- Apply natural language processing (NLP) techniques to analyze the sentiments expressed in the text.
- Use sentiment analysis models to classify the sentiments as positive, negative, or neutral.
- Utilize machine learning or deep learning models to improve the accuracy of sentiment classification.

4. Trend Analysis:

- Analyze the sentiment trends over time to identify patterns and correlations with gold price movements.
- Study how different events and news impact public sentiment and subsequently gold prices.

5. Visualization:

- Create visualizations to represent the sentiment trends and their relationship with gold prices.
- Use dashboards and charts to present the findings in an easily interpretable format.

6. Insights and Predictions:

- Derive actionable insights from the sentiment data.
- Develop predictive models to forecast gold price movements based on sentiment trends and other relevant factors.

7. **Report and Recommendations**:

- Compile a comprehensive report summarizing the findings, methodologies, and recommendations.
- Provide strategies for investors, traders, and stakeholders to leverage sentiment analysis in their decision-making processes.

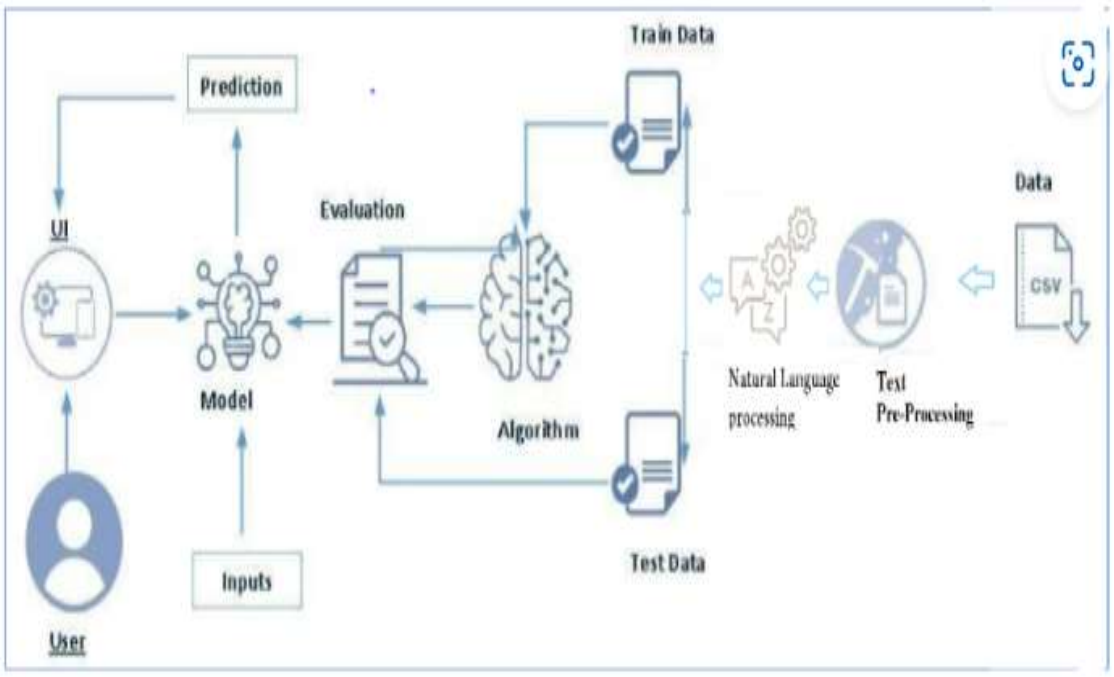
Outcome:

By achieving these objectives, the project would aim to provide a deeper understanding of how public sentiment influences the gold market and offer tools for better investment and trading strategies.

THEORITICAL ANALYSIS

SENTIMENTAL ANALYSIS OF COMMODITY NEWS(GOLD)

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.



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

- Specify the business problem
- Business requirements
- Literature Survey

- Social or Business Impact.

