CODING

MODEL-01

#DATA PREPROCESSING AND DATA CLEANING:

In [ ]:

```
import pandas as pd

import numpy as np
```

In [ ]:

```
Data = pd.read_csv('STOCK.csv')

Data.head()
```

In [ ]:

```
Data.tail()
```

In [ ]:

```
Data.shape
```

In [ ]:

```
Data = Data.dropna()
```

In [ ]:

```
Data.shape
```

In [ ]:

```
Data.size
```

In [ ]:

```
Data.isnull().sum()
```

In [ ]:

```
Data.info()
```

In [ ]:

```
Data.columns
```

In [ ]:

```
Data['label'].unique()
```

In [ ]:

```
Data['label'].value_counts()
```

In [ ]:

```
Data.groupby('label').describe()
```

BEFORE LABEL ENCODER

In [ ]:

```
Data.head()
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
var_mod = ['label','text']
le = LabelEncoder()
for i in var_mod:
    Data[i] = le.fit_transform(Data[i]).astype(int)
```

AFTER LABEL ENCODER

In [ ]:

```
Data.head()
```

In [ ]:

```
Data.duplicated()
```

In [ ]:

```
Data.duplicated().sum()
```

In [ ]:

```
Data = Data.drop_duplicates()
```

In [ ]:

```
Data.duplicated().sum()
```

In [ ]:

MODEL-02

#DATA VISUALIZATION AND DATA ANALYSIS

In [ ]:

```
import pandas as pd
```

```
import numpy as np
```

In [ ]:

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

In [ ]:

```
Data = pd.read_csv('STOCK.csv')
```

```
Data.head()
```

In [ ]:

```
Data.tail()
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
```

```
var_mod = ['label']
```

```
le = LabelEncoder()
```

```
for i in var_mod:
```

```
    Data[i] = le.fit_transform(Data[i]).astype(int)
```

In [ ]:

```
sns.countplot(x='label',data=Data)
```

In [ ]:

```
plt.hist(Data['label'],color='green')
```

In [ ]:

```
Data['label'].plot(kind='density')
```

In [ ]:

```
sns.displot(Data['label'], color='purple')
```

In [ ]:

```
sns.violinplot(Data['label'], color='yellow')
```

In [ ]:

```
sns.ecdfplot(Data['label'], color='blue')
```

In [ ]:

```
sns.distplot(Data['label'], color='RED')
```

In [ ]:

```
import re
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
```

In [ ]:

```
# Define preprocess function for text preprocessing


def preprocess_text(text):
```

```
    # Check for NaN values and handle them
    if pd.isnull(text):
        return ""
    # Convert to lowercase
    text = text.lower()
    # Remove special characters and digits
    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Tokenization and remove stop words

    stop_words = set(stopwords.words('english'))

    words = [word for word in word_tokenize(text) if word not in stop_words]

    # Stemming

    ps = PorterStemmer()

    words = [ps.stem(word) for word in words]

    # Join the preprocessed words back into a single string

    preprocessed_text = ' '.join(words)
```

```
    return preprocessed_text
```

In [ ]:

```
Data['text'] = Data['text'].apply(preprocess_text)
```

In [ ]:

```
from sklearn.model_selection import train_test_split


X,X_test,y,y_test = train_test_split(Data.loc[:,'text':],Data['label'],test_size=0.2)
```

In [ ]:

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

In [ ]:

```
ham=' '.join(X.loc[y==0,'text'].values)
ham_text = WordCloud(background_color='RED',max_words=2000,width = 800, height = 800).generate(ham)


plt.figure(figsize=[10,30])


plt.imshow(ham_text,interpolation='bilinear')
plt.title('NEGATIVE')
plt.axis('off')
```

In [ ]:

```
B=' '.join(X.loc[y==1 ,'text'].values)
B = WordCloud(background_color='green',max_words=2000,width = 800, height = 800).generate(B)
```

```
plt.figure(figsize=[10,30])


plt.imshow(B, interpolation='bilinear')

plt.axis('off')

plt.title('NEUTRAL')
```

In [ ]:

```
C=' '.join(X.loc[y==2 ,'text'].values)

C = WordCloud(background_color='YELLOW',max_words=2000,width = 800, height = 800).generate(C)


plt.figure(figsize=[10,30])


plt.imshow(C, interpolation='bilinear')

plt.axis('off')

plt.title('POSITIVE')
```

MODEL-03

#MULTINOMIALNB ALGORITHM

In [ ]:

```
import pandas as pd

import numpy as np
```

In [ ]:

```
Data = pd.read_csv('STOCK.csv')

Data.head()
```

In [ ]:

```
Data.tail()
```

In [ ]:

```
Data['label'].value_counts()
```

In [ ]:

```
import re
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
```

In [ ]:

```
# Define preprocess function for text preprocessing

def preprocess_text(text):

    # Check for NaN values and handle them

    if pd.isnull(text):
        return ""

    # Convert to lowercase

    text = text.lower()

    # Remove special characters and digits
```

```
    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Tokenization and remove stop words

    stop_words = set(stopwords.words('english'))

    words = [word for word in word_tokenize(text) if word not in stop_words]

    # Stemming

    ps = PorterStemmer()

    words = [ps.stem(word) for word in words]

    # Join the preprocessed words back into a single string

    preprocessed_text = ' '.join(words)

    return preprocessed_text
```

In [ ]:

```
# Step 2: Data Preprocessing

Data['text'] = Data['text'].apply(preprocess_text)
```

