**SOURCE CODE**

**User side views.py**

**from** django.shortcuts **import** render,HttpResponse  
**from** django.contrib **import** messages  
**from** users.forms **import** UserRegistrationForm  
**from** users.models **import** UserRegistrationModel,FlightDataModel  
**import** io,csv  
**from** django.conf **import** settings  
  
**from** .FlightDataPreproces **import** DPDataPrePRocess  
**from** .models **import** FlightDataModel  
**from** django\_pandas.io **import** read\_frame  
*# Create your views here.***def** UserRegisterAction(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'**)  
 *# return HttpResponseRedirect('./CustLogin')* form = UserRegistrationForm()  
 **return** render(request, **'UserRegister.html'**, {**'form'**: form})  
 **else**:  
 print(**"Invalid form"**)  
 **else**:  
 form = UserRegistrationForm()  
 **return** render(request, **'UserRegister.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/UserHome.html'**, {})  
 **else**:  
 messages.success(request, **'Your Account Not at activated'**)  
 **return** render(request, **'UserLogin.html'**)  
 *# return render(request, 'user/userpage.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** UserUploadForm(request):  
 **return** render(request,**'users/uploadform.html'**,{})  
  
**def** UserDataUpload(request):  
 *# declaring template* template = **"users/UserHome.html"** data = FlightDataModel.objects.all()  
 *# prompt is a context variable that can have different values depending on their context* prompt = {  
 **'order'**: **'Order of the CSV should be name, email, address, phone, profile'**,  
 **'profiles'**: data  
 }  
 *# GET request returns the value of the data with the specified key.* **if** request.method == **"GET"**:  
 **return** render(request, template, prompt)  
 csv\_file = request.FILES[**'file'**]  
 *# let's check if it is a csv file* **if not** csv\_file.name.endswith(**'.csv'**):  
 messages.error(request, **'THIS IS NOT A CSV FILE'**)  
 data\_set = csv\_file.read().decode(**'UTF-8'**)  
 **try**:  
 *# setup a stream which is when we loop through each line we are able to handle a data in a stream* io\_string = io.StringIO(data\_set)  
 next(io\_string)  
 **for** column **in** csv.reader(io\_string, delimiter=**','**, quotechar=**"|"**):  
 \_, created = FlightDataModel.objects.update\_or\_create(  
 DAY = column[1],  
 DEPARTURE\_TIME= column[2],  
 FLIGHT\_NUMBER= column[3],  
 DESTINATION\_AIRPORT= column[4],  
 ORIGIN\_AIRPORT= column[5],  
 DAY\_OF\_WEEK= column[6],  
 TAXI\_OUT= column[7]  
 )  
 **except** Exception **as** ex:  
 print(**'error at'**, ex)  
 context = {}  
  
 **return** render(request, **'users/UserHome.html'**, context)  
  
**def** DataPreProcessing(request):  
 *#dataset = settings.MEDIA\_ROOT + "\\" + 'flightsdata.csv'* qs = FlightDataModel.objects.all()  
 dataset = read\_frame(qs)  
 print(**"Dataset "**,dataset)  
 x = DPDataPrePRocess()  
 data = x.process\_data(datasetname = dataset)  
  
 **return** render(request, **'users/PreProcessData.html'**,{**'data'**:qs})  
  
**def** UsermachineLearning(request):  
 qs = FlightDataModel.objects.all()  
 dataset = read\_frame(qs)  
 x = DPDataPrePRocess()  
 lg\_dict = x.MyLogiSticregression(dataset)  
 *#lg\_dict = {}* dt\_dict = x.MyDecisionTree(dataset)  
 rf\_dict = x.MyRandomForest(dataset)  
 br\_dict = x.MyBayesianRidge(dataset)  
 gbr\_dict = x.MyGradientBoostingRegressor(dataset)  
  
 **return** render(request,**'users/UsrMachineLearningRslt.html'**,{**'lg\_dict'**:lg\_dict,**'dt\_dict'**:dt\_dict,**'rf\_dict'**:rf\_dict,**'br\_dict'**:br\_dict,**'gbr\_dict'**:gbr\_dict})  
  
**def** UserGraphs(request):  
 qs = FlightDataModel.objects.all()  
 dataset = read\_frame(qs)  
 x = DPDataPrePRocess()  
 *#lg\_dict = x.MyLogiSticregression(dataset)* lg\_dict = {}  
 dt\_dict = x.MyDecisionTree(dataset)  
 rf\_dict = x.MyRandomForest(dataset)  
 br\_dict = x.MyBayesianRidge(dataset)  
 gbr\_dict = x.MyGradientBoostingRegressor(dataset)  
  
 **return** render(request, **'users/UserGraphs.html'**,  
 {**'lg\_dict'**: lg\_dict, **'dt\_dict'**: dt\_dict, **'rf\_dict'**: rf\_dict, **'br\_dict'**: br\_dict,  
 **'gbr\_dict'**: gbr\_dict})

