POLITECNICO
DI TORINO

ICT for Smart Societies

"Interdisciplinary Project"

2018-2019

Technical Steps

of

"Smartphone application to collect mobility patterns and give mobility information to users"

1. Firstly, data must be downloaded from the application “Mobilità Dinamica”.
   The following link corresponds to the API to access to the data:

   https://my-moby.com/mobilita/api/v1/data/position?nItem=100&nPage=1&from=1547551363000&to=1550229763000

   In the fields "from=" and "to=" the period of data that you want to see must be inserted. Both fields are in UNIX millisecond format.
   Data is saved in a JSON format.

   In this project the data used is the one collected last year (2017-2018).

2. A collection is created in MongoDB containing the data downloaded, then some fields (columns) of this data are chosen implementing the code showed below. Finally, data is extracted as csv file.

```
var result=db.users.aggregate([{

$project: {

_id:0,

user_Id:"$properties.UserId",

orig_Id:"$properties.origin_zone.census_zone_id",

date_orig:"$properties.origin_zone.date",

longitud_orig:{$arrayElemAt: [ "$properties.origin_zone.loc.coordinates", 0]},

latitude_orig:{$arrayElemAt: [ "$properties.origin_zone.loc.coordinates", 1]},

dest_Id:"$properties.destination_zone.census_zone_id",

date_dest:"$properties.destination_zone.date",

longitud_dest:{$arrayElemAt: [ "$properties.destination_zone.loc.coordinates", 0]},

latitude_dest:{$arrayElemAt: [ "$properties.destination_zone.loc.coordinates", 1]}

},}])

while (result.hasNext()){

o = result.next()

print
(o["user_Id"],o["orig_Id"],o["date_orig"],o["longitud_orig"],o["latitude_orig"],o["dest_Id"],o["d
ate_dest"],o["longitud_dest"],o["latitude_dest"])}
```

3. Data extracted from MongoDB (csv file) are imported in Python with the lines of code shown in *Figure 1*.

```
In [3]: import numpy as np
        import pandas as pd
        import datetime

        folder="F:\ICT\Interdisiplinary project\Prj"
        Data= folder + '/' + 'newDataofUser.csv'
        UserData=pd.read_csv(Data,sep=" ")
        UserData
```

Figure 1 - Import data in Python

After filtering data in MongoDB, a data frame of 9 columns (table below) is created.

Title	Description
userId	Id of user
origin_id	Id of origin zone
origin_date	Date of start trip
longitud_orig	Longitude of location where a person start trip
latitude_orig	latitude of location where a person start trip
destination_id	Id of destination zone
destination_date	Date of finish trip
longitud_dest	Longitude of location where a person finish trip
latitud_dest	latitude of location where a person finish trip

Dataframe of trips obtained in Python is showed in *Figure 2*.

Out[3]:

	userId	origin_id	origin_date	longitud_orig	latitude_orig	destination_id	destination_date	longitud_dest	latitud_dest
0	594994d9c9e77c0001b73c33	231	1507830408139	7.691520	45.065525	0	1507830435469	7.695906	45.066072
1	594994d9c9e77c0001b73c33	237	1507838470195	7.692031	45.064449	232	1507838663869	7.692731	45.065297
2	594994d9c9e77c0001b73c33	231	1507830259171	7.691520	45.065525	231	1507830323839	7.691520	45.065525
3	594994d9c9e77c0001b73c33	0	1507833854788	7.695906	45.066072	227	1507833880303	7.692144	45.066354
4	594994d9c9e77c0001b73c33	227	1507837692961	7.692144	45.066354	237	1507837818921	7.692031	45.064449
5	59d649fcc9e77c0001badd8d	2832	1507830802152	7.697783	45.097775	1770	1507831960029	7.696186	45.097878
6	594994d9c9e77c0001b73c33	232	1507838699036	7.692731	45.065297	0	1507838747479	7.695906	45.066072
7	594994d9c9e77c0001b73c33	0	1507839352865	7.695906	45.066072	0	1507839368545	7.695906	45.066072
8	59d649fcc9e77c0001badd8d	1770	1507846688557	7.696186	45.097878	1618	1507876766930	7.695708	45.097205
9	594994d9c9e77c0001b73c33	362	1507828214796	7.685500	45.054393	231	1507829740868	7.691520	45.065525
10	594994d9c9e77c0001b73c33	362	1507896294924	7.686718	45.054100	362	1507896295115	7.686823	45.053511
11	594994d9c9e77c0001b73c33	362	1507893628112	7.685500	45.054393	362	1507893816661	7.685500	45.054393
12	594994d9c9e77c0001b73c33	362	1507889745811	7.687096	45.054185	362	1507890548129	7.685500	45.054393
13	594994d9c9e77c0001b73c33	362	1507894690140	7.685500	45.054393	362	1507895587557	7.686718	45.054100
14	594994d9c9e77c0001b73c33	362	1507896608083	7.686823	45.053511	797	1507898707522	7.660934	45.062339
15	594994d9c9e77c0001b73c33	237	1507882898987	7.691812	45.065180	362	1507884644148	7.687096	45.054185
16	594994d9c9e77c0001b73c33	797	1507911514891	7.660934	45.062339	232	1507913707946	7.692274	45.065163
17	594994d9c9e77c0001b73c33	232	1507935597127	7.692274	45.065163	425	1507938260091	7.699909	45.069548

