Pre - Ordenado

June 19, 2017

1 TP2: Machine Learning

1.0.1 Imports

```
In [ ]: import pandas as pd
        from datetime import datetime
        import scipy.spatial
        from sklearn import preprocessing
        import numpy as np
        import matplotlib.pyplot as plt
```

1.0.2 Data loading

```
In [ ]: print "* * * * * Data loading * * * * *"
        print "loading station csv"
        stationDF = pd.read_csv('../CSVs/station.csv')
        print "loading trip train csv"
        trainingSet = pd.read_csv('../CSVs/trip_train.csv')
        print "loading trip test csv"
        testingSet = pd.read_csv('../CSVs/trip_test.csv')

        # DF DEL TP1
        # (0 = Monday, 1 = Tuesday...)
        print "loading dfSF_Bay csv"
        dfSF_Bay = pd.read_csv('../CSVs/dfSF_Bay.csv')
```

1.1 Basic data analysis

```
In [ ]: print "stationDF.shape: ", stationDF.shape
        print "trainingSet.shape: ", trainingSet.shape
        print "testingSet.shape: ", testingSet.shape
        print "dfSF_Bay.shape: ", dfSF_Bay.shape
```

stationDF.shape: (70, 7) trainingSet.shape: (549961, 11) testingSet.shape: (119998, 10) dfSF_Bay.shape: (733, 33)

1.1.1 Distancias entre estaciones

```
In [ ]: print "* * * * * Working on station distances * * * * *"
        # Create new temporary dataframe with distances
        distancesDF = pd.DataFrame(columns=["start_station_id", "end_station_id", "distance"])

        # Calculate distances between stations
        for station, lat, lon in zip(stationDF.id, stationDF.lat, stationDF.long):
            for station2, lat2, lon2 in zip(stationDF.id, stationDF.lat, stationDF.long):
                distancesDF = distancesDF.append({
                    "start_station_id": station,
                    "end_station_id": station2,
                    "distance": scipy.spatial.distance.cityblock([lat, lon], [lat2, lon2])
                }, ignore_index=True)

        distancesDF['start_station_id'] = distancesDF.start_station_id.astype(int)
        distancesDF['end_station_id'] = distancesDF.end_station_id.astype(int)

        # Merge this new data to training and testing sets
        trainingSet = pd.merge(trainingSet,distancesDF,on =['start_station_id','end_station_id']
        testingSet = pd.merge(testingSet,distancesDF,on =['start_station_id','end_station_id'],h

        # delete auxiliary distances df
        del distancesDF
```

