**SAMPLE CODE**

**User side views.py**

from django.contrib import messages

from django.shortcuts import render, HttpResponse

from django.conf import settings

import os

from .forms import UserRegistrationForm

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 Not at activated')

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 user\_view\_dataset(request):

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

import pandas as pd

df = pd.read\_csv(path)

df = df.drop(['name', 'MDVP:Fo(Hz)','MDVP:Fhi(Hz)','MDVP:Flo(Hz)','MDVP:Jitter(%)','MDVP:Jitter(Abs)','MDVP:RAP','MDVP:PPQ','Jitter:DDP','spread1','spread2','MDVP:Shimmer'], axis=1)

df = df.to\_html

return render(request, 'users/view\_data.html', {'df': df})

def user\_model\_evaluations(request):

from .utility.parkinson\_utility import start\_models

result = start\_models()

print(result)

return render(request, 'users/model\_results.html',result)

def user\_predict\_form(request):

if request.method == 'POST':

nhr = float(request.POST.get('NHR'))

hnr = float(request.POST.get('HNR'))

rpde = float(request.POST.get('RPDE'))

dfa = float(request.POST.get('DFA'))

ppe = float(request.POST.get('PPE'))

test\_data = [nhr, hnr,rpde,dfa, ppe]

from .utility import process\_user\_input

test\_pred = process\_user\_input.get\_result(test\_data)

print("Test Result is:", test\_pred)

if test\_pred[0] == 0:

rslt = False

else:

rslt = True

return render(request, "users/parkinson\_form.html", {"test\_data": test\_data, "result": rslt})

else:

return render(request, 'users/parkinson\_form.html', {})

forms.py

from django import forms

from .models import UserRegistrationModel

class UserRegistrationForm(forms.ModelForm):

name = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[a-zA-Z]+'}), required=True, max\_length=100)

loginid = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[a-zA-Z]+'}), required=True, max\_length=100)

password = forms.CharField(widget=forms.PasswordInput(attrs={'pattern': '(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{8,}',

'title': 'Must contain at least one number and one uppercase and lowercase letter, and at least 8 or more characters'}),

required=True, max\_length=100)

mobile = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[56789][0-9]{9}'}), required=True,

max\_length=100)

email = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,}$'}),

required=True, max\_length=100)

locality = forms.CharField(widget=forms.TextInput(), required=True, max\_length=100)

address = forms.CharField(widget=forms.Textarea(attrs={'rows': 4, 'cols': 22}), required=True, max\_length=250)

city = forms.CharField(widget=forms.TextInput(

attrs={'autocomplete': 'off', 'pattern': '[A-Za-z ]+', 'title': 'Enter Characters Only '}), required=True,

max\_length=100)

state = forms.CharField(widget=forms.TextInput(

attrs={'autocomplete': 'off', 'pattern': '[A-Za-z ]+', 'title': 'Enter Characters Only '}), required=True,

max\_length=100)

status = forms.CharField(widget=forms.HiddenInput(), initial='waiting', max\_length=100)

class Meta():

model = UserRegistrationModel

fields = '\_\_all\_\_'

Model Building and results

import os

import pandas as pd

from django.conf import settings

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

a = pd.read\_csv(path)

a.head()

a.shape

a.dtypes

# print(a.head())

# sns.catplot(x='status', kind='count', data=a)

# for i in a:

# if i != 'status' and i != 'name':

# sns.catplot(x='status', y=i, kind='box', data=a)

b = a.drop(['name'], axis=1)

from sklearn.preprocessing import MinMaxScaler

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.metrics import accuracy\_score

features = a.drop(['status', 'name'], axis=1)

labels = a['status']

scaler = MinMaxScaler((-1, 1))

x = scaler.fit\_transform(features)

y = labels

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=5)

def start\_models():

from sklearn.linear\_model import LogisticRegression

from xgboost import XGBRFClassifier, XGBClassifier

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import AdaBoostClassifier, BaggingClassifier, ExtraTreesClassifier, \