Figure 2 - Trips Dataframe

"Date" field present in origin/destination of the trips is a *NumberLong*, so it is necessary to remove 3 last numbers of two columns of the date ("origin_date", "destination_date") in order to make it readable, as shown in *Figure 3*.

```
In [ ]: UserData['origin_date'] = UserData['origin_date'].astype(str).str[:-3].astype(np.int64)
        UserData['destination_date'] = UserData['destination_date'].astype(str).str[:-3].astype(np.int64)
```

Figure 3- From NumberLong to date

4. Next step consists in separating transit trip from final trip, home trips and suburb trips according to the algorithm depicted in *Figure 4*.

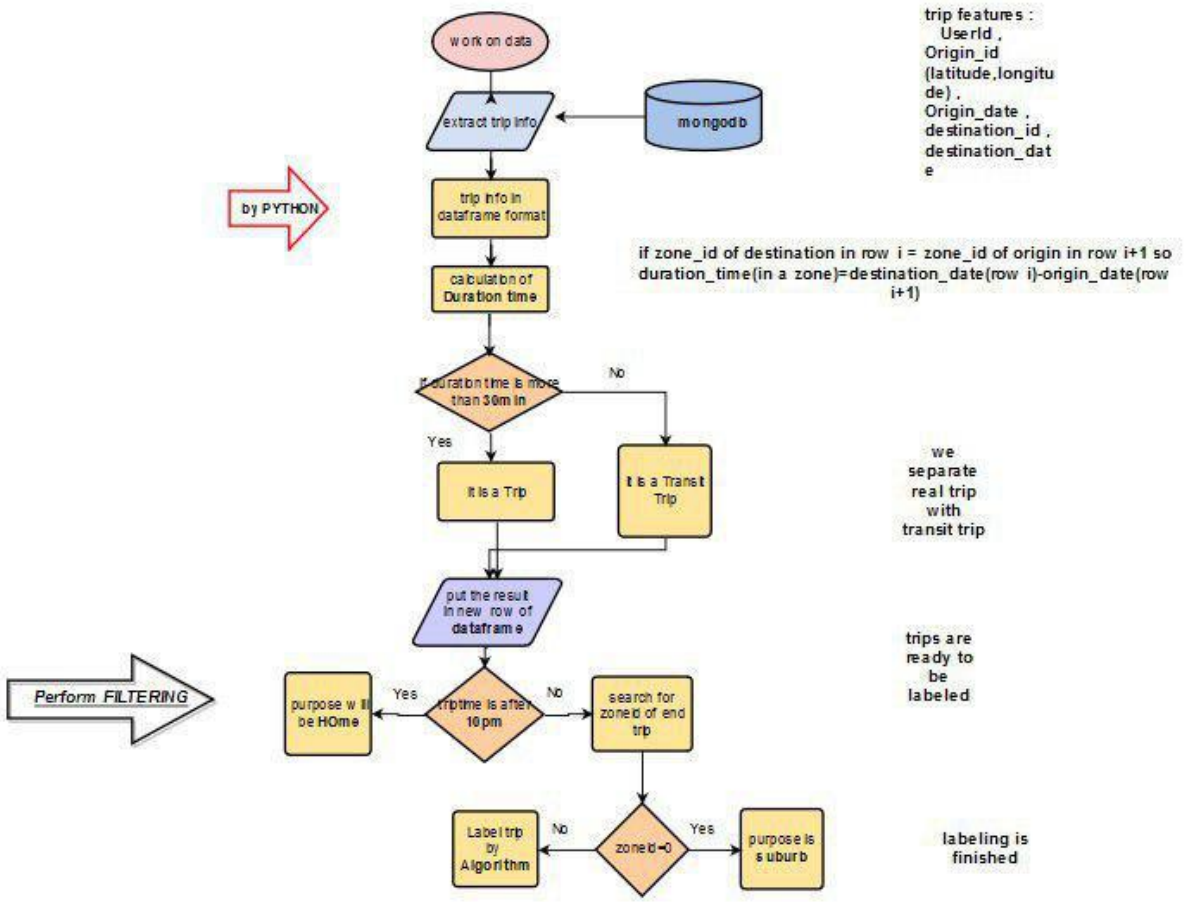

Figure 4 - Separate trips by type

The code present in *Figure 5* is used to define if trips are “transit” or “final”.

```
#seprate transit and final trips
for i in range(0,UserData.shape[0]-1):
        UserData.loc[i,'trip_duration']=(UserData.loc[i,'destination_date']-UserData.loc[i,'origin_date']).astype(int)
UserData['trip_duration']=abs(UserData['trip_duration'])

for i in range(0,UserData.shape[0]-1):
    if (UserData.loc[i,'destination_id']==UserData.loc[i+1,'origin_id']):
        UserData.loc[i,'duration']=(UserData.loc[i,'destination_date']-UserData.loc[i+1,'origin_date']).astype(int)
UserData['duration']=abs(UserData['duration'])

for i in range(0,UserData.shape[0]-1):
    UserData.loc[i,'duration_allTrips']=(UserData.loc[i,'destination_date']-UserData.loc[i+1,'origin_date']).astype(int)