Data Collection & Preparation

- Collect the dataset
- Data Preparation

Exploratory Data Analysis

- Descriptive statistical
- Visual Analysis

Model Building

- Training the model in multiple algorithms
- Testing the model

Performance Testing

- Testing model with multiple evaluation metrics

Model Deployment

- Save the best model
- Integrate with Web Framework

Prior Knowledge:

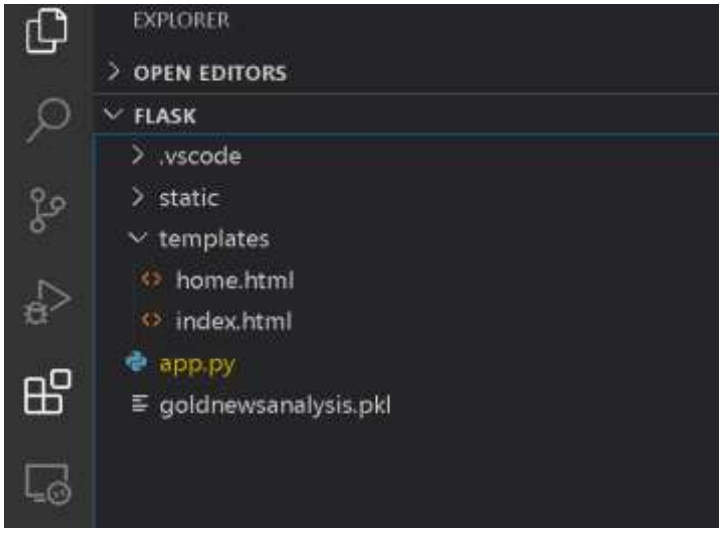
You must have prior knowledge of following topics to complete this project.

ML Concepts

Supervised learning: https://www.javatpoint.com/supervised-machine-learning
Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification- algorithm
Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to- understand-the-math-behind-xgboost/
Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model- evaluation-error-metrics/
NLP:-https://www.javatpoint.com/nlp
Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create the Project folder which contains files as shown below



We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
goldnewsanalysis.pkl is our saved model. Further we will use this model for flask integration.
Training folder contains a model training file.

Define Problem / Problem Understanding

Problem understanding is the process of gaining a clear understanding of the problem, its causes, and its impact, setting the stage for effective problem-solving.

Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Collect The Dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.
Note: There are a number of techniques for understanding the data. But here we have used

some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image.

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import re
        import string
        from sklearn.model_selection import train_test_split
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import accuracy_score
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.svm import SVC
        import pickle
```

Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

```
data.head()
```

	Dates	URL	News	Price Direction Up	Price Direction Constant	Price Direction Down	Asset Comparision	Past Information	Future Information	Price Sentiment
0	28-01-2016	http://www.marketwatch.com/story/april-gold-dc...	april gold down 20 cents to settle at $1,116.1...	0	0	1	0	1	0	negative
1	13-09-2017	http://www.marketwatch.com/story/gold-prices-s...	gold suffers third straight daily decline	0	0	1	0	1	0	negative
2	26-07-2016	http://www.marketwatch.com/story/gold-futures-...	Gold futures edge up after two-session decline	1	0	0	0	1	0	positive
3	28-02-2018	https://www.metalsdaily.com/link/277199/dent-r...	dent research : is gold's day in the sun comin...	0	0	0	0	0	1	none
4	06-09-2017	http://www.marketwatch.com/story/gold-steadies...	Gold snaps three-day rally as Trump, lawmakers...	0	0	1	0	1	0	negative

```
data.tail()
```

	Dates	URL	News	Price Direction Up	Price Direction Constant	Price Direction Down	Asset Comparision	Past Information	Future Information	Price Sentiment
10565	07-01-2013	https://www.moneycontrol.com/news/business/mar...	gold seen falling from 3-week high this week	0	0	1	0	1	0	negative
10566	27-09-2018	https://www.metalsdaily.com/link/284468/domini...	dominic frisby : now looks like a good time to...	1	0	0	0	0	1	positive
10567	03-03-2017	https://www.thehindubusinessline.com/markets/g...	Gold heading for worst week since November on ...	0	0	1	0	1	0	negative
10568	11-06-2008	http://www.marketwatch.com/story/august-gold-u...	august gold up 7.60at 878.80 an ounce on nymex	1	0	0	0	1	0	positive

Data Preparation

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling Outliers

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 2.1: Handling missing values

Let’s find the shape of our dataset first. To find the shape of our data, the df.shape method is used. To find the data type, df.info() function is used

```
In [6]: #to check shape of data set
        df.shape

Out[6]: (10570, 10)
```

```
In [7]: #checking the information of dataset
        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
```

For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function.

From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
In [4]: df.isnull().sum()

Out[4]: 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
```

Activity 2.2: Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using manual encoding with the help of list comprehension.

