**SAMPLE CODE:**

**User side views:**

from django.shortcuts import render, HttpResponse

from .forms import UserRegistrationForm

from django.contrib import messages

from .models import UserRegistrationModel

# Create your views here.

def UserRegisterActions(request):

if request.method == 'POST':

form = UserRegistrationForm(request.POST)

if form.is\_valid():

print('Data is Valid')

form.save()

messages.success(request, 'You have been successfully registered')

form = UserRegistrationForm()

return render(request, 'UserRegistrations.html', {'form': form})

else:

messages.success(request, 'Email or Mobile Already Existed')

print("Invalid form")

else:

form = UserRegistrationForm()

return render(request, 'UserRegistrations.html', {'form': form})

def UserLoginCheck(request):

if request.method == "POST":

loginid = request.POST.get('loginname')

pswd = request.POST.get('pswd')

print("Login ID = ", loginid, ' Password = ', pswd)

try:

check = UserRegistrationModel.objects.get(loginid=loginid, password=pswd)

status = check.status

print('Status is = ', status)

if status == "activated":

request.session['id'] = check.id

request.session['loggeduser'] = check.name

request.session['loginid'] = loginid

request.session['email'] = check.email

print("User id At", check.id, status)

return render(request, 'users/UserHome.html', {})

else:

messages.success(request, 'Your Account has not been activated by Admin.')

return render(request, 'UserLogin.html')

except Exception as e:

print('Exception is ', str(e))

pass

messages.success(request, 'Invalid Login id and password')

return render(request, 'UserLogin.html', {})

def UserHome(request):

return render(request, 'users/UserHome.html', {})

def TrainModel(request):

import os

import tensorflow as tf

import pandas as pd

import numpy as np

from django.conf import settings

from matplotlib import pyplot as plt

activity\_codes\_mapping = {'A': 'walking',

'B': 'jogging',

'C': 'stairs',

'D': 'sitting',

'E': 'standing',

'F': 'typing',

'G': 'brushing teeth',

'H': 'eating soup',

'I': 'eating chips',

'J': 'eating pasta',

'K': 'drinking from cup',

'L': 'eating sandwich',

'M': 'kicking soccer ball',

'O': 'playing catch tennis ball',

'P': 'dribbling basket ball',

'Q': 'writing',

'R': 'clapping',

'S': 'folding clothes'}

activity\_color\_map = {activity\_codes\_mapping['A']: 'lime',

activity\_codes\_mapping['B']: 'red',

activity\_codes\_mapping['C']: 'blue',

activity\_codes\_mapping['D']: 'orange',

activity\_codes\_mapping['E']: 'yellow',

activity\_codes\_mapping['F']: 'lightgreen',

activity\_codes\_mapping['G']: 'greenyellow',

activity\_codes\_mapping['H']: 'magenta',

activity\_codes\_mapping['I']: 'gold',

activity\_codes\_mapping['J']: 'cyan',

activity\_codes\_mapping['K']: 'purple',

activity\_codes\_mapping['L']: 'lightgreen',

activity\_codes\_mapping['M']: 'violet',

activity\_codes\_mapping['O']: 'limegreen',

activity\_codes\_mapping['P']: 'deepskyblue',

activity\_codes\_mapping['Q']: 'mediumspringgreen',

activity\_codes\_mapping['R']: 'plum',

activity\_codes\_mapping['S']: 'olive'}

def show\_accel\_per\_activity(device, df, act, interval\_in\_sec = None):

''' Plots acceleration time history per activity '''

df1 = df.loc[df.activity == act].copy()

df1.reset\_index(drop = True, inplace = True)

df1['duration'] = (df1['timestamp'] - df1['timestamp'].iloc[0])/1000000000 # nanoseconds --> seconds

if interval\_in\_sec == None:

ax = df1[:].plot(kind='line', x='duration', y=['x','y','z'], figsize=(25,7), grid = True) # ,title = act)

else:

ax = df1[:interval\_in\_sec\*20].plot(kind='line', x='duration', y=['x','y','z'], figsize=(25,7), grid = True) # ,title = act)

ax.set\_xlabel('duration (sec)', fontsize = 15)

ax.set\_ylabel('acceleration (m/sec^2)',fontsize = 15)

ax.set\_title('Acceleration: Device: ' + device + ' Activity: ' + act, fontsize = 15)