UserData['duration_allTrips']=abs(UserData['duration_allTrips'])

UserData['transit']='final trip'
UserData.loc[
    UserData['duration'] <= 1800,
    ('transit')
] = (1)


for i in range(0,UserData.shape[0]-1):
    if (UserData.loc[i,'transit']==1):
         UserData.loc[i,'transit']='transit'
```

Figure 5 - Transit/final trip definition

The code present in *Figure 6* is used to define if trips are “home” or “suburb”.

```
In [7]: # define home and suburb trips
for i in range(0,UserData.shape[0]):
    if(UserData.loc[i,'destination_id']==0):
         UserData.loc[i,'Final_purpose']=UserData.loc[i,'purpose']+'(Suburb)'
    if (UserData.loc[i,'transit']=='transit'):
        UserData.loc[i,'Final_purpose']='Transit'

    a=UserData.loc[i,'destination_date'].hour
    if(a>=22 or a<=5):
        UserData.loc[i,'Final_purpose']='Home'
```

Figure 6 - Home/suburb trip definition

5. Census zones and Points of Interest collection

The set of Census zone (of last year) is imported in MondoDB into a collection.
Two different POIs csv file are imported in two different dataframes: weekday and weekend.
Each POI is characterized by the fields showed in the table below. ZoneId is the census zone the POI belongs to.

Title	**Description**
PointId	Id of point of interests
Name	The names
ZoneId	Id of zone where locate
Longitude	Longitude of coordinate of POIs
Latitude	Latitude of coordinate of POIs

