LABORATORIO 14

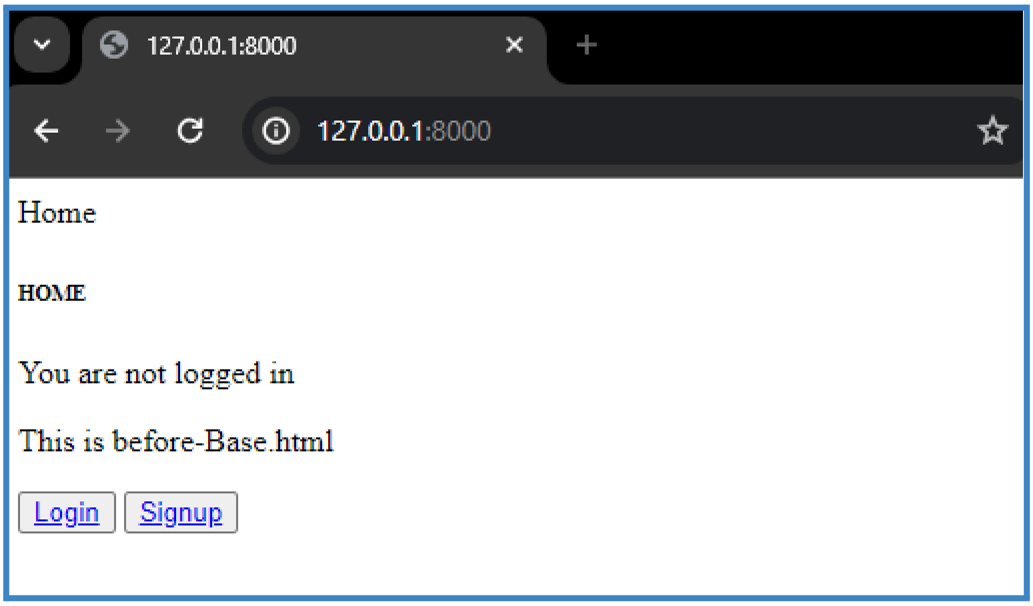
“Avance Proyecto Final”

Alumno(s):	- *Alvaro Huamani Jair Imanol* - *Castro Vilchez Estefany* - *Dávila Vargas Randy Blasco* - *Fontela Vilcasa Rodrigo Alejandro* - *Rojas Huayhua Yesica Nancy*		***Nota***	
Grupo:	*C24-B*		***Ciclo: V***	

Descripción:

En el siguiente link https://github.com/shr1911/Tourism-Recommendation.git podemos encontrar un proyecto el cual contiene un sistema de recomendación de restaurantes en base a precio, cercanía y hora de disponibilidad, rating y preferencias del usuario establecidas.

Paso 1:



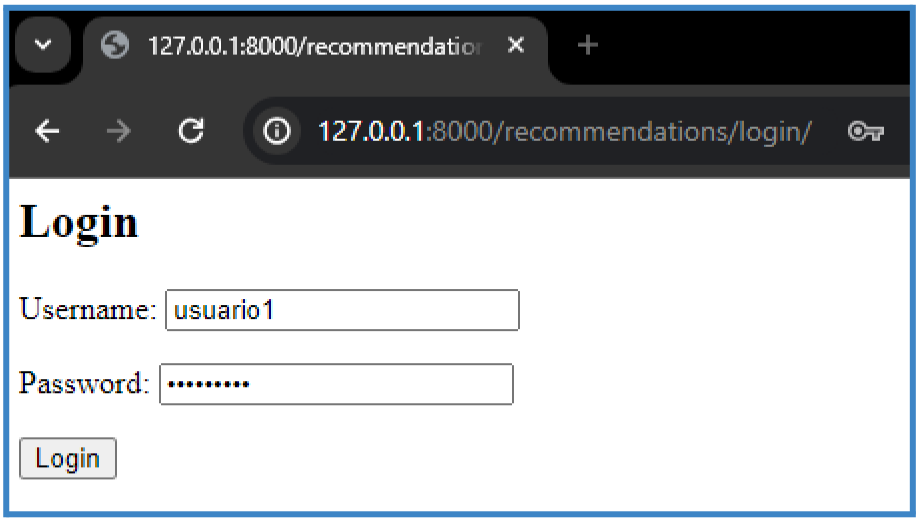
Home

HOME

You are not logged in

This is before-Base.html

Paso2:

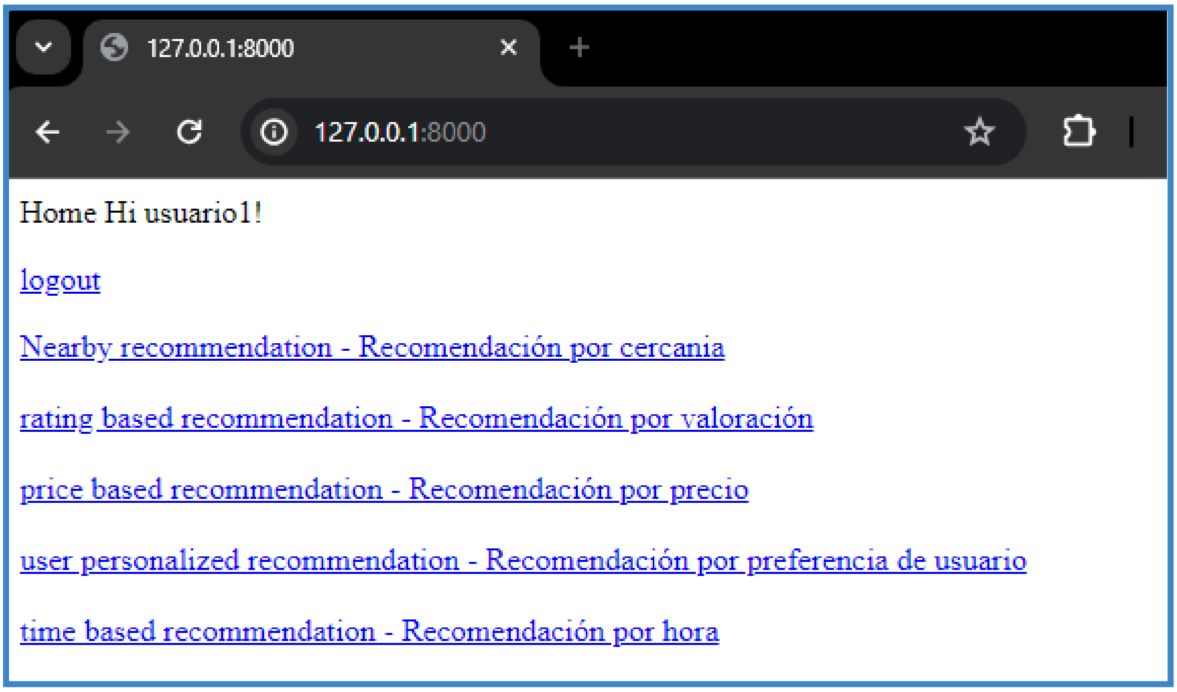

127.0.0.1:8000/recommendatio
127.0.0.1:8000/recommendations/login/

Login

Username: usuario1

Password: •••••••••

Paso 3:



Home Hi usuario1!

logout

Nearby recommendation - Recomendación por cercania

rating based recommendation - Recomendación por valoración

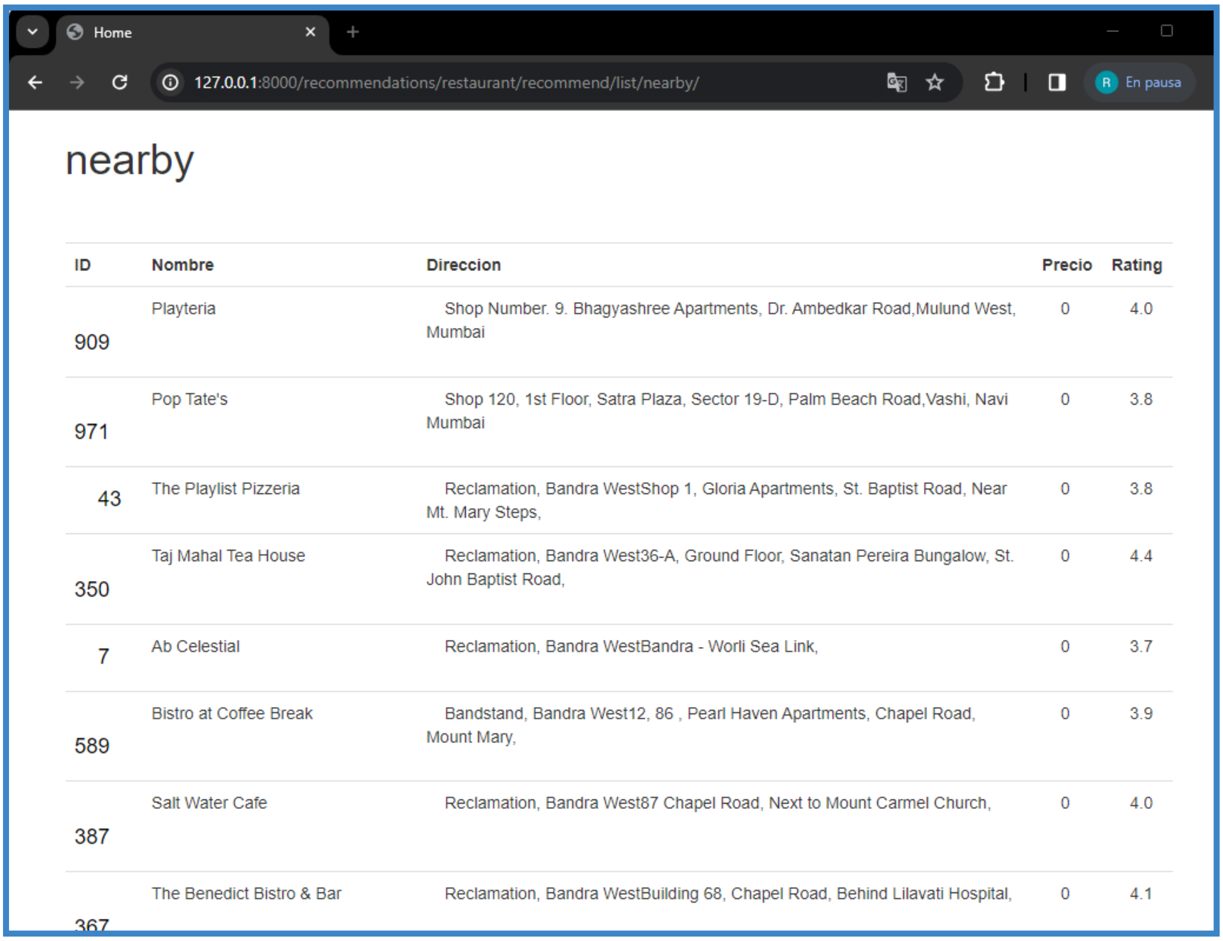
price based recommendation - Recomendación por precio

user personalized recommendation - Recomendación por preferencia de usuario

time based recommendation - Recomendación por hora

Paso 4:

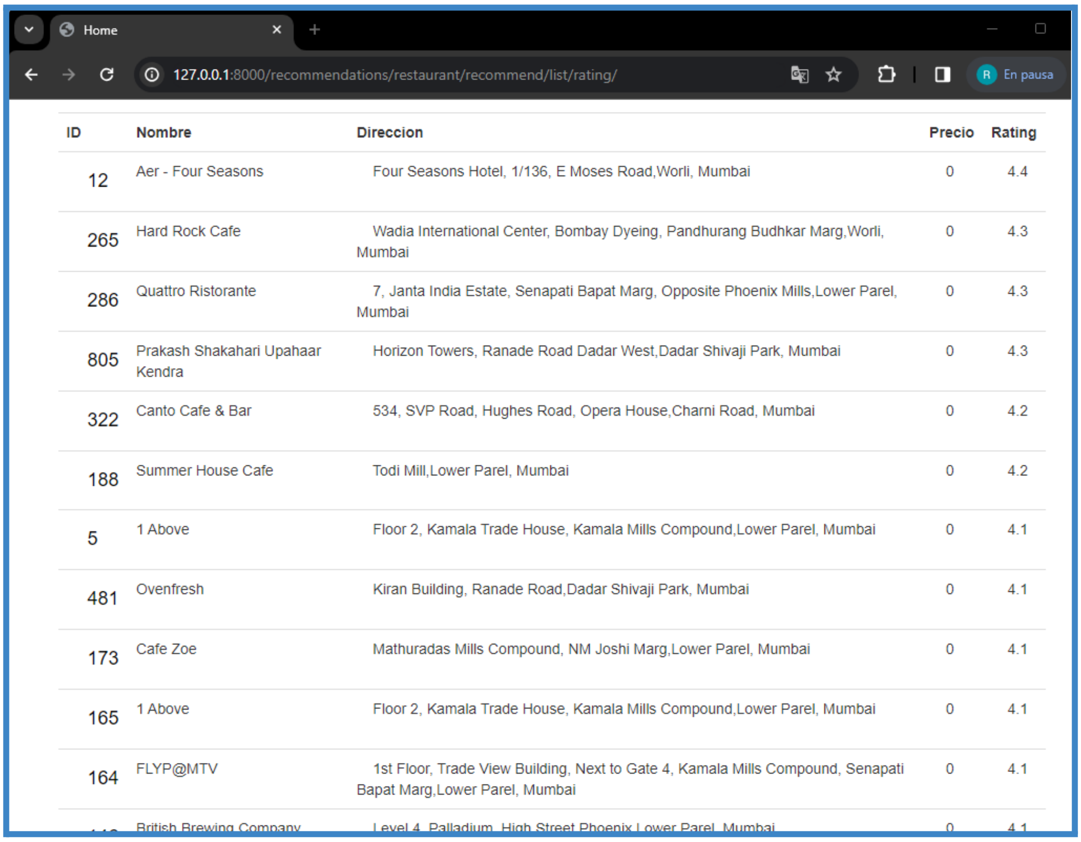
CERCANÍA


Home
127.0.0.1:8000/recommendations/restaurant/recommend/list/nearby/
R En pausa

nearby

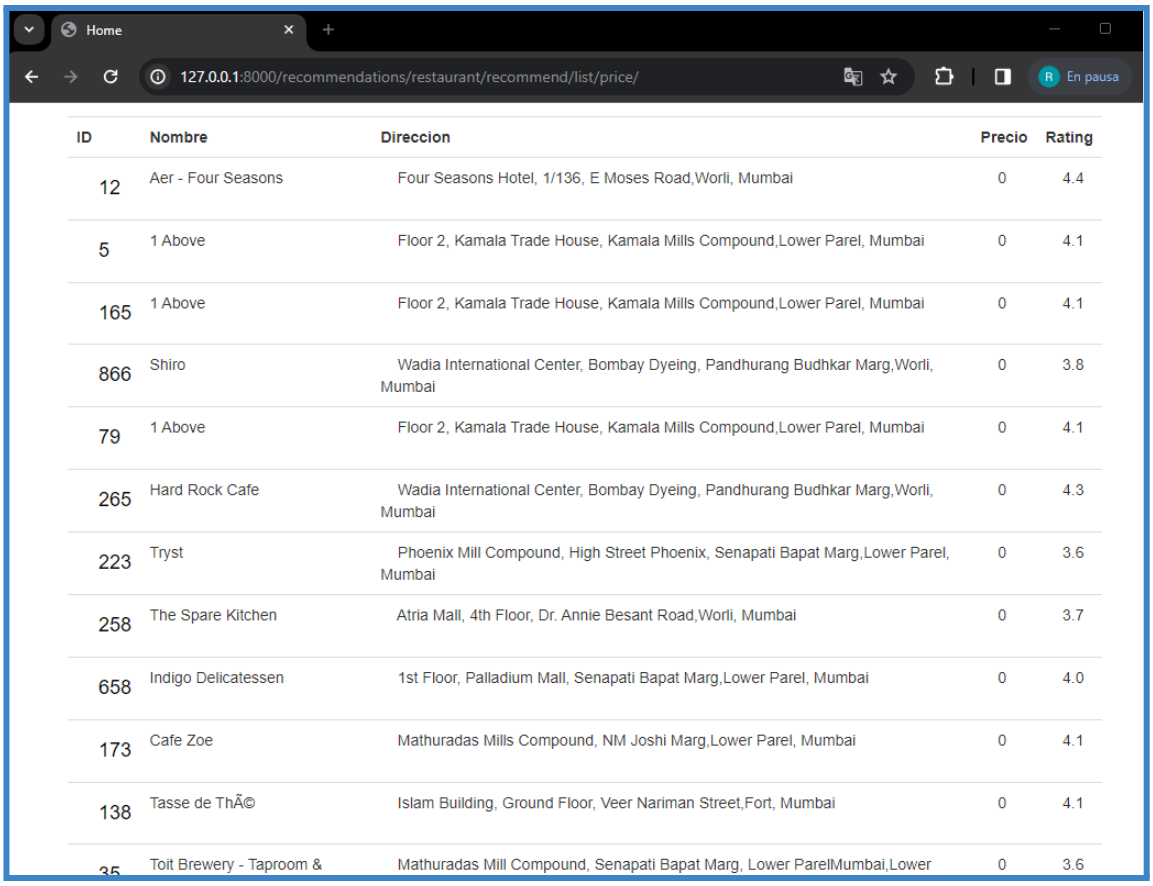
ID	Nombre	Direccion	Precio	Rating
909	Playteria	Shop Number. 9. Bhagyashree Apartments, Dr. Ambedkar Road,Mulund West, Mumbai	0	4.0
971	Pop Tate's	Shop 120, 1st Floor, Satra Plaza, Sector 19-D, Palm Beach Road,Vashi, Navi Mumbai	0	3.8
43	The Playlist Pizzeria	Reclamation, Bandra WestShop 1, Gloria Apartments, St. Baptist Road, Near Mt. Mary Steps,	0	3.8
350	Taj Mahal Tea House	Reclamation, Bandra West36-A, Ground Floor, Sanatan Pereira Bungalow, St. John Baptist Road,	0	4.4
7	Ab Celestial	Reclamation, Bandra WestBandra - Worli Sea Link,	0	3.7
589	Bistro at Coffee Break	Bandstand, Bandra West12, 86 , Pearl Haven Apartments, Chapel Road, Mount Mary,	0	3.9
387	Salt Water Cafe	Reclamation, Bandra West87 Chapel Road, Next to Mount Carmel Church,	0	4.0
367	The Benedict Bistro & Bar	Reclamation, Bandra WestBuilding 68, Chapel Road, Behind Lilavati Hospital,	0	4.1

VALORACIÓN


Home
127.0.0.1:8000/recommendations/restaurant/recommend/list/rating/
En pausa

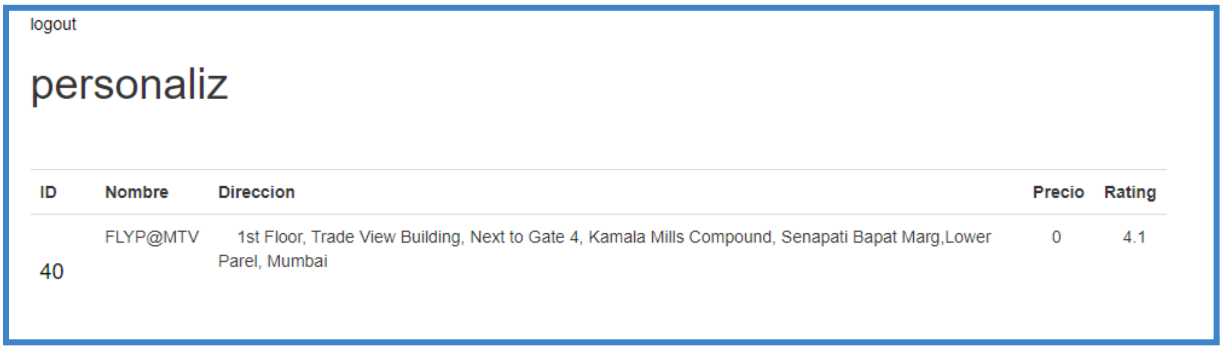
ID	Nombre	Direccion	Precio	Rating
12	Aer - Four Seasons	Four Seasons Hotel, 1/136, E Moses Road,Worli, Mumbai	0	4.4
265	Hard Rock Cafe	Wadia International Center, Bombay Dyeing, Pandhurang Budhkar Marg,Worli, Mumbai	0	4.3
286	Quattro Ristorante	7, Janta India Estate, Senapati Bapat Marg, Opposite Phoenix Mills,Lower Parel, Mumbai	0	4.3
805	Prakash Shakahari Upahaar Kendra	Horizon Towers, Ranade Road Dadar West,Dadar Shivaji Park, Mumbai	0	4.3
322	Canto Cafe & Bar	534, SVP Road, Hughes Road, Opera House,Charni Road, Mumbai	0	4.2
188	Summer House Cafe	Todi Mill,Lower Parel, Mumbai	0	4.2
5	1 Above	Floor 2, Kamala Trade House, Kamala Mills Compound,Lower Parel, Mumbai	0	4.1
481	Ovenfresh	Kiran Building, Ranade Road,Dadar Shivaji Park, Mumbai	0	4.1
173	Cafe Zoe	Mathuradas Mills Compound, NM Joshi Marg,Lower Parel, Mumbai	0	4.1
165	1 Above	Floor 2, Kamala Trade House, Kamala Mills Compound,Lower Parel, Mumbai	0	4.1
164	FLYP@MTV	1st Floor, Trade View Building, Next to Gate 4, Kamala Mills Compound, Senapati Bapat Marg,Lower Parel, Mumbai	0	4.1
[illegible]	British Brewing Company	Level 4, Palladium, High Street Phoenix,Lower Parel, Mumbai	0	4.1

PRECIO


Home
127.0.0.1:8000/recommendations/restaurant/recommend/list/price/
En pausa

ID	Nombre	Direccion	Precio	Rating
12	Aer - Four Seasons	Four Seasons Hotel, 1/136, E Moses Road,Worli, Mumbai	0	4.4
5	1 Above	Floor 2, Kamala Trade House, Kamala Mills Compound,Lower Parel, Mumbai	0	4.1
165	1 Above	Floor 2, Kamala Trade House, Kamala Mills Compound,Lower Parel, Mumbai	0	4.1
866	Shiro	Wadia International Center, Bombay Dyeing, Pandhurang Budhkar Marg,Worli, Mumbai	0	3.8
79	1 Above	Floor 2, Kamala Trade House, Kamala Mills Compound,Lower Parel, Mumbai	0	4.1
265	Hard Rock Cafe	Wadia International Center, Bombay Dyeing, Pandhurang Budhkar Marg,Worli, Mumbai	0	4.3
223	Tryst	Phoenix Mill Compound, High Street Phoenix, Senapati Bapat Marg,Lower Parel, Mumbai	0	3.6
258	The Spare Kitchen	Atria Mall, 4th Floor, Dr. Annie Besant Road,Worli, Mumbai	0	3.7
658	Indigo Delicatessen	1st Floor, Palladium Mall, Senapati Bapat Marg,Lower Parel, Mumbai	0	4.0
173	Cafe Zoe	Mathuradas Mills Compound, NM Joshi Marg,Lower Parel, Mumbai	0	4.1
138	Tasse de ThÃ©	Islam Building, Ground Floor, Veer Nariman Street,Fort, Mumbai	0	4.1
35	Toit Brewery - Taproom &	Mathuradas Mill Compound, Senapati Bapat Marg, Lower ParelMumbai,Lower	0	3.6

