\*AIML-PROJECT REPORT\*

INTRODUCTION:

PROJECT OVERVIEW: The project aims to revolutionize ecommerce shipping by leveraging machine learning to predict delivery times accurately. Focused on enhancing customer satisfaction and operational efficiency, the model analyzes diverse datasets to optimize shipping logistics. By addressing the challenges of timely deliveries, the project seeks to reduce costs and establish a more reliable and transparent ecommerce shipping process. With a scope covering various order types and regions, the project's expected outcomes include improved delivery accuracy and a positive impact on the overall ecommerce experience. Ethical considerations are paramount, ensuring fairness and responsible use of the predictive model in ecommerce operations.

PURPOSE: The purpose of this project is to enhance ecommerce shipping operations through the implementation of a machine learning model. By accurately predicting delivery times, the project aims to improve customer satisfaction, optimize logistics, and reduce shipping costs. The ultimate goal is to establish a more efficient and transparent shipping process in the ecommerce industry

LITERATURE SURVEY:

EXISTING PROBLEM: The existing problem in ecommerce shipping lies in the difficulty of accurately predicting delivery times. Inconsistent estimations can lead to customer dissatisfaction, increased operational costs, and a lack of transparency in the shipping process. Addressing this issue is crucial for improving overall customer experience and optimizing logistics in the ecommerce industry.

REFERENCES: ><https://scholar.google.com/>

><https://pubmed.ncbi.nlm.nih.gov/>

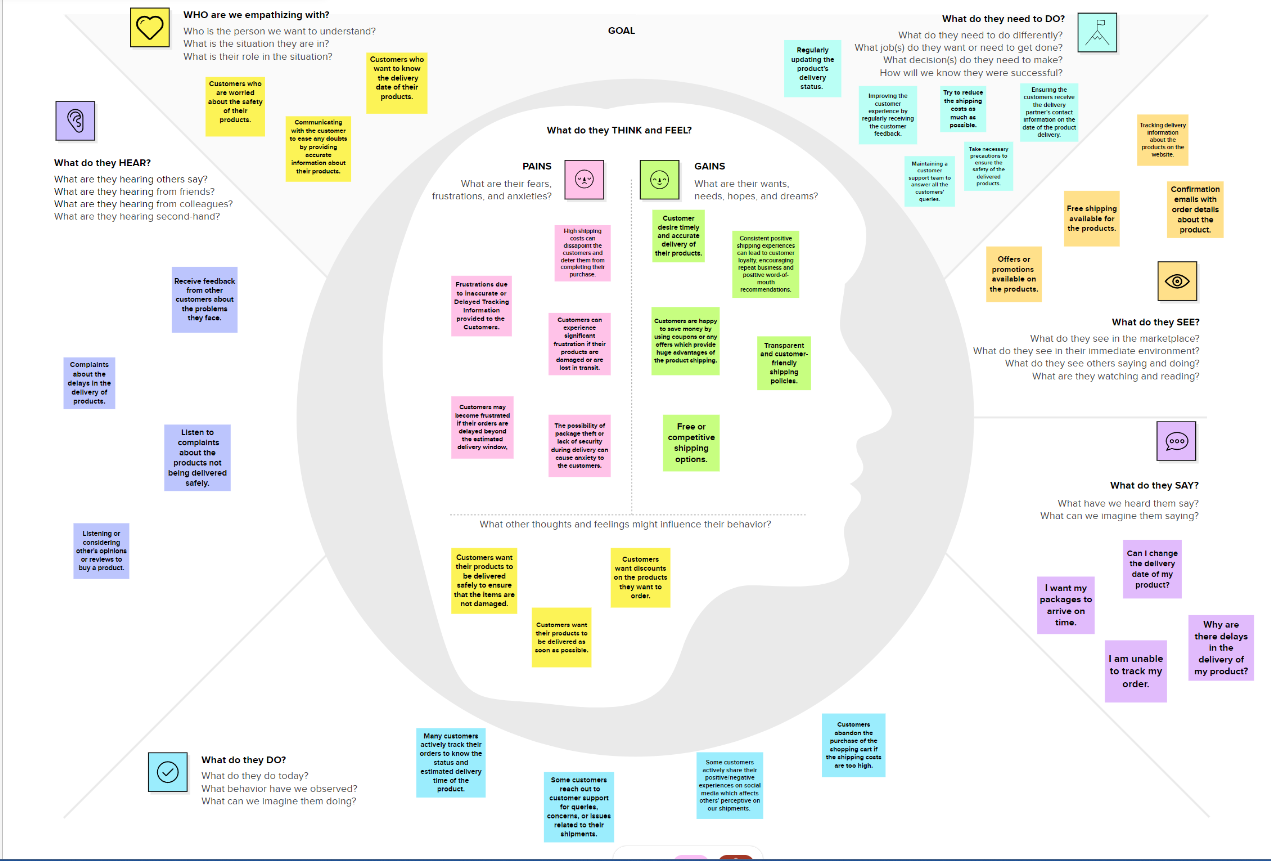
><https://ieeexplore.ieee.org/Xplore/home.jsp>