# plt.show()

def show\_ang\_velocity\_per\_activity(device, df, act, interval\_in\_sec = None):

''' Plots angular volocity time history per activity '''

df1 = df.loc[df.activity == act].copy()

df1.reset\_index(drop = True, inplace = True)

df1['duration'] = (df1['timestamp'] - df1['timestamp'].iloc[0])/1000000000 # nanoseconds --> seconds

if interval\_in\_sec == None:

ax = df1[:].plot(kind='line', x='duration', y=['x','y','z'], figsize=(25,7), grid = True) # ,title = act)

else:

ax = df1[:interval\_in\_sec\*20].plot(kind='line', x='duration', y=['x','y','z'], figsize=(25,7), grid = True) # ,title = act)

ax.set\_xlabel('duration (sec)', fontsize = 15)

ax.set\_ylabel('angular velocity (rad/sec)',fontsize = 15)

ax.set\_title('Angular velocity: Device: ' + device + ' Activity: ' + act, fontsize = 15)

datasetpath = os.path.join(settings.MEDIA\_ROOT,'wisdm-dataset')

#accel\_phone

raw\_par\_10\_phone\_accel = pd.read\_csv(datasetpath + '/' + 'raw/phone/accel/data\_1610\_accel\_phone.txt', names = ['participant\_id' , 'activity\_code' , 'timestamp', 'x', 'y', 'z'], index\_col=None, header=None)

print('-'\*100)

print(raw\_par\_10\_phone\_accel)

raw\_par\_10\_phone\_accel.z = raw\_par\_10\_phone\_accel.z.str.strip(';')

raw\_par\_10\_phone\_accel.z = pd.to\_numeric(raw\_par\_10\_phone\_accel.z)

raw\_par\_10\_phone\_accel['activity'] = raw\_par\_10\_phone\_accel['activity\_code'].map(activity\_codes\_mapping)

raw\_par\_10\_phone\_accel = raw\_par\_10\_phone\_accel[['participant\_id', 'activity\_code', 'activity', 'timestamp', 'x', 'y', 'z']]

print(raw\_par\_10\_phone\_accel)

for key in activity\_codes\_mapping:

show\_accel\_per\_activity('Phone', raw\_par\_10\_phone\_accel, activity\_codes\_mapping[key], 10)

#accel\_watch

raw\_par\_20\_watch\_accel = pd.read\_csv(datasetpath + '/' + 'raw/watch/accel/data\_1620\_accel\_watch.txt', names = ['participant\_id' , 'activity\_code' , 'timestamp', 'x', 'y', 'z'], index\_col=None, header=None)

raw\_par\_20\_watch\_accel.z = raw\_par\_20\_watch\_accel.z.str.strip(';')

raw\_par\_20\_watch\_accel.z = pd.to\_numeric(raw\_par\_20\_watch\_accel.z)

raw\_par\_20\_watch\_accel['activity'] = raw\_par\_20\_watch\_accel['activity\_code'].map(activity\_codes\_mapping)

raw\_par\_20\_watch\_accel = raw\_par\_20\_watch\_accel[['participant\_id', 'activity\_code', 'activity', 'timestamp', 'x', 'y', 'z']]

print(raw\_par\_20\_watch\_accel)

for key in activity\_codes\_mapping:

show\_accel\_per\_activity('Watch', raw\_par\_20\_watch\_accel, activity\_codes\_mapping[key], 50)

