SAMPLE CODE:

**User side views:**

# Create your views here.

from django.shortcuts import render, HttpResponse

from django.contrib import messages

from .forms import UserRegistrationForm

from .models import UserRegistrationModel

from django.conf import settings

# 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('loginid')

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/UserHomePage.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/UserHomePage.html', {})

def DatasetView(request):

path = settings.MEDIA\_ROOT + "//" + 'DataSet.csv'

import pandas as pd

df = pd.read\_csv(path, nrows=100,index\_col=False)

df.reset\_index()

df = df.to\_html

return render(request, 'users/viewdataset.html', {'data': df})

def preProcessData(request):

from .utility.PreprocessedData import preProcessed\_data\_view

data = preProcessed\_data\_view()

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

def Model\_Results(request):

from .utility import PreprocessedData

nb\_report = PreprocessedData.build\_naive\_bayes()

knn\_report = PreprocessedData.build\_knn()

dt\_report = PreprocessedData.build\_decsionTree()

rf\_report = PreprocessedData.build\_randomForest()

svm\_report = PreprocessedData.build\_svm()

mlp\_report = PreprocessedData.build\_mlp()

return render(request, 'users/ml\_reports.html', {'nb': nb\_report,"knn":knn\_report, 'dt': dt\_report, 'rf': rf\_report, 'svm': svm\_report,'mlp':mlp\_report})

def user\_input\_prediction(request):

if request.method=='POST':

from .utility import PreprocessedData

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

result = PreprocessedData.predict\_userInput(joninfo)

print(request)

return render(request, 'users/testform.html', {'result': result})

else:

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

**base.html:**

{%load static%}

<!DOCTYPE html>

<html>

<head>

<!-- /.website title -->

<title>Clouds html5 Multipurpose Landing Page for Apps</title>

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">

<!-- CSS Files -->

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

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

<link href="{%static 'fonts/icon-7-stroke/css/pe-icon-7-stroke.css'%}" rel="stylesheet">

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

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

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

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

<!-- Google Fonts -->

<link href='http://fonts.googleapis.com/css?family=Open+Sans:400,300,600,700' rel='stylesheet' type='text/css'>

<link href='http://fonts.googleapis.com/css?family=Alegreya+Sans:100,300,400,700' rel='stylesheet' type='text/css'>

<!-- Font Awesome -->

<link href="http://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-awesome.min.css" rel="stylesheet">

</head>

<body data-spy="scroll" data-target="#navbar-scroll">

<div id="top"></div>

<!-- NAVIGATION -->

<div id="menu">

<nav class="navbar-wrapper navbar-default" role="navigation">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-themers">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand site-name" href="#top" style="COlor:WHITE"><h2>Fake Job Posting</h2></a>

</div>

<div id="navbar-scroll" class="collapse navbar-collapse navbar-themers navbar-right">

<ul class="nav navbar-nav">

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

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

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

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

</ul>

</div>

</div>

</nav>

</div>

{%block contents%}

{%endblock%}

<!-- /.footer -->

<footer id="footer">

<div class="container">

<div class="col-sm-4 col-sm-offset-4">

<!-- /.social links -->

<div class="text-center wow fadeInUp" style="font-size: 14px;">Copyright Alex Corporations Template by <a

href="#">Alex Hales</a></div>

<a href="#" class="scrollToTop"><i class="fa fa-arrow-circle-o-up"></i></a>

</div>

</div>

</footer>

<!-- /.javascript files -->

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

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

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

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

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

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

<script src="{%static 'js/ekko-lightbox-min.js'%}"></script>

<script type="text/javascript">

$(document).delegate('\*[data-toggle="lightbox"]', 'click', function (event) {

event.preventDefault();

$(this).ekkoLightbox();

});

</script>

<script>

new WOW().init();

</script>

</body>

</html>

**Index.html:**

{%extends 'base.html'%}

{%load static%}

{%block contents%}

<!-- /.parallax full screen background image -->

<div class="fullscreen landing parallax banner" style="background-image:url('{%static 'images/bg.jpg'%}');"

data-img-width="2000" data-img-height="1325" data-diff="100">

<div class="overlay">

<div class="container">

<div class="row">

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

<!-- /.main title -->

<h1 class="wow fadeInLeft">

Common types of Job Scam

</h1>

<!-- /.header paragraph -->

<div class="landing-text wow fadeInLeft">

<p>

Fraudsters who want to gain other people’s personal information like insurance details, bank

details, income tax details, date of birth, national id create fake job advertisements.

Advance fee scams occur when frauds ask for money showing reasons like admin charges,

information security checking cost, management cost etc. Sometimes fraudsters act them-

selves as employers and ask people about passport details, bank statements, driving license

etc. as pre-employment check. Illegal money mulling scams occur when they convince students

to pay money into their accounts and then transfer it back.

</p>

</div>

</div>

<!-- /.phone image -->

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

<img src="{%static 'images/header-phone.png'%}" alt="phone"

class="header-phone img-responsive wow fadeInRight">

</div>

</div>

</div>

</div>

</div>

<!-- /.feature section -->

<div id="feature">

<div class="container">

<div class="row">

<div class="col-md-10 col-md-offset-1 col-sm-12 text-center feature-title">

<!-- /.feature title -->

<h2>Related Works</h2>

<p>

Many researches occurred to predict if a job post is real or

fake. A good number of research works are to check online

fraud job advertiser. Vidros [1] et al. identified job scammers

as fake online job advertiser. They found statistics about many

real and renowned companies and enterprises who produced

fake job advertisements or vacancy posts with ill-motive. They

experimented on EMSCAD dataset using several classification

algorithms like naive bayes classifier, random forest classifier,Zero R, One R etc. Random Forest

Classifier showed the

best performance on the dataset with 89.5% classification

accuracy

</p>

</div>

</div>

</div>

</div>

{%endblock%}

**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 AdminCartResults(request):

from users.utility.ProcessCart import start\_process\_cart

rslt\_dict = start\_process\_cart()

return render(request, "admins/admincartresults.html", rslt\_dict)

def AdminGBDTResults(request):

from users.utility.ProcessCart import start\_process\_gbdt

rslt\_dict = start\_process\_gbdt()

return render(request, "admins/admingbdtresults.html", rslt\_dict)

def classification\_report(request):

from users.utility import PreprocessedData

nb\_report = PreprocessedData.build\_naive\_bayes()

knn\_report = PreprocessedData.build\_knn()

dt\_report = PreprocessedData.build\_decsionTree()

rf\_report = PreprocessedData.build\_randomForest()

svm\_report = PreprocessedData.build\_svm()

mlp\_report = PreprocessedData.build\_mlp()

return render(request, 'admins/reports.html',

{'nb': nb\_report, "knn": knn\_report, 'dt': dt\_report, 'rf': rf\_report, 'svm': svm\_report,

'mlp': mlp\_report})