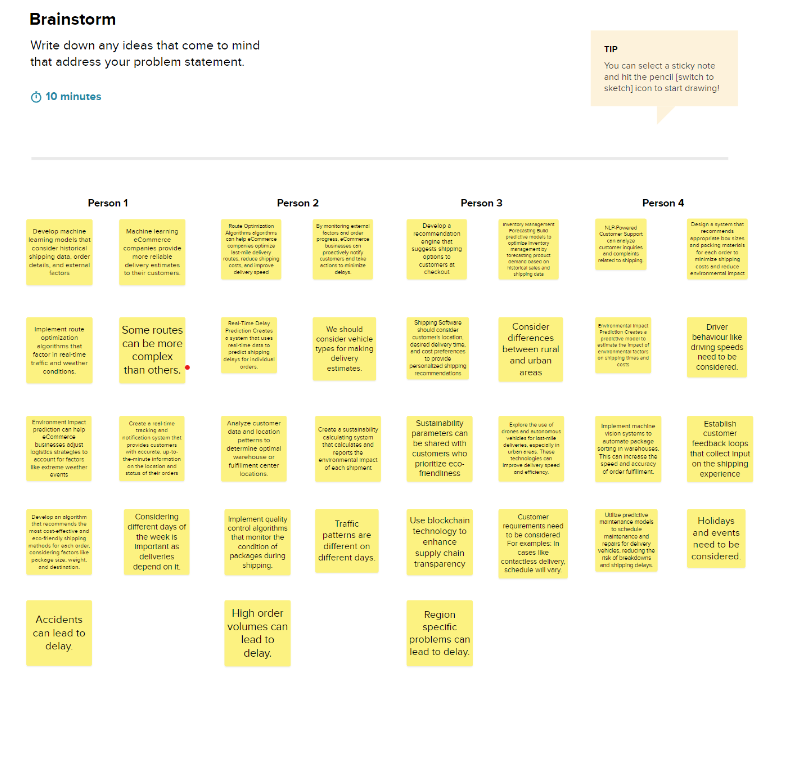
><https://www.researchgate.net/>

PROBLEM STATEMENT DEFINITION: Inconsistent and inaccurate shipping time estimates in ecommerce lead to customer dissatisfaction, increased operational costs, and a lack of transparency. This project addresses these challenges by implementing a machine learning model to predict shipping times accurately. The goal is to enhance customer satisfaction, optimize operational efficiency, and establish a more transparent and reliable ecommerce shipping process.

IDEATIOIN&PROPOSED SOLUTION:

EMPATHY MAP CANVAS:



IDEATION & BRAINSTROMING: 

REQUIREMENT ANLYSIS:   
Functional requirements : It describe the specific functionalities or features that a system, software, or product must have to meet its intended purpose. For an ecommerce shipping predictions project, some functional requirements could include:

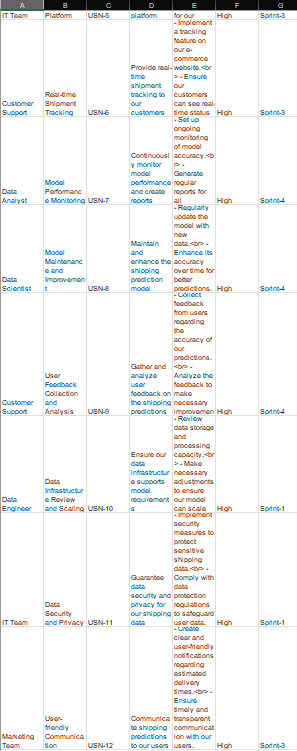
1. **Prediction Accuracy:**
   * The system should accurately predict shipping times for individual orders based on historical and real-time data.
2. **Scalability:**
   * The model should be scalable to handle varying order volumes and adapt to the growing demands of an ecommerce platform.
3. **Integration:**
   * The system must seamlessly integrate with existing ecommerce platforms to retrieve order data and provide accurate predictions.
4. **Real-time Updates:**
   * Provide real-time updates on shipping status and potential delays, ensuring customers are informed promptly.
5. **User Authentication:**
   * Implement secure user authentication mechanisms to control access to sensitive shipping data and prediction functionalities.
6. **Customization:**
   * Allow for customization of prediction models based on specific product categories, geographical regions, or other relevant parameters.
7. **Performance Monitoring:**
   * Include mechanisms for monitoring the performance of the prediction model, with the ability to identify and address issues promptly.

