International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 2 Volume No.7 March - April – 2023
DOI:10.46647/ijetms.2023.v07i02.066 ISSN: 2581-4621

DATA POISON DETECTION USING MACHINE LEARNING

Buvaneswari M[1],Ramalakshmanan Alias Manikandan U[2] , Ram Kumar K[3] , Selvabharathi S[4]
[1] Assistant Professor - Computer Science Engineering, Paavai Engineering College, Namakkal, Tamil Nadu
[2,3,4] UG - Computer Science Engineering, Paavai Engineering College, Namakkal, Tamil Nadu

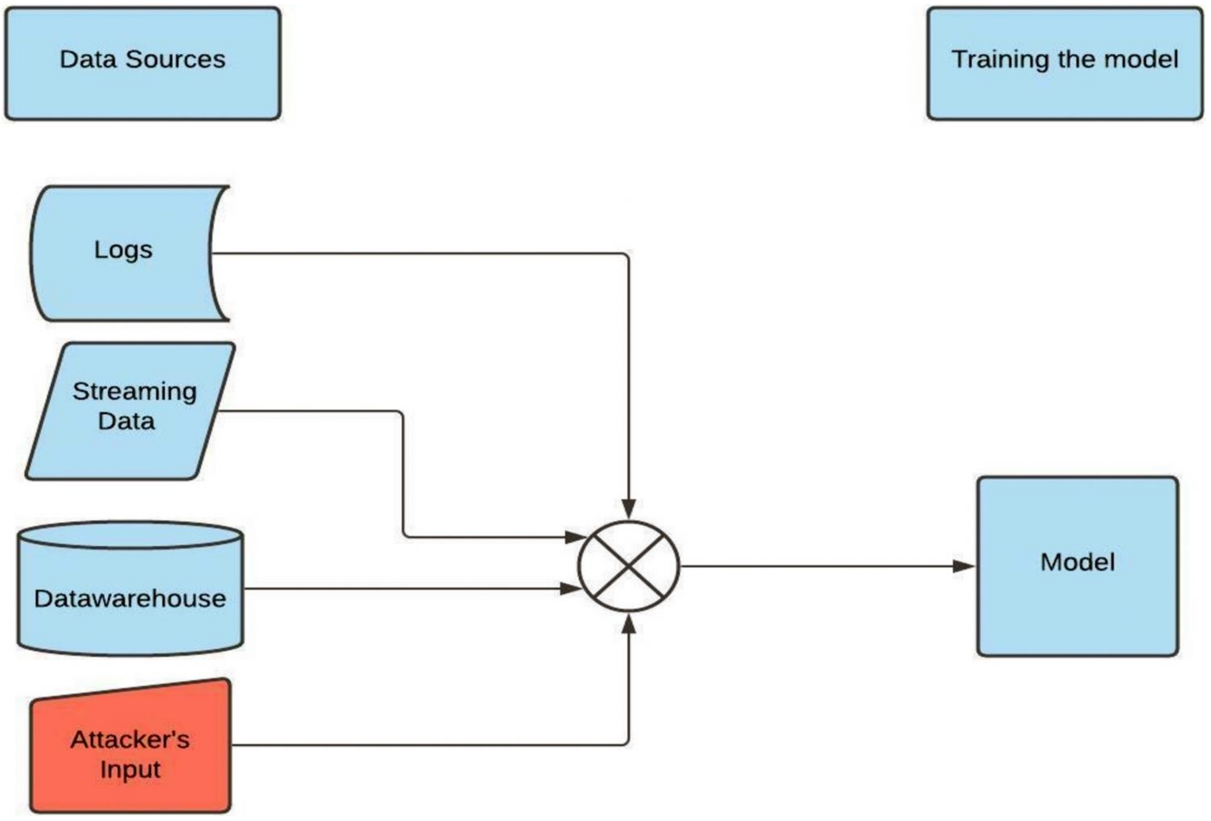
ABSTRACT
Distributed machine learning (DML) can realize massive dataset training when no single node can work out the accurate results within an acceptable time. However, this will inevitably expose more potential targets to attackers compared with the non-distributed environment. In this paper, we classify DML into basic-DML and semi-DML. In basic-DML, the center server dispatches learning tasks to distributed machines and aggregates their learning results. While in semi-DML, the center server further devotes resources into dataset learning in addition to its duty in basic-DML. We firstly put forward a novel data poison detection scheme for basic-DML, which utilizes a cross-learning mechanism to find out the poisoned data. We prove that the proposed cross-learning mechanism would generate training loops, based on which a mathematical model is established to find the optimal number of training loops. Then, for semi-DML, we present an improved data poison detection scheme to provide better learning protection with the aid of the central resource. To efficiently utilize the system resources, an optimal resource allocation approach is developed. Simulation results show that the proposed scheme can significantly improve the accuracy of the final model by up to 20% for support vector machine and 60% for logistic regression in the basic-DML scenario. Moreover, in the semi-DML scenario, the improved data poison detection scheme with optimal resource allocation can decrease the wasted resources for 20-100%.
Keywords—Distributed machine learning,demi-DM