**FlightDataProcess.py**

**import** pandas **as** pd  
**import** matplotlib.pyplot **as** plt  
**import** matplotlib  
matplotlib.use(**'TkAgg'**)  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.linear\_model **import** LogisticRegression  
**from** sklearn.tree **import** DecisionTreeRegressor  
**from** sklearn.metrics **import** accuracy\_score  
**from** sklearn.ensemble **import** RandomForestRegressor  
**from** sklearn.linear\_model **import** BayesianRidge  
**from** sklearn.ensemble **import** GradientBoostingRegressor  
**from** sklearn **import** metrics  
**import** numpy **as** np  
**class** DPDataPrePRocess:  
 **def** process\_data(self,datasetname):  
 *#dataset = pd.read\_csv(datasetname)* dataset = datasetname  
 dataset = dataset[[**'DAY'**, **'DEPARTURE\_TIME'**, **'FLIGHT\_NUMBER'**, **'DESTINATION\_AIRPORT'**, **'ORIGIN\_AIRPORT'**, **'DAY\_OF\_WEEK'**,**'TAXI\_OUT'**]]  
 *#print(dataset.dtypes)* dataset.fillna  
 dataset.dropna()  
 dataset = dataset.fillna(0)  
 dataset.fillna(method=**'ffill'**)  
 print(dataset.isnull().values.any())  
 *#print(dataset.head(10))  
 #plot\_corr(dataset)  
 #plt.show()* print(dataset.dtypes)  
 dataset.to\_csv(**'file1.csv'**)  
 data\_dict = dataset.to\_dict()  
 **return** data\_dict  
  
 **def** MyLogiSticregression(self,dataset):  
 print(**"###Logistic Regression####"**)  
 *#print('Have a great day ',dataset)  
 #dataset = pd.read\_csv(dataset)* dataset = dataset[[**'DAY'**,**'DEPARTURE\_TIME'**,**'FLIGHT\_NUMBER'**,**'DESTINATION\_AIRPORT'**,**'ORIGIN\_AIRPORT'**,**'DAY\_OF\_WEEK'**,**'TAXI\_OUT'**]]  
 *#print(dataset.head())* X = dataset.iloc[:,:1].values  
 y = dataset.iloc[:,2].values  
 X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y, test\_size=1/3,random\_state=0)  
  
 model = LogisticRegression()  
 model.fit(X\_train,y\_train)  
 y\_pred = model.predict(X\_test)  
  
 *#acuracy = accuracy\_score(y\_pred,y\_test)  
 #print(acuracy)* lgDict = {}  
 lg\_MAE = metrics.mean\_absolute\_error(y\_pred.round(), y\_test)  
 lg\_MSE = metrics.mean\_squared\_error(y\_pred.round(), y\_test)  
 lg\_EVS = metrics.explained\_variance\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
 lg\_MedianAE = metrics.median\_absolute\_error(y\_test, y\_pred)  
 lg\_R2Score = metrics.r2\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
  
 lgDict.update({**'lg\_MAE'**:lg\_MAE,**'lg\_MSE'**:lg\_MSE,**'lg\_EVS'**:lg\_EVS,**'lg\_MedianAE'**:lg\_MedianAE,**'lg\_R2Score'**:lg\_R2Score})  
  
 print(**"MAE="**,lg\_MAE )  
 print(**"MSE="**,lg\_MSE )  
 print(**"RMSE="**, np.sqrt(metrics.mean\_squared\_error(y\_pred.round(), y\_test)))  
 print(**"Variance Score "**,lg\_EVS)  
 print(**"Median Absalute Error="**,lg\_MedianAE)  
 print(**"R2\_Score"**, lg\_R2Score)  
  
 **return** lgDict  
  
 **def** MyDecisionTree(self, dataset):  
 print(**"###Decesion Treee####"**)  
 *#print('Have a great day ', dataset)  
 #dataset = pd.read\_csv(dataset)* dataset = dataset[  
 [**'DAY'**, **'DEPARTURE\_TIME'**, **'FLIGHT\_NUMBER'**, **'DESTINATION\_AIRPORT'**, **'ORIGIN\_AIRPORT'**, **'DAY\_OF\_WEEK'**,  
 **'TAXI\_OUT'**]]  
 *#print(dataset.head())* X = dataset.iloc[:, :1].values  
 y = dataset.iloc[:, 2].values  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=1 / 3, random\_state=0)  
  
 model = DecisionTreeRegressor()  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
  