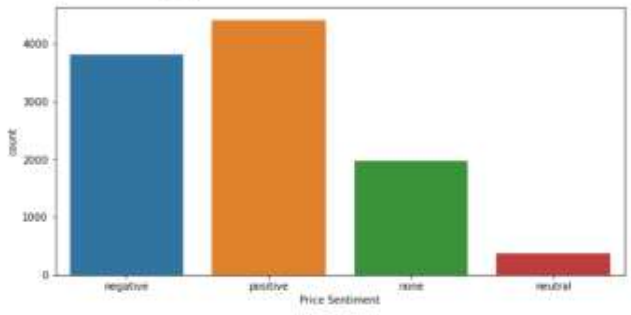
• In our project, categorical features is Price sentiment column but no need to convert in this project because the we are doing here natural language processing but I have shared how to convert the categorical values to numerical

```
In [19]: df['Price Sentiment'].value_counts()

Out[19]: positive    4412
         negative    3814
         none        1968
         neutral      376
         Name: Price Sentiment, dtype: int64

In [20]: df['Price Sentiment'].unique()

Out[20]: array(['negative', 'positive', 'none', 'neutral'], dtype=object)

In [21]: df['Price Sentiment']=df['Price Sentiment'].map({'negative':1,'positive':2,'neutral':3,'none':2})
```

Activity 2.3: Handling Imbalance Data

With the help of count plot are visualized. And here we are going to find bars with imbalance data of Price sentiment. but in this dataset no need to perfom any analysis as we are using data as it is

```
In [12]: plt.figure(figsize=(10,5))
         sns.countplot(df['Price Sentiment'])
         df['Price Sentiment'].value_counts()

Out[12]: positive    4412
         negative    3814
         none        1968
         neutral      376
         Name: Price Sentiment, dtype: int64
```



Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a crucial step in data analysis that focuses on understanding and visualizing the data to extract meaningful insights. By examining the dataset, its structure, and the relationships between variables, EDA helps identify patterns, outliers, and trends. Through summary statistics, visualizations, and statistical techniques, EDA enables data scientists to gain a deeper understanding of the data's distribution, central tendencies, and dispersion. It also helps in identifying missing values, outliers, or anomalies that may impact subsequent analyses. EDA serves as a foundation for hypothesis generation, feature selection, and model building, guiding the data analysis process towards more informed decision-making.

Descriptive Analysis

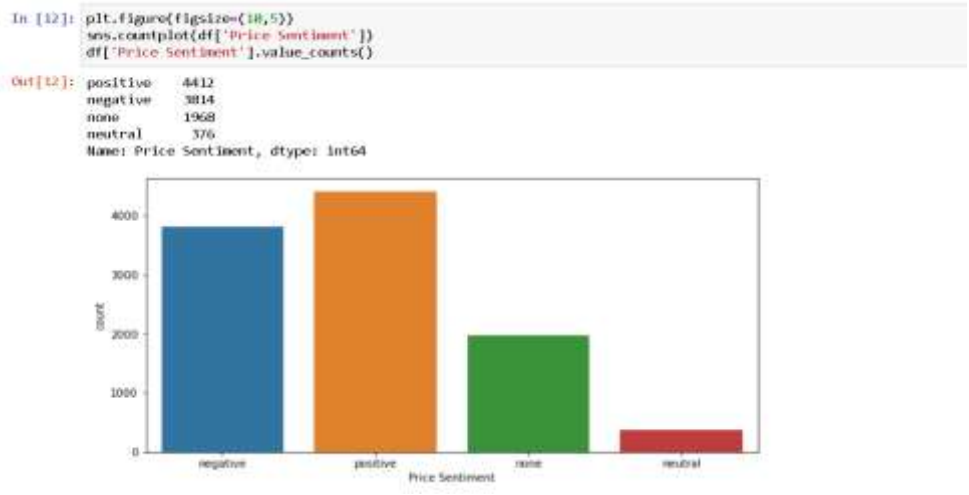
Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features

Visual Analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Activity 2.1: Univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and countplot.
In our dataset we have some categorical features. With the countplot function, we are going to count the unique category in those feature

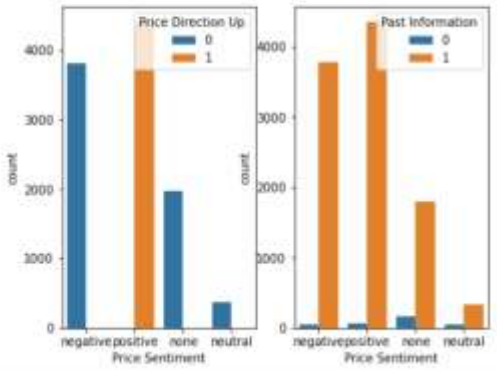


Activity 2.2: Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we are visualizing the relationship between price sentimnet & price direction up, price sentimnet & past Infromation.

