```
from django.shortcuts import render

# Create your views here.
from django.http import HttpResponse, HttpRequest
from django.shortcuts import render, redirect
#from .forms import *
from django.contrib import messages
from django.shortcuts import render
from django.urls import reverse_lazy
from django.urls import reverse
from django.http import HttpResponse
from django.views.generic import (View,TemplateView,
ListView,DetailView,
CreateView,DeleteView,
UpdateView)
from . import models
from .forms import *
from django.core.files.storage import FileSystemStorage
#from topicApp.Topicfun import Topic
#from ckdApp.funckd import ckd
#from sklearn.tree import export_graphviz #plot tree
#from sklearn.metrics import roc_curve, auc #for model evaluation
#from sklearn.metrics import classification_report #for model evaluation
##from sklearn.model_selection import train_test_split
#X_train, X_test, y_train, y_test =
train_test_split(df2.drop('classification_yes', 1), df2['classification_yes'],
test_size = .2, random_state=10)

import time
import pandas as pd
import numpy as np
#from sklearn.preprocessing import StandardScaler
#from sklearn.feature_selection import SelectKBest
#from sklearn.feature_selection import chi2
#from sklearn.model_selection import train_test_split
#from sklearn.decomposition import PCA
#from sklearn.feature_selection import RFE
#from sklearn.linear_model import LogisticRegression
import pickle
import matplotlib.pyplot as plt
#import eli5 #for purmutation importance
#from eli5.sklearn import PermutationImportance
#import shap #for SHAP values
#from pdpbox import pdp, info_plots #for partial plots
np.random.seed(123) #ensure reproduc
class dataUploadView(View):
    form_class = ckdForm
    success_url = reverse_lazy('success')
    template_name = 'create.html'
    failure_url= reverse_lazy('fail')
    filenot_url= reverse_lazy('filenot')
    def get(self, request, *args, **kwargs):
```

```
        form = self.form_class()
        return render(request, self.template_name, {'form': form})
    def post(self, request, *args, **kwargs):
        #print('inside post')
        form = self.form_class(request.POST, request.FILES)
        #print('inside form')
        if form.is_valid():
            form.save()
            data_a= request.POST.get('Age')
            data_d=request.POST.get('DailyRate')
            data_m=request.POST.get('MonthlyIncome')
            data_mr=request.POST.get('MonthlyRate')
            data_wy=request.POST.get('TotalWorkingYears')
            data_y= request.POST.get('YearsAtCompany')
            data_yc=request.POST.get('YearsInCurrentRole')
            data_cm=request.POST.get('YearsWithCurrManager')            #print
(data)
            #dataset1=pd.read_csv("prep.csv",index_col=None)
            dicc={'yes':1,'no':0}
            filename = 'finalized_model_rf.sav'
            classifier = pickle.load(open(filename, 'rb'))

            data =
np.array([data_a,data_d,data_m,data_mr,data_wy,data_y,data_yc,data_cm])
            #sc = StandardScaler()
            #data = sc.fit_transform(data.reshape(-1,1))
            out=classifier.predict(data.reshape(1,-1))
# providing an index
            #ser = pd.DataFrame(data, index =['bgr','bu','sc','pcv','wbc'])

            #ss=ser.T.squeeze()
#data_for_prediction = X_test1.iloc[0,:].astype(float)

#data_for_prediction =obj.pca(np.array(data_for_prediction),y_test)
            #obj=ckd()
            ##plt.savefig("static/force_plot.png",dpi=150, bbox_inches='tight')




            return render(request, "succ_msg.html",
{'data_a':data_a,'data_d':data_d,'data_m':data_m,'data_mr':data_mr,'data_wy':dat
a_wy,'data_y':data_y,'data_yc':data_yc,'data_cm':data_cm,
                                                        'out':out})


        else:
            return redirect(self.failure_url)
```