In [ ]:

```
# Step 3: Feature Extraction (TF-IDF)


from sklearn.feature_extraction.text import TfidfVectorizer


tfidf_vectorizer = TfidfVectorizer()
x1 = tfidf_vectorizer.fit_transform(Data['text'])
```

In [ ]:

```
# Assuming you have a column named 'label' containing the target labels


y1 = Data['label']
```

In [ ]:

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter


ros =RandomOverSampler(random_state=42)
x,y=ros.fit_resample(x1,y1)


print("OUR DATASET COUNT         : ", Counter(y1))
print("OVER SAMPLING DATA COUNT  : ", Counter(y))
```

In [ ]:

```
# Step 5: Splitting Data
```

```
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=42)
```

In [ ]:

```
from sklearn.naive_bayes import MultinomialNB
```

In [ ]:

```
# Step 6: Machine Learning Model (Naive Bayes)


MNB = MultinomialNB()
```

In [ ]:

```
# Step 7: Training the Model


MNB.fit(x_train, y_train)
```

In [ ]:

```
# Step 8: Evaluation

predicted = MNB.predict(x_test)
```

In [ ]:

```
from sklearn.metrics import accuracy_score


AC = accuracy_score(y_test,predicted)


print("THE ACCURACY SCORE OF MULTINOMIALNB IS :",AC*100)
```

In [ ]:

```
from sklearn.metrics import hamming_loss
```

```
HL = hamming_loss(y_test,predicted)


print("THE HAMMING LOSS OF MULTINOMIALNB IS :",HL*100)
```

In [ ]:

```
from sklearn.metrics import classification_report


CL = classification_report(y_test,predicted)


print('THE CLASSIFICATION REPORT OF MULTINOMIALNB:\n\n',CL)
```

In [ ]:

```
from sklearn.metrics import confusion_matrix


CM = confusion_matrix(y_test,predicted)


print('THE CONFUSION MATRIX SCORE OF MULTINOMIALNB:\n\n\n',CM)
```

In [ ]:

```
import matplotlib.pyplot as plt


cm=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF MULTINOMIALNB:\n\n')
print(cm)
print("\n\nDISPLAY CONFUSION MATRIX OF MULTINOMIALNB: \n\n")
```

```
from sklearn.metrics import ConfusionMatrixDisplay


cm = confusion_matrix(y_test, predicted, labels=MNB.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=MNB.classes_)
disp.plot()
plt.show()
In [ ]:
def graph():
    import matplotlib.pyplot as plt
    data=[AC]
    alg="MULTINOMIALNB"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("YELLOWGREEN"))
    plt.title("THE ACCURACY SCORE OF MULTINOMIALNB IS\n\n\n")
    plt.legend(b,data,fontsize=9)
graph()
```

MODEL-04

#LOGISTIC REGRESSION ALGORITHM

```
In [ ]:
import pandas as pd
import numpy as np
In [ ]:
Data = pd.read_csv('STOCK.csv')
```

```
Data.head()
```

In [ ]:

```
Data.tail()
```

In [ ]:

```
Data['label'].value_counts()
```

In [ ]:

```
import re
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
```

In [ ]:

```
# Define preprocess function for text preprocessing

def preprocess_text(text):

    # Check for NaN values and handle them

    if pd.isnull(text):
        return ""

    # Convert to lowercase

    text = text.lower()
```

```
    # Remove special characters and digits

    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Tokenization and remove stop words

    stop_words = set(stopwords.words('english'))

    words = [word for word in word_tokenize(text) if word not in stop_words]

    # Stemming

    ps = PorterStemmer()

    words = [ps.stem(word) for word in words]

    # Join the preprocessed words back into a single string

    preprocessed_text = ' '.join(words)

    return preprocessed_text
```

In [ ]:

```
# Step 2: Data Preprocessing
```

```
Data['text'] = Data['text'].apply(preprocess_text)
```

In [ ]:

```
# Step 3: Feature Extraction (TF-IDF)


from sklearn.feature_extraction.text import TfidfVectorizer


tfidf_vectorizer = TfidfVectorizer()
x1 = tfidf_vectorizer.fit_transform(Data['text'])
```

In [ ]:

```
# Assuming you have a column named 'label' containing the target labels


y1 = Data['label']
```

In [ ]:

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter


ros =RandomOverSampler(random_state=42)
x,y=ros.fit_resample(x1,y1)


print("OUR DATASET COUNT         : ", Counter(y1))
print("OVER SAMPLING DATA COUNT  : ", Counter(y))
```

In [ ]:

```
# Step 5: Splitting Data
```

```
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=42)
```

In [ ]:

```
from sklearn.linear_model import LogisticRegression
```

In [ ]:

```
# Step 6: Machine Learning Model (Naive Bayes)


LOG = LogisticRegression()
```

In [ ]:

```
# Step 7: Training the Model


LOG.fit(x_train, y_train)
```

In [ ]:

```
# Step 8: Evaluation

predicted = LOG.predict(x_test)
```

In [ ]:

```
from sklearn.metrics import accuracy_score


AC = accuracy_score(y_test,predicted)


print("THE ACCURACY SCORE OF LOGISTIC REGRESSION IS :",AC*100)
```

In [ ]:

```
from sklearn.metrics import hamming_loss


HL = hamming_loss(y_test,predicted)


print("THE HAMMING LOSS OF LOGISTIC REGRESSION IS :",HL*100)
```