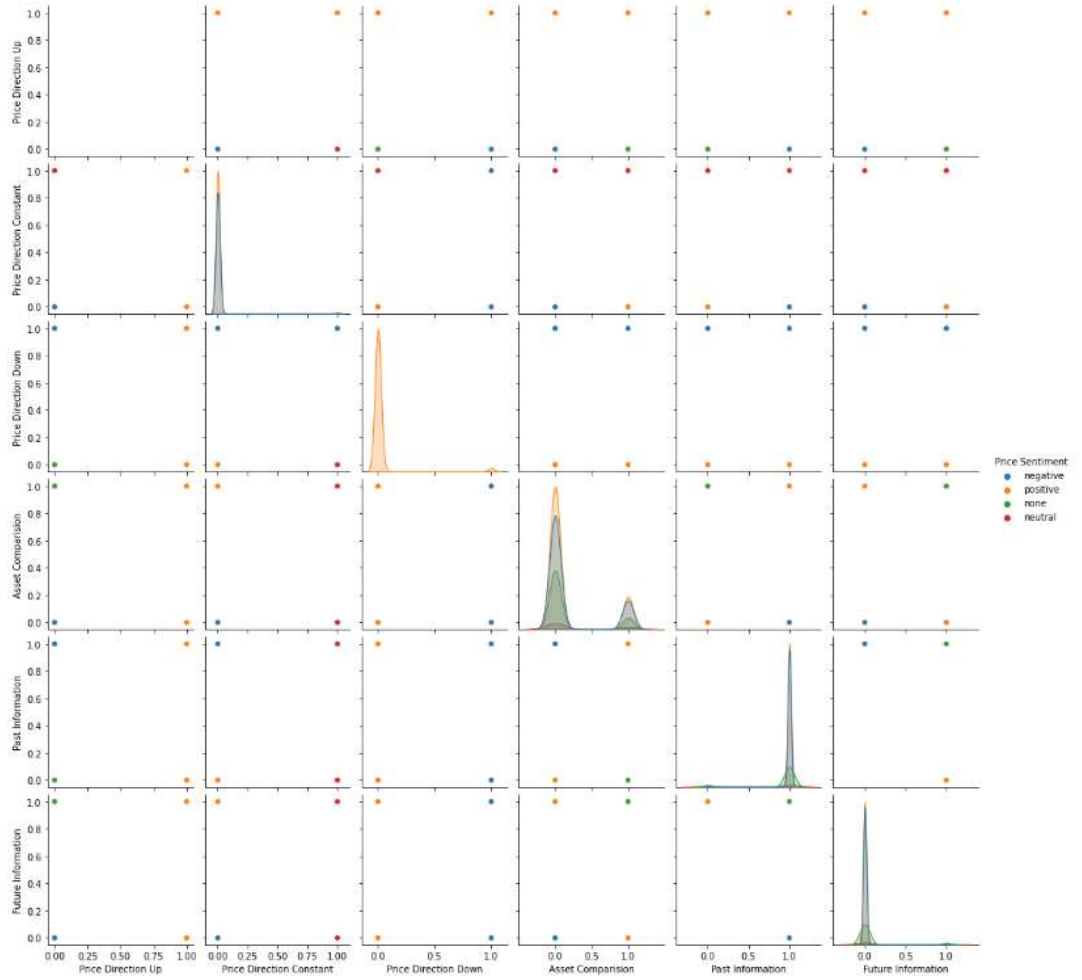
Countplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

```
In [9]: #visualising two columns against each other
        plt.figure(figsize=(10,5))
        plt.subplot(131)
        sns.countplot(df['Price Sentiment'], hue=df['Price Direction Up'])
        plt.subplot(132)
        sns.countplot(df['Price Sentiment'], hue=df['Past Information'])
        #plt.show
Out[9]: <AxesSubplot:xlabel='Price Sentiment', ylabel='count'>
```



Activity 2.3: Multivariate analysis

```
In [8]: sns.pairplot(df, hue='Price Sentiment')
Out[8]: <seaborn.axisgrid.PairGrid at 0x27739237820>
```



Text pre-processing (python packages)
Text pre-processing is a crucial step in Natural Language Processing (NLP) and Information Retrieval (IR) tasks. The goal is to convert raw text into a more meaningful and manageable representation for further analysis.