1.1.2 Process date & time data

```
In [ ]: print "* * * * * Converting necessary data to dateTime * * * * *"
        # Convert necessary data to dateTime
        dfSF_Bay['date'] = pd.to_datetime(dfSF_Bay.date)

        trainingSet['start_date'] = pd.to_datetime(trainingSet.start_date)
        trainingSet['end_date'] = pd.to_datetime(trainingSet.end_date)

        testingSet['start_date'] = pd.to_datetime(testingSet.start_date)
        testingSet['end_date'] = pd.to_datetime(testingSet.end_date)

        # Create new features related to date & time based on the unique 'date' feature
        # Work with training set
        trainingSet['start_dayOfWeek'] = trainingSet.start_date.dt.dayofweek
        trainingSet['start_week'] = trainingSet.start_date.dt.week
        trainingSet['start_quarter'] = trainingSet.start_date.dt.quarter
        trainingSet['start_time'] = trainingSet.start_date.dt.time
        trainingSet['start_hour'] = trainingSet.start_date.dt.hour
        trainingSet['start_minute'] = trainingSet.start_date.dt.minute
        trainingSet['start_year'] = trainingSet.start_date.dt.year
        trainingSet['start_month'] = trainingSet.start_date.dt.month
        trainingSet['start_day'] = trainingSet.start_date.dt.day
        trainingSet['start_date'] = trainingSet.start_date.dt.date

        trainingSet['end_dayOfWeek'] = trainingSet.end_date.dt.dayofweek
        trainingSet['end_week'] = trainingSet.end_date.dt.week
        trainingSet['end_quarter'] = trainingSet.end_date.dt.quarter
        trainingSet['end_time'] = trainingSet.end_date.dt.time
        trainingSet['end_hour'] = trainingSet.end_date.dt.hour
        trainingSet['end_minute'] = trainingSet.end_date.dt.minute
        trainingSet['end_year'] = trainingSet.end_date.dt.year
        trainingSet['end_month'] = trainingSet.end_date.dt.month
        trainingSet['end_day'] = trainingSet.end_date.dt.day
        trainingSet['end_date'] = trainingSet.end_date.dt.date

        trainingSet['year'] = pd.to_datetime(trainingSet['start_date']).dt.year
        trainingSet['month'] = pd.to_datetime(trainingSet['start_date']).dt.month
        trainingSet['weekday'] = pd.to_datetime(trainingSet['start_date']).dt.weekday

        # Work with testing set
        testingSet['start_dayOfWeek'] = testingSet.start_date.dt.dayofweek
        testingSet['start_week'] = testingSet.start_date.dt.week
        testingSet['start_quarter'] = testingSet.start_date.dt.quarter
        testingSet['start_time'] = testingSet.start_date.dt.time
        testingSet['start_hour'] = testingSet.start_date.dt.hour
        testingSet['start_minute'] = testingSet.start_date.dt.minute
        testingSet['start_year'] = testingSet.start_date.dt.year
        testingSet['start_month'] = testingSet.start_date.dt.month
        testingSet['start_day'] = testingSet.start_date.dt.day
        testingSet['start_date'] = testingSet.start_date.dt.date

        testingSet['end_dayOfWeek'] = testingSet.end_date.dt.dayofweek
        testingSet['end_week'] = testingSet.end_date.dt.week
        testingSet['end_quarter'] =testingSet.end_date.dt.quarter
        testingSet['end_time'] = testingSet.end_date.dt.time
        testingSet['end_hour'] = testingSet.end_date.dt.hour
        testingSet['end_minute'] = testingSet.end_date.dt.minute
        testingSet['end_year'] = testingSet.end_date.dt.year
        testingSet['end_month'] = testingSet.end_date.dt.month
        testingSet['end_day'] = testingSet.end_date.dt.day
        testingSet['end_date'] = testingSet.end_date.dt.date

        testingSet['year'] = pd.to_datetime(testingSet['start_date']).dt.year
        testingSet['month'] = pd.to_datetime(testingSet['start_date']).dt.month
        testingSet['weekday'] = pd.to_datetime(testingSet['start_date']).dt.weekday
```

```
In [ ]: print "trainingSet cols values", list(trainingSet.columns.values)
```

```
In [ ]: print "testingSet cols values", list(testingSet.columns.values)
```

1.2 Feature Historico

In [ ]:
```
print "* * * * * Working on historic feature * * * * *"
print "* * * Calculating historic feature * * *"
import math
listaStart = []
listaEnd = []
for i in list(trainingSet.start_station_id.values):
    if i not in listaStart:
        listaStart.append(i)
for i in list(trainingSet.end_station_id.values):
    if i not in listaEnd:
        listaEnd.append(i)
listaHistorico = []
for i in listaStart:
    for j in listaEnd:
        df = trainingSet[(trainingSet['start_station_id'] == i) & (trainingSet['end_stat
        historico = df.duration.mean()
        if (not(math.isnan(historico))):
            listaHistorico.append([i,j,historico])
        else:
            listaHistorico.append([i,j,0])

listaHistorico
```

In [ ]:
```
starStationId = []
endStationId = []
historical = []
for x in listaHistorico:
    starStationId.append(x[0])
    endStationId.append(x[1])
    historical.append(x[2])

data = {
    'start_station_id' : starStationId,
    'end_station_id' : endStationId,
    'historical' : historical,
}

dfData = pd.DataFrame(data,columns = ['start_station_id','end_station_id','historical'])
dfData
```