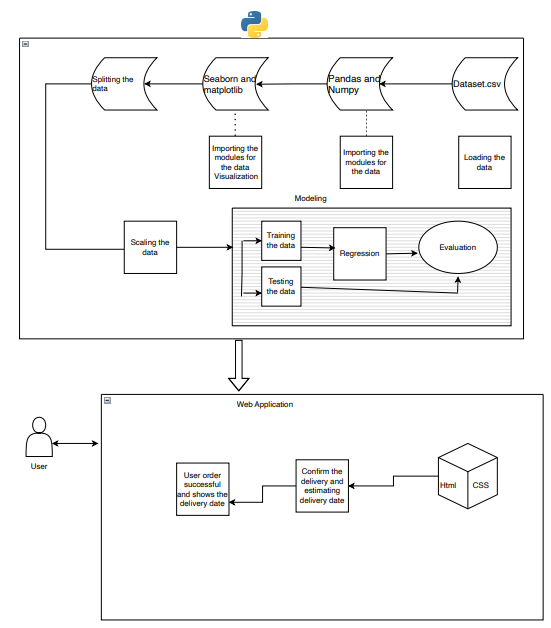
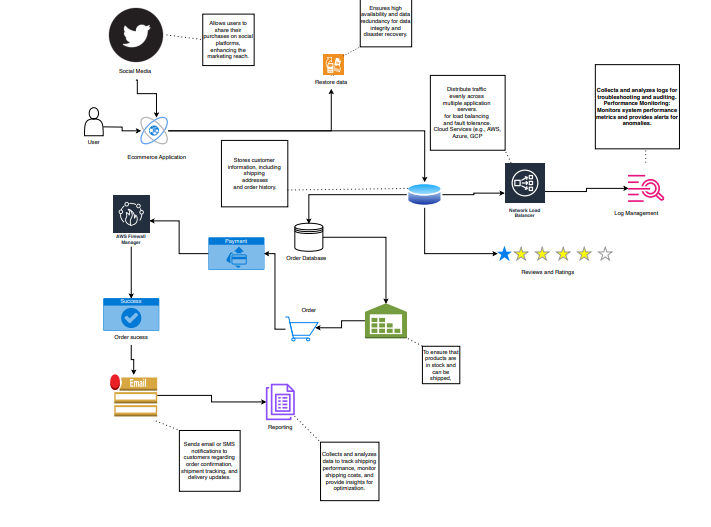
NON-FUNCTIONAL REQUIREMENTS: It specify the characteristics that the system must have, but they do not relate directly to the system's functionality. For an ecommerce shipping predictions project, non-functional requirements might include:

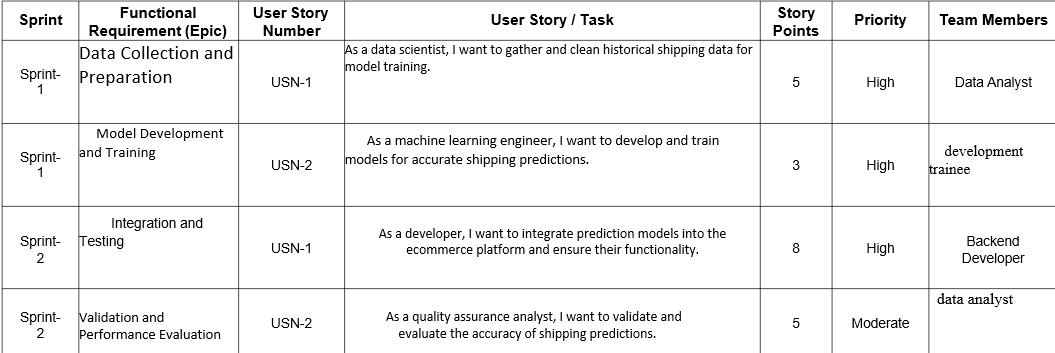
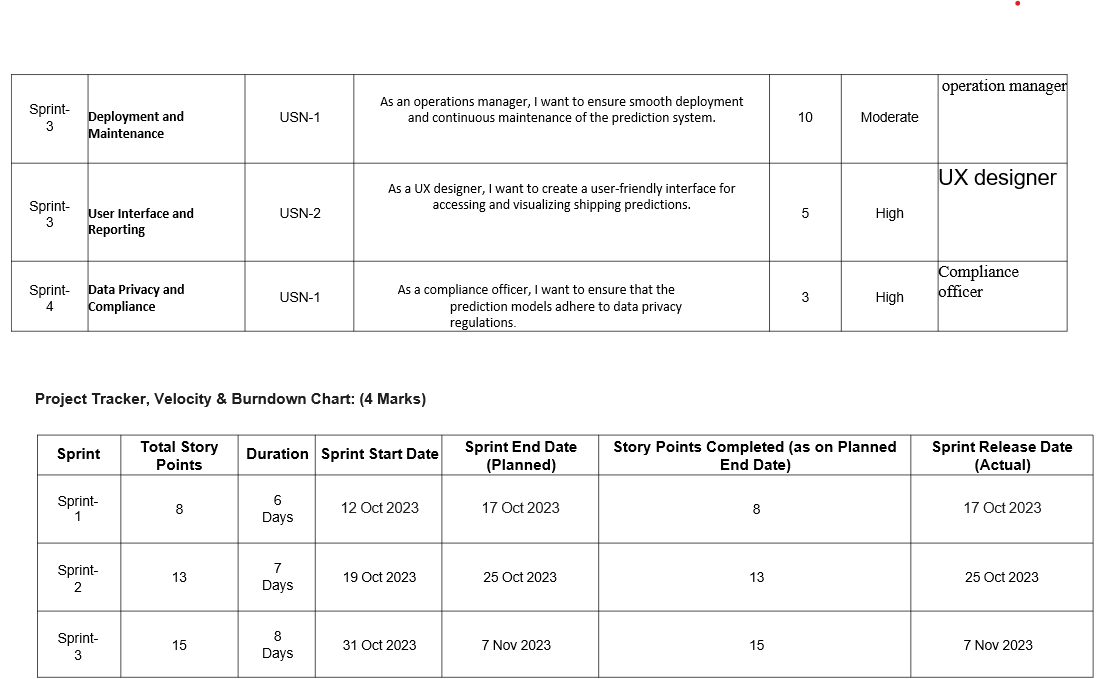
1. **Performance:**
   * The system should provide accurate predictions within a specified response time, even under peak order loads.
2. **Reliability:**
   * The prediction model must be highly reliable, minimizing downtime and ensuring consistent performance under various conditions.
3. **Scalability:**
   * The system should scale efficiently to handle an increasing volume of orders without compromising prediction accuracy.
4. **Usability:**
   * The user interface must be intuitive, ensuring that both ecommerce operators and customers can easily access and interpret shipping predictions.
5. **Availability:**
   * The system should be available 24/7, ensuring that users can access shipping predictions at any time without significant downtime.
6. **Security:**
   * Implement robust security measures to protect sensitive customer and shipping data, complying with industry standards and regulations.