In [ ]:

```
from sklearn.metrics import classification_report


CL = classification_report(y_test,predicted)


print('THE CLASSIFICATION REPORT OF LOGISTIC REGRESSION:\n\n',CL)
```

In [ ]:

```
from sklearn.metrics import confusion_matrix


CM = confusion_matrix(y_test,predicted)


print('THE CONFUSION MATRIX SCORE OF LOGISTIC REGRESSION:\n\n\n',CM)
```

In [ ]:

```
import matplotlib.pyplot as plt


cm=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF LOGISTIC REGRESSION:\n\n')
print(cm)
print("\n\nDISPLAY CONFUSION MATRIX OF LOGISTIC REGRESSION: \n\n")
```

```
from sklearn.metrics import ConfusionMatrixDisplay


cm = confusion_matrix(y_test, predicted, labels=LOG.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=LOG.classes_)

disp.plot()

plt.show()
```

In [ ]:

```
def graph():
    import matplotlib.pyplot as plt
    data=[AC]
    alg="LOGISTIC REGRESSION"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("CORAL"))
    plt.title("THE ACCURACY SCORE OF LOGISTIC REGRESSION IS\n\n\n")
    plt.legend(b,data,fontsize=9)
graph()
```

In [ ]:

MODEL-05

```
# RANDOM FOREST CLASSIFIER ALGORITHM
```

In [ ]:

```
import pandas as pd

import numpy as np
```

In [ ]:

```
Data = pd.read_csv('STOCK.csv')
```

```
Data.head()
```

In [ ]:

```
Data.tail()
```

In [ ]:

```
Data['label'].value_counts()
```

In [ ]:

```
import re
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
```

In [ ]:

```
# Define preprocess function for text preprocessing

def preprocess_text(text):

    # Check for NaN values and handle them

    if pd.isnull(text):
        return ""

    # Convert to lowercase
```

```
    text = text.lower()

    # Remove special characters and digits

    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Tokenization and remove stop words

    stop_words = set(stopwords.words('english'))

    words = [word for word in word_tokenize(text) if word not in stop_words]

    # Stemming

    ps = PorterStemmer()

    words = [ps.stem(word) for word in words]

    # Join the preprocessed words back into a single string

    preprocessed_text = ' '.join(words)

    return preprocessed_text
```

In [ ]:

Step 2: Data Preprocessing

Data['text'] = Data['text'].apply(preprocess_text)

In [ ]:

Step 3: Feature Extraction (TF-IDF)

from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer = TfidfVectorizer()

x1 = tfidf_vectorizer.fit_transform(Data['text'])

In [ ]:

Assuming you have a column named 'label' containing the target labels

y1 = Data['label']

In [ ]:

import imblearn

from imblearn.over_sampling import RandomOverSampler

from collections import Counter

ros =RandomOverSampler(random_state=42)

x,y=ros.fit_resample(x1,y1)

print("OUR DATASET COUNT : ", Counter(y1))

print("OVER SAMPLING DATA COUNT : ", Counter(y))

In [ ]:

```
# Step 5: Splitting Data
```

```
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=42)
```

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [ ]:

```
# Step 6: Machine Learning Model (Naive Bayes)
```

```
RFC = RandomForestClassifier()
```

In [ ]:

```
# Step 7: Training the Model
```

```
RFC.fit(x_train, y_train)
```

In [ ]:

```
# Step 8: Evaluation
predicted = RFC.predict(x_test)
```

In [ ]:

```
from sklearn.metrics import accuracy_score
```

```
AC = accuracy_score(y_test,predicted)
```

```
print("THE ACCURACY SCORE OF RANDOM FOREST CLASSIFIER IS :",AC*100)
```

In [ ]:

```
from sklearn.metrics import hamming_loss

HL = hamming_loss(y_test,predicted)

print("THE HAMMING LOSS OF RANDOM FOREST CLASSIFIER IS :",HL*100)
```

In [ ]:

```
from sklearn.metrics import classification_report

CL = classification_report(y_test,predicted)

print('THE CLASSIFICATION REPORT OF RANDOM FOREST CLASSIFIER:\n\n',CL)
```

In [ ]:

```
from sklearn.metrics import confusion_matrix

CM = confusion_matrix(y_test,predicted)

print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST
CLASSIFIER:\n\n\n',CM)
```

In [ ]:

```
import matplotlib.pyplot as plt

cm=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST CLASSIFIER:\n\n')
```

```
print(cm)
print("\n\nDISPLAY CONFUSION MATRIX OF RANDOM FOREST CLASSIFIER: \n\n")

from sklearn.metrics import ConfusionMatrixDisplay

cm = confusion_matrix(y_test, predicted, labels=RFC.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=RFC.classes_)
disp.plot()
plt.show()
```

In [ ]:

```
def graph():
    import matplotlib.pyplot as plt
    data=[AC]
    alg="RANDOM FOREST CLASSIFIER"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("VIOLET"))
    plt.title("THE ACCURACY SCORE OF RANDOM FOREST CLASSIFIER IS\n\n\n")
    plt.legend(b,data,fontsize=9)
graph()
```

In [ ]:

In [ ]:

```
import joblib
joblib.dump(RFC, 'MODEL.pkl')
```

In [ ]:

```
import joblib

joblib.dump(tfidf_vectorizer, 'VECTOR.pkl')
```

View.py

```
from django.shortcuts import render, redirect

from . models import UserPersonalModel

from . forms import UserPersonalForm, UserRegisterForm

from django.contrib.auth import authenticate, login,logout

from django.contrib import messages

import numpy as np


import re

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

from nltk.tokenize import word_tokenize

from sklearn.feature_extraction.text import TfidfVectorizer

import joblib

import pandas as pd




def Landing_1(request):

    return render(request, '1_Landing.html')
```

```
def Register_2(request):
    form = UserRegisterForm()
    if request.method =='POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            user = form.cleaned_data.get('username')
            messages.success(request, 'Account was successfully created. ' + user)
            return redirect('Login_3')

    context = {'form':form}
    return render(request, '2_Register.html', context)



def Login_3(request):
    if request.method =='POST':
        username = request.POST.get('username')
        password = request.POST.get('password')

        user = authenticate(username=username, password=password)

        if user is not None:
            login(request, user)
```

```
            return redirect('Home_4')
        else:
            messages.info(request, 'Username OR Password incorrect')

    context = {}
    return render(request,'3_Login.html', context)

def Home_4(request):
    return render(request, '4_Home.html')

def Teamates_5(request):
    return render(request,'5_Teamates.html')

def Domain_Result_6(request):
    return render(request,'6_Domain_Result.html')

def Problem_Statement_7(request):
    return render(request,'7_Problem_Statement.html')


def Per_Info_8(request):
    if request.method == 'POST':
        fieldss = ['firstname','lastname','age','address','phone','city','state','country']
        form = UserPersonalForm(request.POST)
```

```
        if form.is_valid():
            print('Saving data in Form')
            form.save()
        return render(request, '4_Home.html', {'form':form})
    else:
        print('Else working')
        form = UserPersonalForm(request.POST)
        return render(request, '8_Per_Info.html', {'form':form})


vectorizer = joblib.load('C:/Users/harin/Music/MAIN_PROJECT/CODE/DEPLOYMENT/PROJECT/APP/VECTOR.pkl')
model = joblib.load('C:/Users/harin/Music/MAIN_PROJECT/CODE/DEPLOYMENT/PROJECT/APP/MODEL.pkl')


def Deploy_9(request):
    if request.method == "POST":
        int_features = [x for x in request.POST.values()]
        input_text2 = int_features[1:]
        print(input_text2)


        if isinstance(input_text2[0], str):
            result = input_text2[0]
        else:
```

```
    result = None

print(result)

def preprocess_text(text):
    if pd.isnull(text):
        return ""
    text = text.lower()
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    stop_words = set(stopwords.words('english'))
    words = [word for word in word_tokenize(text) if word not in stop_words]
    ps = PorterStemmer()
    words = [ps.stem(word) for word in words]
    preprocessed_text = ' '.join(words)
    return preprocessed_text

preprocessed_input = preprocess_text(result)

input_features = vectorizer.transform([preprocessed_input])

predicted_label = model.predict(input_features)[0]

print(f"Predicted Label: {predicted_label}")
if predicted_label == -1:
```

```
            return render(request, '9_Deploy.html', {"prediction_text":f"THE NEGATIVE CONTENTS DETECTED IN THIS REVIEWS IN STOCK MARKET."})
        elif predicted_label == 1:
            return render(request, '9_Deploy.html', {"prediction_text":f"THE POSITIVE CONTENTS DETECTED IN THIS REVIEWS IN STOCK MARKET ."})

    else:
        return render (request, '9_Deploy.html')


def Per_Database_10(request):
    models = UserPersonalModel.objects.all()
    return render(request, '10_Per_Database.html', {'models':models})

def Logout(request):
    logout(request)
    return redirect('Landing_1')
```

Landing.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<style>
            @import
url('https://fonts.googleapis.com/css?family=Poppins:200,300,400,500,600,700,800,900&dis
play=swap');
           *
           {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: 'Poppins', sans-serif;
           }
          header
           {
            position: absolute;
            top: 0;
            left: 0;
            width: 100%;
            padding: 40px 100px;
            z-index: 1000;
            display: flex;
            justify-content: space-between;
            align-items: center;
```

```
}
header .logo
{
  color: #fff;
  text-transform: uppercase;
  cursor: pointer;
}
.toggle
{
  position: relative;
  width: 60px;
  height: 60px;
  background: url(https://i.ibb.co/HrfVRcx/menu.png);
  background-repeat: no-repeat;
  background-size: 30px;
  background-position: center;
  cursor: pointer;
}
.toggle.active
{
  background: url(https://i.ibb.co/rt3HybH/close.png);
  background-repeat: no-repeat;
  background-size: 25px;
  background-position: center;
```

```
  cursor: pointer;
}
.showcase
{
  position: absolute;
  right: 0;
  width: 100%;
  min-height: 100vh;
  padding: 100px;
  display: flex;
  justify-content: space-between;
  align-items: center;
  background: #111;
  transition: 0.5s;
  z-index: 2;
}
.showcase.active
{
  right: 300px;
}

.showcase video
{
  position: absolute;
```

```
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  object-fit: cover;
  opacity: 0.8;
}
.overlay
{
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: #03a9f4;
  mix-blend-mode: overlay;
}
.text
{
  position: relative;
  z-index: 10;
}

.text h2
```

```
{
  font-size: 5em;
  font-weight: 800;
  color: #fff;
  line-height: 1em;
  text-transform: uppercase;
}
.text h3
{
  font-size: 4em;
  font-weight: 700;
  color: #fff;
  line-height: 1em;
  text-transform: uppercase;
}
.text p
{
  font-size: 1.1em;
  color: #fff;
  margin: 20px 0;
  font-weight: 400;
  max-width: 700px;
}
.text a
```