In [ ]:
```
print "* * Merging historic feature * *"
# Merge this new data to training and testing dfs
# Training
trainingSet = pd.merge(trainingSet,dfData,on=['start_station_id', 'end_station_id'],how

trainingSet['historical'] = trainingSet.historical.astype(int)
```

```
        # Testing
        testingSet = pd.merge(testingSet, dfData, on=['start_station_id', 'end_station_id'], how

        testingSet['historical'] = testingSet.historical.astype(int)

        # delete auxiliar dataframe
        del dfData
```

```
In [ ]: print "trainingSet.shape: ", trainingSet.shape
        print "testingSet.shape: ", testingSet.shape
```

The difference in the shapes is due to the duration feature used in the training set, which was used to calculate the historical feature.

1.2.1 Trabajamos con dfSF_Bay

```
In [ ]: # Convert necessary data to dateTime
        dfSF_Bay['date'] = pd.to_datetime(dfSF_Bay.date)

        trainingSet['start_date'] = pd.to_datetime(trainingSet.start_date)
        trainingSet['end_date'] = pd.to_datetime(trainingSet.end_date)

        testingSet['start_date'] = pd.to_datetime(testingSet.start_date)
        testingSet['end_date'] = pd.to_datetime(testingSet.end_date)
```

```
In [ ]: print "***Merging dfSF_Bay data***"
        # Merge trainingSet with new data

        testingSet = pd.merge(testingSet,dfSF_Bay,left_on ='start_date',right_on='date',how = 'i
        trainingSet = pd.merge(trainingSet,dfSF_Bay,left_on ='start_date',right_on='date',how =
```

```
In [ ]: print "Saving pre - Discretizacion csvs..."
        trainingSet.to_csv('../CSVs/preDiscretizationTraining.csv', index=False)
        testingSet.to_csv('../CSVs/preDiscretizationTesting.csv', index=False)
```