PERSONALIZADO



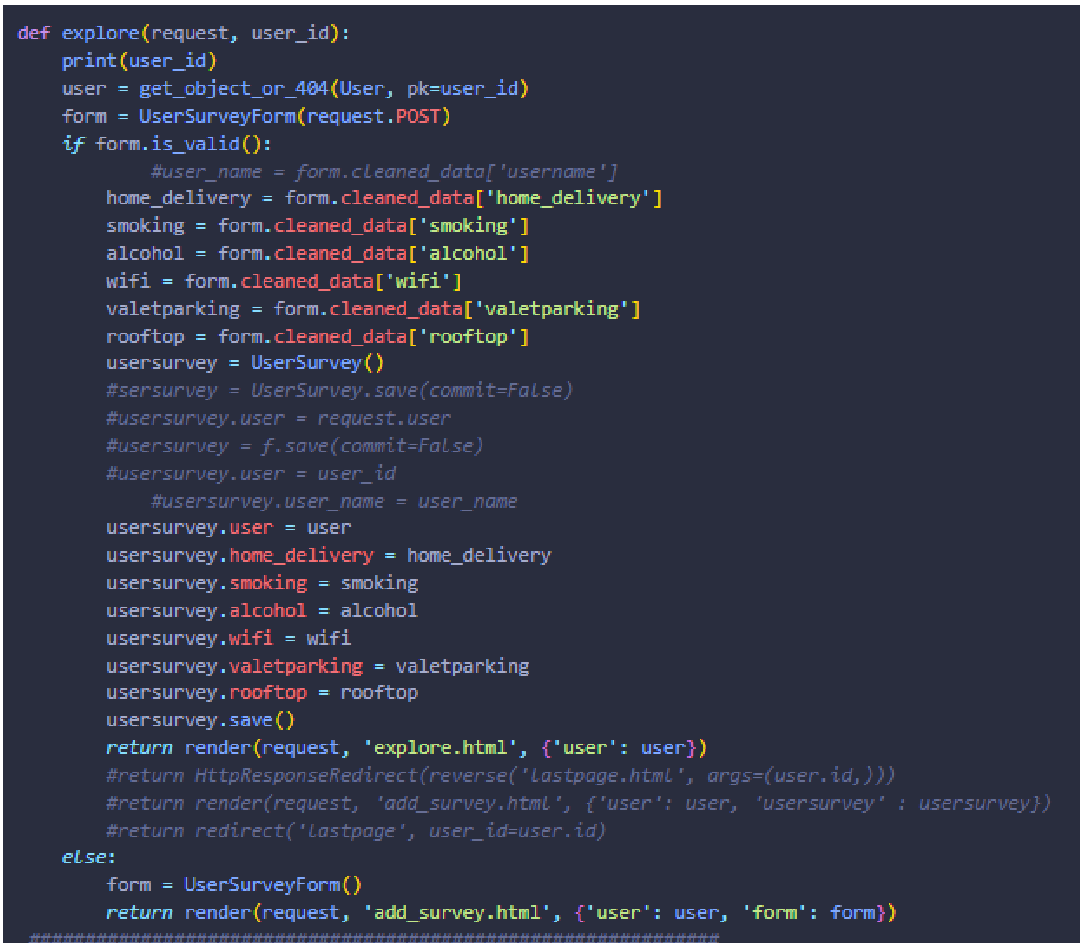
logout

personaliz

ID	Nombre	Direccion	Precio	Rating
40	FLYP@MTV	1st Floor, Trade View Building, Next to Gate 4, Kamala Mills Compound, Senapati Bapat Marg,Lower Parel, Mumbai	0	4.1

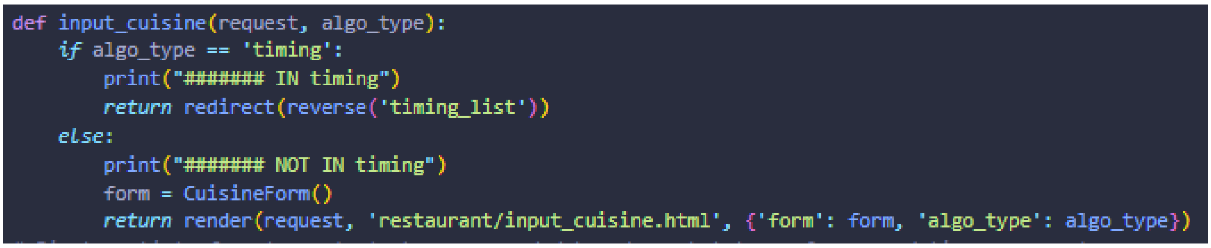
CÓDIGO:

PREFERENCIAS

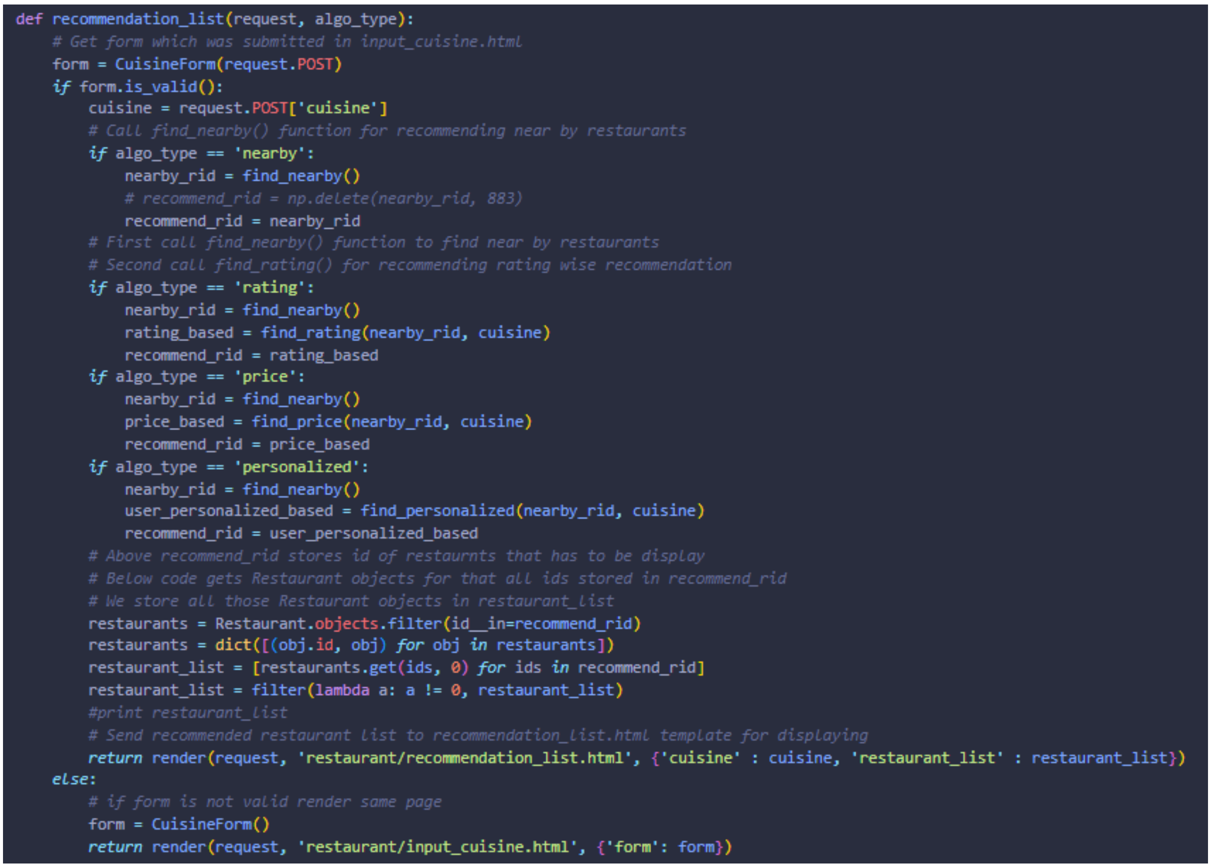


```
def explore(request, user_id):
    print(user_id)
    user = get_object_or_404(User, pk=user_id)
    form = UserSurveyForm(request.POST)
    if form.is_valid():
        #user_name = form.cleaned_data['username']
        home_delivery = form.cleaned_data['home_delivery']
        smoking = form.cleaned_data['smoking']
        alcohol = form.cleaned_data['alcohol']
        wifi = form.cleaned_data['wifi']
        valetparking = form.cleaned_data['valetparking']
        rooftop = form.cleaned_data['rooftop']
        usersurvey = UserSurvey()
        #sersurvey = UserSurvey.save(commit=False)
        #usersurvey.user = request.user
        #usersurvey = f.save(commit=False)
        #usersurvey.user = user_id
            #usersurvey.user_name = user_name
        usersurvey.user = user
        usersurvey.home_delivery = home_delivery
        usersurvey.smoking = smoking
        usersurvey.alcohol = alcohol
        usersurvey.wifi = wifi
        usersurvey.valetparking = valetparking
        usersurvey.rooftop = rooftop
        usersurvey.save()
        return render(request, 'explore.html', {'user': user})
        #return HttpResponseRedirect(reverse('lastpage.html', args=(user.id,)))
        #return render(request, 'add_survey.html', {'user': user, 'usersurvey' : usersurvey})
        #return redirect('lastpage', user_id=user.id)
    else:
        form = UserSurveyForm()
        return render(request, 'add_survey.html', {'user': user, 'form': form})
##############################################################
```

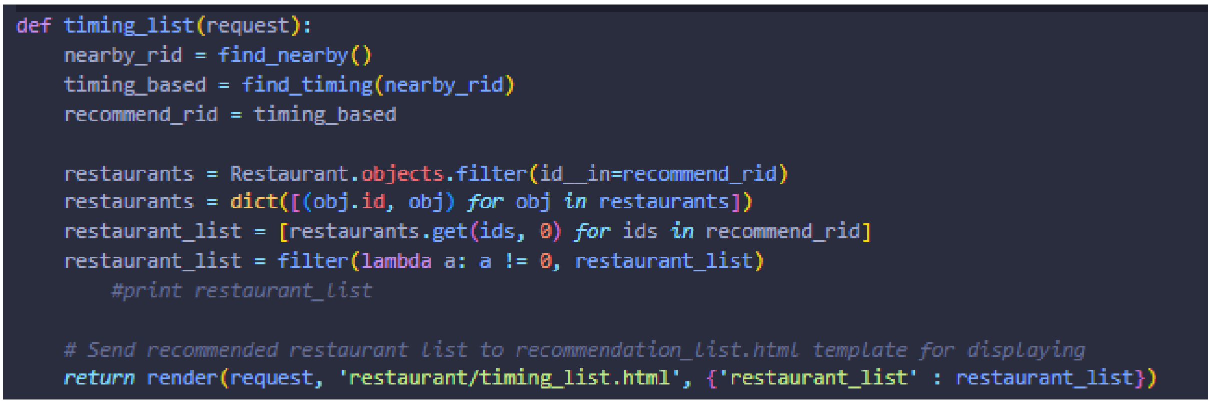
RECOMENDACIÓN



```
def input_cuisine(request, algo_type):
    if algo_type == 'timing':
        print("####### IN timing")
        return redirect(reverse('timing_list'))
    else:
        print("####### NOT IN timing")
        form = CuisineForm()
        return render(request, 'restaurant/input_cuisine.html', {'form': form, 'algo_type': algo_type})
```

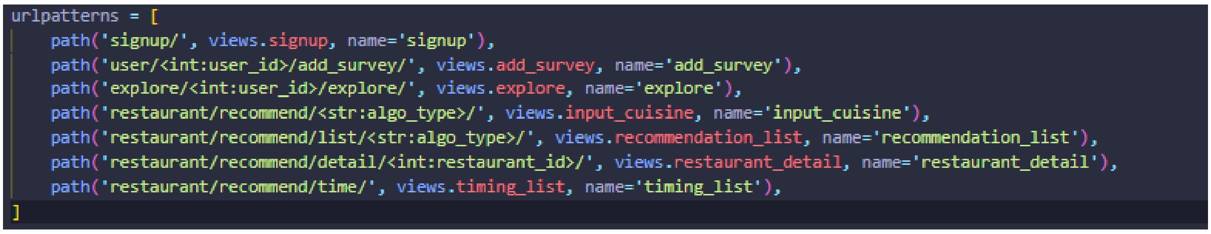


```python
def recommendation_list(request, algo_type):
    # Get form which was submitted in input_cuisine.html
    form = CuisineForm(request.POST)
    if form.is_valid():
        cuisine = request.POST['cuisine']
        # Call find_nearby() function for recommending near by restaurants
        if algo_type == 'nearby':
            nearby_rid = find_nearby()
            # recommend_rid = np.delete(nearby_rid, 883)
            recommend_rid = nearby_rid
        # First call find_nearby() function to find near by restaurants
        # Second call find_rating() for recommending rating wise recommendation
        if algo_type == 'rating':
            nearby_rid = find_nearby()
            rating_based = find_rating(nearby_rid, cuisine)
            recommend_rid = rating_based
        if algo_type == 'price':
            nearby_rid = find_nearby()
            price_based = find_price(nearby_rid, cuisine)
            recommend_rid = price_based
        if algo_type == 'personalized':
            nearby_rid = find_nearby()
            user_personalized_based = find_personalized(nearby_rid, cuisine)
            recommend_rid = user_personalized_based
        # Above recommend_rid stores id of restaurnts that has to be display
        # Below code gets Restaurant objects for that all ids stored in recommend_rid
        # We store all those Restaurant objects in restaurant_list
        restaurants = Restaurant.objects.filter(id__in=recommend_rid)
        restaurants = dict([(obj.id, obj) for obj in restaurants])
        restaurant_list = [restaurants.get(ids, 0) for ids in recommend_rid]
        restaurant_list = filter(lambda a: a != 0, restaurant_list)
        #print restaurant_list
        # Send recommended restaurant list to recommendation_list.html template for displaying
        return render(request, 'restaurant/recommendation_list.html', {'cuisine' : cuisine, 'restaurant_list' : restaurant_list})
    else:
        # if form is not valid render same page
        form = CuisineForm()
        return render(request, 'restaurant/input_cuisine.html', {'form': form})
```