Then, as shown in *Figure 7*, Euclidean distance is calculated in Python to get the shortest one.

```
In [2]: #reading POIs file for weekday
folder="F:\ICT\Interdisiplinary project\Prj"
po= folder + '/' + 'poi.csv'
point=pd.read_csv(po,sep=",")

#calculate euclidean distance between coordinate of points(data from last year) and POIs
for i in range(0,UserData.shape[0]):
#     point2.loc[j,'distance_long']=((UserData.loc[i,'longitud_dest']-point2['longitude'])**2)
#     point2.loc[j,'distance_lat']=((UserData.loc[i,'latitud_dest']-point2['latitude'])**2)
    point['distance']=np.sqrt(((UserData.loc[i,'longitud_dest']-point['longitude'])**2)+((UserData.loc[i,'latitud_dest']-point['l
    c=point['distance'].idxmin() #get the index of minimum distance
    UserData.loc[i,'purpose']=point.loc[c,'PointId']
```

```
In [3]: #reading POIs file for weekend
folder="F:\ICT\Interdisiplinary project\Prj"
po= folder + '/' + 'POI_weekend.csv'
point2=pd.read_csv(po,sep=",")

#calculate euclidean distance between coordinate of points(data from last year) and POIs
for i in range(0,UserData.shape[0]):

#             point2.loc[j,'distance_long']=(UserData.loc[i,'longitud_dest']-point2.loc[j,'longitude'])**2
#             point2.loc[j,'distance_lat']=(UserData.loc[i,'latitud_dest']-point2.loc[j,'latitude'])**2
    point2['distance']=np.sqrt(((UserData.loc[i,'longitud_dest']-point2['longitude'])**2)+((UserData.loc[i,'latitud_dest']-point2
    c2=point2['distance'].idxmin() #get the index of minimum distance
    UserData.loc[i,'purpose2']=point2.loc[c2,'PointId']
```

Figure 7 - Euclidean distance calculation

6. Labelling of trips

   After importing the data, timestamp is converted to readable date and then time and day are separated. Furthermore, the purpose of trips is determined according to the day of the week and the time of the day.

It means that if the trip happens on Saturday, or on Sunday, or on Friday after 19:00, POIs of weekend are used. Labelling of the trips according to the day of the week is shown in the code of *Figure 8*.

```
In [4]: #convert timestamp to readable data
UserData['origin_date']=pd.to_datetime(UserData['origin_date'], unit='s')
UserData['destination_date']=pd.to_datetime(UserData['destination_date'], unit='s')

#separate date and time in two different column
UserData['dest_date']=UserData['destination_date'].dt.date
UserData['dest_time']=UserData['destination_date'].dt.time

UserData['orig_date']=UserData['origin_date'].dt.date
UserData['orig_time']=UserData['origin_date'].dt.time


UserData['dayOFweek']=UserData['destination_date'].dt.weekday_name #add one column that determine name of week ex:sunday

#determine purpose of trips based on day of week and time
for i in range(0,UserData.shape[0]):
    a=UserData.loc[i,'dayOFweek']
    b=UserData.loc[i,'destination_date'].hour
    if (a is 'Saturday' or a is 'Sunday'):
        UserData.loc[i,'Final_purpose']=UserData.loc[i,'purpose2']
        UserData.loc[i,'dayOFweek']='weekend'
    elif (a is 'Friday' and b>=19 ):
        UserData.loc[i,'Final_purpose']=UserData.loc[i,'purpose2']
        UserData.loc[i,'dayOFweek']='weekend'
    else:
        UserData.loc[i,'Final_purpose']=UserData.loc[i,'purpose']
        UserData.loc[i,'dayOFweek']='weekday'
```

Figure 8 - Labelling of trips according to day of the week

Trips are also labelled according to the “portion_of_day”: daytime (between 8:00 Am and 7:00 Pm), morning and night (otherwise), visible in *Figure 9*.

```
#separate day to daytime and morn&nigth according to hours
for i in range(0,UserData.shape[0]):
    x= UserData.loc[i,'dest_time'].hour
    if (x>=8 and x<=19):
        UserData.loc[i,'portion_of_day']='daytime'
    else:
        UserData.loc[i,'portion_of_day']='morn & nigth'
```

Figure 9 – Labelling of trips according to portion of the day

7. Labelling of zones

According to destination, day of the week, portion of the day and purpose the label of zones are selected. The code is shown in *Figure 10*.

```
In [11]: #groupby zone and count trip in each zone
         ourzone=pd.DataFrame();
         ourzone=UserData.groupby(
             ['destination_id','dayOFweek','portion_of_day','Final_purpose']
         ).agg(
             {

                 # find the number of destination id
                'Final_purpose': "count",



             }
         )

In [12]: ourzone.to_csv(r'F:\ICT\Interdisiplinary project\Prj\ourzone.csv')
         ourzone.dropna();

In [13]: #Label zone
         folder="F:\ICT\Interdisiplinary project\Prj"
         Data= folder + '/' + 'ourzone.csv'
         ourzone=pd.DataFrame();
         ourzone=pd.read_csv(Data,sep=",")
         ourzone=ourzone.loc[ourzone.groupby(["destination_id", "dayOFweek","portion_of_day"])["Final_purpose.1"].idxmax()]
         ourzone.to_csv(r'F:\ICT\Interdisiplinary project\Prj\ourzone.csv')
         ourzone.to_json(r'F:\ICT\Interdisiplinary project\Prj\ourzone.json')
         ourzone
```

Figure 10 - Labelling of zones

The obtained labelled zones are visible in the table of *Figure 11*, where the label corresponds to "Final_purpose".