```
{
  display: inline-block;
  font-size: 1em;
  background: #fff;
  padding: 10px 30px;
  text-transform: uppercase;
  text-decoration: none;
  font-weight: 500;
  margin-top: 10px;
  color: #111;
  letter-spacing: 2px;
  transition: 0.2s;
}
.text a:hover
{
  letter-spacing: 6px;
}
.social
{
  position: absolute;
  z-index: 10;
  bottom: 20px;
  display: flex;
  justify-content: center;
```

```
  align-items: center;
}
.social li
{
  list-style: none;
}
.social li a
{
  display: inline-block;
  margin-right: 20px;
  filter: invert(1);
  transform: scale(0.5);
  transition: 0.5s;
}
.social li a:hover
{
  transform: scale(0.5) translateY(-15px);
}
.menu
{
  position: absolute;
  top: 0;
  right: 0;
  width: 300px;
```

```
  height: 100%;
  display: flex;
  justify-content: center;
  align-items: center;
}
.menu ul
{
  position: relative;
}
.menu ul li
{
  list-style: none;
}
.menu ul li a
{
  text-decoration: none;
  font-size: 24px;
  color: #111;
}
.menu ul li a:hover
{
  color: #03a9f4;
}
#one{
```

```
    color: black;
    border-radius: 20px;
  }

  @media (max-width: 991px)
  {
    .showcase,
    .showcase header
    {
      padding: 40px;
    }
    .text h2
    {
      font-size: 3em;
    }
    .text h3
    {
      font-size: 2em;
    }

  }
</style>

<body>
```

```
<section class="showcase">
  <header>
    <h2 class="logo">FOR SOCIAL DOMAIN</h2>
    <div class="toggle"></div>
  </header>
  <video src="static/images/S0.mp4" muted loop autoplay></video>
  <div class="overlay"></div>
  <div class="text">
    <h2>TO ANALYSE</h2>
    <h3>STOCK MARKET SENTIMENT ANALYSIS</h3>
    <p>The stock market is a financial marketplace where investors buy and sell shares of publicly traded companies. It provides a platform for companies to raise capital by selling ownership stakes to the public and allows investors to trade these shares, with prices influenced by supply and demand, economic conditions, and company performance.</p>
    <a href="{% url 'Register_2' %}" id="one">TO REGISTER YOUR ACCOUNT</a>
  </div>
</section>
<div class="menu">
  <ul>
    <li><a href="https://twitter.com/StocksResearch">TWITTER</a></li>
    <li><a href="https://www.facebook.com/groups/195040057198116/">FACEBOOK</a></li>
    <li><a href="https://www.instagram.com/thestockmarketindia/">INSTAGRAM</a></li>
    <li><a href="https://en.wikipedia.org/wiki/Stock_market">WIKIPEDIA</a></li>
```

```
          <li><a href="https://www.justdial.com/Chennai/Stock-Brokers/nct-
10458461">CONTACT</a></li>
        </ul>
      </div>
      <script>
        const menuToggle = document.querySelector('.toggle');
          const showcase = document.querySelector('.showcase');

          menuToggle.addEventListener('click', () => {
            menuToggle.classList.toggle('active');
            showcase.classList.toggle('active');
          })
    </script>
</body>
</html>
```

Register.html

```
{% load static %}
<!DOCTYPE html>

<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Document</title>
<style>
        html {
        background-image: url("/static/images/S1.webp");
        background-repeat: no-repeat;
        background-size: cover;
    }

    h1 {
        font-family: 'Kanit', sans-serif;
        color: #fff;
        font-size: 40px;
    }

    #login_form {
        width: 608px;
        height: 756px;
        border-color: #fff;
        border-style: groove;
        border-radius: 25%;
        text-align: center;
        margin: auto;
        margin-top: 65px;
        backdrop-filter: blur(10px);
```

```
    transition: 0.3s;
}

#login_form:hover {
    box-shadow: 6px 10px 10px 10px rgba(255, 255, 255, 0.781);
}

h2 {
    font-family: 'Aoboshi One', serif;
    color: #fff;
    font-size: 25px;
    font-style: italic;
    text-align: center;
    cursor: pointer;
}

input {
    padding: 27px 23px;
    border-radius: 25px;
    background: inherit;
    border-style: none;
    box-shadow: 10px 10px 40px 20px rgba(0, 0, 0, 0.377);
    font-family: 'Work Sans', sans-serif;
    text-align: center;
```

```
    font-size: 19px;
    color: #fff;
    margin-bottom: 25px;
    transition: 0.2s;
}

input:hover {
    box-shadow: 10px 10px 40px 20px rgba(0, 0, 0, 0.616);
}

::placeholder {
    color: #ffffffa8;
    font-family: 'Work Sans', sans-serif;
    font-size: 20px;
    text-align: center;
    transition: 0.2s;
}

button {
    padding: 25px 40px;
    color: #000;
    background-color: #fff;
    border-style: none;
    border-radius: 25%;
```

```
            font-family: 'Nunito', sans-serif;
            font-size: 20px;
            transition: 0.3s;
        }

        button:hover {
            background-color: rgb(142, 196, 48);
            font-size: 24px;
            transform: scale(1, 1);
            box-shadow: 6px 6px 6px 6px rgba(255, 255, 255, 0.747);
            color: #fff;
            cursor: pointer;
        }