 *# acuracy = accuracy\_score(y\_pred,y\_test)  
 # print(acuracy)* dtDict = {}  
 dt\_MAE = metrics.mean\_absolute\_error(y\_pred.round(), y\_test)  
 dt\_MSE = metrics.mean\_squared\_error(y\_pred.round(), y\_test)  
 dt\_EVS = metrics.explained\_variance\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
 dt\_MedianAE = metrics.median\_absolute\_error(y\_test, y\_pred)  
 dt\_R2Score = metrics.r2\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
  
 dtDict.update({**'dt\_MAE'**:dt\_MAE,**'dt\_MSE'**:dt\_MSE,**'dt\_EVS'**:dt\_EVS,**'dt\_MedianAE'**:dt\_MedianAE,**'dt\_R2Score'**:dt\_R2Score})  
  
 print(**"MAE="**, dt\_MAE)  
 print(**"MSE="**, dt\_MAE)  
 print(**"RMSE="**, np.sqrt(metrics.mean\_squared\_error(y\_pred.round(), y\_test)))  
 print(**"Variance Score "**,dt\_EVS)  
 print(**"Median Absalute Error="**, dt\_MedianAE)  
 print(**"R2\_Score"**, dt\_R2Score)  
 **return** dtDict  
  
 **def** MyRandomForest(self, dataset):  
 print(**"###RadomForest####"**)  
 *#print('Have a great day ', dataset)  
 #dataset = pd.read\_csv(dataset)* dataset = dataset[  
 [**'DAY'**, **'DEPARTURE\_TIME'**, **'FLIGHT\_NUMBER'**, **'DESTINATION\_AIRPORT'**, **'ORIGIN\_AIRPORT'**, **'DAY\_OF\_WEEK'**,  
 **'TAXI\_OUT'**]]  
 *#print(dataset.head())* X = dataset.iloc[:, :1].values  
 y = dataset.iloc[:, 2].values  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=1 / 3, random\_state=0)  
  
 model = RandomForestRegressor()  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
  
 *# acuracy = accuracy\_score(y\_pred,y\_test)  
 # print(acuracy)* rfDict = {}  
 rf\_MAE = metrics.mean\_absolute\_error(y\_pred.round(), y\_test)  
 rf\_MSE = metrics.mean\_squared\_error(y\_pred.round(), y\_test)  
 rf\_EVS = metrics.explained\_variance\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
 rf\_MedianAE = metrics.median\_absolute\_error(y\_test, y\_pred)  
 rf\_R2Score = metrics.r2\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
  
 rfDict.update({**'rf\_MAE'**: rf\_MAE, **'rf\_MSE'**: rf\_MSE, **'rf\_EVS'**: rf\_EVS, **'rf\_MedianAE'**: rf\_MedianAE,  
 **'rf\_R2Score'**: rf\_R2Score})  
  
 print(**"MAE="**, rf\_MAE)  
 print(**"MSE="**, rf\_MSE)  
 print(**"RMSE="**, np.sqrt(metrics.mean\_squared\_error(y\_pred.round(), y\_test)))  
 print(**"Variance Score "**, rf\_EVS)  
 print(**"Median Absalute Error="**, rf\_MedianAE)  
 print(**"R2\_Score"**, rf\_R2Score)  
 **return** rfDict  
  
 **def** MyBayesianRidge(self, dataset):  
 print(**"###RadomForest####"**)  
 *#print('Have a great day ', dataset)  
 #dataset = pd.read\_csv(dataset)* dataset = dataset[  
 [**'DAY'**, **'DEPARTURE\_TIME'**, **'FLIGHT\_NUMBER'**, **'DESTINATION\_AIRPORT'**, **'ORIGIN\_AIRPORT'**, **'DAY\_OF\_WEEK'**,  
 **'TAXI\_OUT'**]]  
 *#print(dataset.head())* X = dataset.iloc[:, :1].values  
 y = dataset.iloc[:, 2].values  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=1 / 3, random\_state=0)  
  
 model = BayesianRidge()  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
  
 *# acuracy = accuracy\_score(y\_pred,y\_test)  
 # print(acuracy)* brDict = {}  
 br\_MAE = metrics.mean\_absolute\_error(y\_pred.round(), y\_test)  
 br\_MSE = metrics.mean\_squared\_error(y\_pred.round(), y\_test)  
 br\_EVS = metrics.explained\_variance\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
 br\_MedianAE = metrics.median\_absolute\_error(y\_test, y\_pred)  
 br\_R2Score = metrics.r2\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
  
 brDict.update({**'br\_MAE'**: br\_MAE, **'br\_MSE'**: br\_MSE, **'br\_EVS'**: br\_EVS, **'br\_MedianAE'**: br\_MedianAE,  
 **'br\_R2Score'**: br\_R2Score})  
  
 print(**"MAE="**, br\_MAE)  
 print(**"MSE="**, br\_MSE)  
 print(**"RMSE="**, np.sqrt(metrics.mean\_squared\_error(y\_pred.round(), y\_test)))  
 print(**"Variance Score "**, br\_EVS)  
 print(**"Median Absalute Error="**, br\_MedianAE)  
 print(**"R2\_Score"**, br\_R2Score)  
 **return** brDict  
  