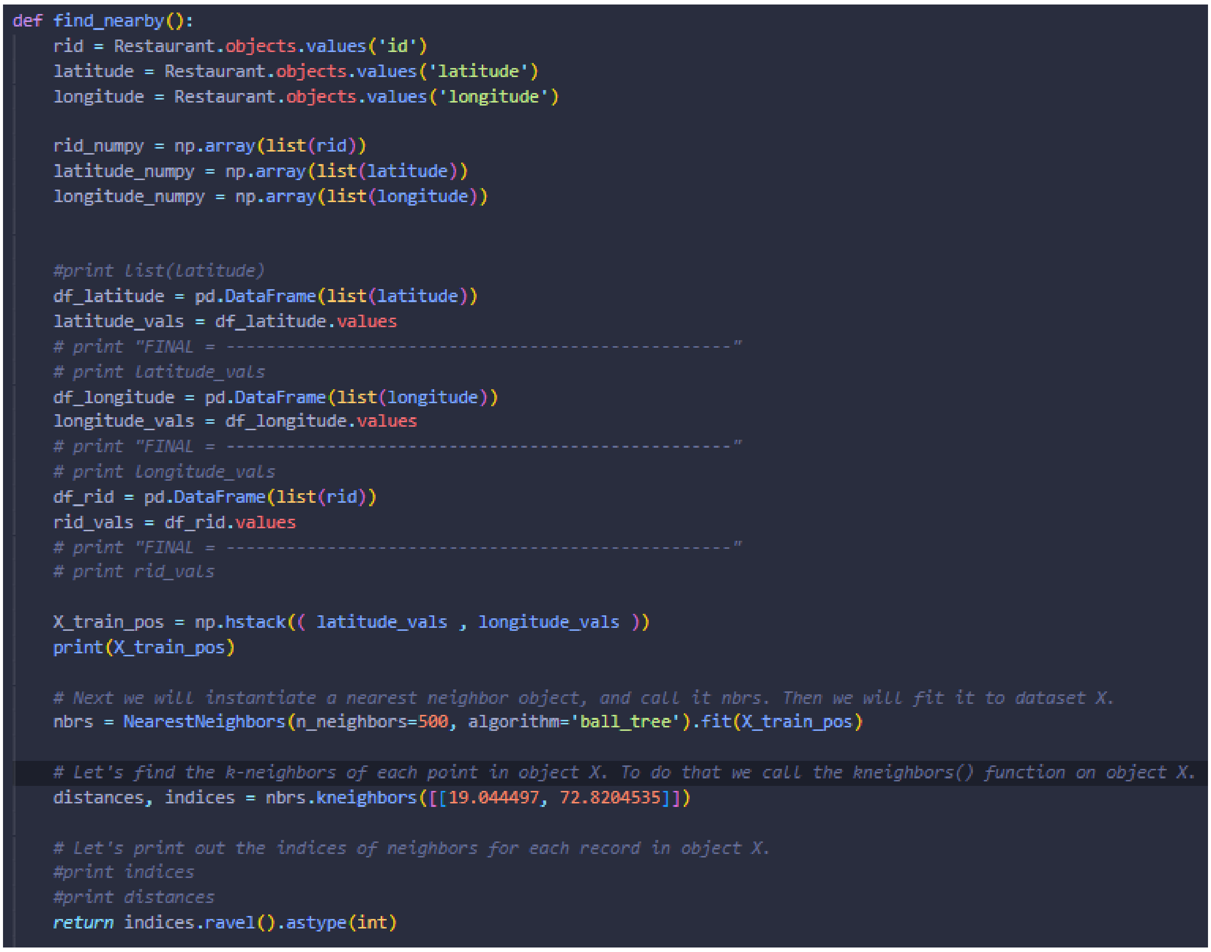
```python
def timing_list(request):
    nearby_rid = find_nearby()
    timing_based = find_timing(nearby_rid)
    recommend_rid = timing_based

    restaurants = Restaurant.objects.filter(id__in=recommend_rid)
    restaurants = dict([(obj.id, obj) for obj in restaurants])
    restaurant_list = [restaurants.get(ids, 0) for ids in recommend_rid]
    restaurant_list = filter(lambda a: a != 0, restaurant_list)
        #print restaurant_list

    # Send recommended restaurant list to recommendation_list.html template for displaying
    return render(request, 'restaurant/timing_list.html', {'restaurant_list' : restaurant_list})
```

RUTAS



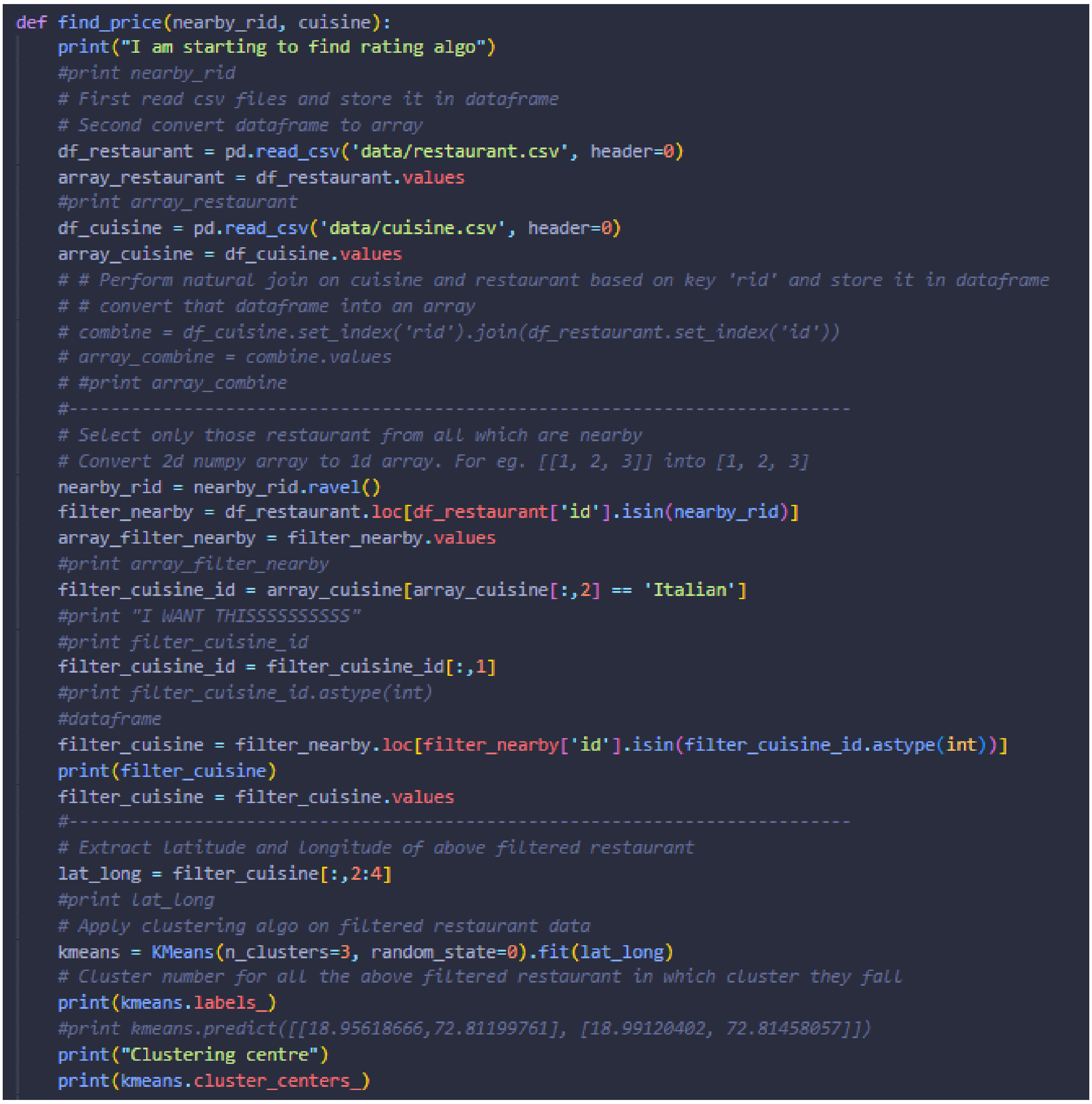
```python
urlpatterns = [
    path('signup/', views.signup, name='signup'),
    path('user/<int:user_id>/add_survey/', views.add_survey, name='add_survey'),
    path('explore/<int:user_id>/explore/', views.explore, name='explore'),
    path('restaurant/recommend/<str:algo_type>/', views.input_cuisine, name='input_cuisine'),
    path('restaurant/recommend/list/<str:algo_type>/', views.recommendation_list, name='recommendation_list'),
    path('restaurant/recommend/detail/<int:restaurant_id>/', views.restaurant_detail, name='restaurant_detail'),
    path('restaurant/recommend/time/', views.timing_list, name='timing_list'),
]
```