Out[13]:

	destination_id	dayOFweek	portion_of_day	Final_purpose	Final_purpose.1
1	0	weekday	daytime	Train	2
2	0	weekday	morn & nigth	Academic	1
4	0	weekend	daytime	Entertainment	1
5	0	weekend	morn & nigth	Entertainment	1
8	28	weekend	morn & nigth	touristic	1
9	44	weekday	daytime	touristic	2
10	67	weekend	daytime	touristic	1
11	80	weekend	daytime	touristic	1
12	97	weekday	morn & nigth	Home	1
13	104	weekday	morn & nigth	Home	1
14	105	weekday	morn & nigth	Home	1
15	117	weekday	daytime	job	1
16	125	weekday	daytime	job	1
17	133	weekend	daytime	touristic	1
18	148	weekday	daytime	Train	1
19	227	weekday	daytime	touristic	2
21	228	weekday	morn & nigth	Home	2
22	228	weekend	morn & nigth	Home	2

Figure 11 - Labelled zones

8. Tracking

Filtering on the users' data is performed, considering only the fields that are useful to tracking. The goal is to understand users' behavior and maybe to do some prediction about users' destination in the future, which can be an alternative to costly surveys.

In *Figure 12* the aggregation of useful fields is shown (in Python).

In [12]:
```
#tracking: group by persons and track trips for each of them
tracking=pd.DataFrame();
tracking=UserData.groupby(
    ['userId','dayOFweek','origin_date','longitud_orig','latitude_orig','portion_of_day_orig','destination_date','longitud_dest',
).agg(
    {
         # find the number of destination id
        'destination_id': "count",

    }
)
tracking
tracking.to_csv(r'F:\ICT\Interdisiplinary project\Prj\trackingOFuser.csv')
```

Figure 12 - Tracking

9. Machine Learning

The two specific approaches were used in order to increase prediction value:

1. Feature Engineering: remove the exact time or day of trips to make it more general, shown in *Figure 13*;

```
In [11]: #making x and y as for traning and test model
         x=UserData[0:446]

         #prepare data for doing machine learning(set the string with numbers ex: final trips is 0 ,otherwise is 1)
         x['transit'] = x['transit'].map({'final trip': 0, 'transit': 1})
         x['dayOFweek'] = x['dayOFweek'].map({'weekday': 1, 'weekend': 2})
         x['orig_dayOFweek'] = x['orig_dayOFweek'].map({'weekday': 1, 'weekend': 2})
         x['portion_of_day'] = x['portion_of_day'].map({'daytime': 0, 'morn & nigth': 1})
         x['orig_portion_of_day'] = x['orig_portion_of_day'].map({'daytime': 0, 'morn & nigth': 1})

         #get y column for making model... y = final purpose(labels of trips)
         y=x.loc[:,'Final_purpose']

         #remove the exact time or day of trips to make it more general and the other coulumn which is not necessary to make model
         x=x.drop(['userId','Final_purpose','nextp','purpose','purpose2','dest_date','dest_time','duration','longitud_orig','latitude_orig
         x = x.fillna(0) #replace nan value with 0

         #some title with same purpose start with different letter... so we arrang them in same letters.
         y = y.replace('Job ', 'job')
         y = y.replace('Shopping', 'shopping')
         y = y.replace('Entertainment ', 'Entertainment')
         y = y.replace('Turistic ', 'touristic')
         y = y.replace('university', 'Academic ')
         y = y.replace('stadio', 'Academic ')
```

Figure 13- Feature Engineering algorithm

The data frame obtained after the application of Feature Engineering is shown in *Figure 14.*