PROJECT DESIGN:

DATA FLOW:



USER STORIES:  
  
SOLUTION ARCHITECTURE:  


PROJECT PLANNING AND SHEDULING:  
  
  


CODING & SOLUTIONING:

Flask code is used as connection between backend and frontend.  
code:  
import pickle

from flask import Flask, request, render\_template

app=Flask(\_name\_)

model=pickle.load(open("bestmodel.pkl","rb"))

data\_normalizer=pickle.load(open("Scaler.pkl","rb"))

@app.route('/')

def home():

return render\_template('home.html')

@app.route('/home')

def home\_direct():

return render\_template('home.html')

@app.route('/about')

def about():

return render\_template('about.html')

@app.route('/predict')

def predict():

return render\_template("predict.html")

@app.route('/output')

def output():

return render\_template("output.html")

@app.route('/submit',methods=['GET','POST'])

def submit():

Discount\_offered=eval(request.form["Discount\_offered"])

Weight\_in\_gms=eval(request.form["Weight\_in\_gms"])

preds=[[Discount\_offered,Weight\_in\_gms]]

predclass=model.predict(data\_normalizer.fit\_transform(preds))

prob=model.predict\_proba(data\_normalizer.fit\_transform(preds))[0]

reach=prob[1]

percent=reach\*100

return render\_template("output.html", output=percent)

if \_name=='\_\_main\_':

app.run(debug=True)

ML CODE: it is used as main sourse code.  
   
code :  
# -\*- coding: utf-8 -\*-

"""Project.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1qJDaAu9CtUBU67HdOTNJMS05mFx-BD2c

"""

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

import pickle as pkl

import numpy as np

import random

from sklearn import svm

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression, LogisticRegressionCV, RidgeClassifier

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from xgboost import XGBClassifier

from sklearn.preprocessing import Normalizer

from sklearn.metrics import accuracy\_score, f1\_score, recall\_score, precision\_score, confusion\_matrix

data = pd.read\_csv("dataset.csv")

data.head()

data.shape

data.info()

data.isnull().sum()

label\_map={}

for i in data.columns:

if str(data[i].dtype) == 'object':

temp={}

cats=data[i].unique()

for index in range(len(cats)):

temp[cats[index]]=index

label\_map[i]=temp

#Labeling

data[i]=data[i].map(temp)

label\_map

c=0

plt.figure(figsize=(18, 10))

for i in data.drop(columns=[

'Warehouse\_block', 'Mode\_of\_Shipment', 'Product\_importance', 'Gender','Reached.on.Time\_Y.N','ID'

]).columns:

if str(data[i].dtype)=="object ":

continue

plt.subplot(2, 3, c+1)

plt.boxplot(data[i])

plt.title(i)

c+=1

plt.show()

def check\_outliers (arr):

Q1= np.percentile(arr, 25, interpolation = 'midpoint')

Q3= np.percentile(arr, 75, interpolation = 'midpoint')

IQR = Q3 - Q1

#Above Upper bound

upper=Q3+1.5\*IQR

upper\_array=np.array(arr>=upper)

print(' '\*3,len(upper\_array[upper\_array == True]), 'are over the upper bound:',upper)

#BeLow Lower bound

lower=Q1-1.5\*IQR

lower\_array=np.array(arr<=lower)

print(' '\*3,len(lower\_array[lower\_array == True]), 'are less than the lower bound:', lower, '\n')

for i in data.drop(columns=[

'Warehouse\_block', 'Mode\_of\_Shipment', 'Product\_importance', 'Gender', 'Reached.on.Time\_Y.N','ID'

]).columns:

if str(data[i].dtype)=='object':

continue

print(i)

check\_outliers(data[i])

data.describe(include='all')

for column in data.columns:

random\_color = "#{:02x}{:02x}{:02x}".format(random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))

plt.figure(figsize=(8, 4))

plt.hist(data[column], bins=50, color=random\_color)

plt.title(f'{column}') # Set the title

plt.xlabel(column) # Set the x-axis label

plt.ylabel('Count') # Set the y-axis label

plt.show() # Show the plot

target\_column = 'Reached.on.Time\_Y.N'

column\_to\_skip = 'ID'

for column in data.columns:

if column != target\_column and column != column\_to\_skip:

plt.figure(figsize=(8, 6))

plt.scatter(data[column], data[target\_column])

plt.title(f'Scatter Plot of {column} vs. {target\_column}')

plt.xlabel(column)

plt.ylabel(target\_column)

plt.show()

correlation\_matrix = data.corr()

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', linewidths=0.5)

plt.title('Correlation Heatmap')

plt.show()

x = data[['Discount\_offered','Weight\_in\_gms']]

y = data['Reached.on.Time\_Y.N']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,

random\_state=1234,test\_size = 0.10,

shuffle=True

)

print(x\_train.shape)

print(x\_test.shape)

print(y\_train.shape)

print(y\_test.shape)

scaler = StandardScaler()

x\_train\_standardized = scaler.fit\_transform(x\_train)

x\_test\_standardized = scaler.transform(x\_test)

from sklearn.preprocessing import StandardScaler

def models\_eval\_mm(x\_train, y\_train, x\_test, y\_test):