    </style>
</head>
<body>
    <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <link rel="stylesheet" href="style.css">
    <link href="https://fonts.googleapis.com/css2?family=Kanit:wght@300&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Aoboshi+One&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Work+Sans:wght@300&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Nunito:wght@300&display=swap" rel="stylesheet">
</head>

<body>
  <form method="POST">
    {% csrf_token %}

        <h2>WELCOME</h2>
        <div id="login_form">
        <h1>REGISTERATION PAGE</h1>
        <input type="text" placeholder="USERNAME" name="username"><br>
        <input type="email" placeholder="EMAIL" name="email"><br>
        <input type="password" placeholder="PASSWORD1" name="password1"><br>
        <input type="password" placeholder="PASSWORD2" name="password2"><br>
        <button type="submit">REGISTER</button>

    </form>
```

```
    <br>
     <footer>

       <h2>Already Have an Account?</h2>
     </footer>
     <div class="foot">
       <button><a href="{% url 'Login_3' %}">LOGIN</a></button>

   </div>
</body>
</html>
```

Login.html

```
{% load static %}
<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
                html {
```

```
    background-image: url("/static/images/S2.jpg");
    background-repeat: no-repeat;
    background-size: cover;
}

h1 {
    font-family: 'Kanit', sans-serif;
    color: #fff;
    font-size: 40px;
}

#login_form {
    width: 608px;
    height: 656px;
    border-color: #fff;
    border-style: groove;
    border-radius: 25%;
    text-align: center;
    margin: auto;
    margin-top: 65px;
    backdrop-filter: blur(10px);
    transition: 0.3s;
}
```

```
#login_form:hover {
    box-shadow: 6px 10px 10px 10px rgba(255, 255, 255, 0.781);
}

h2 {
    font-family: 'Aoboshi One', serif;
    color: #fff;
    font-size: 25px;
    font-style: italic;
    text-align: center;
    cursor: pointer;
}

input {
    padding: 27px 23px;
    border-radius: 25px;
    background: inherit;
    border-style: none;
    box-shadow: 10px 10px 40px 20px rgba(0, 0, 0, 0.377);
    font-family: 'Work Sans', sans-serif;
    text-align: center;
    font-size: 19px;
    color: #fff;
    margin-bottom: 25px;
```

```
    transition: 0.2s;
}

input:hover {
    box-shadow: 10px 10px 40px 20px rgba(0, 0, 0, 0.616);
}

::placeholder {
    color: #ffffffa8;
    font-family: 'Work Sans', sans-serif;
    font-size: 20px;
    text-align: center;
    transition: 0.2s;
}

button {
    padding: 25px 40px;
    color: #000;
    background-color: #fff;
    border-style: none;
    border-radius: 25%;
    font-family: 'Nunito', sans-serif;
    font-size: 20px;
    transition: 0.3s;
```

```
        }

        button:hover {
            background-color: rgb(142, 196, 48);
            font-size: 24px;
            transform: scale(1, 1);
            box-shadow: 6px 6px 6px 6px rgba(255, 255, 255, 0.747);
            color: #fff;
            cursor: pointer;
        }



    </style>
</head>
<body>
    <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <link href="https://fonts.googleapis.com/css2?family=Kanit:wght@300&display=swap"
rel="stylesheet">
```

```
    <link href="https://fonts.googleapis.com/css2?family=Aoboshi+One&display=swap"
rel="stylesheet">
    <link
href="https://fonts.googleapis.com/css2?family=Work+Sans:wght@300&display=swap"
rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Nunito:wght@300&display=swap"
rel="stylesheet">
</head>

<body>
  <form method="POST">
    {% csrf_token %}

        <h2>WELCOME</h2>
        <div id="login_form">
        <h1>LOGIN</h1>
        <h3>USERNAME</h3>
        <input type="text" placeholder="USERNAME" name="username"><br>
        <h3>PASSWORD</h3>
        <input type="password" placeholder="PASSWORD" name="password"><br>
        <button type="submit">LOGIN</button>

    </form>

    <br>
```

```
    <footer>
      <h2>Don't Have an Account?</h2>
    </footer>
    <div class="foot">
      <button><a href="{% url 'Register_2' %}">REGISTER</a></button>

  </div>

</body>

</html>

</body>
</html>
```

Open.html

```
{% load static %}
<!DOCTYPE html>

<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Document</title>
    <style>

        * {
        box-sizing: border-box;
        }

        html {
        margin: 0;
        padding: 0;
        }

        body{
        margin:0;
        padding: 0;
        background: linear-gradient(rgba(0, 0, 0, 0.35), rgba(0, 0, 0, 0.35)),
url("/static/images/S4.jpg");
        background-size: cover;
        color: black;
        font-family: "open sans";
        height: 100vh;
        display: flex;
        flex-direction: column;
        justify-content: space-between;
        }
```

```
/* ------------------------ NAVBAR ----------------*/



header{

display:flex;

justify-content: space-between;

color: white;

background: rgba(0,0,0,0.2)

}


header div{

display:flex;

justify-content: space-between;

align-items: center;

font-family: Pacifico;

margin: 0 2rem;

}


header div i{

font-size: 2rem;

margin: 1rem;