In Python, there are several packages that provide support for text pre-processing operations. Some of the most common ones are:
NLTK (Natural Language Toolkit)- It is one of the most widely used NLP libraries in Python. It provides tools for tokenization, stemming, lemmatization, stop-word removal, and more.

Re (regular expressions) – This module provides support for working with regular expressions. It is commonly used to perform string operations such as removing punctuation, white spaces, and other non-alphanumeric characters.

```
In [14]: #Applying the text cleaning
         import re
         import string

         #This function converts to lower-case, removes square bracket, removes numbers and punctuation
         def text_clean_1(text):
             text = text.lower()      #converts to lower-case or upper case
             text = re.sub('\[.*?\]', '', text)  #it will remove .,?,"", []
             text = re.sub('[%s]' % re.escape(string.punctuation), '', text)     # #it will remove "",.
             text = re.sub('\w*\d\w*', '', text)
             text = re.sub('[‘’“”…]', '', text)  #it will remove ‘’, ‘’, “”, ”“, and …,
             text = re.sub('\n', '', text)
             return text

         Cleaned_News = lambda x: text_clean_1(x)
```

After Text Pre-Processing:- After text pre-processing, the data will typically look much cleaner and more manageable. The specific form it takes will depend on the specific operations performed during pre-processing and here is below fig after cleaning the text

```
In [15]: #the updated text here
         df['Cleaned_News'] = pd.DataFrame(df.News.apply(Cleaned_News))
         df.head(10)
```

Out[15]:

	News	Price Sentiment	Cleaned_News
0	april gold down 20 cents to settle at $1,116.1...	negative	april gold down cents to settle at
1	gold suffers third straight daily decline	negative	gold suffers third straight daily decline
2	Gold futures edge up after two-session decline	positive	gold futures edge up after twosession decline
3	dent research . is gold's day in the sun comin...	none	dent research is golds day in the sun coming ...
4	Gold snaps three-day rally as Trump, lawmakers ...	negative	gold snaps threeday rally as trump lawmakers r...
5	Dec. gold climbs $9.40, or 0.7%, to settle at ...	positive	dec gold climbs or to settle at
6	gold falls by rs 26 on sluggish demand, global ...	negative	gold falls by rs on sluggish demand global cues
7	Gold futures fall for the session, but gain fo...	positive	gold futures fall for the session but gain for ...
8	Gold struggles; silver slides, base metals falter	neutral	gold struggles silver slides base metals falter
9	april gold holds slight gain, up $2.60, or 0.2...	positive	april gold holds slight gain up or at

Splitting data into train and test

Now let’s split the Dataset into train and test sets. First split the dataset into x and y and then split the data set
Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
In [18]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=0)

         print("x_train:",len(x_train))
         print("x_test:",len(x_test))
         print("y_train:",len(y_train))
         print("y_test:",len(y_test))
```

```
x_train: 8456
x_test: 2114
y_train: 8456
y_test: 2114
```

Model Building

Machine Learning Model Building is the process of creating predictive or analytical models using machine learning algorithms. It involves data preprocessing, feature engineering, selecting an appropriate algorithm, training the model, evaluating its performance, and iterating to improve accuracy and robustness.

Training The Model In Multiple Algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying three classification algorithms. The best model is saved based on its performance.

Activity 1.1: Logistic Regression model

A function named Logistic regression is created and train and test data are passed as the parameters. Inside the function, Logistic Regression algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model with confusion matrix and accuracy score

The Tf-idf (Term frequency- inverse document frequency) is a commonly used numerical representation of text data that is used in NLP and IR tasks. The goal of Tf-idf is to represent the importance of a word in a document while taking into account the frequency of the word in the entire corpus.
Using a pipeline has several benefits. First, it makes the process of building a machine learning model more efficient, as the steps can be automated and repeated with ease. Second, it reduces the risk of errors, as each step in the pipeline is clearly defined and less prone to manual mistakes

Model building with Logistic Regression

```
In [20]: from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn.linear_model import LogisticRegression
         tvec = TfidfVectorizer()
         clf2 = LogisticRegression()

In [21]: from sklearn.pipeline import Pipeline
         model = Pipeline([('vectorizer',tvec),('classifier',clf2)])
         model.fit(x_train,y_train)
         from sklearn.metrics import confusion_matrix
         predictions = model.predict(x_test)
         pred_train=model.predict(x_train)
         confusion_matrix(predictions, y_test)