 **def** MyGradientBoostingRegressor(self, dataset):  
 print(**"###GradientBoostingRegressor####"**)  
 *#print('Have a great day ', dataset)  
 #dataset = pd.read\_csv(dataset)* dataset = dataset[  
 [**'DAY'**, **'DEPARTURE\_TIME'**, **'FLIGHT\_NUMBER'**, **'DESTINATION\_AIRPORT'**, **'ORIGIN\_AIRPORT'**, **'DAY\_OF\_WEEK'**,  
 **'TAXI\_OUT'**]]  
 *#print(dataset.head())* X = dataset.iloc[:, :1].values  
 y = dataset.iloc[:, 2].values  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=1 / 3, random\_state=0)  
  
 model = GradientBoostingRegressor()  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
  
 *# acuracy = accuracy\_score(y\_pred,y\_test)  
 # print(acuracy)* gbrDict = {}  
 gbr\_MAE = metrics.mean\_absolute\_error(y\_pred.round(), y\_test)  
 gbr\_MSE = metrics.mean\_squared\_error(y\_pred.round(), y\_test)  
 gbr\_EVS = metrics.explained\_variance\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
 gbr\_MedianAE = metrics.median\_absolute\_error(y\_test, y\_pred)  
 gbr\_R2Score = metrics.r2\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
  
 gbrDict.update({**'gbr\_MAE'**: gbr\_MAE, **'gbr\_MSE'**: gbr\_MSE, **'gbr\_EVS'**: gbr\_EVS, **'gbr\_MedianAE'**: gbr\_MedianAE,  
 **'gbr\_R2Score'**: gbr\_R2Score})  
  
 print(**"MAE="**, gbr\_MAE)  
 print(**"MSE="**, gbr\_MSE)  
 print(**"RMSE="**, np.sqrt(metrics.mean\_squared\_error(y\_pred.round(), y\_test)))  
 print(**"Variance Score "**, gbr\_EVS)  
 print(**"Median Absalute Error="**, gbr\_MedianAE)  
 print(**"R2\_Score"**, gbr\_R2Score)  
 **return** gbrDict  
  
  
  
  
**def** plot\_corr(data\_frame, size=11):  
 corr = data\_frame.corr() *# data frame correlation function* fig, ax = plt.subplots(figsize=(size, size))  
 ax.matshow(corr) *# color code the rectangles by correlation value* plt.xticks(range(len(corr.columns)), corr.columns) *# draw x tick marks* plt.yticks(range(len(corr.columns)), corr.columns) *# draw y tick marks*

**Urls.py**

*"""FlightDelays URL Configuration  
  
The `urlpatterns` list routes URLs to views. For more information please see:  
 https://docs.djangoproject.com/en/2.0/topics/http/urls/  
Examples:  
Function views  
 1. Add an import: from my\_app import views  
 2. Add a URL to urlpatterns: path('', views.home, name='home')  
Class-based views  
 1. Add an import: from other\_app.views import Home  
 2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')  
Including another URLconf  
 1. Import the include() function: from django.urls import include, path  
 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))  
"""***from** django.contrib **import** admin  
**from** django.urls **import** path  
**from** FlightDelays **import** views **as** mainview  
**from** users **import** views **as** usr  
**from** admins **import** views **as** admins  
  
  
urlpatterns = [  
 path(**'admin/'**, admin.site.urls),  
 path(**''**,mainview.index,name=**'index'**),  
 path(**'UserRegister/'**,mainview.UserRegister, name=**'UserRegister'**),  
 path(**'UserLogin/'**,mainview.UserLogin, name=**'UserLogin'**),  
 path(**'AdminLogin/'**, mainview.AdminLogin, name=**'AdminLogin'**),  
 path(**'Logout/'**, mainview.Logout, name=**'Logout'**),  
  
 *### User Based URLS* path(**'UserRegisterAction/'**,usr.UserRegisterAction, name=**'UserRegisterAction'**),  
 path(**'UserLoginCheck/'**,usr.UserLoginCheck, name=**'UserLoginCheck'**),  
 path(**'UserUploadForm/'**, usr.UserUploadForm, name=**'UserUploadForm'**),  
 path(**'UserDataUpload/'**, usr.UserDataUpload, name=**'UserDataUpload'**),  
 path(**'DataPreProcessing/'**,usr.DataPreProcessing, name=**'DataPreProcessing'**),  
 path(**'UsermachineLearning/'**, usr.UsermachineLearning, name=**'UsermachineLearning'**),  
 path(**'UserGraphs/'**, usr.UserGraphs, name=**'UserGraphs'**),  
  