1. Introduction

Suppose you call one of your company suppliers to understand why they stopped sending you emails about promotions. The supplier replies that they continue to send their promotion as usual and invite you to check the spam. The supplier was right! The emails ended up in your spam folder, together with other communications from that company. This could have been happening by accident, but instead as a result of fraud, in which an evil competitor ensures that the email client marks any email from the victim company as spam. To this end, this malicious company could flood you with spam also containing the victim company's name - until your machine learning-based spam filter associates this benign name with the property "spam", thus trashing any future promotions. This scenario is an instance of a machine learning security threat called data poisoning, described already in 2008 by Nelson et al. .
Under this setting, malicious users may cause failures in ML systems (e.g., spam filters) by tampering with their training data, thereby posing real concerns about their trustworthiness. Several sources confirm that poisoning is already carried out in practice . For example, Microsoft's chatbot Tay1 was designed to learn language by interacting with users, but instead learned offensive statements. Alternatively, a group of extremists submitted wrongly-labeled images of portable ovens with wheels tagging them as Jewish baby strollers to poison Google's image search.
In addition to data poisoning, other attacks are threatening the reliability and robustness of ML systems. Evasion attacks have been conceived to force the victim's model to output wrong predictions at test time. For example, a malicious user can craft printable stickers that can confuse a classifier to classify a stop sign, posing different security concerns for the userssafety and self-driving industry.
Privacy attacks have been devised to extract private information about the target system (via model stealing), its users (via model extraction), or its sensitive training data (via membership inference), thus undermining the system's intellectual property or users' privacy. The potential harm that evasion,

privacy, and poisoning attacks can cause, along with other AI-related risks, have led the European Union (EU) to publish the EU AI Act, to start defining proper policies for AI trustworthiness. These regulations require AI systems to provide not only accurate but also human-aligned decisions, which follow the principles of being explainable, fair, robust, and accountable.

2. Experimental Methods or Methodology



Data poisoning can render machine learning models inaccurate, possibly resulting in poor decisions based on faulty outputs. With no easy fixes available, security pros must focus on prevention and detection. Machine learning adoption exploded over the past decade, driven in part by the rise of cloud computing, which has made high performance computing and storage more accessible to all businesses. As vendors integrate machine learning into products across industries, and users rely on the output of its algorithms in their decision making, security experts warn of adversarial attacks designed to abuse the technology. Most social networking platforms, online video platforms, large shopping sites, search engines and other services have some sort of recommendation system based on machine learning. The movies and shows that people like on Netflix, the content that people like or share on Facebook, the hashtags and likes on Twitter, the products consumers buy or view on Amazon, the queries users type in Google Search are all fed back into these sites' machine learning models to make better and more accurate recommendations.

3. Results and Discussion

DATA POISONING

Data poisoning or model poisoning attacks involve polluting a machine learning model's training data. Data poisoning is considered an integrity attack because tampering with the training data impacts the model's ability to output correct predictions.

PREVENT AND DETECT

The difficulties in fixing poisoned models, model developers need to focus on measures that could either block attack attempts or detect malicious inputs before the next training cycle happens things like input validity checking, rate limiting, regression testing, manual moderation and using various statistical techniques to detect anomalies. For example, a small group of accounts, IP addresses, or users shouldn't account for a large portion of the model training data. Restrictions can be placed on how many inputs provided by a unique user are accepted into the training data or with what weight.

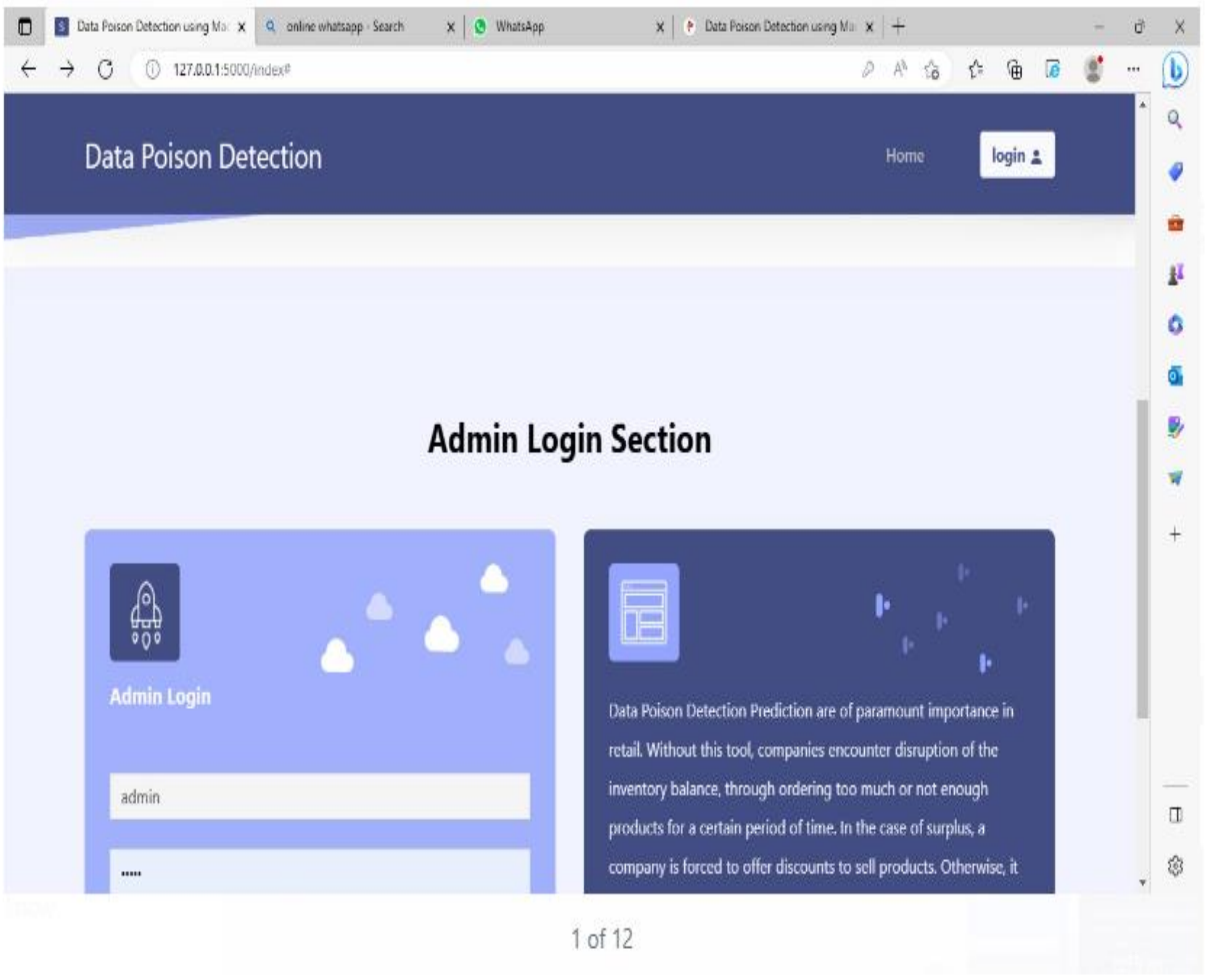
DATA POISONING ATTACKS IN DEEP LEARNING VISION SYSTEMS