# Logistic Regression

lg = LogisticRegression(random\_state=1234)

lg.fit(x\_train\_standardized, y\_train)

print("--Logistic Regression--")

print("Train Score:", lg.score(x\_train\_standardized, y\_train))

print("Test Score:", lg.score(x\_test\_standardized, y\_test))

print()

# Logistic Regression with Cross-Validation

lcv = LogisticRegression(random\_state=1234)

lcv.fit(x\_train\_standardized, y\_train)

print("--Logistic Regression CV--")

print("Train Score:", lcv.score(x\_train\_standardized, y\_train))

print("Test Score:", lcv.score(x\_test\_standardized, y\_test))

print()

# XGBoost

xgb = XGBClassifier(random\_state=1234)

xgb.fit(x\_train\_standardized, y\_train)

print("--XGBoost--")

print("Train Score:", xgb.score(x\_train\_standardized, y\_train))

print("Test Score:", xgb.score(x\_test\_standardized, y\_test))

print()

# Ridge Classifier

rg = RidgeClassifier(random\_state=1234)

rg.fit(x\_train\_standardized, y\_train)

print("--Ridge Classifier--")

print("Train Score:", rg.score(x\_train\_standardized, y\_train))

print("Test Score:", rg.score(x\_test\_standardized, y\_test))

print()

# K-Nearest Neighbors

knn = KNeighborsClassifier()

knn.fit(x\_train\_standardized, y\_train)

print("--KNN--")

print("Train Score:", knn.score(x\_train\_standardized, y\_train))

print("Test Score:", knn.score(x\_test\_standardized, y\_test))

print()

# Random Forest

rf = RandomForestClassifier(random\_state=1234)

rf.fit(x\_train\_standardized, y\_train)

print("--Random Forest--")

print("Train Score:", rf.score(x\_train\_standardized, y\_train))

print("Test Score:", rf.score(x\_test\_standardized, y\_test))

print()

# Support Vector Machine (SVM)

svc = svm.SVC(random\_state=1234)

svc.fit(x\_train\_standardized, y\_train)

print("--SVM Classifier--")

print("Train Score:", svc.score(x\_train\_standardized, y\_train))

print("Test Score:", svc.score(x\_test\_standardized, y\_test))

print()

return lg, lcv, xgb, rg, knn, rf, svc

lg,lcv,xgb,rg,knn,rf,svc = models\_eval\_mm(x\_train\_standardized,y\_train,x\_test\_standardized,y\_test)

# Logistic Regression

lg\_prediction = lg.predict(x\_test.iloc[0].values.reshape(1, -1))

print("Logistic Regression Prediction:", lg\_prediction)

# Logistic Regression with Cross-Validation

lcv\_prediction = lcv.predict(x\_test.iloc[0].values.reshape(1, -1))

print("Logistic Regression with Cross-Validation Prediction:", lcv\_prediction)

# XGBoost

xgb\_prediction = xgb.predict(x\_test.iloc[0].values.reshape(1, -1))

print("XGBoost Prediction:", xgb\_prediction)

# Ridge Classifier

rg\_prediction = rg.predict(x\_test.iloc[0].values.reshape(1, -1))

print("Ridge Classifier Prediction:", rg\_prediction)

# K-Nearest Neighbors

knn\_prediction = knn.predict(x\_test.iloc[0].values.reshape(1, -1))

print("K-Nearest Neighbors Prediction:", knn\_prediction)

# Random Forest

rf\_prediction = rf.predict(x\_test.iloc[0].values.reshape(1, -1))

print("Random Forest Prediction:", rf\_prediction)

# Support Vector Machine (SVM)

svc\_prediction = svc.predict(x\_test.iloc[0].values.reshape(1, -1))

print("Support Vector Machine (SVM) Prediction:", svc\_prediction)

def evaluate\_model(name, model, x\_test, y\_test):

y\_pred = model.predict(x\_test\_standardized)

result = []

result.append(name)

result.append(f"Accuracy: {accuracy\_score(y\_test, y\_pred) \* 100:.2f}%")

result.append(f"F1 Score: {f1\_score(y\_test, y\_pred) \* 100:.2f}%")

result.append(f"Recall: {recall\_score(y\_test, y\_pred) \* 100:.2f}%")

result.append(f"Precision: {precision\_score(y\_test, y\_pred) \* 100:.2f}%")

return result

model\_list = {

"Logistic Regression": lg,

"Logistic Regression CV": lcv,

"XGBoost": xgb,

"Ridge Classifier": rg,

"KNN": knn,

"Random Forest": rf,

"Support Vector Classifier": svc

}

model\_eval\_info = []

for model\_name, model in model\_list.items():

model\_eval\_info.append(evaluate\_model(model\_name, model, x\_test\_standardized, y\_test))

model\_eval\_info = pd.DataFrame(model\_eval\_info, columns=["Name", "Accuracy", "F1 Score", "Recall", "Precision"])

model\_eval\_info.to\_csv("model\_eval.csv")