 *### Admin Based Urls* path(**'AdminLoginCheck/'**,admins.AdminLoginCheck, name=**'AdminLoginCheck'**),  
 path(**'ViewUsers/'**, admins.ViewUsers, name=**'ViewUsers'**),  
 path(**'AdminActivaUsers/'**, admins.AdminActivaUsers, name=**'AdminActivaUsers'**),  
 path(**'AdmimnAddData/'**,admins.AdmimnAddData, name=**'AdmimnAddData'**),  
 path(**'AdminAddingFlightData/'**,admins.AdminAddingFlightData, name=**'AdminAddingFlightData'**),  
 path(**'AdminViewData/'**,admins.AdminViewData, name=**'AdminViewData'**),  
 path(**'AdminFindArrivalDelay/'**, admins.AdminFindArrivalDelay, name=**'AdminFindArrivalDelay'**),  
 path(**'AdminGraphs/'**,admins.AdminGraphs, name=**'AdminGraphs'**),  
  
  
  
  
]

Adminside Views.py

**from** django.shortcuts **import** render,HttpResponse  
**from** django.contrib **import** messages  
**from** users.models **import** UserRegistrationModel,FlightDataModel  
**from** .forms **import** FlightDataForms  
**from** django.core.paginator **import** Paginator, EmptyPage, PageNotAnInteger  
**from** django.conf **import** settings  
*# Create your views here.***from** .CalculationArrivalDelay **import** ArrivalDelay  
  
**def** AdminLoginCheck(request):  
 **if** request.method == **'POST'**:  
 usrid = request.POST.get(**'loginname'**)  
 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** ViewUsers(request):  
 data = UserRegistrationModel.objects.all()  
 **return** render(request, **'admins/ViewUsers.html'**,{**'data'**:data})  
  
**def** AdminActivaUsers(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/ViewUsers.html'**,{**'data'**:data})  
  
**def** AdmimnAddData(request):  
 form = FlightDataForms()  
 **return** render(request,**'admins/AddDataForm.html'**,{**'form'**:form})  
  
**def** AdminAddingFlightData(request):  
 **if** request.method == **'POST'**:  
 form = FlightDataForms(request.POST)  
 **if** form.is\_valid():  
 print(**'Data is Valid'**)  
 form.save()  
 messages.success(request, **'Data Added Successfull'**)  
 form = FlightDataForms()  
 **return** render(request, **'admins/AddDataForm.html'**, {**'form'**: form})  
 **else**:  
 print(**"Invalid form"**)  
 **else**:  
 form = FlightDataForms()  
 **return** render(request, **'admins/AddDataForm.html'**, {**'form'**: form})  
  
  
**def** AdminViewData(request):  
 data\_list = FlightDataModel.objects.all()  
 page = request.GET.get(**'page'**, 1)  
  
 paginator = Paginator(data\_list, 60)  
 **try**:  
 users = paginator.page(page)  
 **except** PageNotAnInteger:  
 users = paginator.page(1)  
 **except** EmptyPage:  
 users = paginator.page(paginator.num\_pages)  
  
 **return** render(request, **'admins/AdminViewFlightData.html'**, {**'users'**: users})  
  
**def** AdminFindArrivalDelay(request):  
 dataset = settings.MEDIA\_ROOT + **"\\"** + **'flightsdata.csv'** obj = ArrivalDelay()  
 lg\_dict = obj.MyLogiSticregression(dataset)  
 *#lg\_dict = {}* dt\_dict = obj.MyDecisionTree(dataset)  
 rf\_dict = obj.MyRandomForest(dataset)  
 br\_dict = obj.MyBayesianRidge(dataset)  
 gbr\_dict = obj.MyGradientBoostingRegressor(dataset)  
  
 **return** render(request, **'admins/AdminMachineLearningRslt.html'**,  
 {**'lg\_dict'**: lg\_dict, **'dt\_dict'**: dt\_dict, **'rf\_dict'**: rf\_dict, **'br\_dict'**: br\_dict,  
 **'gbr\_dict'**: gbr\_dict})  
  
  
**def** AdminGraphs(request):  
 dataset = settings.MEDIA\_ROOT + **"\\"** + **'flightsdata.csv'** obj = ArrivalDelay()  
 *#lg\_dict = x.MyLogiSticregression(dataset)* lg\_dict = {}  
 dt\_dict = obj.MyDecisionTree(dataset)  
 rf\_dict = obj.MyRandomForest(dataset)  
 br\_dict = obj.MyBayesianRidge(dataset)  
 gbr\_dict = obj.MyGradientBoostingRegressor(dataset)  
  
 **return** render(request, **'admins/AdminGraphs.html'**,  
 {**'lg\_dict'**: lg\_dict, **'dt\_dict'**: dt\_dict, **'rf\_dict'**: rf\_dict, **'br\_dict'**: br\_dict,  
 **'gbr\_dict'**: gbr\_dict})