The practice of using deep learning methods in safety critical vision systems such as autonomous driving has come a long way. As vision systems supported by deep learning methods become ubiquitous, the possible security threats faced by these systems have come into greater focus. As it is with any artificial intelligence system, these deep neural vision networks are first trained on a data set of interest, once they start performing well, they are deployed to a real-world environment. In the training stage, deep learning systems are susceptible to data poisoning attacks.

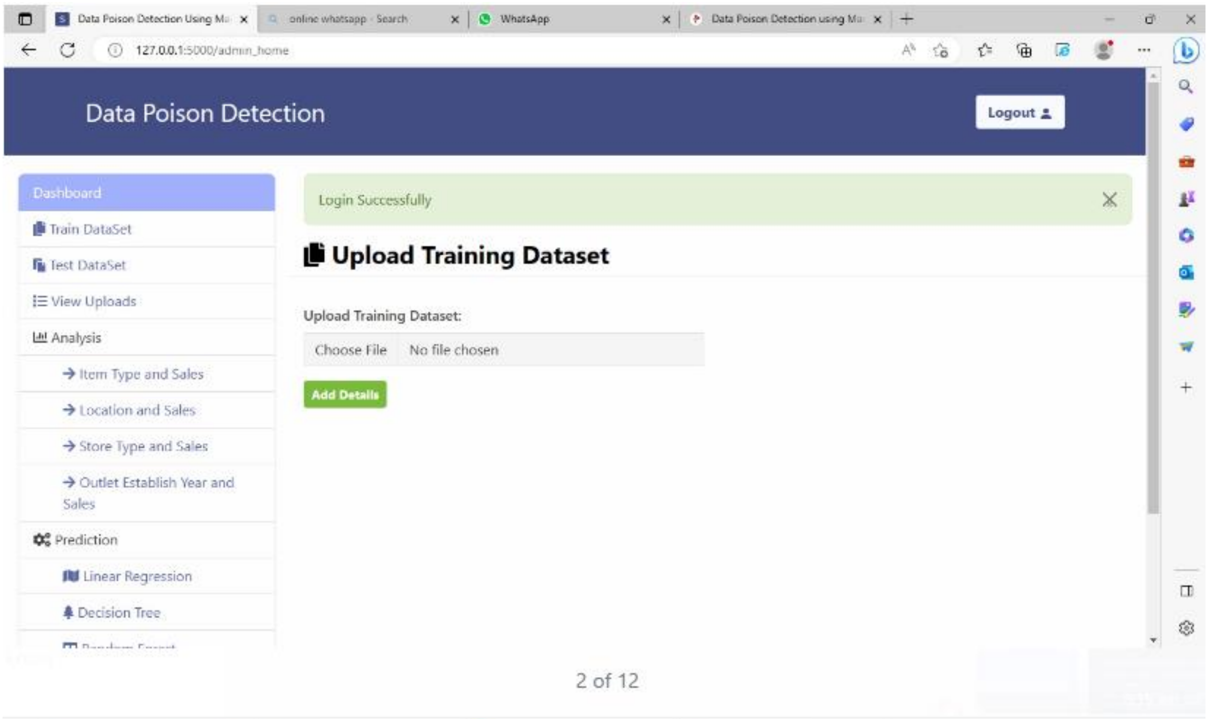
SOFTWARE CODE

```
from flask import Flask,flash, render_template, request,url_for,redirect,Response,redirect,session
from datetime import datetime
from functools import wraps
from flask_mysqldb import MySQL
import os
import numpy as np
import pandas as pd
import RF
import DT
import LR
import XG
app=Flask(__name__)
app.config['MYSQL_HOST']='localhost'
app.config['MYSQL_USER']='root'
app.config['MYSQL_PASSWORD']=''
app.config['MYSQL_DB']='db_pro_inventory'
app.config['MYSQL_CURSORCLASS']='DictCursor'
mysql=MySQL(app)
UPLOAD_FOLDER='static/file'
PREDICT_FOLDER='static/predict'
FILE_EXTENSIONS = {'csv'}
#check file extension
def allowed_extensions(file_name):
    return '.' in file_name and file_name.rsplit('.',1)[1].lower() in FILE_EXTENSIONS
#define function for convert yyyy-mm-dd to dd-mm-yyyy
def format_datetime(value, format="%d-%m-%Y"):
    if value is None:
        return ""
    return datetime.strptime(value,"%Y-%m-%d").strftime(format)
#configured Jinja2 environment with user defined
app.jinja_env.filters['date_format']=format_datetime
#Login
@app.route('/')
@app.route('/index',methods=['POST','GET'])
def index():
    status=True
    if request.method=='POST':
        email=request.form["aname"]
        pwd=request.form["apass"]
        cur=mysql.connection.cursor()
        cur.execute("select * from admin where ANAME=%s and APASS=%s",(email,pwd))
        data=cur.fetchone()
        if data:
            session['logged_in']=True
            session['aname']=data["ANAME"]
            flash('Login Successfully','success')
            return redirect('admin_home')
        else:
            flash('Invalid Login. Try Again','danger')
    return render_template("index.html")
#check if user logged in
def is_logged_in(f):
     @wraps(f)
     def wrap(*args,**kwargs):
            if 'logged_in' in session:
                        return f(*args,**kwargs)
            else:
                        flash('Unauthorized, Please Login','danger')
                        return redirect(url_for('index'))
     return wrap
@app.route('/admin_home',methods=['POST','GET'])
def admin_home():
    if request.method=='POST':
        if 'img' not in request.files:
            flash('No file part','danger')
        file = request.files['img']
        if file.filename == '':
            flash('No file selected','danger')
        if file and allowed_extensions(file.filename):
            filename, file_extension = os.path.splitext(file.filename)
            new_filename = "train.csv"
            file.save(os.path.join(UPLOAD_FOLDER, new_filename))
            flash('Training Dataset Upload Successfully','success')
        else:
            flash('Upload only csv file','danger')
    return render_template("admin_home.html")
@app.route('/test_dataset',methods=['POST','GET'])
def test_dataset():
    if request.method=='POST':
        if 'img' not in request.files:
            flash('No file part','danger')
        file = request.files['img']
        if file.filename == '':
            flash('No file selected','danger')
        if file and allowed_extensions(file.filename):
            filename, file_extension = os.path.splitext(file.filename)
            new_filename = "test.csv"
            file.save(os.path.join(UPLOAD_FOLDER, new_filename))
            flash('Test Dataset Upload Successfully','success')
        else:
            flash('Upload only csv file','danger')
    return render_template("test_dataset.html")
@app.route("/view_uploads",methods=['POST','GET'])
@is_logged_in
def view_uploads():
    path = "static/file/"
    dir_list = os.listdir(path)
    files=[]
    for f in dir_list:
        files.append(f)
    return render_template("view_uploads.html",data=files)
#admin - delete dataset
@app.route('/delete/<string:file>',methods=['POST','GET'])
@is_logged_in
def delete(file):
    os.remove("static/file/"+file)
    flash("Dataset Deleted Successfully","success")
    return redirect(url_for("view_uploads"))
@app.route("/plot_item_type",methods=['POST','GET'])
@is_logged_in
def plot_item_type():
    import matplotlib
    matplotlib.use('Agg')
    import matplotlib.pyplot as plt
    df=pd.read_csv(UPLOAD_FOLDER+'/train.csv')
    plt.figure()
    plt.gcf().subplots_adjust(left=0.25)
    df.groupby(['Item_Type'])['Item_Outlet_Sales'].sum().plot(kind='barh')