Out[21]: array([[701,  15,  26,  29],
                [  1,  50,   1,   2],
                [ 31,   9, 330,  48],
                [ 36,  15,  34, 786]], dtype=int64)
```

Activity 1.2: SVM (Support Vector machine)

A function named svm is created and train and test data are passed as the parameters. Inside the function, svc algorithm is initialized and training data is passed to the model with fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done. and it performed same as logistic regression model here we using the svm algorithms remaining all are the same process as above

Model building with SVM ¶

In [22]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
svm=SVC(kernel='linear')
tvec = TfidfVectorizer()
```

In [23]:
```
# from sklearn.pipeline import Pipeline
model2 = Pipeline([('vectorizer',tvec),('classifier',svm)])
model2.fit(x_train,y_train)
from sklearn.metrics import confusion_matrix
predictions2 = model.predict(x_test)
pred2_train=model.predict(x_train)
confusion_matrix(predictions2, y_test)
```

Out[23]:
```
array([[701,  15,  26,  29],
       [  1,  50,   1,   2],
       [ 31,   9, 330,  48],
       [ 36,  15,  34, 786]], dtype=int64)
```

Testing The Model

Here we have tested with Logistic regression and Svm algorithms. With the help of predict() function.

In [26]:
```
example = ["gold to trade in 28670-29160 range: achievers equities"]
result = model.predict(example)
print(result)
```

```
['neutral']
```

In [27]:
```
example = ["gold to trade in 28670-29160 range: achievers equities"]
result = model2.predict(example)
print(result)
```

```
['neutral']
```

In [28]:
```
example = ["can investment in gold, sensex & ppfs give the same returns?"]
result = model.predict(example)
print(result)
```

```
['none']
```

In [29]:
```
example = ["can investment in gold, sensex & ppfs give the same returns?"]
result = model2.predict(example)
print(result)
```

```
['none']
```

Performance Testing

Machine Learning Performance Testing evaluates the accuracy and effectiveness of trained models by comparing their predictions against known values. It measures performance using metrics like accuracy, precision, recall, and identifies issues like overfitting or underfitting. It helps in selecting the best model and improving its overall performance for real-world applications.

Testing Model With Multiple Evaluation Metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

Activity 1.1: Compare the model

For comparing the below two models, with their accuracy score

```
#for Logistic Regression
from sklearn.metrics import classification_report
# assume y_train and pred_train are your true and predicted labels, respectively
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

    negative       0.91      0.91      0.91       769
     neutral       0.93      0.56      0.70        89
        none       0.79      0.84      0.82       391
    positive       0.90      0.91      0.91       865

    accuracy                           0.88      2114
   macro avg       0.88      0.81      0.83      2114
weighted avg       0.88      0.88      0.88      2114
```

```
#for SVM
from sklearn.metrics import classification_report
# assume y_train and pred2_train are your true and predicted labels, respectively
print(classification_report(y_test, predictions2))
```

```
              precision    recall  f1-score   support

    negative       0.92      0.90      0.91       769
     neutral       0.81      0.64      0.72        89
        none       0.79      0.85      0.82       391
    positive       0.90      0.90      0.90       865

    accuracy                           0.88      2114
   macro avg       0.86      0.82      0.84      2114
weighted avg       0.88      0.88      0.88      2114
```

Comparing Model Accuracy Before & After Applying Hyperparameter Tuning

After seeing, the results of models are displayed as output. From the two models which are Logistic Regression and SVM both models are performing well & Hyperparameter tuning For this project (it is not required)

```
In [26]: example = ["gold to trade in 28670-29160 range: achievers equities"]
         result = model.predict(example)
         print(result)

         ['neutral']

In [27]: example = ["gold to trade in 28670-29160 range: achievers equities"]
         result = model2.predict(example)
         print(result)

         ['neutral']

In [28]: example = ["can investment in gold, sensex & ppfs give the same returns?"]
         result = model.predict(example)
         print(result)

         ['none']

In [29]: example = ["can investment in gold, sensex & ppfs give the same returns?"]
         result = model2.predict(example)
         print(result)

         ['none']
```

```
In [24]: #Logistic Regression
         from sklearn.metrics import accuracy_score
         print("Accuracy_test : ", accuracy_score(predictions, y_test))
         print("Accuracy_train : ", accuracy_score(pred_train, y_train))

         Accuracy_test :  0.8831598864711447
         Accuracy_train :  0.9331835383159887
```

```
In [25]: #SVM
         #from sklearn.metrics import accuracy_score
         print("Accuracy_test: ", accuracy_score(predictions2, y_test))
         print("Accuracy_train : ", accuracy_score(pred2_train, y_train))

         Accuracy_test:  0.8831598864711447
         Accuracy_train :  0.9331835383159887
```

Model Deployment

Flask Model Deployment involves deploying a saved machine learning model using the Flask framework. It includes initializing a Flask application, loading the saved model, defining routes and endpoints, and deploying the application on a web server or cloud platform. This enables users to interact with the deployed model in real-time via a browser or API calls.