}
```

```
header nav{
padding: 1rem 2rem;
}

nav ul li{
list-style: none;
display: inline;
text-transform: uppercase;
font-weight: bold;
letter-spacing: 5px;
}

nav li a{
padding: 1rem;
margin: 1rem;
text-decoration: none;
color: white;
transition: all 250ms ease-in;
}

nav li a:hover{
background: rgba(255,255,255,.3);
color: black;
```

```
}



/* ----------------------- TOP SECTION ----------------*/

.titles{
color: white;
text-align: center;
width: 50vw;
margin: 0 auto;
}

.titles h1{
font-family: Pacifico;
font-size: 5rem;
margin-bottom: 0;
text-shadow: 1px 1px 0 black;

}

.titles p{
letter-spacing: 3px;
text-shadow: 1px 1px 0 black;
```

```
}

/* ----------------------- BOT SECTION ----------------*/




.container-boxes{
margin: 0 auto;
padding: 0;
display:flex;
justify-content: space-around;
align-items: flex-end;
max-width: 80vw;
}

.box{
background: rgba(255,255,255,.5);
margin: 1rem;
padding: .5rem;
display: flex;
justify-content: space-between;
align-items: center;
max-width: 350px;
max-height: 180px;
```

```
min-height: 180px;
transition: all 250ms ease-out;
}

.box:hover{
background: rgba(255,255,255,.7);
transform: translateY(-20%);
}

.box a{
text-decoration: none;
color: black;
}

.icon{
font-size: 3rem;
padding: 1rem;
}

.text h3{
text-transform: uppercase;
letter-spacing: 4px;
margin-bottom: 0;
}
```

```
.text p{
margin-top: 1rem;
line-height: 1.5rem;
text-align: left;
}

.text{
padding: .5rem;
}



/* MEDIAQUERIES */



@media (max-width: 1160px){
.icon{
    font-size: 2rem;
    padding: 0.5rem;
}
.text h3{
    text-transform: uppercase;
    letter-spacing: 4px;
    margin-bottom: 0;
```

```
    font-size: 1rem;
}

.text p{
    margin-top: 1rem;
    line-height: 1.5rem;
    text-align: left;
    font-size: .8rem;
}
nav li a:hover{
background: none;
color: black;
}
}



@media (max-width: 850px){
body{
    height: 100%;
}
.container-boxes{
    flex-direction: column;
}
.box:hover{
```

```
            background: rgba(255,255,255,.7);
            transform: none;
        }
        .titles h1{
        font-size: 3rem;
        }
        }
    </style>
</head>
<body>
    <!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>STOCK</title>
</head>
```

<body>

<header>

STOCK MARKET SENTIMENT ANALYSIS

<nav>

FOUNDATIONS
DOMAIN_RESULT
PROBLEM_STATEMENTS
LOGOUT

</nav>
</header>

<section class="titles">

<h1>STOCK MARKET SENTIMENT ANALYSIS USING NLP</h1>

The stock market is a financial marketplace where individuals and institutions buy and sell shares of publicly traded companies. It provides a platform for raising capital and allows investors to participate in company ownership by purchasing stocks, which represent ownership in a company. Stock prices fluctuate based on supply and demand, economic

conditions, and company performance, making it a key indicator of overall economic health.

<h2>HOW TO PREVENT STOCK MARKET NEGATIVE REVIEWS</h2>

Diversify your portfolio: Spread your investments across different asset classes, industries, and companies to reduce risk.

Conduct research: Thoroughly analyze the companies you invest in, their financial health, and industry trends.

Set realistic goals and risk tolerance: Define your investment objectives and how much risk you're willing to take. Avoid making impulsive decisions based on short-term market fluctuations.

</section>

<section class="container-boxes">


```
    <div class="text">
      <a href="{% url 'Per_Info_8' %}">
        <h3>PERSONAL_INFO</h3>
        <p>Please Enter Your Personal informations.</p>
      </a>
    </div>

  </div>

  <div class="box">

    <div class="icon">
      <a href="#"><i class="fas fa-seedling"></i></a>
    </div>

    <div class="text">
      <a href="{% url 'Deploy_9' %}">
        <h3>DEPLOYMENT</h3>
        <p>
          NLP deployment is the process of implementing natural language processing models and systems into real-world applications for tasks like text classification, sentiment analysis, or chatbots.</p>
      </a>
    </div>
```

```
    </div>

    <div class="box">

      <div class="icon">
        <a href="#"><i class="fas fa-address-card"></i></a>
      </div>

      <div class="text">
        <a href="{% url 'Per_Database_10' %}">
          <h3>INFO_DATABASE</h3>
          <p>
            A database is a structured collection of data organized for efficient storage, retrieval, and management of information.</p>
        </a>
      </div>

    </div>
  </section>

</body>

</html>
</body>
</html>
```

Deployment.html

```
{% load static %}

<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>

        * {
        box-sizing: border-box;
        }

        html {
        margin: 0;
        padding: 0;
        }

        body{
```

```
        margin:0;
        padding: 0;
        background: linear-gradient(rgba(0, 0, 0, 0.35), rgba(0, 0, 0, 0.35)),
url("/static/images/S4.jpg");
        background-size: cover;
        color: black;
        font-family: "open sans";
        height: 100vh;
        display: flex;
        flex-direction: column;
        justify-content: space-between;
        }

        /* ------------------------ NAVBAR ----------------*/


        header{
        display:flex;
        justify-content: space-between;
        color: white;
        background: rgba(0,0,0,0.2)
        }

        header div{
        display:flex;
```

```
justify-content: space-between;

align-items: center;

font-family: Pacifico;

margin: 0 2rem;