Out[54]:

latitude_orig	destination_id	longitud_dest	latitud_dest	trip_duration	duration_allTrips	transit	dayOFweek	orig_dayOFweek	portion_of_day	orig_portion_of_day
45.065525	0	7.695906	45.066072	27.0	8035.0	0	1	1	0	0
45.064449	232	7.692731	45.065297	193.0	8404.0	0	1	1	1	1
45.065525	231	7.691520	45.065525	64.0	3531.0	0	1	1	0	0
45.066072	227	7.692144	45.066354	26.0	3812.0	0	1	1	0	0
45.066354	237	7.692031	45.064449	126.0	7016.0	0	1	1	0	0
45.097775	1770	7.696186	45.097878	1158.0	6739.0	0	1	1	0	0
45.065297	0	7.695906	45.066072	48.0	605.0	1	1	1	1	1
45.066072	0	7.695906	45.066072	16.0	7320.0	0	1	1	1	1
45.097878	1618	7.695708	45.097205	30078.0	48552.0	0	1	1	1	1
45.054393	231	7.691520	45.065525	1526.0	66554.0	0	1	1	0	0
45.054100	362	7.686823	45.053511	1.0	2667.0	0	1	1	0	0
45.054393	362	7.685500	45.054393	188.0	4071.0	0	1	1	0	0
45.054185	362	7.685500	45.054393	803.0	4142.0	0	1	1	0	0
45.054393	362	7.686718	45.054100	897.0	1021.0	1	1	1	0	0
45.053511	797	7.660934	45.062339	2099.0	15809.0	0	1	1	0	0
45.065180	362	7.687096	45.054185	1746.0	26870.0	0	1	1	0	0
45.062339	232	7.692274	45.065163	2193.0	21890.0	0	1	1	0	0
45.065163	425	7.699909	45.069548	2663.0	43472.0	0	2	2	1	1

Figure 14 - Dataframe after Feature Engineering

2. Grid Search: appropriate input parameters of Machine Learning algorithm are selected, in such a way that they can provide better results. The code is present in *Figure 15*.

```
In [14]: # Add important librarys for making model

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearch
from sklearn.cross_validation import StratifiedKFold

#splilit data to train and test
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.1, random_state=0)
parameter_gridsearch = {
                 'max_depth' : [3,4,5,6,7,8,9,10],  #depth of each decision tree
                 'n_estimators': [100,150,180],#count of decision tree
                 'max_features': ['auto', 'log2'],
#                  'min_samples_split': [2],
#                  'min_samples_leaf': [1, 3, 4],
                 'bootstrap': [True, False],
                 }

clf = RandomForestClassifier()
crossvalidation = StratifiedKFold(y_train , n_folds=5)

gridsearch = GridSearchCV(clf,             #grid search for algorithm optimization
                          scoring='accuracy',
                          param_grid=parameter_gridsearch,
                          cv=crossvalidation)
```

Figure 15 - Grid Search algorithm

The data frame obtained after Grid Search use to make model. Furthermore, the test model is necessary to be sure model work correctly. These steps are shown in *Figure 16.*

```
#make model
gridsearch.fit(X_train, y_train)
model = gridsearch
parameters = gridsearch.best_params_
print('Best Score: {}'.format(gridsearch.best_score_))

#test model
preds = gridsearch.predict(X_test)


# preds = clf.predict(X_test)
sum(preds==y_test)/y_test.shape[0] #calculate the error
```

Figure 16 – Test model

10. Visualization

- Tracking map

Firstly, trips related to one user are extracted (csv file) from the dataframe obtained in Python. Then, for each day (with present trips) a new csv file is created containing the trips of that user, with these fields: timestamp (origin/destination – day of the week – date - timestamp), longitude, latitude.

Once the files are imported in Google maps, points become visible in the map. Routes are indicated by lines that are manually depicted from origin points to destination points.

In this way it is possible to analyze users' behavior and suggest their habits.

- Dynamic map

  At the first, labeled zones are extracted from python. On the other hand, in google map, three layers to display census zones (kml format) and four layers for different day of week and different time of the day (csv format) are made. Each level contains "**zones**" and "**label**" and Create a gradient color based on different category of label.