#gyro\_phone

raw\_par\_35\_phone\_ang\_vel = pd.read\_csv(datasetpath + '/' + 'raw/phone/gyro/data\_1635\_gyro\_phone.txt', names = ['participant\_id' , 'activity\_code' , 'timestamp', 'x', 'y', 'z'], index\_col=None, header=None)

raw\_par\_35\_phone\_ang\_vel.z = raw\_par\_35\_phone\_ang\_vel.z.str.strip(';')

raw\_par\_35\_phone\_ang\_vel.z = pd.to\_numeric(raw\_par\_35\_phone\_ang\_vel.z)

raw\_par\_35\_phone\_ang\_vel['activity'] = raw\_par\_35\_phone\_ang\_vel['activity\_code'].map(activity\_codes\_mapping)

raw\_par\_35\_phone\_ang\_vel = raw\_par\_35\_phone\_ang\_vel[['participant\_id', 'activity\_code', 'activity', 'timestamp', 'x', 'y', 'z']]

print(raw\_par\_35\_phone\_ang\_vel)

for key in activity\_codes\_mapping:

show\_ang\_velocity\_per\_activity('Phone', raw\_par\_35\_phone\_ang\_vel, activity\_codes\_mapping[key])

#gyro\_watch

raw\_par\_45\_watch\_ang\_vel = pd.read\_csv(datasetpath + '/' + 'raw/watch/gyro/data\_1635\_gyro\_watch.txt', names = ['participant\_id' , 'activity\_code' , 'timestamp', 'x', 'y', 'z'], index\_col=None, header=None)

raw\_par\_45\_watch\_ang\_vel.z = raw\_par\_45\_watch\_ang\_vel.z.str.strip(';')

raw\_par\_45\_watch\_ang\_vel.z = pd.to\_numeric(raw\_par\_45\_watch\_ang\_vel.z)

raw\_par\_45\_watch\_ang\_vel['activity'] = raw\_par\_45\_watch\_ang\_vel['activity\_code'].map(activity\_codes\_mapping)

raw\_par\_45\_watch\_ang\_vel = raw\_par\_45\_watch\_ang\_vel[['participant\_id', 'activity\_code', 'activity', 'timestamp', 'x', 'y', 'z']]

print(raw\_par\_45\_watch\_ang\_vel)

for key in activity\_codes\_mapping:

show\_ang\_velocity\_per\_activity('Watch', raw\_par\_45\_watch\_ang\_vel, activity\_codes\_mapping[key])

features = ['ACTIVITY',

'X0', # 1st bin fraction of x axis acceleration distribution

'X1', # 2nd bin fraction ...

'X2',

'X3',

'X4',

'X5',

'X6',

'X7',

'X8',

'X9',

'Y0', # 1st bin fraction of y axis acceleration distribution

'Y1', # 2nd bin fraction ...

'Y2',

'Y3',

'Y4',

'Y5',

'Y6',

'Y7',

'Y8',

'Y9',

'Z0', # 1st bin fraction of z axis acceleration distribution

'Z1', # 2nd bin fraction ...

'Z2',

'Z3',

'Z4',

'Z5',

'Z6',

'Z7',

'Z8',

'Z9',

'XAVG', # average sensor value over the window (per axis)

'YAVG',

'ZAVG',

'XPEAK', # Time in milliseconds between the peaks in the wave associated with most activities. heuristically determined (per axis)

'YPEAK',

'ZPEAK',

'XABSOLDEV', # Average absolute difference between the each of the 200 readings and the mean of those values (per axis)

'YABSOLDEV',

'ZABSOLDEV',

'XSTANDDEV', # Standard deviation of the 200 window's values (per axis) \*\*\*BUG!\*\*\*

'YSTANDDEV',

'ZSTANDDEV',

'XVAR', # Variance of the 200 window's values (per axis) \*\*\*BUG!\*\*\*

'YVAR',

'ZVAR',

'XMFCC0', # short-term power spectrum of a wave, based on a linear cosine transform of a log power spectrum on a non-linear mel scale of frequency (13 values per axis)

'XMFCC1',

'XMFCC2',

'XMFCC3',

'XMFCC4',

'XMFCC5',

'XMFCC6',

'XMFCC7',

'XMFCC8',

'XMFCC9',

'XMFCC10',

'XMFCC11',

'XMFCC12',

'YMFCC0', # short-term power spectrum of a wave, based on a linear cosine transform of a log power spectrum on a non-linear mel scale of frequency (13 values per axis)