GradientBoostingClassifier, RandomForestClassifier

lr = cross\_val\_score(LogisticRegression(), x\_train, y\_train)

xgbc = cross\_val\_score(XGBRFClassifier(), x\_train, y\_train)

xgb = cross\_val\_score(XGBClassifier(), x\_train, y\_train)

svm = cross\_val\_score(SVC(), x\_train, y\_train)

dtc = cross\_val\_score(DecisionTreeClassifier(), x\_train, y\_train)

adb = cross\_val\_score(AdaBoostClassifier(), x\_train, y\_train)

bbc = cross\_val\_score(BaggingClassifier(), x\_train, y\_train)

etc = cross\_val\_score(ExtraTreesClassifier(), x\_train, y\_train)

gbc = cross\_val\_score(GradientBoostingClassifier(), x\_train, y\_train)

rfc = cross\_val\_score(RandomForestClassifier(), x\_train, y\_train)

print('log reg', lr, lr.mean())

print('xgbd', xgbc, xgbc.mean())

print('xgb', xgb, xgb.mean())

print('svm', svm, svm.mean())

print('dtc', dtc, dtc.mean())

print('adb', adb, adb.mean())

print('bbc', bbc, bbc.mean())

print('etc', etc, etc.mean())

print('gbc', gbc, gbc.mean())

print('rfc', rfc, rfc.mean())

model = XGBClassifier()

model.fit(x\_train, y\_train)

accuracy\_dict = {}

# y\_predtr = model.predict(x\_train)

# print(accuracy\_score(y\_train, y\_predtr) \* 100)

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

xg\_accuracy = accuracy\_score(y\_test, y\_pred) \* 100

accuracy\_dict.update({'xg\_accuracy': xg\_accuracy})

model = ExtraTreesClassifier()

model.fit(x\_train, y\_train)

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

etc\_accuracy = accuracy\_score(y\_test, y\_pred) \* 100

accuracy\_dict.update({'etc\_accuracy': etc\_accuracy})

from sklearn.metrics import classification\_report, confusion\_matrix

print(classification\_report(y\_test, y\_pred))

print(confusion\_matrix(y\_test, y\_pred))

model = AdaBoostClassifier()

model.fit(x\_train, y\_train)

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

ada\_accuracy = accuracy\_score(y\_test, y\_pred) \* 100

accuracy\_dict.update({'ada\_accuracy': ada\_accuracy})

model = SVC(kernel='rbf')

model.fit(x\_train, y\_train)

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

svc\_accuracy = accuracy\_score(y\_test, y\_pred) \* 100

accuracy\_dict.update({'svc\_accuracy': svc\_accuracy})

model = RandomForestClassifier()

model.fit(x\_train, y\_train)

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

rf\_accuracy = accuracy\_score(y\_test, y\_pred) \* 100

accuracy\_dict.update({'rf\_accuracy': rf\_accuracy})

## ANN

import keras

from keras.models import Sequential

from keras.layers import Dense

classifier = Sequential()

classifier.add(Dense(output\_dim=6, init='uniform', activation='relu', input\_dim=22))

classifier.add(Dense(output\_dim=6, init='uniform', activation='relu'))

classifier.add(Dense(output\_dim=1, init='uniform', activation='sigmoid'))

classifier.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

print(classifier.summary())

classifier.fit(x\_train, y\_train, batch\_size=10, nb\_epoch=100)

y\_pred = classifier.predict(x\_test)

y\_pred = (y\_pred > 0.5)

ann\_accuracy = accuracy\_score(y\_test, y\_pred) \* 100

accuracy\_dict.update({'ann\_accuracy': ann\_accuracy})

return accuracy\_dict

Predictions\_user\_input.py

import os

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

from django.conf import settings

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

a = pd.read\_csv(path)

a = a[['NHR', 'HNR', 'RPDE', 'DFA', 'PPE', 'status']]

a.head()

a.shape

a.dtypes

# print(a.head())

sns.catplot(x='status', kind='count', data=a)

plt.show()

for i in a:

if i != 'status' and i != 'PPE':

sns.catplot(x='status', y=i, kind='box', data=a)

plt.show()

# b = a.drop(['name'], axis=1)

data = a.drop('status', axis=1)

sns.heatmap(data.corr(), annot=True)

plt.show()

from sklearn.preprocessing import MinMaxScaler

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

features = a.drop(['status'], axis=1)

labels = a['status']

scaler = MinMaxScaler((-1, 1))

x = scaler.fit\_transform(features)

y = labels

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=5)

def get\_result(test\_set):

from xgboost import XGBClassifier

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()

model.fit(x\_train, y\_train)

scaler = MinMaxScaler((-1, 1))

x = scaler.fit\_transform([test\_set])

y\_pred = model.predict(x)