model\_eval\_info

import pickle

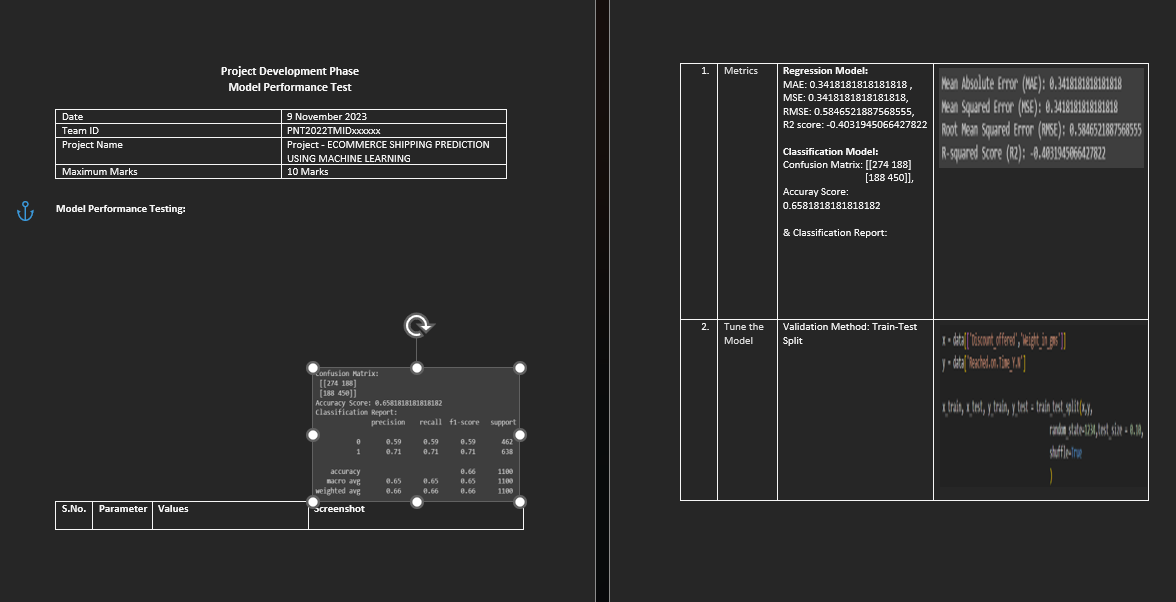
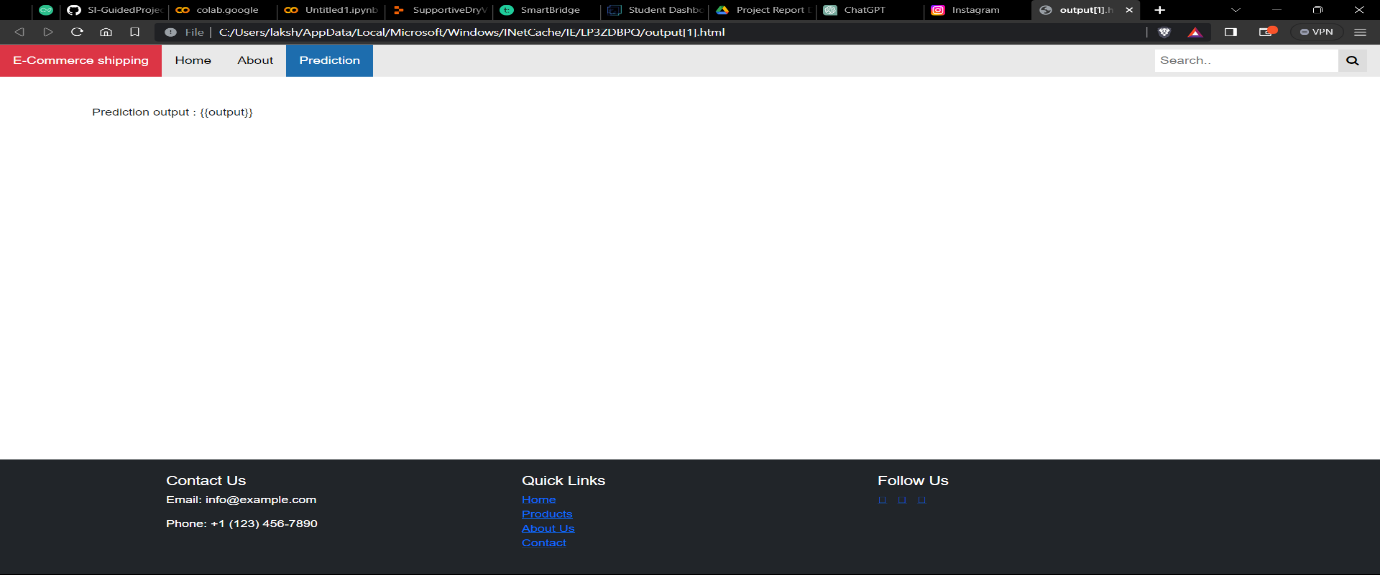
filename="bestmodel.pkl"