'YMFCC1',

'YMFCC2',

'YMFCC3',

'YMFCC4',

'YMFCC5',

'YMFCC6',

'YMFCC7',

'YMFCC8',

'YMFCC9',

'YMFCC10',

'YMFCC11',

'YMFCC12',

'ZMFCC0', # short-term power spectrum of a wave, based on a linear cosine transform of a log power spectrum on a non-linear mel scale of frequency (13 values per axis)

'ZMFCC1',

'ZMFCC2',

'ZMFCC3',

'ZMFCC4',

'ZMFCC5',

'ZMFCC6',

'ZMFCC7',

'ZMFCC8',

'ZMFCC9',

'ZMFCC10',

'ZMFCC11',

'ZMFCC12',

'XYCOS', # The cosine distances between sensor values for pairs of axes (three pairs of axes)

'XZCOS',

'YZCOS',

'XYCOR', # The correlation between sensor values for pairs of axes (three pairs of axes)

'XZCOR',

'YZCOR',

'RESULTANT', # Average resultant value, computed by squaring each matching x, y, and z value, summing them, taking the square root, and then averaging these values over the 200 readings

'PARTICIPANT'] # Categirical: 1600 -1650

import glob

#the duplicate files to be ignored; all identical to 1600

duplicate\_files = [str(i) for i in range(1611, 1618)] # '1611',...'1617'

# path = r'media/wisdm-dataset/arff\_files/phone/accel'

path = datasetpath + '/' + 'arff\_files/phone/accel'

all\_files = glob.glob(path + "/\*.arff")

list\_dfs\_phone\_accel = []

for filename in all\_files:

if any(dup\_fn in filename for dup\_fn in duplicate\_files):

continue #ignore the duplicate files

df = pd.read\_csv(filename, names = features, skiprows = 96, index\_col=None, header=0)

list\_dfs\_phone\_accel.append(df)

all\_phone\_accel = pd.concat(list\_dfs\_phone\_accel, axis=0, ignore\_index=True, sort=False)

print(all\_phone\_accel)

print(all\_phone\_accel.info())

all\_phone\_accel\_breakpoint = all\_phone\_accel.copy()

# all\_phone\_accel['ACTIVITY'].map(activity\_codes\_mapping).value\_counts()

# \_ = all\_phone\_accel['ACTIVITY'].map(activity\_codes\_mapping).value\_counts().plot(kind = 'bar', figsize = (15,5), color = 'purple', title = 'row count per activity', legend = True, fontsize = 15)

all\_phone\_accel.drop('PARTICIPANT', axis = 1, inplace = True)

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

y = all\_phone\_accel.ACTIVITY

X = all\_phone\_accel.drop('ACTIVITY', axis=1)

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

train\_size = 0.75,

test\_size = 0.25,

shuffle = True,

stratify = all\_phone\_accel.ACTIVITY)

print('-----X\_train-------')

print(X\_train)

print('-----Y Train-------')

print(y\_train)

X\_train.insert(0, 'Y', y\_train)

print('-----X\_train-------')

print(X\_train)

y\_train = X\_train['Y']

print('-----Y Train-------')

print(y\_train)

X\_train.drop(['Y'], axis = 1, inplace = True)

from sklearn.preprocessing import MaxAbsScaler

scaling\_transformer = MaxAbsScaler().fit(X\_train[['XAVG', 'YAVG', 'ZAVG', 'XPEAK', 'YPEAK', 'ZPEAK', 'XABSOLDEV', 'YABSOLDEV', 'ZABSOLDEV', 'RESULTANT']])

X\_train[['XAVG', 'YAVG', 'ZAVG', 'XPEAK', 'YPEAK', 'ZPEAK', 'XABSOLDEV', 'YABSOLDEV', 'ZABSOLDEV', 'RESULTANT']] = scaling\_transformer.transform(X\_train[['XAVG', 'YAVG', 'ZAVG', 'XPEAK', 'YPEAK', 'ZPEAK', 'XABSOLDEV', 'YABSOLDEV', 'ZABSOLDEV', 'RESULTANT']])