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

print(y\_pred)

return y\_pred

Base.html

{%load static%}

<!doctype html>

<html class="no-js" lang="en">

<head>

<!-- META DATA -->

<meta charset="utf-8">

<meta content="IE=edge" http-equiv="X-UA-Compatible">

<meta content="width=device-width, initial-scale=1" name="viewport">

<link href="https://fonts.googleapis.com/css?family=Playfair+Display:400,400i,700,700i,900,900i" rel="stylesheet">

<link href="https://fonts.googleapis.com/css?family=Poppins:100,200,300,400,500,600,700,800,900" rel="stylesheet">

<link href="https://fonts.googleapis.com/css?family=Lato:100,300,400,700,900" rel="stylesheet">

<title>Parkinson</title>

<link href="{%static 'images/logo/favicon.png'%}" rel="shortcut icon" type="image/icon"/>

<link href="{%static 'css/font-awesome.min.css'%}" rel="stylesheet">

<link href="https://cdn.linearicons.com/free/1.0.0/icon-font.min.css" rel="stylesheet">

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

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

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

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

<link href="{%static 'css/owl.theme.default.min.css'%}" rel="stylesheet"/>

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

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

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

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

<!--[if lt IE 9]>

<script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>

<script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>

<![endif]-->

</head>

<body>

<!--[if lte IE 9]>

<p class="browserupgrade">You are using an <strong>outdated</strong> browser. Please <a href="https://browsehappy.com/">upgrade

your browser</a> to improve your experience and security.</p>

<![endif]-->

<!--menu start-->

<section id="menu">

<div class="container">

<div class="menubar">

<nav class="navbar navbar-default">

<div class="navbar-header">

<a class="navbar-brand" href="{%url 'index'%}">

<h2><font style="color:WHITE">Parkinson’s Disease</font></h2>

<!--<img src="{%static 'images/logo/logo.png'%}" alt="logo">-->

</a>

</div><!--/.navbar-header -->

<!-- Collect the nav links, forms, and other content for toggling -->

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

<ul class="nav navbar-nav navbar-right">

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

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

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

<li><a href="{%url 'UserRegister'%}">Registrations</a></li>

</ul><!-- / ul -->

</div><!-- /.navbar-collapse -->

</nav><!--/nav -->

</div><!--/.menubar -->

</div>

</section>

{%block contents%}

{%endblock%}

<footer class="footer-copyright">

<div class="container">

<div class="row">

<div class="col-sm-7">

<div class="foot-copyright pull-left">

<p>

&copy; All Rights Reserved. Designed and Developed by

<a href="#">Alex Corp</a>

</p>

</div><!--/.foot-copyright-->

</div><!--/.col-->

</div><!--/.row-->

<div id="scroll-Top">

<i aria-hidden="true" class="fa fa-angle-double-up return-to-top" data-original-title="Back to Top" data-placement="top"

data-toggle="tooltip" id="scroll-top" title=""></i>

</div><!--/.scroll-Top-->

</div><!-- /.container-->

</footer><!-- /.footer-copyright-->

<!-- footer-copyright end -->

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

<script src="https://cdnjs.cloudflare.com/ajax/libs/modernizr/2.8.3/modernizr.min.js" type="text/javascript"></script>

<script src="{%static 'js/bootstrap.min.js'%}" type="text/javascript"></script>

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

<script src="{%static 'js/jquery.hc-sticky.min.js'%}" type="text/javascript"></script>

<script src="{%static 'js/jquery.magnific-popup.min.js'%}"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-easing/1.4.1/jquery.easing.min.js"

type="text/javascript"></script>

<script src="{%static 'js/owl.carousel.min.js'%}" type="text/javascript"></script>

<script src="{%static 'js/jquery.counterup.min.js'%}"></script>

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

<script src="{%static 'js/jak-menusearch.js'%}" type="text/javascript"></script>

<script src="{%static 'js/custom.js'%}" type="text/javascript"></script>

</body>

</html>