}


header div i{

font-size: 2rem;

margin: 1rem;

}



header nav{

padding: 1rem 2rem;

}


nav ul li{

list-style: none;

display: inline;

text-transform: uppercase;

font-weight: bold;

letter-spacing: 5px;

}
```

```
nav li a{
padding: 1rem;
margin: 1rem;
text-decoration: none;
color: white;
transition: all 250ms ease-in;
}

nav li a:hover{
background: rgba(255,255,255,.3);
color: black;
}



/* ------------------------ TOP SECTION ----------------*/

.titles{
color: white;
text-align: center;
width: 50vw;
margin: 0 auto;
}
```

```
.titles h1{
font-family: Pacifico;
font-size: 5rem;
margin-bottom: 0;
text-shadow: 1px 1px 0 black;

}

.titles p{
letter-spacing: 3px;
text-shadow: 1px 1px 0 black;
}

/* ------------------------ BOT SECTION ----------------*/



.container-boxes{
margin: 0 auto;
padding: 0;
display:flex;
justify-content: space-around;
align-items: flex-end;
max-width: 80vw;
```

```
}

.box{
background: rgba(255,255,255,.5);
margin: 1rem;
padding: .5rem;
display: flex;
justify-content: space-between;
align-items: center;
max-width: 350px;
max-height: 180px;
min-height: 180px;
transition: all 250ms ease-out;
}

.box:hover{
background: rgba(255,255,255,.7);
transform: translateY(-20%);
}

.box a{
text-decoration: none;
color: black;
}
```

```
.icon{
font-size: 3rem;
padding: 1rem;
}

.text h3{
text-transform: uppercase;
letter-spacing: 4px;
margin-bottom: 0;
}

.text p{
margin-top: 1rem;
line-height: 1.5rem;
text-align: left;
}

.text{
padding: .5rem;
}


/* MEDIAQUERIES */
```

```
@media (max-width: 1160px){
.icon{
    font-size: 2rem;
    padding: 0.5rem;
}
.text h3{
    text-transform: uppercase;
    letter-spacing: 4px;
    margin-bottom: 0;
    font-size: 1rem;
}

.text p{
    margin-top: 1rem;
    line-height: 1.5rem;
    text-align: left;
    font-size: .8rem;
}
nav li a:hover{
background: none;
color: black;
}
```

```
        }



        @media (max-width: 850px){
        body{
            height: 100%;
        }
        .container-boxes{
            flex-direction: column;
        }
        .box:hover{
            background: rgba(255,255,255,.7);
            transform: none;
        }
        .titles h1{
        font-size: 3rem;
        }
        }
        }
    </style>
</head>
```

```
<body>
    <!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>STOCK</title>
</head>

<body>

  <header>
    <div>
      <i class="fas fa-atom"></i>
      <p>STOCK MARKET SENTIMENT ANALYSIS</p>
    </div>
    <nav>
      <ul>
        <li><a href="{% url 'Teamates_5' %}">FOUNDATIONS</a></li>
        <li><a href="{% url 'Domain_Result_6' %}">DOMAIN_RESULT</a></li>
        <li><a href="{% url 'Problem_Statement_7' %}">PROBLEM_STATEMENTS</a></li>
        <li><a href="{% url 'Login_3' %}">LOGOUT</a></li>
```


</nav>

</header>

<section class="titles">

<h1>STOCK MARKET SENTIMENT ANALYSIS USING NLP</h1>

The stock market is a financial marketplace where individuals and institutions buy and sell shares of publicly traded companies. It provides a platform for raising capital and allows investors to participate in company ownership by purchasing stocks, which represent ownership in a company. Stock prices fluctuate based on supply and demand, economic conditions, and company performance, making it a key indicator of overall economic health.

<h2>HOW TO PREVENT STOCK MARKET NEGATIVE REVIEWS</h2>

Diversify your portfolio: Spread your investments across different asset classes, industries, and companies to reduce risk.

Conduct research: Thoroughly analyze the companies you invest in, their financial health, and industry trends.

Set realistic goals and risk tolerance: Define your investment objectives and how much risk you're willing to take. Avoid making impulsive decisions based on short-term market fluctuations.

</section>

```
<section class="container-boxes">
  <div class="box">

    <div class="icon">
      <a href="#"><i class="fas fa-fire"></i></a>
    </div>

    <div class="text">
      <a href="{% url 'Per_Info_8' %}">
        <h3>PERSONAL_INFO</h3>
        <p>Please Enter Your Personal informations.</p>
      </a>
    </div>

  </div>

  <div class="box">

    <div class="icon">
      <a href="#"><i class="fas fa-seedling"></i></a>
```

```
</div>

<div class="text">
  <a href="{% url 'Deploy_9' %}">
    <h3>DEPLOYMENT</h3>
    <p>
      NLP deployment is the process of implementing natural language processing models and systems into real-world applications for tasks like text classification, sentiment analysis, or chatbots.</p>
  </a>
</div>

</div>

<div class="box">

  <div class="icon">
    <a href="#"><i class="fas fa-address-card"></i></a>
  </div>

  <div class="text">
    <a href="{% url 'Per_Database_10' %}">
      <h3>INFO_DATABASE</h3>
      <p>
```

A database is a structured collection of data organized for efficient storage, retrieval, and management of information.</p>

</section>

</body>

</html>

</body>

</html>