X\_test = X\_test.copy()

X\_test[['XAVG', 'YAVG', 'ZAVG', 'XPEAK', 'YPEAK', 'ZPEAK', 'XABSOLDEV', 'YABSOLDEV', 'ZABSOLDEV', 'RESULTANT']] = scaling\_transformer.transform(X\_test[['XAVG', 'YAVG', 'ZAVG', 'XPEAK', 'YPEAK', 'ZPEAK', 'XABSOLDEV', 'YABSOLDEV', 'ZABSOLDEV', 'RESULTANT']])

print('-------X\_test-------')

print(X\_test)

X\_train.reset\_index(drop = True, inplace = True)

print('-------X\_train--------')

print(X\_train)

X\_test.reset\_index(drop = True, inplace = True)

print('-------X\_test-------')

print(X\_test)

print('len - ',len(X\_test))

print('type - ',type(X\_test))

X\_test.to\_csv('TestDataFrame.csv')

y\_train.reset\_index(drop = True, inplace = True)

print('-----Y Train-------')

print(y\_train)

y\_test.reset\_index(drop = True, inplace = True)

print('-----Y test----------')

print(y\_test)

import pandas as pd

import matplotlib.pyplot as plt

import os

from sklearn.tree import DecisionTreeClassifier

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

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

from sklearn.metrics import classification\_report

from sklearn.tree import DecisionTreeClassifier

import pickle

from sklearn.model\_selection import StratifiedShuffleSplit

\_ = y\_train.value\_counts(sort = False).plot(kind = 'bar', figsize = (15,5), color = 'red', title = 'row count per activity', legend = True, fontsize = 15)

# plt.show()

my\_cv = StratifiedShuffleSplit(n\_splits=5, train\_size=0.7, test\_size=0.3)

dt\_classifier = DecisionTreeClassifier()

my\_param\_grid = {'min\_samples\_leaf': [6, 10, 20, 40],

'min\_weight\_fraction\_leaf': [0.01, 0.02, 0.05],

'criterion': ['entropy'],

'min\_impurity\_decrease': [1e-2, 7e-3]}

dt\_model\_gs = GridSearchCV(estimator=dt\_classifier,

param\_grid=my\_param\_grid,

cv=my\_cv,

scoring='accuracy',

verbose = 0,

return\_train\_score = True)

dt\_model\_gs.fit(X\_train, y\_train)

print('-------Fit Done---------')

print(dt\_model\_gs.best\_params\_)

dt\_best\_classifier = dt\_model\_gs.best\_estimator\_

pickle.dump(dt\_best\_classifier, open('Ajmodel.pkl', 'wb'))

print('-------Pickling Model Dumped------')

y\_test\_pred = dt\_best\_classifier.predict(X\_test)

classification\_report = classification\_report(y\_true=y\_test,y\_pred=y\_test\_pred,output\_dict=True)

classification\_report = pd.DataFrame(classification\_report).transpose().to\_html()

return render(request, 'users/TrainModel.html', {'classification\_report':classification\_report})

def Predict(request):

if request.method == 'POST':

activity\_codes\_mapping = {'A': 'walking',

'B': 'jogging',

'C': 'stairs',

'D': 'sitting',

'E': 'standing',

'F': 'typing',

'G': 'brushing teeth',

'H': 'eating soup',

'I': 'eating chips',

'J': 'eating pasta',

'K': 'drinking from cup',

'L': 'eating sandwich',

'M': 'kicking soccer ball',

'O': 'playing catch tennis ball',

'P': 'dribbling basket ball',

'Q': 'writing',

'R': 'clapping',

'S': 'folding clothes'}

import os

from django.conf import settings

import pickle

import pandas as pd

index\_no = request.POST.get('index\_no')

print(index\_no)

print('type ----> ',type(index\_no))

modelPath = os.path.join(settings.MEDIA\_ROOT,'Ajmodel.pkl')

testDataPath = os.path.join(settings.MEDIA\_ROOT,'TestDataFrame.csv')

pickled\_model = pickle.load(open(modelPath, 'rb'))

testData = pd.read\_csv(testDataPath)

pred\_val = testData.iloc[int(index\_no)]

print(pred\_val)

pred\_result = pickled\_model.predict([pred\_val])