    plt.savefig('static/plot/item_type_and_sales.png')
    return render_template("plot_item_type.html")
@app.route("/plot_item_location",methods=['POST','GET'])
@is_logged_in
def plot_item_location():
    import matplotlib
    matplotlib.use('Agg')
    import matplotlib.pyplot as plt
    df=pd.read_csv(UPLOAD_FOLDER+'/train.csv')
    plt.figure()
    plt.gcf().subplots_adjust(left=0.25)
    df.groupby(['Outlet_Location_Type'])['Item_Outlet_Sales'].sum().plot(kind='bar')
    plt.savefig('static/plot/plot_item_location.png')
    return render_template("plot_item_location.html")
@app.route("/plot_item_store_type",methods=['POST','GET'])
@is_logged_in
def plot_item_store_type():
    import matplotlib
    matplotlib.use('Agg')
    import matplotlib.pyplot as plt
    df=pd.read_csv(UPLOAD_FOLDER+'/train.csv')
    plt.figure()
    plt.gcf().subplots_adjust(left=0.3)
    df.groupby(['Outlet_Type'])['Item_Outlet_Sales'].sum().plot(kind='barh')
    plt.savefig('static/plot/plot_item_store_type.png')
    return render_template("plot_item_store_type.html")
@app.route("/plot_store_establish",methods=['POST','GET'])
@is_logged_in
def plot_store_establish():
    import matplotlib
    matplotlib.use('Agg')
    import matplotlib.pyplot as plt
    df=pd.read_csv(UPLOAD_FOLDER+'/train.csv')
    plt.figure()
    plt.gcf().subplots_adjust(left=0.3)
    df.groupby(['Outlet_Establishment_Year'])['Item_Outlet_Sales'].sum().plot(kind='barh')
    plt.savefig('static/plot/plot_store_establish.png')
    return render_template("plot_store_establish.html")
@app.route("/build_model",methods=['POST','GET'])
@is_logged_in
def build_model():
    return render_template("build_model.html")
@app.route("/ajax_build_model",methods=['POST','GET'])
def ajax_build_model():
    x=main.create_mode()
    return "Accuracy Score is "+str(x)
#Linear Regression
@app.route("/predict_lr",methods=['POST','GET'])
@is_logged_in
def predict_lr():
    score=round(LR.predict()*100,2)
    records=[]
    heading=[]
    import csv
    if os.path.exists(PREDICT_FOLDER+'/predict_lr.csv'):
        with open(PREDICT_FOLDER+'/predict_lr.csv',encoding="utf8") as csv_file:
            csv_reader = csv.reader(csv_file, delimiter=',')
            heading=[]
            for row in csv_reader:
                heading.append(row)
                break
            i=0
            for row in csv_reader:
                if i>0:
                    records.append(row)
                i+=1
    return render_template("predict_lr.html",data=records,heading=heading,score=score)
#Decision tree
@app.route("/predict_dt",methods=['POST','GET'])
def predict_dt():
    score=round(DT.predict()*100,2)
    records=[]
    heading=[]
    import csv
    if os.path.exists(PREDICT_FOLDER+'/predict_dt.csv'):
        with open(PREDICT_FOLDER+'/predict_dt.csv',encoding="utf8") as csv_file:
            csv_reader = csv.reader(csv_file, delimiter=',')
            heading=[]
            for row in csv_reader:
                heading.append(row)
                break
            i=0
            for row in csv_reader:
                if i>0:
                    records.append(row)
                i+=1
    return render_template("predict_dt.html",data=records,heading=heading,score=score)


#Random Forest
@app.route("/predict_rf",methods=['POST','GET'])
@is_logged_in
def predict_rf():
    score=round(RF.predict()*100,2)
    records=[]
    heading=[]
    import csv
    if os.path.exists(PREDICT_FOLDER+'/predict_rf.csv'):
        with open(PREDICT_FOLDER+'/predict_rf.csv',encoding="utf8") as csv_file:
            csv_reader = csv.reader(csv_file, delimiter=',')
            heading=[]
            for row in csv_reader:
                heading.append(row)
                break
            i=0
            for row in csv_reader:
                if i>0:
                    records.append(row)
                i+=1
    return render_template("predict_rf.html",data=records,heading=heading,score=score)

#XGBoost
@app.route("/predict_xg",methods=['POST','GET'])
@is_logged_in
def predict_xg():
    score=round(XG.predict()*100,2)
    records=[]
    heading=[]
    import csv
    if os.path.exists(PREDICT_FOLDER+'/predict_xg.csv'):
        with open(PREDICT_FOLDER+'/predict_xg.csv',encoding="utf8") as csv_file:
            csv_reader = csv.reader(csv_file, delimiter=',')
            heading=[]
            for row in csv_reader:
                heading.append(row)
                break
            i=0
            for row in csv_reader:
                if i>0:
                    records.append(row)
                i+=1
    return render_template("predict_xg.html",data=records,heading=heading,score=score)
#logout
@app.route("/logout")
def logout():
    session.clear()
    flash('You are now logged out','success')
    return redirect(url_for('index'))

if __name__=='__main__':
    app.secret_key='secret123
    app.run(debug=True)
```