**Calculation ArrivalDelay.py**

**import** pandas **as** pd  
**import** matplotlib.pyplot **as** plt  
**import** matplotlib  
matplotlib.use(**'TkAgg'**)  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.linear\_model **import** LogisticRegression  
**from** sklearn.tree **import** DecisionTreeRegressor  
**from** sklearn.metrics **import** accuracy\_score  
**from** sklearn.ensemble **import** RandomForestRegressor  
**from** sklearn.linear\_model **import** BayesianRidge  
**from** sklearn.ensemble **import** GradientBoostingRegressor  
**from** sklearn **import** metrics  
**import** numpy **as** np  
**class** ArrivalDelay:  
 **def** process\_data(self,datasetname):  
 *#dataset = pd.read\_csv(datasetname)* dataset = datasetname  
 dataset = dataset[[**'DAY'**, **'DEPARTURE\_TIME'**, **'FLIGHT\_NUMBER'**, **'DESTINATION\_AIRPORT'**, **'ORIGIN\_AIRPORT'**, **'DAY\_OF\_WEEK'**,**'TAXI\_OUT'**]]  
 *#print(dataset.dtypes)* dataset.fillna  
 dataset.dropna()  
 dataset = dataset.fillna(0)  
 dataset.fillna(method=**'ffill'**)  
 print(dataset.isnull().values.any())  
 *#print(dataset.head(10))  
 #plot\_corr(dataset)  
 #plt.show()* print(dataset.dtypes)  
 dataset.to\_csv(**'file1.csv'**)  
 data\_dict = dataset.to\_dict()  
 **return** data\_dict  
  
 **def** MyLogiSticregression(self,dataset):  
 print(**"###Logistic Regression####"**)  
 *#print('Have a great day ',dataset)* dataset = pd.read\_csv(dataset)  
 dataset = dataset[[**'DAY'**,**'DEPARTURE\_TIME'**,**'FLIGHT\_NUMBER'**,**'ARRIVAL\_DELAY'**,**'DESTINATION\_AIRPORT'**,**'ORIGIN\_AIRPORT'**,**'DAY\_OF\_WEEK'**,**'TAXI\_OUT'**]]  
 *#print(dataset.head())* dataset.fillna  
 dataset.dropna()  
 dataset = dataset.fillna(0)  
 X = dataset.iloc[:,:3].values  
 y = dataset.iloc[:,2].values  
 X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y, test\_size=1/3,random\_state=0)  
  
 model = LogisticRegression()  
 model.fit(X\_train,y\_train)  
 y\_pred = model.predict(X\_test)  
  
 *#acuracy = accuracy\_score(y\_pred,y\_test)  
 #print(acuracy)* lgDict = {}  
 lg\_MAE = metrics.mean\_absolute\_error(y\_pred.round(), y\_test)  
 lg\_MSE = metrics.mean\_squared\_error(y\_pred.round(), y\_test)  
 lg\_EVS = metrics.explained\_variance\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
 lg\_MedianAE = metrics.median\_absolute\_error(y\_test, y\_pred)  
 lg\_R2Score = metrics.r2\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
  
 lgDict.update({**'lg\_MAE'**:round(lg\_MAE,2),**'lg\_MSE'**:round(lg\_MSE,2),**'lg\_EVS'**:round(lg\_EVS,2),**'lg\_MedianAE'**:round(lg\_MedianAE,2),**'lg\_R2Score'**:round(lg\_R2Score,2)})  
  
 print(**"MAE="**,lg\_MAE )  
 print(**"MSE="**,lg\_MSE )  
 print(**"RMSE="**, np.sqrt(metrics.mean\_squared\_error(y\_pred.round(), y\_test)))  
 print(**"Variance Score "**,lg\_EVS)  
 print(**"Median Absalute Error="**,lg\_MedianAE)  
 print(**"R2\_Score"**, lg\_R2Score)  
  
 **return** lgDict  
  
 **def** MyDecisionTree(self, dataset):  
 print(**"###Decesion Treee####"**)  
 *#print('Have a great day ', dataset)* dataset = pd.read\_csv(dataset)  
 dataset = dataset[[**'DAY'**,**'DEPARTURE\_TIME'**,**'FLIGHT\_NUMBER'**,**'ARRIVAL\_DELAY'**,**'DESTINATION\_AIRPORT'**,**'ORIGIN\_AIRPORT'**,**'DAY\_OF\_WEEK'**,**'TAXI\_OUT'**]]  
 *#print(dataset.head())* dataset.fillna  
 dataset.dropna()  
 dataset = dataset.fillna(0)  
 X = dataset.iloc[:, :3].values  
 y = dataset.iloc[:, 2].values  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=1 / 3, random\_state=0)  
  
 model = DecisionTreeRegressor()  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
  