RECOMENDACIÓN POR CERCANÍA

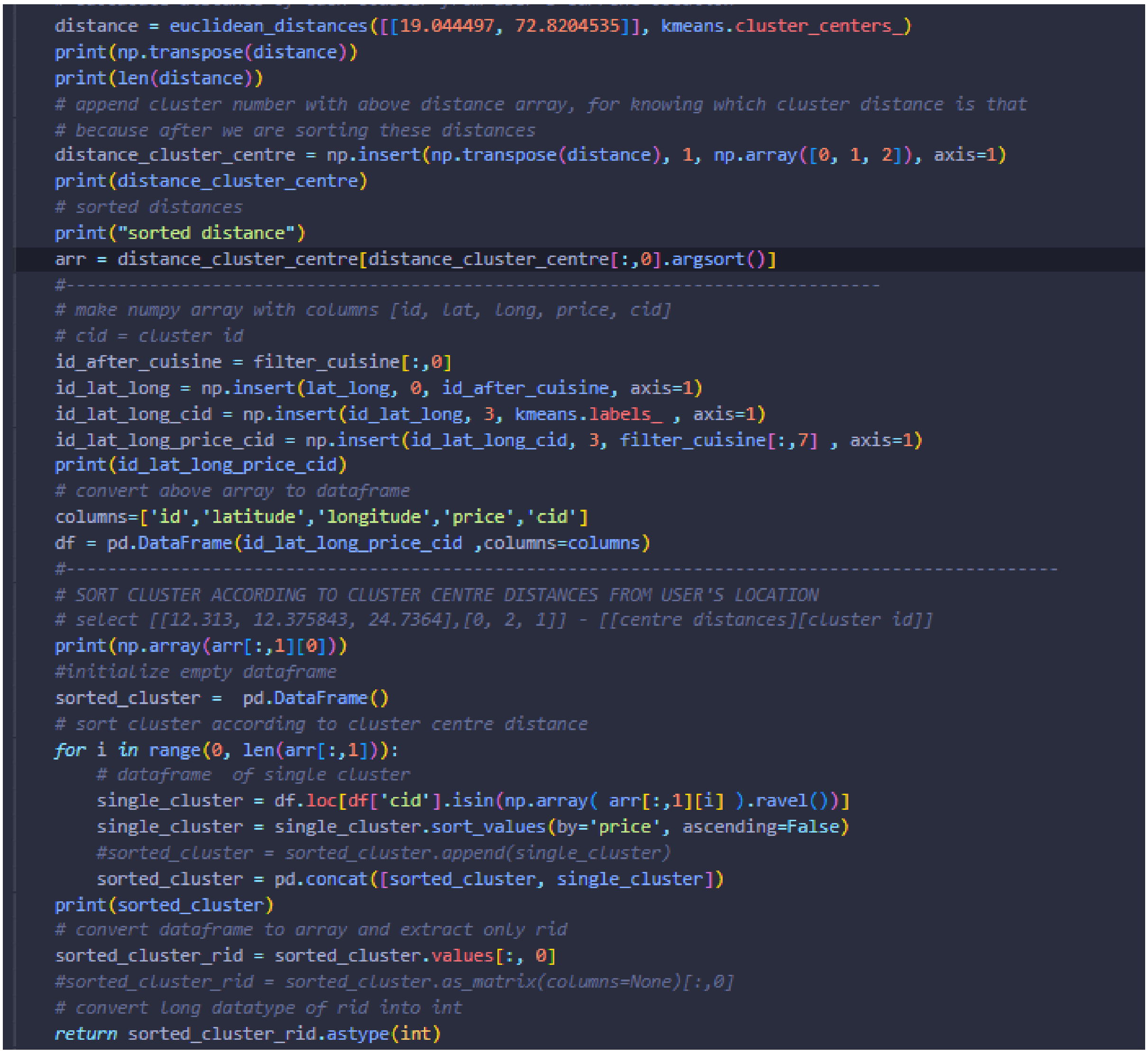


```
def find_nearby():
    rid = Restaurant.objects.values('id')
    latitude = Restaurant.objects.values('latitude')
    longitude = Restaurant.objects.values('longitude')

    rid_numpy = np.array(list(rid))
    latitude_numpy = np.array(list(latitude))
    longitude_numpy = np.array(list(longitude))


    #print list(latitude)
    df_latitude = pd.DataFrame(list(latitude))
    latitude_vals = df_latitude.values
    # print "FINAL = --------------------------------------------------"
    # print latitude_vals
    df_longitude = pd.DataFrame(list(longitude))
    longitude_vals = df_longitude.values
    # print "FINAL = --------------------------------------------------"
    # print longitude_vals
    df_rid = pd.DataFrame(list(rid))
    rid_vals = df_rid.values
    # print "FINAL = --------------------------------------------------"
    # print rid_vals

    X_train_pos = np.hstack(( latitude_vals , longitude_vals ))
    print(X_train_pos)

    # Next we will instantiate a nearest neighbor object, and call it nbrs. Then we will fit it to dataset X.
    nbrs = NearestNeighbors(n_neighbors=500, algorithm='ball_tree').fit(X_train_pos)

    # Let's find the k-neighbors of each point in object X. To do that we call the kneighbors() function on object X.
    distances, indices = nbrs.kneighbors([[19.044497, 72.8204535]])

    # Let's print out the indices of neighbors for each record in object X.
    #print indices
    #print distances
    return indices.ravel().astype(int)
```

RECOMENDACIÓN POR PRECIO

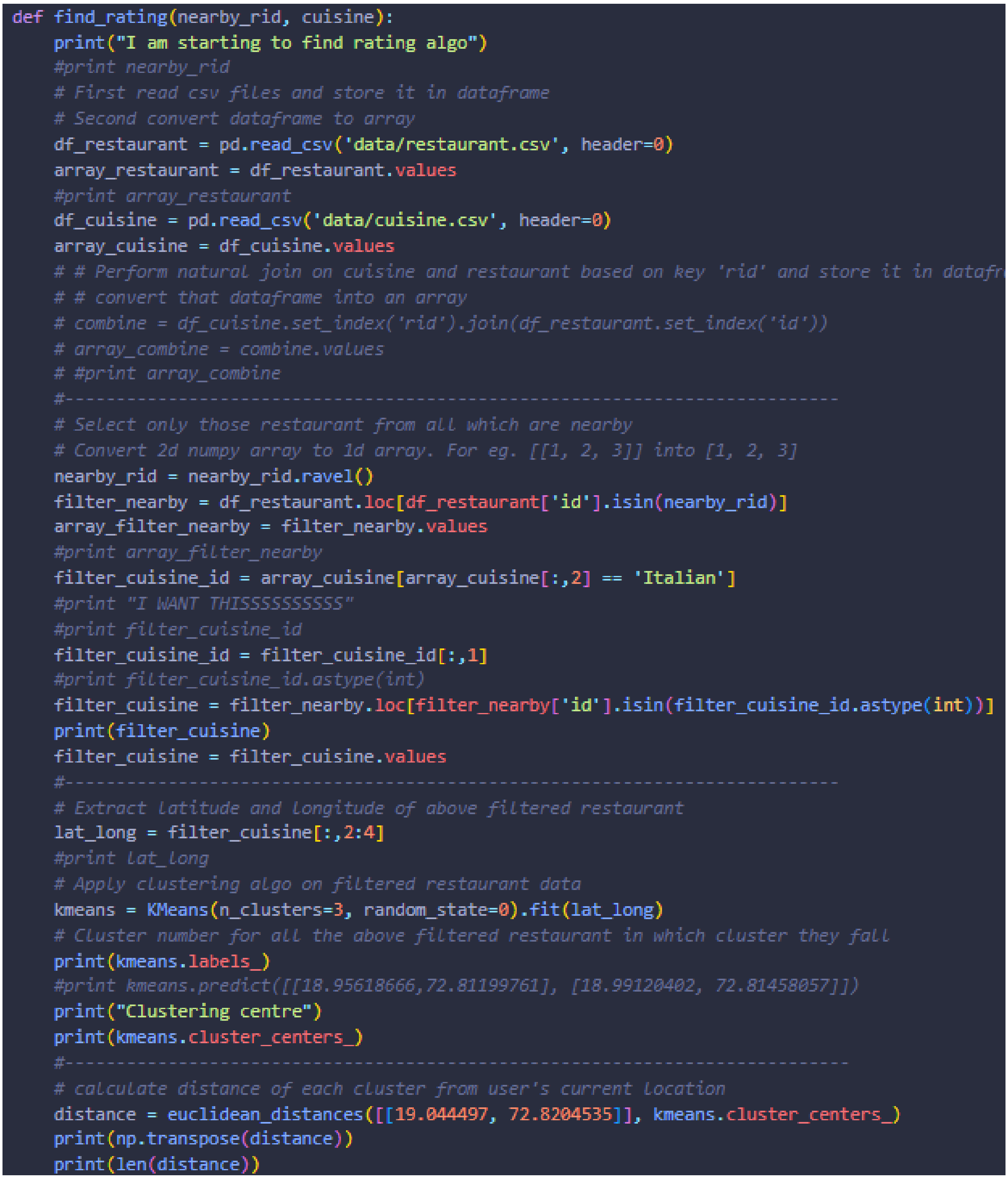


```python
def find_price(nearby_rid, cuisine):
    print("I am starting to find rating algo")
    #print nearby_rid
    # First read csv files and store it in dataframe
    # Second convert dataframe to array
    df_restaurant = pd.read_csv('data/restaurant.csv', header=0)
    array_restaurant = df_restaurant.values
    #print array_restaurant
    df_cuisine = pd.read_csv('data/cuisine.csv', header=0)
    array_cuisine = df_cuisine.values
    # # Perform natural join on cuisine and restaurant based on key 'rid' and store it in dataframe
    # # convert that dataframe into an array
    # combine = df_cuisine.set_index('rid').join(df_restaurant.set_index('id'))
    # array_combine = combine.values
    # #print array_combine
    #---------------------------------------------------------------------------
    # Select only those restaurant from all which are nearby
    # Convert 2d numpy array to 1d array. For eg. [[1, 2, 3]] into [1, 2, 3]
    nearby_rid = nearby_rid.ravel()
    filter_nearby = df_restaurant.loc[df_restaurant['id'].isin(nearby_rid)]
    array_filter_nearby = filter_nearby.values
    #print array_filter_nearby
    filter_cuisine_id = array_cuisine[array_cuisine[:,2] == 'Italian']
    #print "I WANT THISSSSSSSSS"
    #print filter_cuisine_id
    filter_cuisine_id = filter_cuisine_id[:,1]
    #print filter_cuisine_id.astype(int)
    #dataframe
    filter_cuisine = filter_nearby.loc[filter_nearby['id'].isin(filter_cuisine_id.astype(int))]
    print(filter_cuisine)
    filter_cuisine = filter_cuisine.values
    #---------------------------------------------------------------------------
    # Extract latitude and longitude of above filtered restaurant
    lat_long = filter_cuisine[:,2:4]
    #print lat_long
    # Apply clustering algo on filtered restaurant data
    kmeans = KMeans(n_clusters=3, random_state=0).fit(lat_long)
    # Cluster number for all the above filtered restaurant in which cluster they fall
    print(kmeans.labels_)
    #print kmeans.predict([[18.95618666,72.81199761], [18.99120402, 72.81458057]])
    print("Clustering centre")
    print(kmeans.cluster_centers_)
```

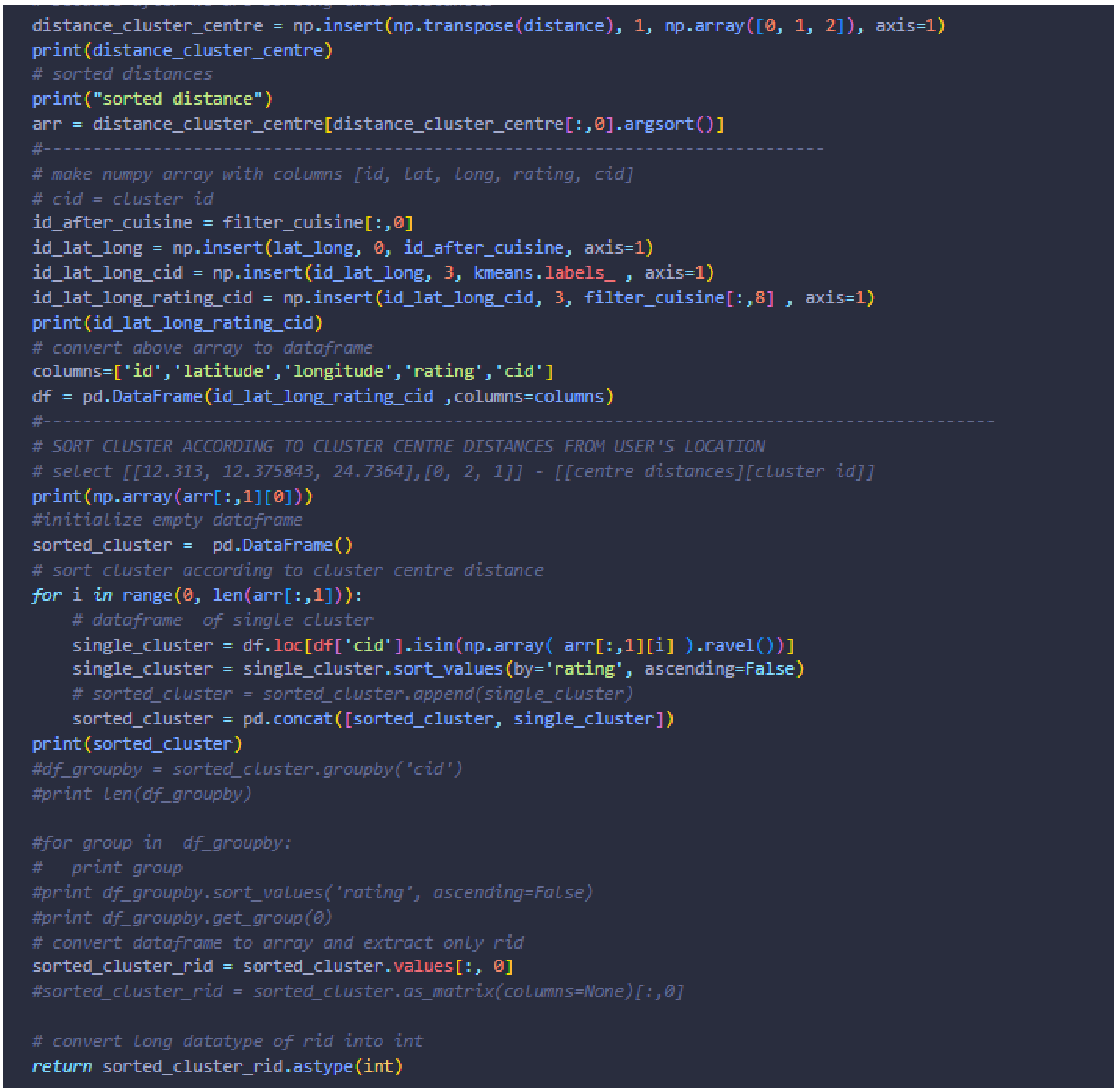


```python
    distance = euclidean_distances([[19.044497, 72.8204535]], kmeans.cluster_centers_)
    print(np.transpose(distance))
    print(len(distance))
    # append cluster number with above distance array, for knowing which cluster distance is that
    # because after we are sorting these distances
    distance_cluster_centre = np.insert(np.transpose(distance), 1, np.array([0, 1, 2]), axis=1)
    print(distance_cluster_centre)
    # sorted distances
    print("sorted distance")
    arr = distance_cluster_centre[distance_cluster_centre[:,0].argsort()]
    #---------------------------------------------------------------------------
    # make numpy array with columns [id, lat, long, price, cid]
    # cid = cluster id
    id_after_cuisine = filter_cuisine[:,0]
    id_lat_long = np.insert(lat_long, 0, id_after_cuisine, axis=1)
    id_lat_long_cid = np.insert(id_lat_long, 3, kmeans.labels_ , axis=1)
    id_lat_long_price_cid = np.insert(id_lat_long_cid, 3, filter_cuisine[:,7] , axis=1)
    print(id_lat_long_price_cid)
    # convert above array to dataframe
    columns=['id','latitude','longitude','price','cid']
    df = pd.DataFrame(id_lat_long_price_cid ,columns=columns)
    #-----------------------------------------------------------------------------------------------
    # SORT CLUSTER ACCORDING TO CLUSTER CENTRE DISTANCES FROM USER'S LOCATION
    # select [[12.313, 12.375843, 24.7364],[0, 2, 1]] - [[centre distances][cluster id]]
    print(np.array(arr[:,1][0]))
    #initialize empty dataframe
    sorted_cluster =  pd.DataFrame()
    # sort cluster according to cluster centre distance
    for i in range(0, len(arr[:,1])):
        # dataframe  of single cluster
        single_cluster = df.loc[df['cid'].isin(np.array( arr[:,1][i] ).ravel())]
        single_cluster = single_cluster.sort_values(by='price', ascending=False)
        #sorted_cluster = sorted_cluster.append(single_cluster)
        sorted_cluster = pd.concat([sorted_cluster, single_cluster])
    print(sorted_cluster)
    # convert dataframe to array and extract only rid
    sorted_cluster_rid = sorted_cluster.values[:, 0]
    #sorted_cluster_rid = sorted_cluster.as_matrix(columns=None)[:,0]
    # convert long datatype of rid into int
    return sorted_cluster_rid.astype(int)
```