Save The Best Model

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

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 entire files link for templates, static and python file

Activity 2.2: Build Python code:

Import the libraries

```
from flask import Flask, render_template, url_for,request,redirect, session
import pickle
import os
import re
from newsapi import NewsApiClient
```

Render HTML page:

```
@app.route('/',methods=['POST', 'GET'])
def homepage():
    return render_template('index.html')
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.
In the above example, '/' URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered.
Whenever you enter the values from the html page the values can be retrieved using POST Method.
Retrieves the value from UI:

```
@app.route('/prediction',methods=['POST', 'GET'])
def predictionpage():
    if request.method == 'POST':
        newsline = request.form["newheadline"]
        pred=[newsline]
        output=model.predict(pred)
        print(output)
        if output==['positive']:
            return render_template('index.html',output='upward movement in gold price')
        if output==['negative']:
            return render_template('index.html',output='downward movement in gold price')
        if output==['neutral']:
            return render_template('index.html',output='steady movement in gold price')
        if output==['none']:
            return render_template('index.html',output='this news headline is not related to gold news')
    return render_template('index.html')
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.
Main Function:

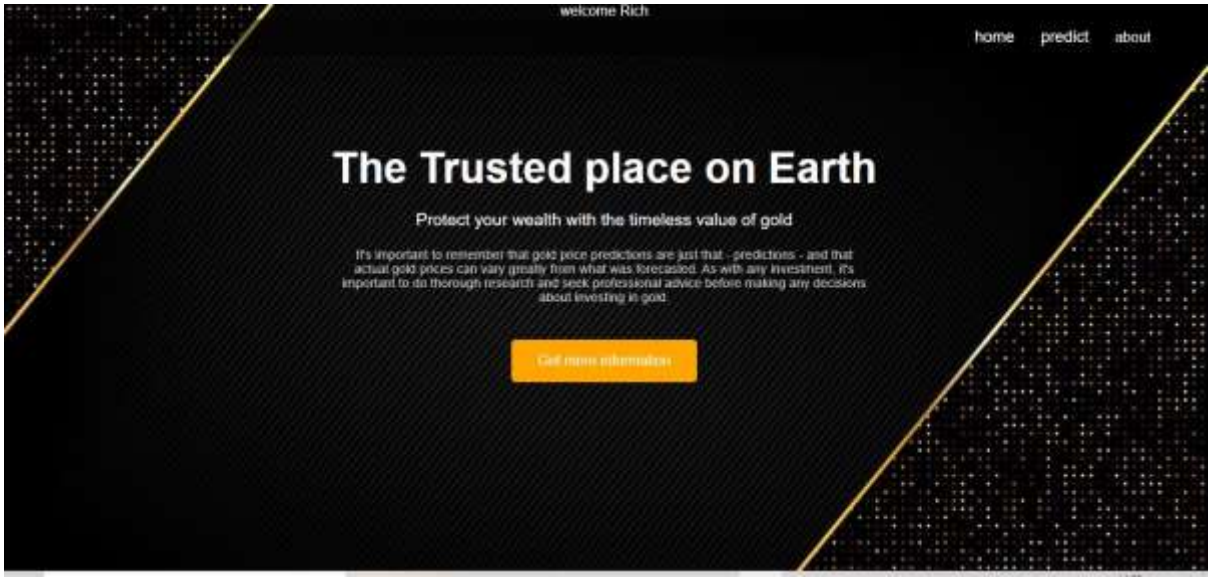
```
if __name__=='__main__':
    app.run(debug=True)
```