 *# acuracy = accuracy\_score(y\_pred,y\_test)  
 # print(acuracy)* dtDict = {}  
 dt\_MAE = metrics.mean\_absolute\_error(y\_pred.round(), y\_test)  
 dt\_MSE = metrics.mean\_squared\_error(y\_pred.round(), y\_test)  
 dt\_EVS = metrics.explained\_variance\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
 dt\_MedianAE = metrics.median\_absolute\_error(y\_test, y\_pred)  
 dt\_R2Score = metrics.r2\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
  
 dtDict.update({**'dt\_MAE'**:round(dt\_MAE,2),**'dt\_MSE'**:round(dt\_MSE,2),**'dt\_EVS'**:round(dt\_EVS,2),**'dt\_MedianAE'**:round(dt\_MedianAE,2),**'dt\_R2Score'**:round(dt\_R2Score,2)})  
  
 print(**"MAE="**, dt\_MAE)  
 print(**"MSE="**, dt\_MAE)  
 print(**"RMSE="**, np.sqrt(metrics.mean\_squared\_error(y\_pred.round(), y\_test)))  
 print(**"Variance Score "**,dt\_EVS)  
 print(**"Median Absalute Error="**, dt\_MedianAE)  
 print(**"R2\_Score"**, dt\_R2Score)  
 **return** dtDict  
  
 **def** MyRandomForest(self, dataset):  
 print(**"###RadomForest####"**)  
 *#print('Have a great day ', dataset)* dataset = pd.read\_csv(dataset)  
 dataset = dataset[  
 [**'DAY'**, **'DEPARTURE\_TIME'**, **'FLIGHT\_NUMBER'**, **'ARRIVAL\_DELAY'**, **'DESTINATION\_AIRPORT'**, **'ORIGIN\_AIRPORT'**,  
 **'DAY\_OF\_WEEK'**, **'TAXI\_OUT'**]]  
 *# print(dataset.head())* dataset.fillna  
 dataset.dropna()  
 dataset = dataset.fillna(0)  
 X = dataset.iloc[:, :3].values  
 y = dataset.iloc[:, 2].values  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=1 / 3, random\_state=0)  
  
 model = RandomForestRegressor()  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
  
 *# acuracy = accuracy\_score(y\_pred,y\_test)  
 # print(acuracy)* rfDict = {}  
 rf\_MAE = metrics.mean\_absolute\_error(y\_pred.round(), y\_test)  
 rf\_MSE = metrics.mean\_squared\_error(y\_pred.round(), y\_test)  
 rf\_EVS = metrics.explained\_variance\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
 rf\_MedianAE = metrics.median\_absolute\_error(y\_test, y\_pred)  
 rf\_R2Score = metrics.r2\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
  
 rfDict.update({**'rf\_MAE'**: round(rf\_MAE,2), **'rf\_MSE'**: round(rf\_MSE,2), **'rf\_EVS'**: round(rf\_EVS,2), **'rf\_MedianAE'**: round(rf\_MedianAE,2),  
 **'rf\_R2Score'**: round(rf\_R2Score,2)})  
  
 print(**"MAE="**, rf\_MAE)  
 print(**"MSE="**, rf\_MSE)  
 print(**"RMSE="**, np.sqrt(metrics.mean\_squared\_error(y\_pred.round(), y\_test)))  
 print(**"Variance Score "**, rf\_EVS)  
 print(**"Median Absalute Error="**, rf\_MedianAE)  
 print(**"R2\_Score"**, rf\_R2Score)  
 **return** rfDict  
  
 **def** MyBayesianRidge(self, dataset):  
 print(**"###RadomForest####"**)  
 *#print('Have a great day ', dataset)* dataset = pd.read\_csv(dataset)  
 dataset = dataset[  
 [**'DAY'**, **'DEPARTURE\_TIME'**, **'FLIGHT\_NUMBER'**, **'ARRIVAL\_DELAY'**, **'DESTINATION\_AIRPORT'**, **'ORIGIN\_AIRPORT'**,  
 **'DAY\_OF\_WEEK'**, **'TAXI\_OUT'**]]  
 *# print(dataset.head())* dataset.fillna  
 dataset.dropna()  
 dataset = dataset.fillna(0)  
 X = dataset.iloc[:, :3].values  
 y = dataset.iloc[:, 2].values  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=1 / 3, random\_state=0)  
  
 model = BayesianRidge()  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
  
 *# acuracy = accuracy\_score(y\_pred,y\_test)  
 # print(acuracy)* brDict = {}  
 br\_MAE = metrics.mean\_absolute\_error(y\_pred.round(), y\_test)  
 br\_MSE = metrics.mean\_squared\_error(y\_pred.round(), y\_test)  
 br\_EVS = metrics.explained\_variance\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
 br\_MedianAE = metrics.median\_absolute\_error(y\_test, y\_pred)  
 br\_R2Score = metrics.r2\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
  
 brDict.update({**'br\_MAE'**: round(br\_MAE,2), **'br\_MSE'**: round(br\_MSE,2), **'br\_EVS'**: round(br\_EVS,2), **'br\_MedianAE'**: round(br\_MedianAE,2),  
 **'br\_R2Score'**: round(br\_R2Score,2)})  
  
 print(**"MAE="**, br\_MAE)  
 print(**"MSE="**, br\_MSE)  
 print(**"RMSE="**, np.sqrt(metrics.mean\_squared\_error(y\_pred.round(), y\_test)))  
 print(**"Variance Score "**, br\_EVS)  
 print(**"Median Absalute Error="**, br\_MedianAE)  
 print(**"R2\_Score"**, br\_R2Score)  
 **return** brDict  
  