RECOMENDACIÓN POR RATING

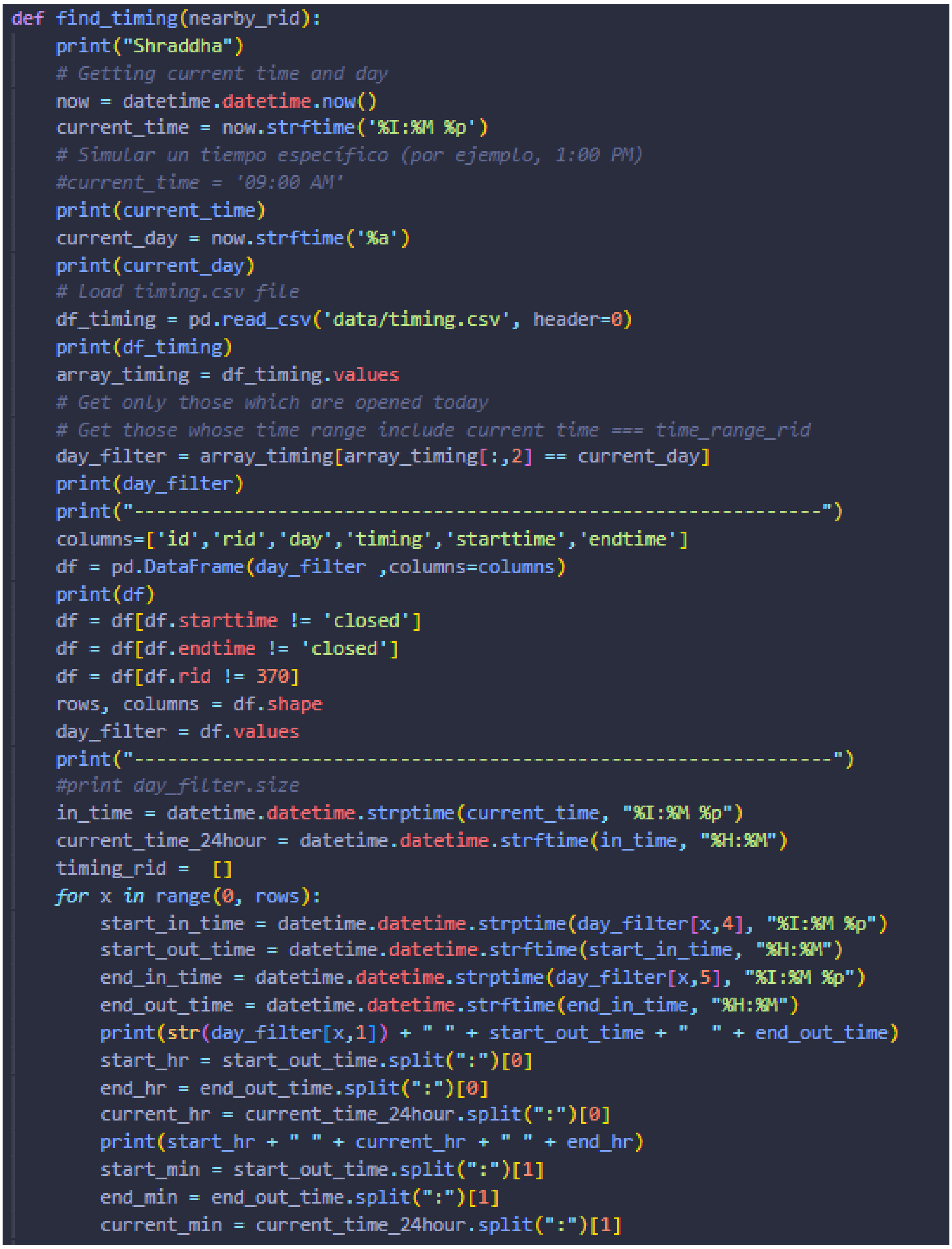


```python
def find_rating(nearby_rid, cuisine):
    print("I am starting to find rating algo")
    #print nearby_rid
    # First read csv files and store it in dataframe
    # Second convert dataframe to array
    df_restaurant = pd.read_csv('data/restaurant.csv', header=0)
    array_restaurant = df_restaurant.values
    #print array_restaurant
    df_cuisine = pd.read_csv('data/cuisine.csv', header=0)
    array_cuisine = df_cuisine.values
    # # Perform natural join on cuisine and restaurant based on key 'rid' and store it in datafr
    # # convert that dataframe into an array
    # combine = df_cuisine.set_index('rid').join(df_restaurant.set_index('id'))
    # array_combine = combine.values
    # #print array_combine
    #-----------------------------------------------------------------------------
    # Select only those restaurant from all which are nearby
    # Convert 2d numpy array to 1d array. For eg. [[1, 2, 3]] into [1, 2, 3]
    nearby_rid = nearby_rid.ravel()
    filter_nearby = df_restaurant.loc[df_restaurant['id'].isin(nearby_rid)]
    array_filter_nearby = filter_nearby.values
    #print array_filter_nearby
    filter_cuisine_id = array_cuisine[array_cuisine[:,2] == 'Italian']
    #print "I WANT THISSSSSSSSSS"
    #print filter_cuisine_id
    filter_cuisine_id = filter_cuisine_id[:,1]
    #print filter_cuisine_id.astype(int)
    filter_cuisine = filter_nearby.loc[filter_nearby['id'].isin(filter_cuisine_id.astype(int))]
    print(filter_cuisine)
    filter_cuisine = filter_cuisine.values
    #-----------------------------------------------------------------------------
    # Extract latitude and longitude of above filtered restaurant
    lat_long = filter_cuisine[:,2:4]
    #print lat_long
    # Apply clustering algo on filtered restaurant data
    kmeans = KMeans(n_clusters=3, random_state=0).fit(lat_long)
    # Cluster number for all the above filtered restaurant in which cluster they fall
    print(kmeans.labels_)
    #print kmeans.predict([[18.95618666,72.81199761], [18.99120402, 72.81458057]])
    print("Clustering centre")
    print(kmeans.cluster_centers_)
    #------------------------------------------------------------------------------
    # calculate distance of each cluster from user's current location
    distance = euclidean_distances([[19.044497, 72.8204535]], kmeans.cluster_centers_)
    print(np.transpose(distance))
    print(len(distance))
```



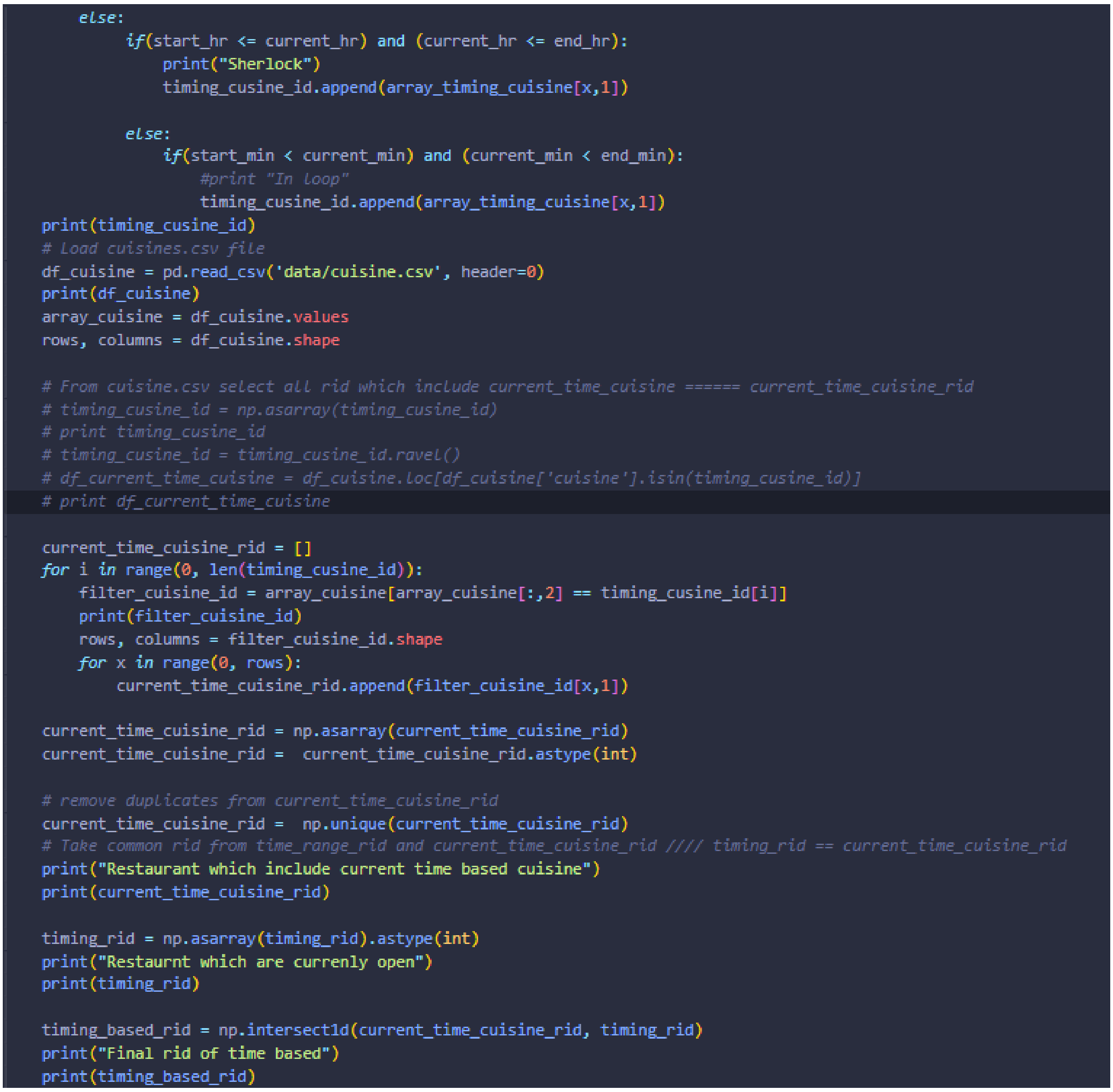
```
distance_cluster_centre = np.insert(np.transpose(distance), 1, np.array([0, 1, 2]), axis=1)
print(distance_cluster_centre)
# sorted distances
print("sorted distance")
arr = distance_cluster_centre[distance_cluster_centre[:,0].argsort()]
#------------------------------------------------------------------------------
# make numpy array with columns [id, lat, long, rating, cid]
# cid = cluster id
id_after_cuisine = filter_cuisine[:,0]
id_lat_long = np.insert(lat_long, 0, id_after_cuisine, axis=1)
id_lat_long_cid = np.insert(id_lat_long, 3, kmeans.labels_ , axis=1)
id_lat_long_rating_cid = np.insert(id_lat_long_cid, 3, filter_cuisine[:,8] , axis=1)
print(id_lat_long_rating_cid)
# convert above array to dataframe
columns=['id','latitude','longitude','rating','cid']
df = pd.DataFrame(id_lat_long_rating_cid ,columns=columns)
#-----------------------------------------------------------------------------------------------
# SORT CLUSTER ACCORDING TO CLUSTER CENTRE DISTANCES FROM USER'S LOCATION
# select [[12.313, 12.375843, 24.7364],[0, 2, 1]] - [[centre distances][cluster id]]
print(np.array(arr[:,1][0]))
#initialize empty dataframe
sorted_cluster =  pd.DataFrame()
# sort cluster according to cluster centre distance
for i in range(0, len(arr[:,1])):
    # dataframe  of single cluster
    single_cluster = df.loc[df['cid'].isin(np.array( arr[:,1][i] ).ravel())]
    single_cluster = single_cluster.sort_values(by='rating', ascending=False)
    # sorted_cluster = sorted_cluster.append(single_cluster)
    sorted_cluster = pd.concat([sorted_cluster, single_cluster])
print(sorted_cluster)
#df_groupby = sorted_cluster.groupby('cid')
#print len(df_groupby)

#for group in  df_groupby:
#   print group
#print df_groupby.sort_values('rating', ascending=False)
#print df_groupby.get_group(0)
# convert dataframe to array and extract only rid
sorted_cluster_rid = sorted_cluster.values[:, 0]
#sorted_cluster_rid = sorted_cluster.as_matrix(columns=None)[:,0]

# convert long datatype of rid into int
return sorted_cluster_rid.astype(int)
```