11. Results (Website)

    Use Microsoft visual studio as editor to make a html page for web site. It is shown in following code.

```
<!DOCTYPE html>

<html xmlns="">
<head>
    <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, height=device-height, initial-
scale=1.0, viewport-fit=cover">
        <meta name="apple-mobile-web-app-capable" content="yes" />

        <!-- Page Title -->
        <title>mobilita DynAmica</title>

        <!-- Compressed Styles -->
        <link href="css/slides.min.css" rel="stylesheet" type="text/css">

        <!-- Custom Styles -->
        <!-- <link href="css/custom.css" rel="stylesheet" type="text/css"> -->

        <!-- jQuery 3.3.1 -->
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

        <!-- Compressed Scripts -->
        <script src="js/slides.min.js" type="text/javascript"></script>

        <!-- Custom Scripts -->
        <!-- <script src="js/custom.js" type="text/javascript"></script> -->

        <!-- Fonts and Material Icons -->
        <link rel="stylesheet" as="font"
href="https://fonts.googleapis.com/css?family=Roboto:100,300,400,500,600,700|Material+Ico
ns"/>

</head>
<body class="slides horizontal simplifiedMobile animated">
```

```
 <nav class="panel top">
  <div class="sections desktop">
    <div class="left"><a href="#" class="opacity-8"><!--<img width="80"
src="assets/img/icon.png" height="80"/>--><svg style="height:21px;"></svg>Politecnico di
torino </a></div>
    <div class="center"><a  class="opacity-8">Interdisciplinary project</a></div>
    <div class="right"><span data-dropdown-id="2" class="button actionButton
dropdownTrigger"><svg><use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#share"></use></svg></span></div>
  </div>

</nav>


<!-- Panel Top #05 -->
<nav class="panel top">
  <div class="sections desktop">
    <div class="left"><a href="" title="Interdisciplinary project"><svg
style="width:82px;height:24px"><use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#logo"></use></svg></a></div>
    <div class="center">
      <ul class="menu">
        <li><a href="#"></a></li>
        <li><a href="#"></a></li>
        <li><a href="#"></a></li>
        <li><a href="#"></a></li>
      </ul>
    </div>
    <div class="right"><a class="button blue gradient"
href="https://www.google.com/maps/d/viewer?mid=1XzgDJgLPDUJ9j0189KJULS_oOqW5xn7G&vomp=1&c
id=mp&cv=hCpcgyE7Vd0.en." target="_blank">Dynamic Map</a><a class="button green gradient"
href="https://drive.google.com/open?id=1b02wt2WNORwp_4HWP6TkDk16B9-Iw6xP&usp=sharing"
target="_blank">Tracking</a></div>
  </div>
  <div class="sections compact hidden">
    <div class="left"><a href="#" title="Slides Framework"><svg
style="width:82px;height:24px"><use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#logo"></use></svg></a></div>
    <div class="right"><span class="button actionButton sidebarTrigger" data-sidebar-
id="1"><svg><use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#menu"></use></svg></span</div>
  </div>
</nav>

<!-- Sidebar -->
<nav class="sidebar" data-sidebar-id="1">
  <div class="close"><svg><use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#close"></use></svg></div>
  <div class="content">
    <a href="#" class="logo"><svg width="37" height="30"><use
xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="#logo-icon"></use></svg></a>
      <p>
           </p>
  </div>
</nav>

<!-- Slide 1 (#34) -->
<section class="slide fade-6 kenBurns">
  <div class="content">
    <div class="container">
      <div class="wrap">
```

```
        <div class="fix-12-12">
          <ul class="flex fixedSpaces verticalCenter reverse">
            <li class="col-6-12 left middle">
              <h1 class="ae-1 fromLeft">Smartphone application to collect mobility
patterns and give mobility information to users</h1>
              <p class="ae-2 fromLeft"><span class="opacity-8"> Mersedeh Kooshki<br
/>Zahra Arshadi<br />Elisa Elmazaj <br /> Berta Mazuecos Velázquez</span></p>
              <!--<a class="button blue gradient ae-3 fromCenter cropLeft">Get
Started</a><a class="button white ae-4 fromCenter">Learn more</a>-->
            </li>
            <li class="col-6-12">
              <img class="ae-4" width="605" src="assets/img/phone.png" alt="iPhones
Thumbnail" data-action="zoom"/>
            </li>
          </ul>
        </div>

      </div>
    </div>
  </div>
  <div class="background" style="background-
image:url(assets/img/background/modern.jpg)"></div>
</section>




<!-- Popup Video -->
<div class="popup autoplay" data-popup-id="60-1">
  <div class="close"><svg><use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#close"></use></svg></div>