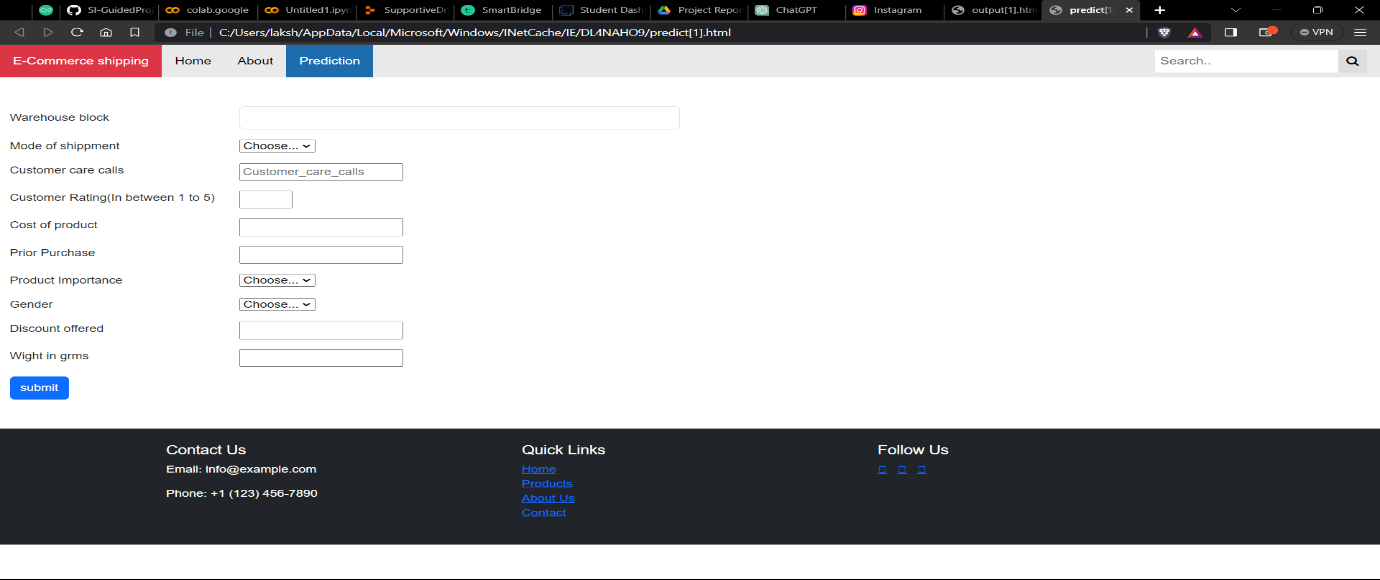
pickle.dump(knn,open(filename,'wb'))

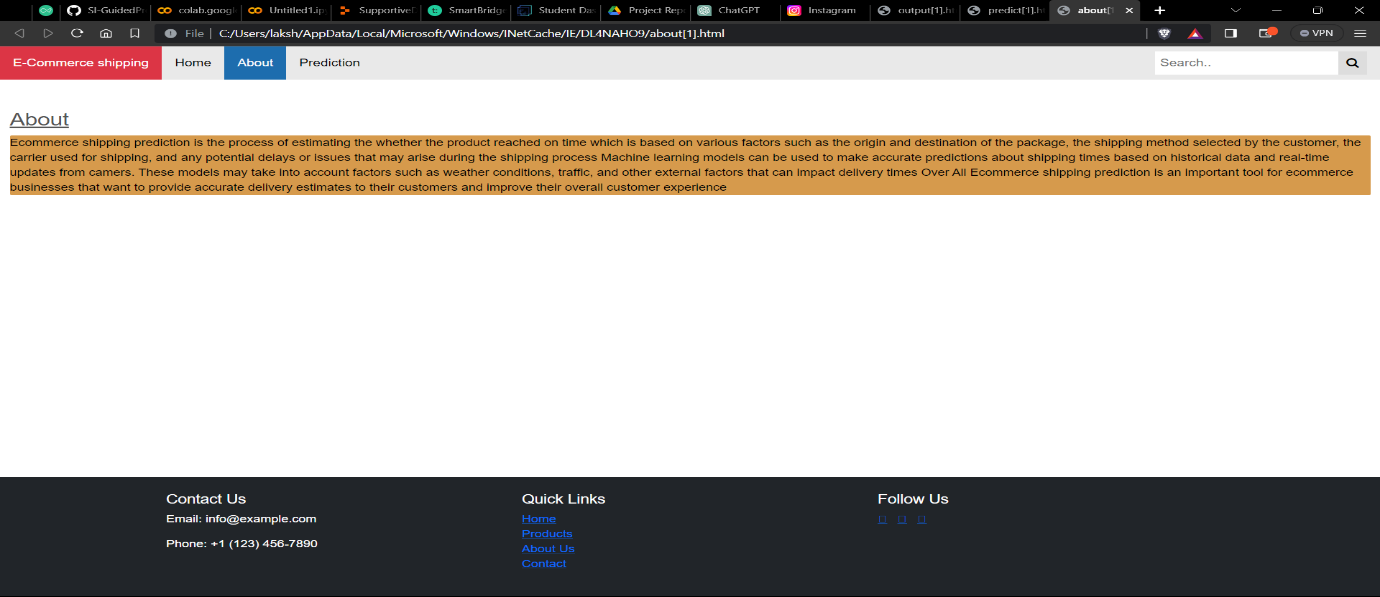
from sklearn.preprocessing import StandardScaler