Activity 2.3: Run the web application

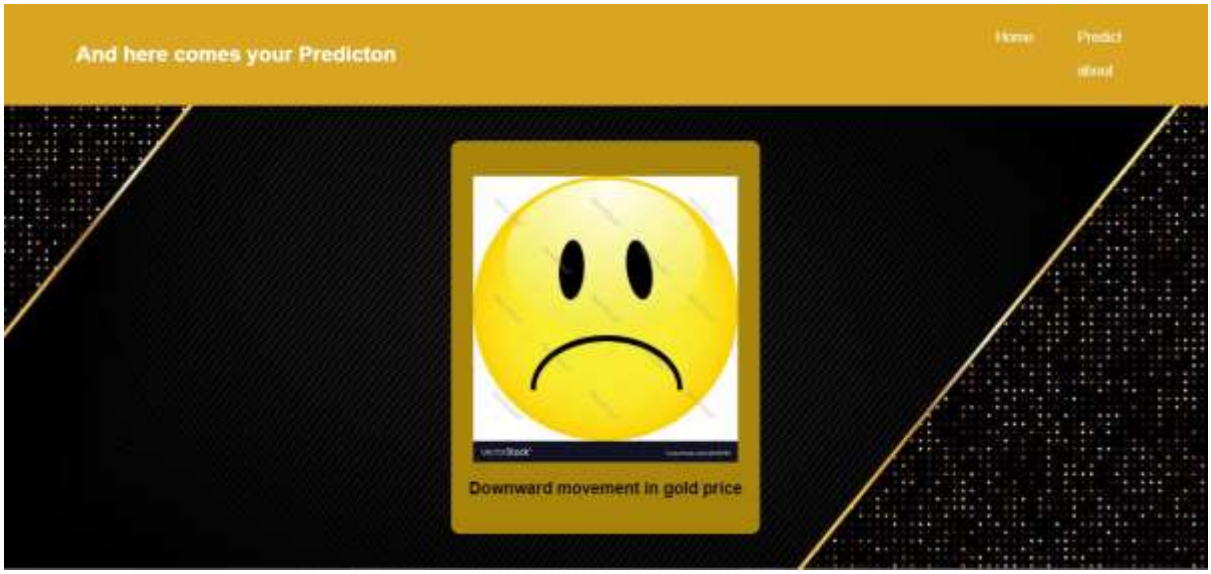
Open anaconda prompt from the start menu
Navigate to the folder where your python script is.
Now type “python app.py” command
Navigate to the localhost where you can view your web page.
Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
base) D:\TheSmartBridge\Projects\2. DrugClassification\Drug c
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a p
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

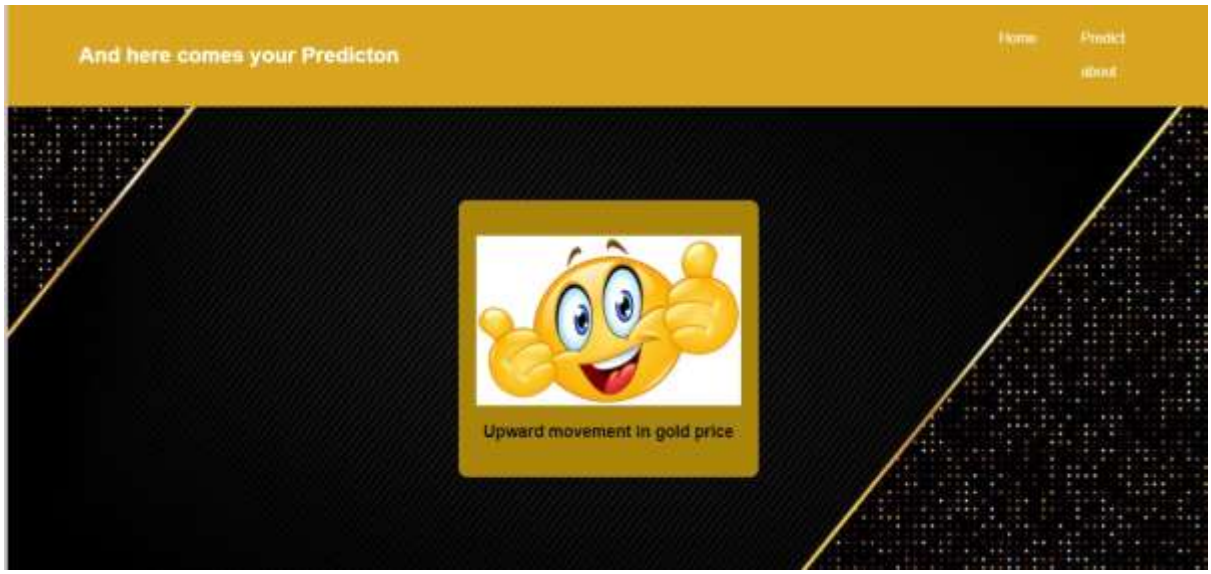
Now, Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result



Downward movement results: -



Upward Movement results:-



Title is not related results:-

PROJECT COMPLETED BY:-