RECOMENDACIÓN POR HORA DE DISPONIBILIDAD Y HORA ACTUAL

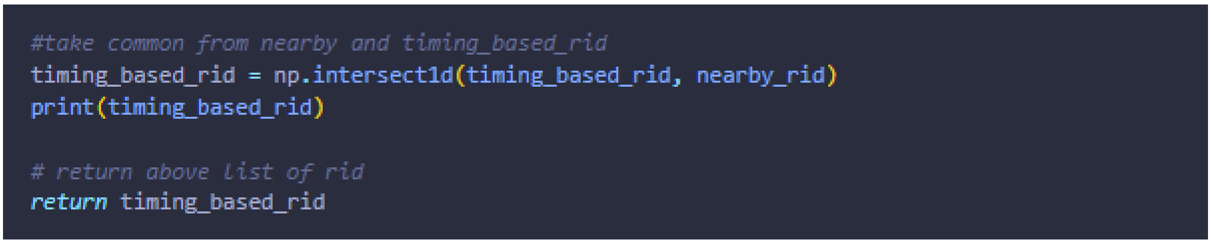


```python
def find_timing(nearby_rid):
    print("Shraddha")
    # Getting current time and day
    now = datetime.datetime.now()
    current_time = now.strftime('%I:%M %p')
    # Simular un tiempo específico (por ejemplo, 1:00 PM)
    #current_time = '09:00 AM'
    print(current_time)
    current_day = now.strftime('%a')
    print(current_day)
    # Load timing.csv file
    df_timing = pd.read_csv('data/timing.csv', header=0)
    print(df_timing)
    array_timing = df_timing.values
    # Get only those which are opened today
    # Get those whose time range include current time === time_range_rid
    day_filter = array_timing[array_timing[:,2] == current_day]
    print(day_filter)
    print("--------------------------------------------------------------")
    columns=['id','rid','day','timing','starttime','endtime']
    df = pd.DataFrame(day_filter ,columns=columns)
    print(df)
    df = df[df.starttime != 'closed']
    df = df[df.endtime != 'closed']
    df = df[df.rid != 370]
    rows, columns = df.shape
    day_filter = df.values
    print("---------------------------------------------------------------")
    #print day_filter.size
    in_time = datetime.datetime.strptime(current_time, "%I:%M %p")
    current_time_24hour = datetime.datetime.strftime(in_time, "%H:%M")
    timing_rid =  []
    for x in range(0, rows):
        start_in_time = datetime.datetime.strptime(day_filter[x,4], "%I:%M %p")
        start_out_time = datetime.datetime.strftime(start_in_time, "%H:%M")
        end_in_time = datetime.datetime.strptime(day_filter[x,5], "%I:%M %p")
        end_out_time = datetime.datetime.strftime(end_in_time, "%H:%M")
        print(str(day_filter[x,1]) + " " + start_out_time + "  " + end_out_time)
        start_hr = start_out_time.split(":")[0]
        end_hr = end_out_time.split(":")[0]
        current_hr = current_time_24hour.split(":")[0]
        print(start_hr + " " + current_hr + " " + end_hr)
        start_min = start_out_time.split(":")[1]
        end_min = end_out_time.split(":")[1]
        current_min = current_time_24hour.split(":")[1]
```

```
    if(start_hr > end_hr):
         #print("SHRADDHA")
         if((start_hr < current_hr) and (start_hr < "23")) or ((end_hr > current_hr) and (end_hr > "00")):
            #print("IN THIS LOOP")
            timing_rid.append(day_filter[x,1])
         else:
             if(start_min < current_min) and (current_min < end_min):
                 #print("In loop")
                 timing_rid.append(day_filter[x,1])
    else:
         if(start_hr < current_hr) and (current_hr < end_hr):
             #print("Sherlock")
             timing_rid.append(day_filter[x,1])
         else:
             if(start_min < current_min) and (current_min < end_min):
                 #print("In loop")
                 timing_rid.append(day_filter[x,1])
print(timing_rid)
# Load timing_cuisine.csv file
df_timing_cuisine = pd.read_csv('data/timing_cuisine.csv', header=0)
print(df_timing_cuisine)
array_timing_cuisine = df_timing_cuisine.values
rows, columns = df_timing_cuisine.shape

# Get all names of cuisines which comes under current_time ===== current_time_cuisine
timing_cusine_id =  []
for x in range(0, rows):
    start_in_time = datetime.datetime.strptime(array_timing_cuisine[x,2], "%I:%M %p")
    start_out_time = datetime.datetime.strftime(start_in_time, "%H:%M")
    end_in_time = datetime.datetime.strptime(array_timing_cuisine[x,3], "%I:%M %p")
    end_out_time = datetime.datetime.strftime(end_in_time, "%H:%M")
    print(start_out_time + "  " + end_out_time)
    start_hr = start_out_time.split(":")[0]
    end_hr = end_out_time.split(":")[0]
    current_hr = current_time_24hour.split(":")[0]
    print(array_timing_cuisine[x,1] + " " + start_hr + " " + current_hr + " " + end_hr)
    start_min = start_out_time.split(":")[1]
    end_min = end_out_time.split(":")[1]
    current_min = current_time_24hour.split(":")[1]
    if(start_hr >= end_hr):
         #print "SHRADDHA"
         if((start_hr <= current_hr) and (start_hr <= "23")) or ((end_hr > current_hr) and (end_hr >= "00")):
            #print "IN THIS LOOP"
            timing_cusine_id.append(array_timing_cuisine[x,1])
         else:
             if(start_min < current_min) and (current_min < end_min):
                 #print "In loop"
                 timing_cusine_id.append(array_timing_cuisine[x,1])
```

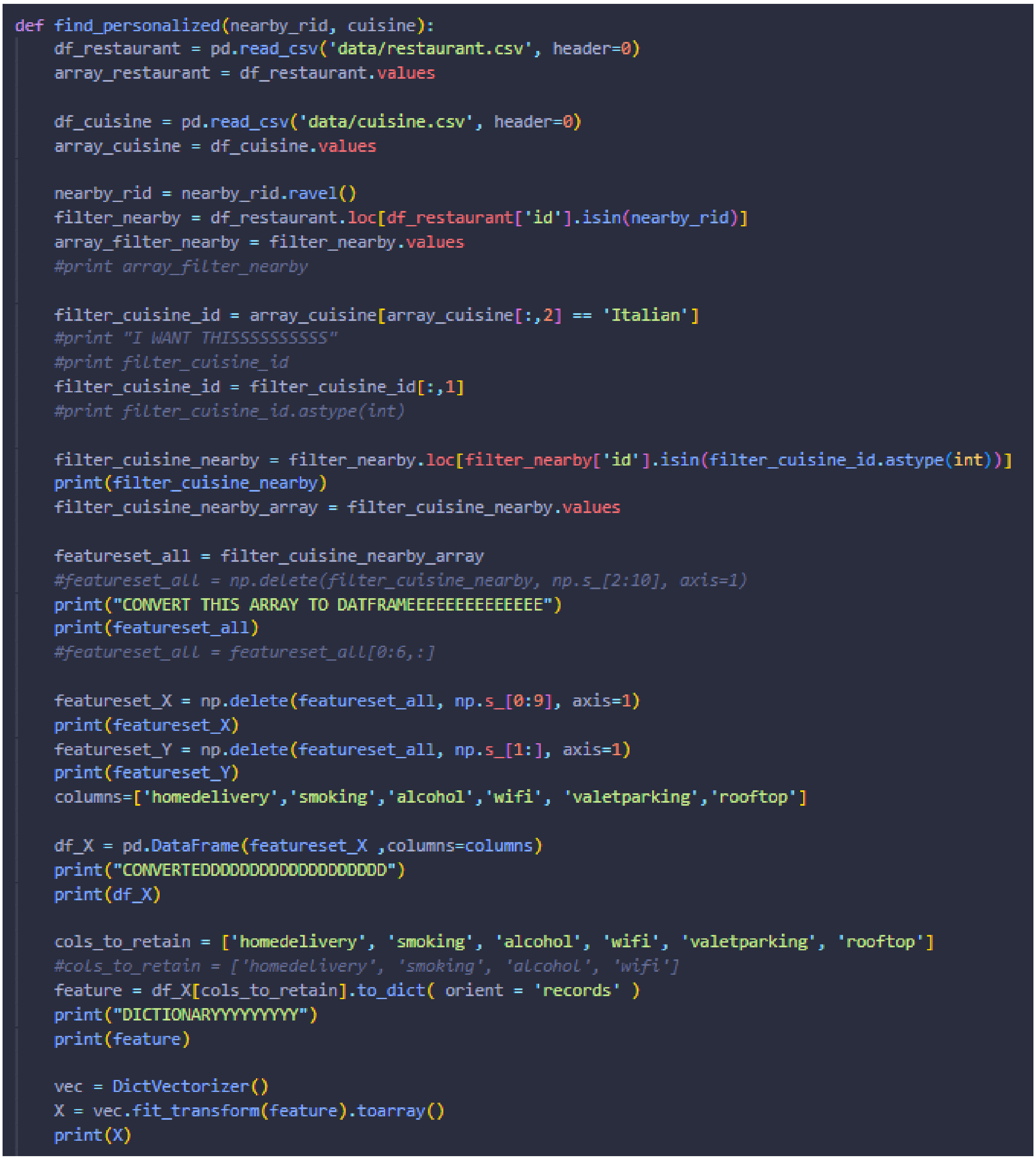


```python
        else:
             if(start_hr <= current_hr) and (current_hr <= end_hr):
                 print("Sherlock")
                 timing_cusine_id.append(array_timing_cuisine[x,1])

             else:
                 if(start_min < current_min) and (current_min < end_min):
                     #print "In loop"
                     timing_cusine_id.append(array_timing_cuisine[x,1])
    print(timing_cusine_id)
    # Load cuisines.csv file
    df_cuisine = pd.read_csv('data/cuisine.csv', header=0)
    print(df_cuisine)
    array_cuisine = df_cuisine.values
    rows, columns = df_cuisine.shape

    # From cuisine.csv select all rid which include current_time_cuisine ====== current_time_cuisine_rid
    # timing_cusine_id = np.asarray(timing_cusine_id)
    # print timing_cusine_id
    # timing_cusine_id = timing_cusine_id.ravel()
    # df_current_time_cuisine = df_cuisine.loc[df_cuisine['cuisine'].isin(timing_cusine_id)]
    # print df_current_time_cuisine

    current_time_cuisine_rid = []
    for i in range(0, len(timing_cusine_id)):
        filter_cuisine_id = array_cuisine[array_cuisine[:,2] == timing_cusine_id[i]]
        print(filter_cuisine_id)
        rows, columns = filter_cuisine_id.shape
        for x in range(0, rows):
            current_time_cuisine_rid.append(filter_cuisine_id[x,1])

    current_time_cuisine_rid = np.asarray(current_time_cuisine_rid)
    current_time_cuisine_rid =  current_time_cuisine_rid.astype(int)

    # remove duplicates from current_time_cuisine_rid
    current_time_cuisine_rid =  np.unique(current_time_cuisine_rid)
    # Take common rid from time_range_rid and current_time_cuisine_rid //// timing_rid == current_time_cuisine_rid
    print("Restaurant which include current time based cuisine")
    print(current_time_cuisine_rid)

    timing_rid = np.asarray(timing_rid).astype(int)
    print("Restaurnt which are currenly open")
    print(timing_rid)

    timing_based_rid = np.intersect1d(current_time_cuisine_rid, timing_rid)
    print("Final rid of time based")
    print(timing_based_rid)
```