2 Discretizacion

```
In [ ]: (trainingSet.dtypes)
```

```
In [ ]: list(testingSet.columns.values)
```

```
In [ ]: print "* * * * * Discretizacion y Normalizacion * * * * *"
        print "* * * * * Discretizacion * * * * *"
```

```
In [ ]: def crearLista (listadoCompleto):
            listaReducida = []
            for i in listadoCompleto:
                if i not in listaReducida:
                    listaReducida.append(i)
            listaReducida.sort()
            return listaReducida
```

```
In [ ]: def discretizar(columna,nombre, df):
            listaReducida = crearLista(columna)
            v = list(range(len(columna)))
            listaCompleta = list(columna)
            for i in listaReducida:
                for j in range(len(listaCompleta)):
                    if(listaCompleta[j] == i):
                        v[j] = 1
                    else:
                        v[j] = 0
                df[nombre+str(i)] = v
```

```
In [ ]: print "Discretizando start_station_name..."
        discretizar(trainingSet.start_station_name,'start ', trainingSet)
        discretizar(testingSet.start_station_name,'start ', testingSet)

        print "Discretizando end_station_name..."
        discretizar(trainingSet.end_station_name,'end ', trainingSet)
        discretizar(testingSet.end_station_name,'end ', testingSet)

        print "Discretizando start_dayOfWeek..."
        discretizar(trainingSet.start_dayOfWeek,'start_dayOfWeek_id', trainingSet)
        discretizar(testingSet.start_dayOfWeek,'start_dayOfWeek_id', testingSet)

        print "Discretizando end_dayOfWeek..."
        discretizar(trainingSet.end_dayOfWeek,'end_dayOfWeek_id', trainingSet)
        discretizar(testingSet.end_dayOfWeek,'end_dayOfWeek_id', testingSet)

        print "Discretizando subscription_type_..."
        discretizar(trainingSet.subscription_type,'subscription_type_', trainingSet)
        discretizar(testingSet.subscription_type,'subscription_type_', testingSet)

        print "Discretizando start_year..."
        discretizar(trainingSet.start_year,'start_year_', trainingSet)
        discretizar(testingSet.start_year,'start_year_', testingSet)

        print "Discretizando end_year_..."
        discretizar(trainingSet.end_year,'end_year_', trainingSet)
        discretizar(testingSet.end_year,'end_year_', testingSet)

        print "Discretizando start_month..."
        discretizar(trainingSet.start_month,'start_month_', trainingSet)
        discretizar(testingSet.start_month,'start_month_', testingSet)

        print "Discretizando end_month..."
        discretizar(trainingSet.end_month,'end_month_', trainingSet)
        discretizar(testingSet.end_month,'end_month_', testingSet)
        print "Discretizando start_day..."
        discretizar(trainingSet.start_day,'start_day_', trainingSet)
        discretizar(testingSet.start_day,'start_day_', testingSet)

        print "Discretizando end_day..."
        discretizar(trainingSet.end_day,'end_day_', trainingSet)
        discretizar(testingSet.end_day,'end_day_', testingSet)

        print "Discretizando start_quarter..."
        discretizar(trainingSet.start_quarter,'start_quarter_', trainingSet)
        discretizar(testingSet.start_quarter,'start_quarter_', testingSet)

        print "Discretizando end_quarter..."
        discretizar(trainingSet.end_quarter,'end_quarter_', trainingSet)
        discretizar(testingSet.end_quarter,'end_quarter_', testingSet)

        print "Discretizando start_hour..."
        discretizar(trainingSet.start_hour,'start_hour_', trainingSet)
        discretizar(testingSet.start_hour,'start_hour_', testingSet)

        print "Discretizando end_hour..."
        discretizar(trainingSet.end_hour,'end_hour', trainingSet)
        discretizar(testingSet.end_hour,'end_hour', testingSet)

In [ ]: print "Dropping trash columns..."
        trainingSet = trainingSet.drop(labels = ['start_date',
                                                  'end_station_name',
                                                  'start_station_name',
                                                  'end_date',
                                                  'subscription_type',
                                                  'zip_code',
                                                  'start_time',
                                                  'end_time',
                                                  'start_dayOfWeek',
                                                  'end_dayOfWeek',
                                                  'start_year',
                                                  'end_year',
                                                  'start_month',
                                                  'end_month',
                                                  'start_day',
                                                  'end_day',
                                                  'start_quarter',
                                                  'end_quarter',
                                                  'start_hour',
                                                  'end_hour'
                                                 ],axis = 1)

        testingSet = testingSet.drop(labels = ['start_date',
                                                  'end_station_name',
                                                  'start_station_name',
                                                  'end_date',
                                                  'subscription_type',
                                                  'zip_code',
                                                  'start_time',
                                                  'end_time',
                                                  'start_dayOfWeek',
                                                  'end_dayOfWeek',
                                                  'start_year',
                                                  'end_year',
                                                  'start_month',
                                                  'end_month',
                                                  'start_day',
                                                  'end_day',
                                                  'start_quarter',
                                                  'end_quarter',
                                                  'start_hour',
                                                  'end_hour'
                                                 ],axis = 1)
```

```
In [ ]: print "trainingSet.shape: ", trainingSet.shape
        print "testingSet.shape: ", testingSet.shape
```

```
In [ ]: # THIS CELL SHOULD NOT BE COPIED. THIS CELL IS ALLOWED TO BE PRESENT ONLY ONCE IN ALL TH
        print "Saving temp csvs..."
        trainingSet.to_csv('../CSVs/tempTraining.csv', index=False)
        testingSet.to_csv('../CSVs/tempTesting.csv', index=False)
```

3 Binarizacion

```
In [ ]: def binarizar(columna,name,df):
            lista = []
            for i in list(columna):
                numero = int(i)
                lista.append(int(bin(numero)[2:]))
            df[name] = lista
```

```
In [ ]: print "Binarizando start_station_id..."
        binarizar(trainingSet.start_station_id,'start_station_id', trainingSet)
        binarizar(testingSet.start_station_id,'start_station_id', testingSet)

        print "Binarizando end_station_id..."
        binarizar(trainingSet.end_station_id,'end_station_id', trainingSet)
        binarizar(testingSet.end_station_id,'end_station_id', testingSet)
```