SampleTestData = testData.head(100).to\_html

print('type of pred\_result --> ', type(pred\_result))

print(pred\_result)

print('type ---> ',type(activity\_codes\_mapping))

activity = activity\_codes\_mapping.get(pred\_result[0])

return render(request, 'users/prediction.html', {'testData':SampleTestData(index=False),'activity':activity})

else:

return render(request, 'users/prediction.html', {})

**Base.html:**

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta content="width=device-width, initial-scale=1.0" name="viewport">

<title>offensive langauge</title>

<meta content="" name="description">

<meta content="" name="keywords">

<!-- Favicons -->

<link href="{% static 'img/favicon.png'%}" rel="icon">

<link href="{% static 'img/apple-touch-icon.png'%}" rel="apple-touch-icon">

<!-- Google Fonts -->

<link href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Muli:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300,300i,400,400i,500,500i,600,600i,700,700i" rel="stylesheet">

<!-- Vendor CSS Files -->

<link href="{% static 'vendor/animate.css/animate.min.css'%}" rel="stylesheet">

<link href="{% static 'vendor/aos/aos.css'%}" rel="stylesheet">

<link href="{% static 'vendor/bootstrap/css/bootstrap.min.css'%}" rel="stylesheet">

<link href="{% static 'vendor/bootstrap-icons/bootstrap-icons.css'%}" rel="stylesheet">

<link href="{% static 'vendor/boxicons/css/boxicons.min.css'%}" rel="stylesheet">

<link href="{% static 'vendor/glightbox/css/glightbox.min.css'%}" rel="stylesheet">

<link href="{% static 'vendor/swiper/swiper-bundle.min.css'%}" rel="stylesheet">

<!-- Template Main CSS File -->

<link href="{% static 'css/style.css'%}" rel="stylesheet">

<!-- =======================================================

\* Template Name: Flattern

\* Updated: May 30 2023 with Bootstrap v5.3.0

\* Template URL: https://bootstrapmade.com/flattern-multipurpose-bootstrap-template/

\* Author: BootstrapMade.com

\* License: https://bootstrapmade.com/license/

======================================================== -->

</head>

<body>

<!-- ======= Header ======= -->

<header id="header" class="d-flex align-items-center">

<div class="container d-flex justify-content-between">

<div class="logo">

<h1 class="text-light"><a href="/index">OFFENSIVE LANGUAGE DETECTION</a></h1>

<!-- Uncomment below if you prefer to use an image logo -->

<!-- <a href="index.html"><img src="assets/img/logo.png" alt="" class="img-fluid"></a>-->

</div>

<nav id="navbar" class="navbar">

<ul>

<li><a href="{% url 'index'%}">Home</a></li>

<li><a href="{% url 'UserLogin' %}">User-Login</a></li>

<li><a href="{% url 'AdminLogin' %}">Admin-Login</a></li>

<li><a href="{% url 'UserRegister' %}">User-Register</a></li>

</ul>

<i class="bi bi-list mobile-nav-toggle"></i>

</nav><!-- .navbar -->

</div>

</header><!-- End Header -->

<!-- ======= Hero Section ======= -->

<section id="hero">

<div id="heroCarousel" data-bs-interval="5000" class="carousel slide carousel-fade" data-bs-ride="carousel">

<div class="carousel-inner" role="listbox">

<!-- Slide 1 -->

<div class="carousel-item active" style="background-image: url('../static/img/slide/slide-1.jpg');">

<div class="carousel-container">

<div class="carousel-content animate\_\_animated animate\_\_fadeInUp">

<h2>Offensive Language Detection on Social<span>Media Based on Text Classification</span></h2>

<p>Detecting offensive language can be a challenging task due to the evolving nature of language and the context-dependent nature of offensiveness. However, there are some common approaches and techniques that can be used to identify offensive language. Here are a few methods.</p>