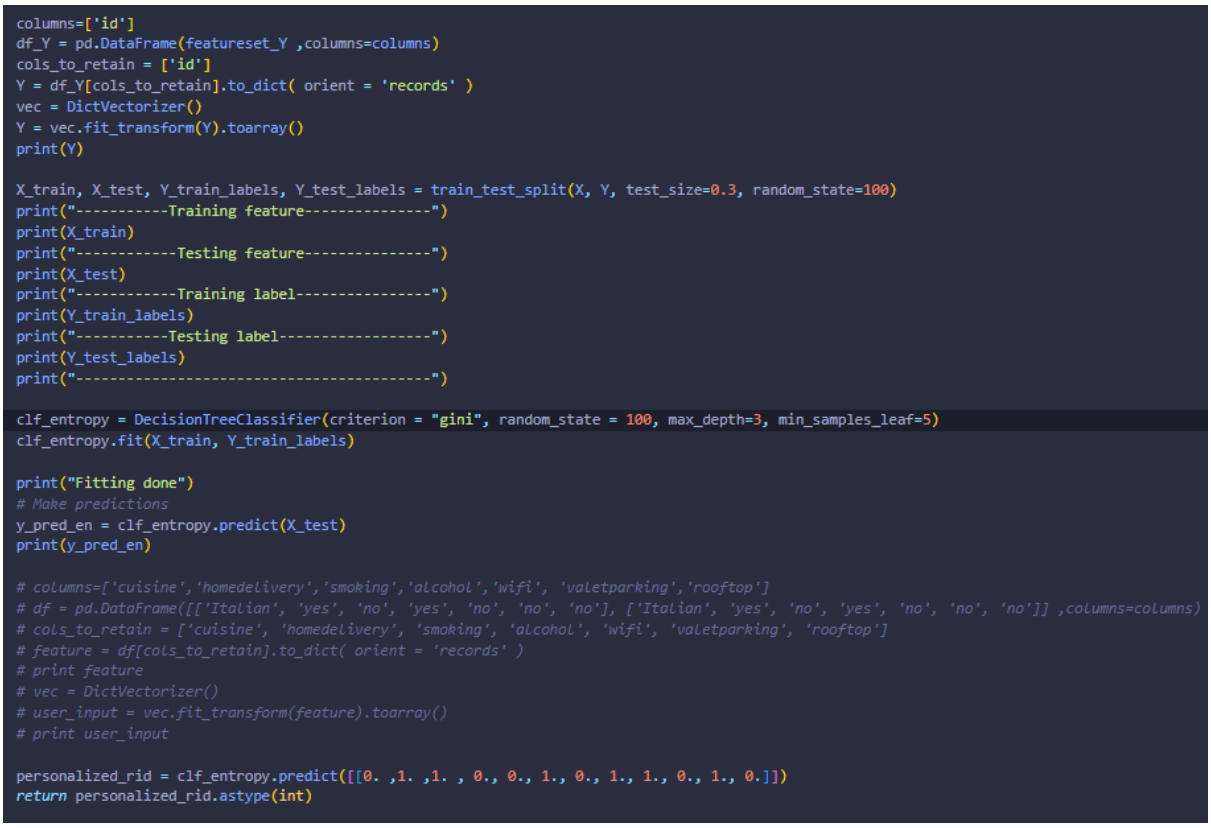


```python
#take common from nearby and timing_based_rid
timing_based_rid = np.intersect1d(timing_based_rid, nearby_rid)
print(timing_based_rid)

# return above list of rid
return timing_based_rid
```

RECOMENDACIÓN PERSONALIZADA

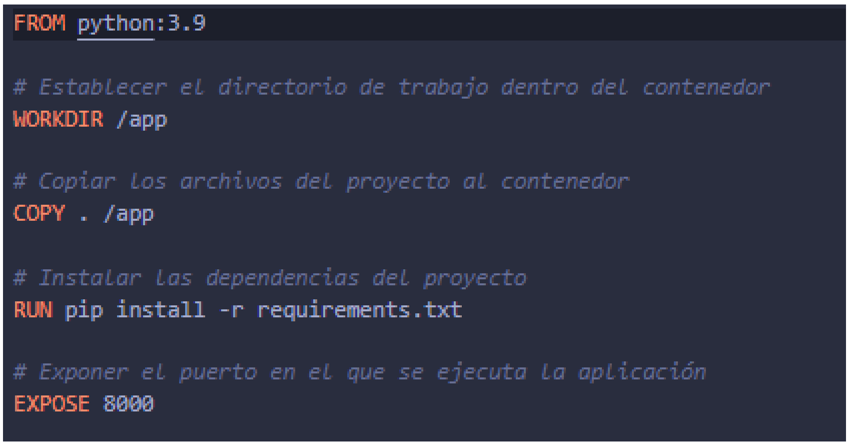


```
def find_personalized(nearby_rid, cuisine):
    df_restaurant = pd.read_csv('data/restaurant.csv', header=0)
    array_restaurant = df_restaurant.values

    df_cuisine = pd.read_csv('data/cuisine.csv', header=0)
    array_cuisine = df_cuisine.values

    nearby_rid = nearby_rid.ravel()
    filter_nearby = df_restaurant.loc[df_restaurant['id'].isin(nearby_rid)]
    array_filter_nearby = filter_nearby.values
    #print array_filter_nearby

    filter_cuisine_id = array_cuisine[array_cuisine[:,2] == 'Italian']
    #print "I WANT THISSSSSSSSSSS"
    #print filter_cuisine_id
    filter_cuisine_id = filter_cuisine_id[:,1]
    #print filter_cuisine_id.astype(int)

    filter_cuisine_nearby = filter_nearby.loc[filter_nearby['id'].isin(filter_cuisine_id.astype(int))]
    print(filter_cuisine_nearby)
    filter_cuisine_nearby_array = filter_cuisine_nearby.values

    featureset_all = filter_cuisine_nearby_array
    #featureset_all = np.delete(filter_cuisine_nearby, np.s_[2:10], axis=1)
    print("CONVERT THIS ARRAY TO DATFRAMEEEEEEEEEEEEEEE")
    print(featureset_all)
    #featureset_all = featureset_all[0:6,:]

    featureset_X = np.delete(featureset_all, np.s_[0:9], axis=1)
    print(featureset_X)
    featureset_Y = np.delete(featureset_all, np.s_[1:], axis=1)
    print(featureset_Y)
    columns=['homedelivery','smoking','alcohol','wifi', 'valetparking','rooftop']

    df_X = pd.DataFrame(featureset_X ,columns=columns)
    print("CONVERTEDDDDDDDDDDDDDDDDDDDD")
    print(df_X)

    cols_to_retain = ['homedelivery', 'smoking', 'alcohol', 'wifi', 'valetparking', 'rooftop']
    #cols_to_retain = ['homedelivery', 'smoking', 'alcohol', 'wifi']
    feature = df_X[cols_to_retain].to_dict( orient = 'records' )
    print("DICTIONARYYYYYYYYY")
    print(feature)

    vec = DictVectorizer()
    X = vec.fit_transform(feature).toarray()
    print(X)
```



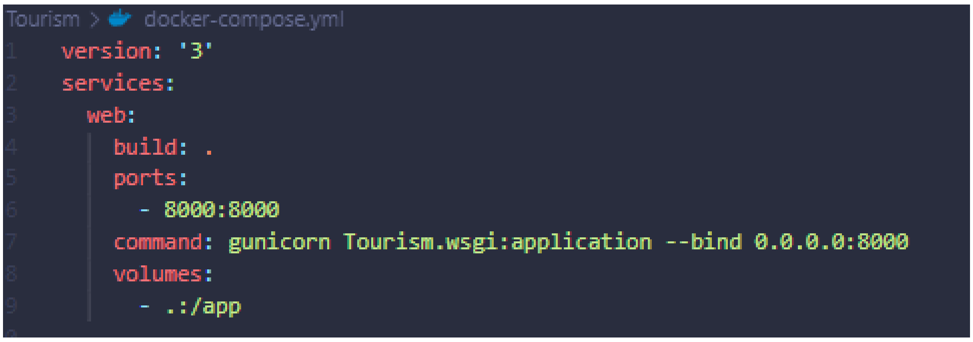
```python
columns=['id']
df_Y = pd.DataFrame(featureset_Y ,columns=columns)
cols_to_retain = ['id']
Y = df_Y[cols_to_retain].to_dict( orient = 'records' )
vec = DictVectorizer()
Y = vec.fit_transform(Y).toarray()
print(Y)

X_train, X_test, Y_train_labels, Y_test_labels = train_test_split(X, Y, test_size=0.3, random_state=100)
print("-----------Training feature---------------")
print(X_train)
print("------------Testing feature---------------")
print(X_test)
print("------------Training label----------------")
print(Y_train_labels)
print("-----------Testing label------------------")
print(Y_test_labels)
print("-------------------------------------------")

clf_entropy = DecisionTreeClassifier(criterion = "gini", random_state = 100, max_depth=3, min_samples_leaf=5)
clf_entropy.fit(X_train, Y_train_labels)

print("Fitting done")
# Make predictions
y_pred_en = clf_entropy.predict(X_test)
print(y_pred_en)

# columns=['cuisine','homedelivery','smoking','alcohol','wifi', 'valetparking','rooftop']
# df = pd.DataFrame([['Italian', 'yes', 'no', 'yes', 'no', 'no', 'no'], ['Italian', 'yes', 'no', 'yes', 'no', 'no', 'no']] ,columns=columns)
# cols_to_retain = ['cuisine', 'homedelivery', 'smoking', 'alcohol', 'wifi', 'valetparking', 'rooftop']
# feature = df[cols_to_retain].to_dict( orient = 'records' )
# print feature
# vec = DictVectorizer()
# user_input = vec.fit_transform(feature).toarray()
# print user_input

personalized_rid = clf_entropy.predict([[0. ,1. ,1. , 0., 0., 1., 0., 1., 1., 0., 1., 0.]])
return personalized_rid.astype(int)
```

DOCKERFILE



```dockerfile
FROM python:3.9

# Establecer el directorio de trabajo dentro del contenedor
WORKDIR /app

# Copiar los archivos del proyecto al contenedor
COPY . /app

# Instalar las dependencias del proyecto
RUN pip install -r requirements.txt

# Exponer el puerto en el que se ejecuta la aplicación
EXPOSE 8000
```

DOCKER-COMPOSE.YML



```yaml
version: '3'
services:
  web:
    build: .
    ports:
      - 8000:8000
    command: gunicorn Tourism.wsgi:application --bind 0.0.0.0:8000
    volumes:
      - .:/app
```