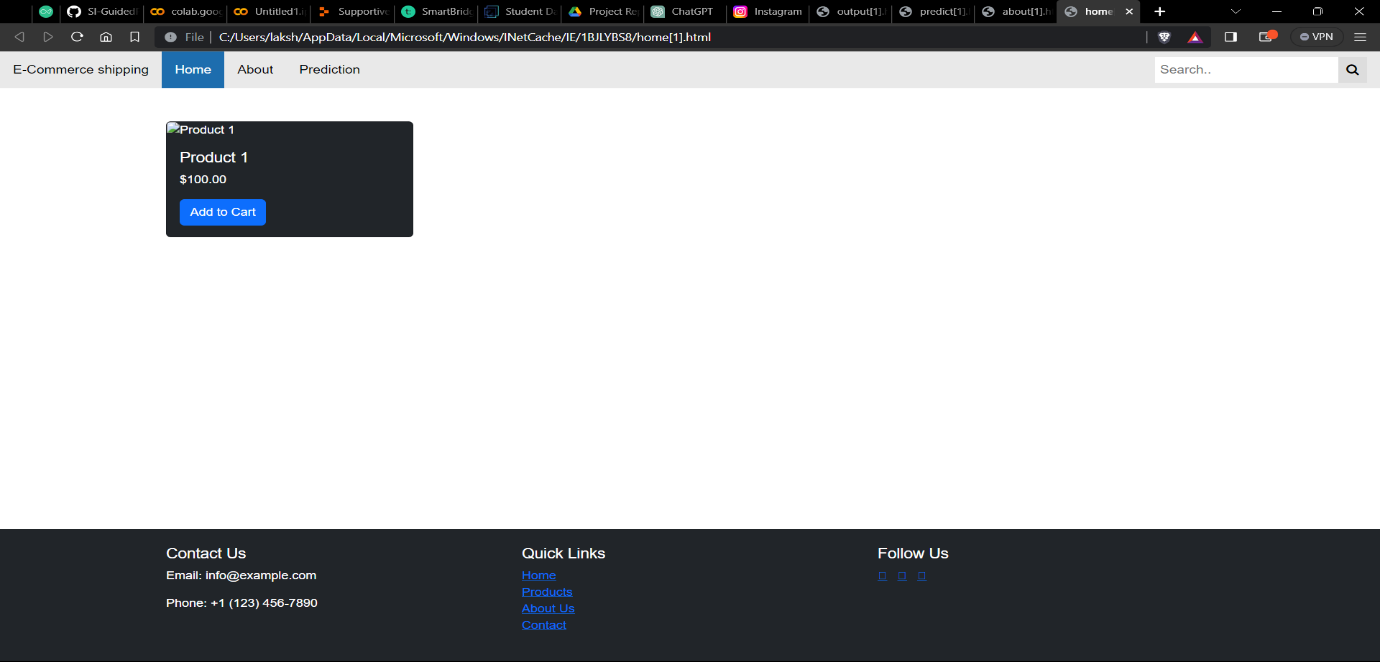
scaler2=StandardScaler()

filename2="Scaler.pkl"

pickle.dump(scaler2,open(filename2,'wb'))  
  
PERFORMANCE TESTING:  
  
OUTPUTS :  








1. ADVANTAGES:  
   **Improved Customer Satisfaction:**
   * Accurate shipping predictions lead to more reliable delivery timelines, enhancing overall customer satisfaction and loyalty.
2. **Cost Reduction:**
   * Streamlining shipping processes through accurate predictions can reduce operational costs by optimizing resource allocation and minimizing delays.
3. **Enhanced Operational Efficiency:**
   * A well-implemented shipping predictions system can improve the efficiency of logistics operations, reducing bottlenecks and optimizing routes.

**4.Increased Transparency:**

Accurate predictions provide transparency into the shipping process, allowing customers to track their orders and receive real-time updates.

1. DISADVANTAGES:  
   **Data Privacy Concerns:**
   * Collecting and processing customer data for shipping predictions may raise concerns about data privacy, requiring robust security measures and compliance with regulations.
2. **Algorithmic Bias:**
   * Machine learning models may exhibit biases based on historical data, potentially leading to unfair predictions. It's crucial to address and mitigate bias issues.
3. Initial Implementation Costs:
   * Implementing a machine learning model and integrating it into existing systems may involve upfront costs for technology, training, and infrastructure.

CONCLUSION:  
In conclusion, the implementation of a machine learning model for ecommerce shipping predictions offers significant advantages such as enhanced customer satisfaction, operational efficiency, and cost reduction. However, challenges like algorithmic bias, data privacy concerns, and initial implementation costs must be addressed to ensure the success and ethical deployment of the system. Striking a balance between technological innovation and user adaptation is paramount for the sustainable improvement of ecommerce logistics.

1. FEATURE SCOPE:  
   **Order Data Integration:**
   * Seamless integration with ecommerce platforms to import and analyze order data for accurate shipping predictions.
2. **Machine Learning Model:**
   * Implementation of a robust machine learning model capable of predicting shipping times based on historical and real-time data.
3. **Real-time Tracking:**
   * Providing customers and ecommerce operators with real-time tracking capabilities for enhanced visibility into the shipping process.
4. **Customizable Parameters:**
   * Flexibility in customizing prediction parameters such as delivery destinations, product categories, and shipping methods.
5. **Security and Authentication:**
   * Implementation of secure user authentication and authorization mechanisms, along with data security measures, to protect sensitive shipping data.

GITHUB LINK:  
  
https://smartinternz.com/Student/guided\_project\_workspace/593046

PROJECT DEMO LIMK:  
  
https://drive.google.com/drive/folders/113olfI\_PeIXI1DYjH9zZ\_heQcY9B\_d2i?usp=sharing