OUTPUT



Data Poison Detection Prediction are of paramount importance in retail. Without this tool, companies encounter disruption of the inventory balance, through ordering too much or not enough products for a certain period of time. In the case of surplus, a company is forced to offer discounts to sell products. Otherwise, it

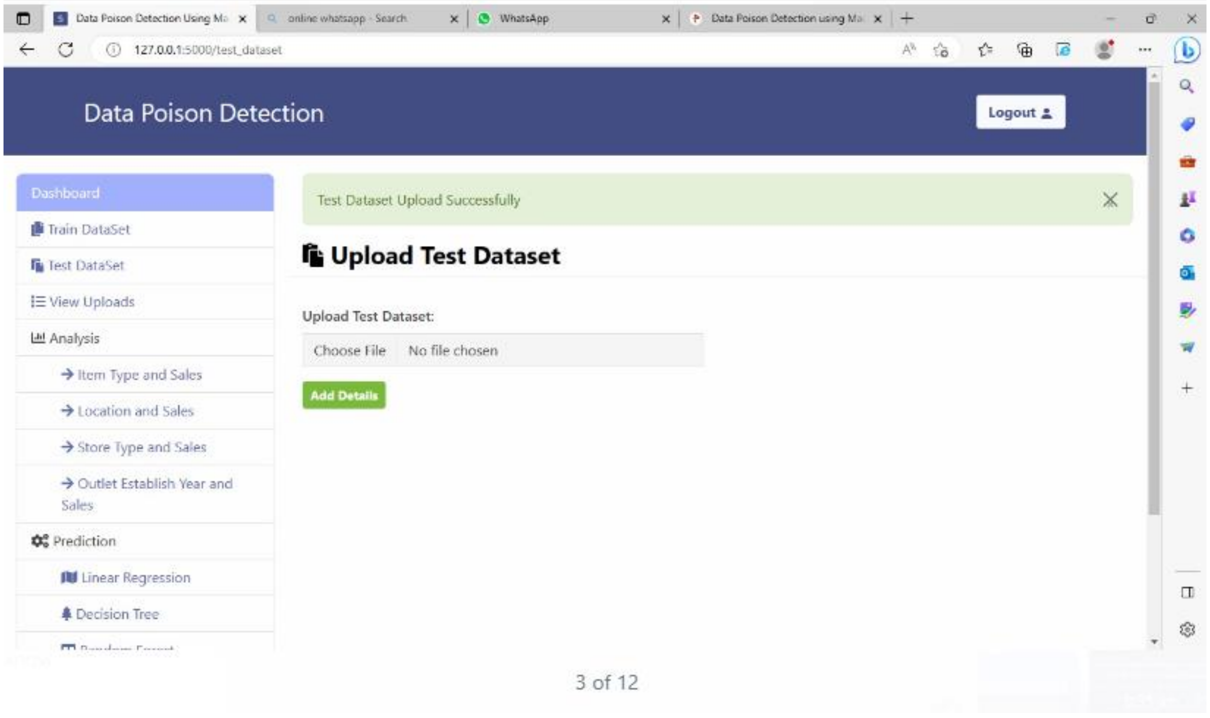


Login Successfully

Upload Training Dataset

Upload Training Dataset:

2 of 12

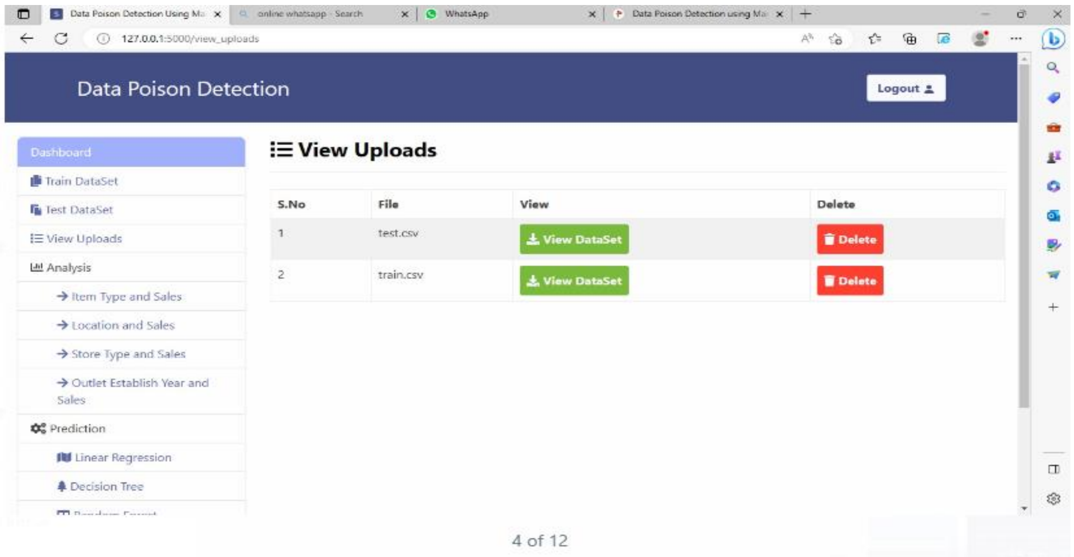


Test Dataset Upload Successfully

Upload Test Dataset

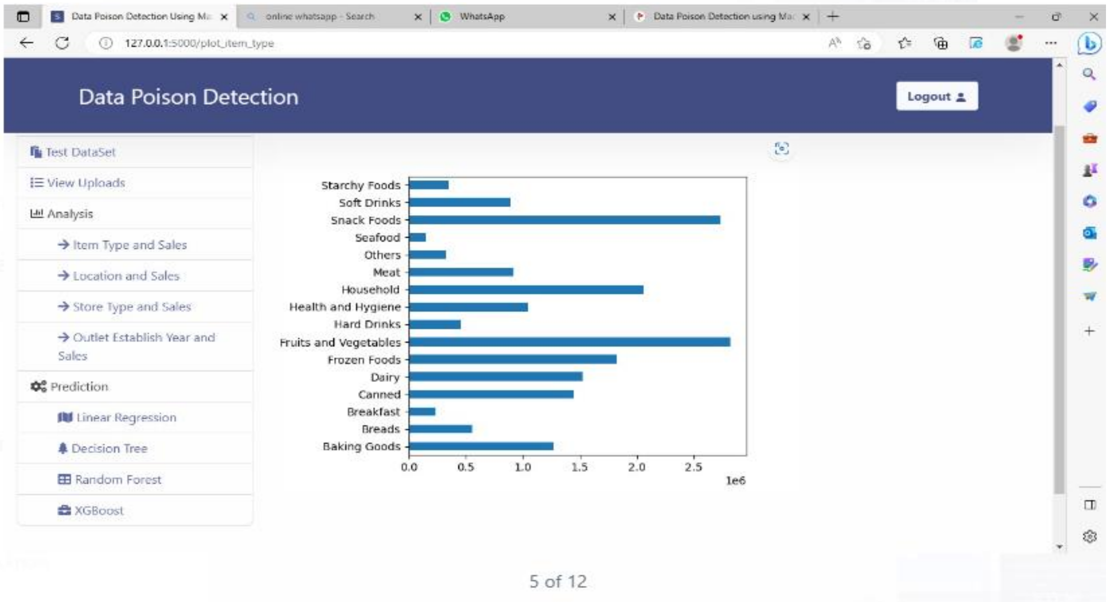
Upload Test Dataset:

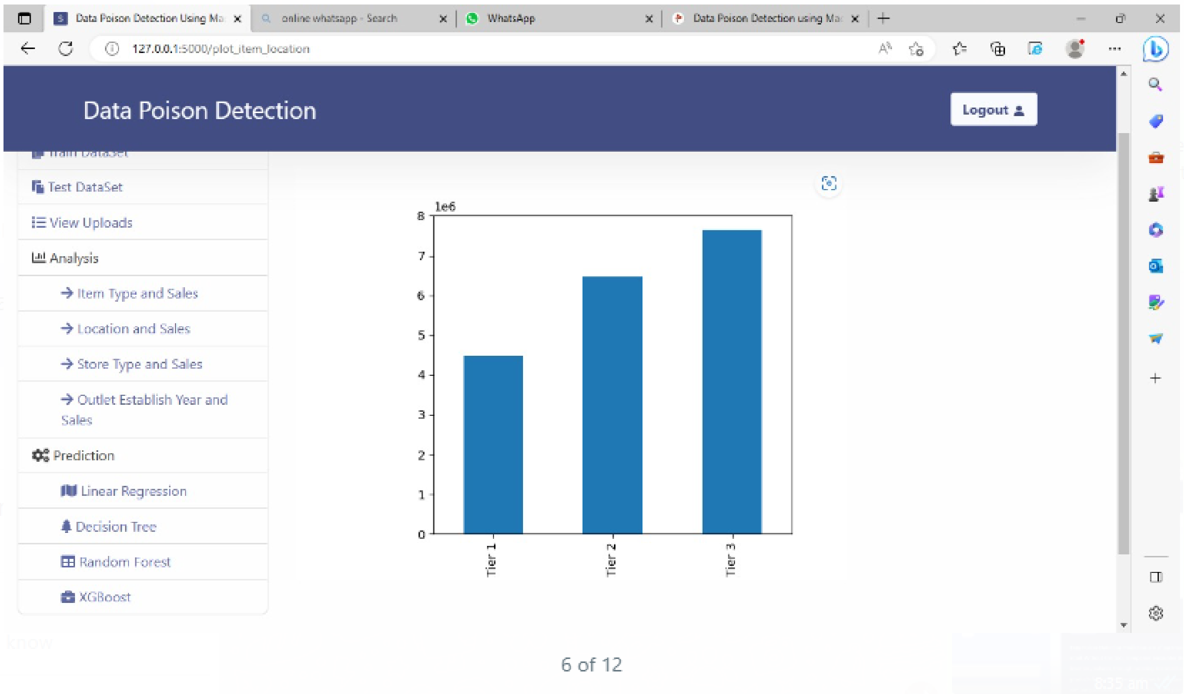
3 of 12



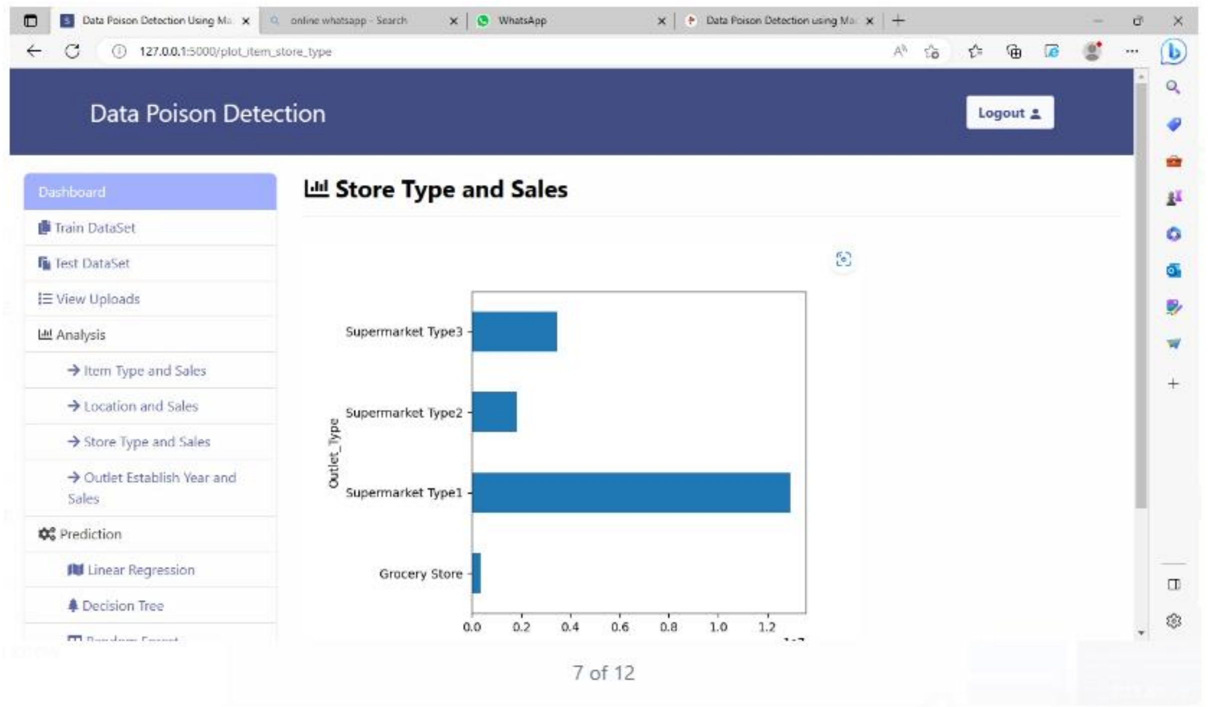
View Uploads

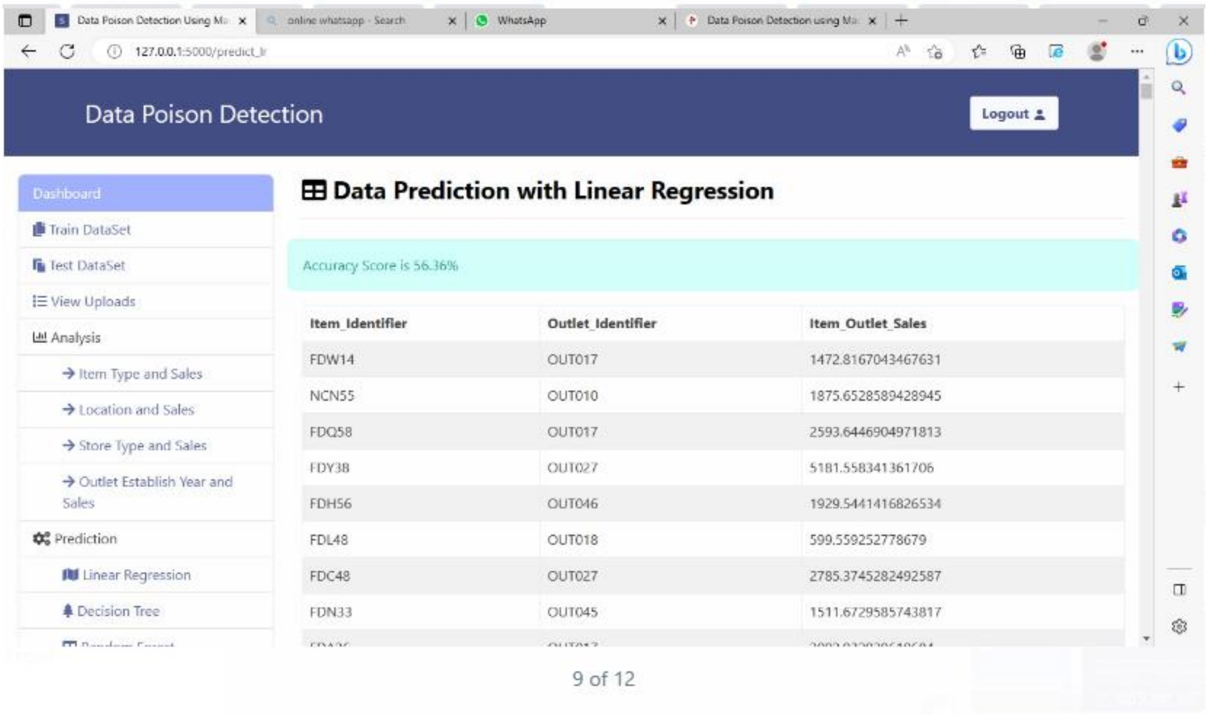
S.No	File	View	Delete
1	test.csv	View DataSet	Delete
2	train.csv	View DataSet	Delete