<div class="text-center"><a href="" class="btn-get-started">Read More</a></div>

</div>

</div>

</div>

<!-- Slide 2 -->

<div class="carousel-item" style="background-image: url('../static/img/slide/slide-2.jpg');">

<div class="carousel-container">

<div class="carousel-content animate\_\_animated animate\_\_fadeInUp">

<h2>Offensive language</h2>

<p>It's important to note that offensive language detection is a complex task, and no single method can be fully effective due to the nuances and context-dependency of offensive language. Additionally, the definition of offensive language may vary across cultures and contexts, making it essential to consider these factors when designing an offensive language detection system.</p>

<div class="text-center"><a href="" class="btn-get-started">Read More</a></div>

</div>

</div>

</div>

<!-- Slide 3 -->

<div class="carousel-item" style="background-image: url('../static/img/slide/slide-3.jpg');">

<div class="carousel-container">

<div class="carousel-content animate\_\_animated animate\_\_fadeInUp">

<h2>Machine learning approach: </h2>

<p>Train a machine learning model using labeled data to classify text as offensive or non-offensive. This typically involves extracting features from the text, such as n-grams, word embeddings, or syntactic features, and using algorithms like Guassian Naive Bayes,Decision Tree(DT),Support Vector Machines (SVM),Random Forest (RF), Logistic Regresiiopn(LR),Multi Layer Preceptron (MLP), Gradient Bossting(GB),Ada Boost.</p>

<div class="text-center"><a href="" class="btn-get-started">Read More</a></div>

</div>

</div>

</div>

</div>

<a class="carousel-control-prev" href="#heroCarousel" role="button" data-bs-slide="prev">

<span class="carousel-control-prev-icon bx bx-left-arrow" aria-hidden="true"></span>

</a>

<a class="carousel-control-next" href="#heroCarousel" role="button" data-bs-slide="next">

<span class="carousel-control-next-icon bx bx-right-arrow" aria-hidden="true"></span>

</a>

<ol class="carousel-indicators" id="hero-carousel-indicators"></ol>

</div>

</section><!-- End Hero -->

{% block contents %}

{% endblock %}

<main id="main">

<!-- ======= Services Section ======= -->

<section id="services" class="services">

<div class="container">

<div class="row">

<div class="col-lg-4 col-md-6">

<div class="icon-box" data-aos="fade-up">

<div class="icon"><i class="bi bi-briefcase"></i></div>

<h4 class="title"><a href="">Machine Learning</a></h4>

<p class="description">Machine Learning is a program that analyses data and learns to predict the outcome.Machine Learning is making the computer learn from studying data and statistics.</p>

</div>

</div>

<div class="col-lg-4 col-md-6">

<div class="icon-box" data-aos="fade-up" data-aos-delay="100">

<div class="icon"><i class="bi bi-card-checklist"></i></div>

<h4 class="title"><a href="">Data preprocessing</a></h4>

<p class="description">It is a crucial step in data analysis and machine learning tasks. It involves preparing the raw data to make it suitable for further analysis or model training.</p>

</div>

</div>

<div class="col-lg-4 col-md-6">

<div class="icon-box" data-aos="fade-up" data-aos-delay="200">

<div class="icon"><i class="bi bi-bar-chart"></i></div>

<h4 class="title"><a href="">Handling Missing Values</a></h4>

<p class="description"> Identify and handle missing values in the dataset. This can involve techniques such as imputation (replacing missing values with estimated values) or deletion (removing rows or columns with missing values).</p>

</div>

</div>

<div class="col-lg-4 col-md-6">

<div class="icon-box" data-aos="fade-up" data-aos-delay="200">

<div class="icon"><i class="bi bi-binoculars"></i></div>

<h4 class="title"><a href="">Training</a></h4>

<p class="description">The training set is used to train the machine learning model. It typically consists of a large portion of the available data, usually around 70-80%.</p>