 **def** MyGradientBoostingRegressor(self, dataset):  
 print(**"###GradientBoostingRegressor####"**)  
 *#print('Have a great day ', dataset)* dataset = pd.read\_csv(dataset)  
 dataset = dataset[  
 [**'DAY'**, **'DEPARTURE\_TIME'**, **'FLIGHT\_NUMBER'**, **'ARRIVAL\_DELAY'**, **'DESTINATION\_AIRPORT'**, **'ORIGIN\_AIRPORT'**,  
 **'DAY\_OF\_WEEK'**, **'TAXI\_OUT'**]]  
 *# print(dataset.head())* dataset.fillna  
 dataset.dropna()  
 dataset = dataset.fillna(0)  
 X = dataset.iloc[:, :3].values  
 y = dataset.iloc[:, 2].values  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=1 / 3, random\_state=0)  
  
 model = GradientBoostingRegressor()  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
  
 *# acuracy = accuracy\_score(y\_pred,y\_test)  
 # print(acuracy)* gbrDict = {}  
 gbr\_MAE = metrics.mean\_absolute\_error(y\_pred.round(), y\_test)  
 gbr\_MSE = metrics.mean\_squared\_error(y\_pred.round(), y\_test)  
 gbr\_EVS = metrics.explained\_variance\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
 gbr\_MedianAE = metrics.median\_absolute\_error(y\_test, y\_pred)  
 gbr\_R2Score = metrics.r2\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)  
  
 gbrDict.update({**'gbr\_MAE'**: round(gbr\_MAE,2), **'gbr\_MSE'**: round(gbr\_MSE,2), **'gbr\_EVS'**: round(gbr\_EVS,2), **'gbr\_MedianAE'**: round(gbr\_MedianAE,2),  
 **'gbr\_R2Score'**: round(gbr\_R2Score,2)})  
  
 print(**"MAE="**, gbr\_MAE)  
 print(**"MSE="**, gbr\_MSE)  
 print(**"RMSE="**, np.sqrt(metrics.mean\_squared\_error(y\_pred.round(), y\_test)))  
 print(**"Variance Score "**, gbr\_EVS)  
 print(**"Median Absalute Error="**, gbr\_MedianAE)  
 print(**"R2\_Score"**, gbr\_R2Score)  
 **return** gbrDict  
  
  
  
  
**def** plot\_corr(data\_frame, size=11):  
 corr = data\_frame.corr() *# data frame correlation function* fig, ax = plt.subplots(figsize=(size, size))  
 ax.matshow(corr) *# color code the rectangles by correlation value* plt.xticks(range(len(corr.columns)), corr.columns) *# draw x tick marks* plt.yticks(range(len(corr.columns)), corr.columns) *# draw y tick marks*

models.py

**from** django.db **import** models  
  
  
*# Create your models here.***class** UserRegistrationModel(models.Model):  
 name = models.CharField(max\_length=100)  
 loginid = models.CharField(unique=**True**, max\_length=100)  
 password = models.CharField(max\_length=100)  
 mobile = models.CharField(max\_length=100)  
 email = models.CharField(max\_length=100)  
 locality = models.CharField(max\_length=100)  
 address = models.CharField(max\_length=1000)  
 city = models.CharField(max\_length=100)  
 state = models.CharField(max\_length=100)  
 status = models.CharField(max\_length=100)  
  
 **def** \_\_str\_\_(self):  
 **return** self.loginid  
  
 **class** Meta:  
 db\_table = **'AviationUsers'  
  
  
class** FlightDataModel(models.Model):  
 DAY =models.IntegerField(default=0)  
 DEPARTURE\_TIME =models.FloatField(default=0.0)  
 FLIGHT\_NUMBER =models.IntegerField(default=0)  
 DESTINATION\_AIRPORT =models.CharField(max\_length=100)  
 ORIGIN\_AIRPORT =models.CharField(max\_length=100)  
 DAY\_OF\_WEEK =models.IntegerField(default=0)  
 TAXI\_OUT =models.FloatField(default=0.0)  
 **def** \_\_str\_\_(self):  
 **return** self.id  
  
 **class** Meta:  
 db\_table = **"FlighDelayData"**