  <div class="content">
    <div class="container">
      <div class="wrap">
        <div class="fix-10-12">
          <div class="embedVideo popupContent">
            <iframe
src="https://player.vimeo.com/video/101231747?color=ff0179&portrait=0" frameborder="0"
webkitallowfullscreen mozallowfullscreen allowfullscreen></iframe>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>


<!-- Popup Video -->
<div class="popup autoplay" data-popup-id="60-2">
  <div class="close"><svg><use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#close"></use></svg></div>
  <div class="content">
    <div class="container">
      <div class="wrap">
        <div class="fix-10-12">
          <div class="embedVideo popupContent">
            <iframe
src="https://player.vimeo.com/video/101231747?color=ff0179&portrait=0" frameborder="0"
webkitallowfullscreen mozallowfullscreen allowfullscreen></iframe>
          </div>
        </div>
      </div>
```

```
    </div>
  </div>
</div>


<!-- Popup Video -->
<div class="popup autoplay" data-popup-id="60-3">
  <div class="close"><svg><use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#close"></use></svg></div>
  <div class="content">
    <div class="container">
      <div class="wrap">
        <div class="fix-10-12">
          <div class="embedVideo popupContent">
            <iframe
src="https://player.vimeo.com/video/101231747?color=ff0179&portrait=0" frameborder="0"
webkitallowfullscreen mozallowfullscreen allowfullscreen></iframe>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>






<!-- Slide 7 (#95) -->
<section id="next" class="slide fade-6 kenBurns">
  <div class="content">
    <div class="container">
      <div class="wrap">

        <div class="fix-6-12">
          <h1 class="huge ae-1 margin-bottom-2">Thanks to </h1>
          <p class="hero ae-2 margin-bottom-3"><span class="opacity-8">Prof. Cristina
Pronello   &   Prof. Fabio Dovis</span></p>

          <!--<form action="#" autocomplete="off" class="slides-form margin-bottom-3">
            <input type="email" class="ae-3" name="email" placeholder="E-mail address"/>
            <button type="submit" class="button blue gradient ae-4" name="submit">Try it
free</button>
          </form>-->

          <a href="https://apps.apple.com/it/app/mobilit%C3%A0-dinamica/id1364437932"
target="_blank" class="button hollow ae-5"><img src="assets/img/appstore.jpg"
height="63"/></a><a href="" target="_blank" class="button hollow ae-6"><img
src="assets/img/googleplay.jpg" height="63"/></a>
        </div>

      </div>
    </div>
  </div>
  <div class="background" style="background-
image:url(assets/img/background/min.jpg)"></div>
</section>

<!-- Panel Bottom #01 -->
<nav class="panel bottom forceMobileView">
  <div class="sections desktop">
```

```
    <div class="left"><a href="#next" class="opacity-8"><img width="40"
src="assets/img/play_color.png" height="40"/><svg style="height:21px;"><use
xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="#apple"></use></svg></a></div>
    <div class="center"><span class="nextSlide"><svg><use
xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="#arrow-
down"></use></svg></span></div>
    <div class="right"><span data-dropdown-id="2" class="button actionButton
dropdownTrigger"><svg><use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#share"></use></svg></span></div>
  </div>
  <div class="sections compact hidden">
    <div class="right">
      <span data-dropdown-id="2" class="button actionButton dropdownTrigger"><svg><use
xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="#share"></use></svg></span>
    </div>
  </div>
</nav>


<!-- Share Window -->
<div class="dropdown share bottom right" data-dropdown-id="2" data-text="Take a look at
this" data-url="https://designmodo.com" data-pinterest-image="https://designmodo.com/wp-
content/uploads/2015/10/Presentation.jpg">
  <div class="center padding-2">
    <div class="title">Share</div>
    <a href="#">Contact us</a>
  </div>
  <ul>
    <li class="social-facebook"><svg><use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#fb-like"></use></svg></li>
    <li class="social-twitter"><svg><use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#twitter"></use></svg></li>
    <li class="social-googlePlus"><svg><use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#googlePlus"></use></svg></li>
    <li class="social-linkedin"><svg><use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#linkedin"></use></svg></li>
    <li class="mail" data-subject="Subject" data-body="Body">share by email</li>
  </ul>
</div>

<!-- Loading Progress Bar -->
<div class="progress-bar blue"></div>

</body>
</html>
```