International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 2 Volume No.7 March - April – 2023
DOI:10.46647/ijetms.2023.v07i02.066 ISSN: 2581-4621





Data Prediction with Linear Regression

Accuracy Score is 56.36%

Item_Identifier	Outlet_Identifier	Item_Outlet_Sales
FDW14	OUT017	1472.8167043467631
NCN55	OUT010	1875.6528589428945
FDQ58	OUT017	2593.6446904971813
FDY38	OUT027	5181.558341361706
FDH56	OUT046	1929.5441416826534
FDL48	OUT018	599.559252778679
FDC48	OUT027	2785.3745282492587
FDN33	OUT045	1511.6729585743817

CONCLUSION

In this paper, we discussed the data poison detection schemes in both basic-DML and semi-DML scenarios. The data poison detection scheme in the basic-DML scenario utilizes a threshold of parameters to find out the poisoned sub-datasets. Moreover, we established a mathematical model to analyze the probability of finding threats with different numbers of training loops. Furthermore, we presented an improved data poison detection scheme and the optimal resource allocation in the semi-DML scenario. Simulation results show that in the basic-DML scenario, the proposed scheme can increase the model accuracy by up to 20% for support vector machine and 60% for logistic regression, respectively. As to the semi-DML scenario, the improved data poison detection scheme with optimal resource allocation can decrease wasted resources for 20-100% compared to the other two schemes without the optimal resource allocation. In the future, the data poison detection scheme can be extended to a more dynamic pattern to fit the changing application environment and attacking intensity. Besides, since the multi-training of sub-datasets would increase the resource consumption of the system, the trade-off between security and resource cost is another topic that needs to be studied further.

References

[1] Jason Brownlee, “What is Deep Learning?” August 16, 2019, https: //machinelearningmastery.com/what-is-deep-learning/
[2] Mathworks, “What Is Deep Learning?” https://www.mathworks.com/di scovery/deep-learning.html
[3] Computer Science, University of Maryland, “Poison Frogs! Targeted Poisoning Attacks on Neural Networks,” https://www.cs.umd.edu/∼tom g/projects/poison/
[4] Keith D. Foote, “A Brief History of Deep Learning,” Feburary 7, 2017, https://www.dataversity.net/brief-history-deep-learning/
[5] Reportlinker, “Global Deeping Learning Industry,” July 2020, https: //www.reportlinker.com/p05798338/Global-Deep-Learning-Industry.h tml?utm source=GNW
[6] Larry Hardesty, “Explained: Neural networks,” April 14, 2017, https: //news.mit.edu/2017/explained-neural-networks-deep-learning-0414
[7] Jeff Dean, “Large-Scale Deep Learning for Intelligent Computer Systems,” https://static.googleusercontent.com/media/research.google.com /en//people/jeff/BayLearn2015.pdf
[8] DeepAI, “Feature Extraction,” https://deepai.org/machine-learning-glos sary-and-terms/feature-extraction
[9] Artem Oppermann, “Artificial Intelligence vs. Machine Learning vs. Deep Learning,” October 29, 2019, https://towardsdatascience.com/artif icial-intelligence-vs-machine-learning-vs-deep-learning-2210ba8cc4ac
[10] Alexander Polyakov, “How to attack Machine Learning (Evasion, Poisoning, Inference, Trojans, Backdoors),” August 6, 2019, https://toward sdatascience.com/how-to-attack-machine-learning-evasion-poisoninginference-trojans-backdoors-a7cb5832595c
[11] Ilja Moisejevs, “Poisoning attacks on Machine Learning,” July 14, 2019, https://towardsdatascience.com/poisoning-attacks-on-machine-learning -1ff247c254db
[12] Daniel Lowd, Christopher Meek, “Good Word Attacks on Statistical Spam Filters,” Semantic Scholar, https://www.semanticscholar.org/pape r/Good-Word-Attacks-on-Statistical-Spam-Filters-Lowd-Meek/16358a 75a3a6561d042e6874d128d82f5b0bd4b3
[13] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, Tom Goldstein, “Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks,” in Proceedings of 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montreal, Canad, https://papers.nips.cc/paper/7849-po ´ ison-frogs-targeted-clean-label-poisoning-attacks-on-neural-networks.p df
[14] Luis Munoz-Gonz ˜ alez, Battista Biggio, Ambra Demontis, Andrea Pau- ´ dice, Vasin Wongrassamee, Emil C. Lupu, Fabio Roli, “Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization,” August 29, 2017, https://arxiv.org/abs/1708.08689
[15] Keras, https://keras.io/
[16] TensorFlow, https://www.tensorflow.org/