</div>

</div>

<div class="col-lg-4 col-md-6">

<div class="icon-box" data-aos="fade-up" data-aos-delay="300">

<div class="icon"><i class="bi bi-brightness-high"></i></div>

<h4 class="title"><a href="">Testing Set</a></h4>

<p class="description">

The testing set is used to evaluate the performance of the trained model. It should be independent of the training set and should not be used during the training process.</p>

</div>

</div>

<div class="col-lg-4 col-md-6">

<div class="icon-box" data-aos="fade-up" data-aos-delay="400">

<div class="icon"><i class="bi bi-calendar4-week"></i></div>

<h4 class="title"><a href="">After Traing and Testing</a></h4>

<p class="description">The predicted values are compared with the actual target values in the testing set to assess the model's accuracy, precision, recall, F1 score, or other performance metrics, depending on the specific task.</p>

</div>

</div>

</div>

</div>

</section><!-- End Services Section -->

</main><!-- End #main -->

<!-- ======= Footer ======= -->

<footer id="footer">

<div class="footer-top">

<div class="container">

<div class="row">

</div>

</div>

</div>

<div class="container d-md-flex py-4">

<div class="me-md-auto text-center text-md-start">

<div class="copyright">

&copy; Copyright <strong><span>Offensive Language Detection</span></strong>. All Rights Reserved

</div>

<div class="credits">

<!-- All the links in the footer should remain intact. -->

<!-- You can delete the links only if you purchased the pro version. -->

<!-- Licensing information: https://bootstrapmade.com/license/ -->

<!-- Purchase the pro version with working PHP/AJAX contact form: https://bootstrapmade.com/flattern-multipurpose-bootstrap-template/ -->

Designed by <a href="https://bootstrapmade.com/">Alex</a>

</div>

</div>

</div>

</footer><!-- End Footer -->

<a href="#" class="back-to-top d-flex align-items-center justify-content-center"><i class="bi bi-arrow-up-short"></i></a>

<!-- Vendor JS Files -->

<script src="{% static 'vendor/aos/aos.js' %}"></script>

<script src="{% static 'vendor/bootstrap/js/bootstrap.bundle.min.js' %}"></script>

<script src="{% static 'vendor/glightbox/js/glightbox.min.js' %}"></script>

<script src="{% static 'vendor/isotope-layout/isotope.pkgd.min.js' %}"></script>

<script src="{% static 'vendor/swiper/swiper-bundle.min.js' %}"></script>

<script src="{% static 'vendor/waypoints/noframework.waypoints.js' %}"></script>

<script src="{% static 'vendor/php-email-form/validate.js' %}"></script>

<!-- Template Main JS File -->

<script src="{% static 'js/main.js' %}"></script>

</body>

</html>

**Admin side views:**

from django.shortcuts import render, HttpResponse

from django.contrib import messages

from users.models import UserRegistrationModel

# Create your views here.

def AdminLoginCheck(request):

if request.method == 'POST':

usrid = request.POST.get('loginid')

pswd = request.POST.get('pswd')

print("User ID is = ", usrid)

if usrid == 'admin' and pswd == 'admin':

return render(request, 'admins/AdminHome.html')

else:

messages.success(request, 'Please Check Your Login Details')

return render(request, 'AdminLogin.html', {})

def AdminHome(request):

return render(request, 'admins/AdminHome.html')

def RegisterUsersView(request):

data = UserRegistrationModel.objects.all()

return render(request,'admins/viewregisterusers.html',{'data':data})

def ActivaUsers(request):

if request.method == 'GET':

id = request.GET.get('uid')

status = 'activated'

print("PID = ", id, status)

UserRegistrationModel.objects.filter(id=id).update(status=status)

data = UserRegistrationModel.objects.all()

return render(request,'admins/viewregisterusers.html',{'data':data})

def DeleteUsers(request):

if request.method == 'GET':

id = request.GET.get('uid')

status = 'activated'

print("PID = ", id, status)

UserRegistrationModel.objects.filter(id=id).delete()

data = UserRegistrationModel.objects.all()

return render(request,'admins/viewregisterusers.html',{'data':data})