**Project Development Phase**

**Model Performance Test**

|  |  |
| --- | --- |
| Date | 16 NOvember 2022 |
| Team ID | Team-592275 |
| Project Name | Gilded Emotions: Unearthing Market Sentiments In Gold News |
| Maximum Marks | 10 Marks |

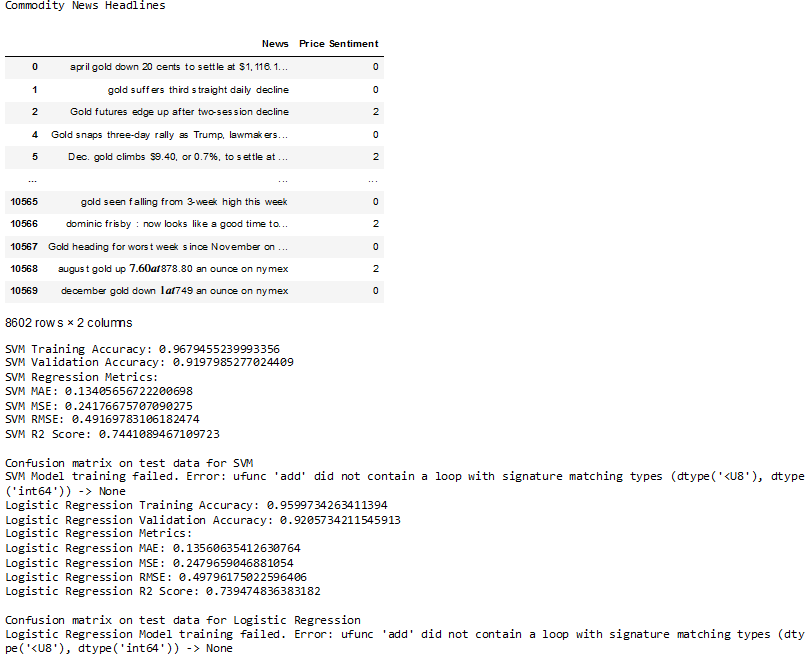
**Model Performance Testing:**

Project team shall fill the following information in model performance testing template.

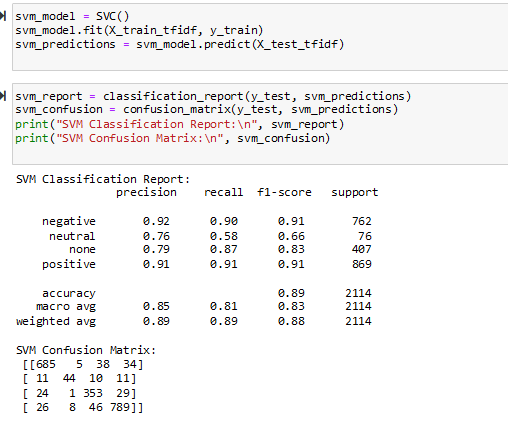
|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Parameter** | **Values** | **Screenshot** |
|  | Metrics | **Regression Model:** SVM Training Accuracy: 0.9679455239993356  SVM Validation Accuracy: 0.9197985277024409  SVM Regression Metrics:  SVM MAE: 0.13405656722200698  SVM MSE: 0.24176675707090275  SVM RMSE: 0.49169783106182474  SVM R2 Score: 0.7441089467109723  **Classification Model:** Confusion Matrix - , Accuray Score- & Classification Report – | [SC-1] in the next page  [SC-2] in the next page |
|  | Tune the Model | Hyperparameter Tuning -  Validation Method - |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Parameter** | **Values** | **Screenshot** |
|  | Metrics | **Regression Model:** Logistic Regression Training Accuracy: 0.9599734263411394  Logistic Regression Validation Accuracy: 0.9205734211545913  Logistic Regression Metrics:  Logistic Regression MAE: 0.13560635412630764  Logistic Regression MSE: 0.2479659046881054  Logistic Regression RMSE: 0.49796175022596406  Logistic Regression R2 Score: 0.739474836383182  **Classification Model:** Confusion Matrix - , Accuray Score- & Classification Report – | [SC-1] in the next page  [SC-3] in the next page |

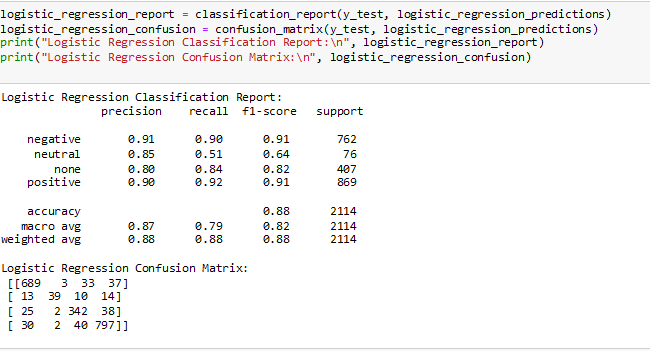
**[SC-1]**



**[SC-2]**



**[SC-3]**



**CODE ::**

# Load packages

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.svm import SVC

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

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

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

# Convert 'Price Sentiment' to numeric values

df['Price Sentiment'] = pd.Categorical(df['Price Sentiment'])

df['Price Sentiment'] = df['Price Sentiment'].cat.codes

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 impure\_data:

# 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

# Function to calculate accuracy

def calculate\_accuracy(model, X, y):

y\_pred = model.predict(X)

accuracy = accuracy\_score(y, y\_pred)

return accuracy

# Function to calculate regression metrics

def calculate\_regression\_metrics(y\_true, y\_pred):

mae = mean\_absolute\_error(y\_true, y\_pred)

mse = mean\_squared\_error(y\_true, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_true, y\_pred)

return mae, mse, rmse, r2

# Updated headline\_sentiment function for SVM

def headline\_sentiment\_svm(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')

try:

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

SVM\_L.fit(train\_corpus\_tf\_idf, y\_train)

# Calculate training accuracy

training\_accuracy = calculate\_accuracy(SVM\_L, train\_corpus\_tf\_idf, y\_train)

# Calculate validation accuracy

validation\_accuracy = calculate\_accuracy(SVM\_L, test\_corpus\_tf\_idf, y\_test)

# This prints accuracy score for the test dataset

print("SVM Training Accuracy:", training\_accuracy)

print("SVM Validation Accuracy:", validation\_accuracy)

# Predictions for SVM model

y\_pred\_svm = SVM\_L.predict(test\_corpus\_tf\_idf)

# Convert sparse matrices to 1D arrays

y\_test\_svm\_array = np.array(y\_test)

y\_pred\_svm\_array = np.array(y\_pred\_svm)

# Calculate regression metrics for SVM

mae\_svm, mse\_svm, rmse\_svm, r2\_svm = calculate\_regression\_metrics(y\_test\_svm\_array, y\_pred\_svm\_array)

print("SVM Regression Metrics:")

print("SVM MAE:", mae\_svm)

print("SVM MSE:", mse\_svm)

print("SVM RMSE:", rmse\_svm)

print("SVM R2 Score:", r2\_svm)

# This prints confusion matrix for the test dataset

labels\_svm = np.unique(y\_test)

m\_svm = confusion\_matrix(y\_test\_svm\_array, SVM\_L.predict(test\_corpus\_tf\_idf), labels=labels\_svm)

print("\nConfusion matrix on test data for SVM")

cm\_svm = pd.DataFrame(m\_svm, index=labels\_svm, columns=labels\_svm)

cm\_svm.index = "Actual: " + cm\_svm.index

cm\_svm.columns = "Predicted: " + cm\_svm.columns

display(cm\_svm)

# Saving the data into a csv file in the current folder

temp\_df\_svm = pd.DataFrame()

temp\_df\_svm["News"] = X\_test

temp\_df\_svm["Actual Price Sentiment"] = y\_test\_svm\_array

temp\_df\_svm["Predicted Sentiment"] = SVM\_L.predict(test\_corpus\_tf\_idf)

temp\_df\_svm.to\_csv("predicted\_svm.csv")

print('Predictions on Test Data for SVM are as follows:')

display(temp\_df\_svm)

except Exception as e:

print(f"SVM Model training failed. Error: {e}")

return tf\_idfvectorizer, SVM\_L

# Updated headline\_sentiment function for Logistic Regression

def headline\_sentiment\_logistic(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)

# Initializing Logistic Regression model

log\_reg = LogisticRegression()

try:

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

log\_reg.fit(train\_corpus\_tf\_idf, y\_train)

# Calculate training accuracy

training\_accuracy\_log = calculate\_accuracy(log\_reg, train\_corpus\_tf\_idf, y\_train)

# Calculate validation accuracy

validation\_accuracy\_log = calculate\_accuracy(log\_reg, test\_corpus\_tf\_idf, y\_test)

# This prints accuracy score for the test dataset

print("Logistic Regression Training Accuracy:", training\_accuracy\_log)

print("Logistic Regression Validation Accuracy:", validation\_accuracy\_log)

# Predictions for Logistic Regression model

y\_pred\_log = log\_reg.predict(test\_corpus\_tf\_idf)

# Convert sparse matrices to 1D arrays

y\_test\_log\_array = np.array(y\_test)

y\_pred\_log\_array = np.array(y\_pred\_log)

# Calculate regression metrics for Logistic Regression

mae\_log, mse\_log, rmse\_log, r2\_log = calculate\_regression\_metrics(y\_test\_log\_array, y\_pred\_log\_array)

print("Logistic Regression Metrics:")

print("Logistic Regression MAE:", mae\_log)

print("Logistic Regression MSE:", mse\_log)

print("Logistic Regression RMSE:", rmse\_log)

print("Logistic Regression R2 Score:", r2\_log)

# This prints confusion matrix for the test dataset

labels\_log = np.unique(y\_test)

m\_log = confusion\_matrix(y\_test\_log\_array, log\_reg.predict(test\_corpus\_tf\_idf), labels=labels\_log)

print("\nConfusion matrix on test data for Logistic Regression")

cm\_log = pd.DataFrame(m\_log, index=labels\_log, columns=labels\_log)

cm\_log.index = "Actual: " + cm\_log.index

cm\_log.columns = "Predicted: " + cm\_log.columns

display(cm\_log)

# Saving the data into a csv file in the current folder

temp\_df\_log = pd.DataFrame()

temp\_df\_log["News"] = X\_test

temp\_df\_log["Actual Price Sentiment"] = y\_test\_log\_array

temp\_df\_log["Predicted Sentiment"] = log\_reg.predict(test\_corpus\_tf\_idf)

temp\_df\_log.to\_csv("predicted\_logistic.csv")

print('Predictions on Test Data for Logistic Regression are as follows:')

display(temp\_df\_log)

except Exception as e:

print(f"Logistic Regression Model training failed. Error: {e}")

return tf\_idfvectorizer, log\_reg

# Testing the SVM model

vectorizer\_svm, model\_svm = headline\_sentiment\_svm(df)

# Testing the Logistic Regression model

vectorizer\_logistic, model\_logistic = headline\_sentiment\_logistic(df)