4 Data filtering

```
In [ ]: trainingSet.drop(['Unnamed: 0'],1,inplace=True)
        testingSet.drop(['Unnamed: 0'],1,inplace=True)
```

```
In [ ]: # Delete repeated data
        trainingSet.drop(['date','year_y','month_y','weekday_y'],1,inplace=True)
        trainingSet = trainingSet.rename(columns={'year_x':'year','month_x':'month','weekday_x':

        testingSet.drop(['date','year_y','month_y','weekday_y'],1,inplace=True)
        testingSet = testingSet.rename(columns={'year_x':'year','month_x':'month','weekday_x': '
```

```
In [ ]: # Delete:
        #     id: el id que identifica univocamente cada uno de los viajes
        #         no proporciona informacion con la que el algoritmo pueda aprender
        #     start_station_id y end_station_id: las estaciones ya estan discretizadas por nombr
        trainingSet.drop(['id', 'start_station_id','end_station_id'],1,inplace=True)

        testingSet.drop(['id', 'start_station_id','end_station_id'],1,inplace=True)
```

```
In [ ]: # Delete:
        #     bike_id: la duracion del viaje es independiente de la bicicleta,
        #         ya que son todas iguales ("") y se entregan sin juicio alguno ("")
        trainingSet.drop(['bike_id'],1,inplace=True)

        testingSet.drop(['bike_id'],1,inplace=True)
```

```
In [ ]: # Delete:
        #     los dias como numero no aportan nada. E.g. 1 puede ser cualquier dia de la semana.
        trainingSet.drop(['start_day_1','start_day_2', 'start_day_3','start_day_4', 'start_day_5
                          'start_day_8','start_day_9', 'start_day_10','start_day_11', 'start_day
                          'start_day_14','start_day_15','start_day_16', 'start_day_17','start_da
                          'start_day_20','start_day_21','start_day_22', 'start_day_23','start_da
                          'start_day_26','start_day_27','start_day_28','start_day_29','start_day
                          'start_day_31'],1,inplace=True)
        trainingSet.drop(['end_day_1','end_day_2','end_day_3','end_day_4','end_day_5','end_day_6
                          'end_day_9','end_day_10', 'end_day_11','end_day_12', 'end_day_13','end
                          'end_day_16','end_day_17','end_day_18','end_day_19','end_day_20','end_
                          'end_day_23','end_day_24','end_day_25','end_day_26','end_day_27','end_
                          'end_day_30','end_day_31',],1,inplace=True)

        testingSet.drop(['start_day_1','start_day_2', 'start_day_3','start_day_4', 'start_day_5'
                          'start_day_8','start_day_9', 'start_day_10','start_day_11', 'start_day
                          'start_day_14','start_day_15','start_day_16', 'start_day_17','start_da
                          'start_day_20','start_day_21','start_day_22', 'start_day_23','start_da
                          'start_day_26','start_day_27','start_day_28','start_day_29','start_day
                          'start_day_31'],1,inplace=True)
        testingSet.drop(['end_day_1','end_day_2','end_day_3','end_day_4','end_day_5','end_day_6'
                          'end_day_9','end_day_10', 'end_day_11','end_day_12', 'end_day_13','end
```

```
                 'end_day_16','end_day_17','end_day_18','end_day_19','end_day_20','end_
                 'end_day_23','end_day_24','end_day_25','end_day_26','end_day_27','end_
                 'end_day_30','end_day_31',],1,inplace=True)
```

```
In [ ]: # Delete:
        #     La duracion del viaje no puede depender de algo del final del mismo.
        #     De la misma manera, razonando analogamente, podemos concluir que contrario a esto,
        #     si influye el instante inicial del mismo
        #     Retiro lo dicho para la estacion final, quedando valido el razonamiento unicamente
        #     para cuestiones temporales. Aun asi esto esta abierto a discusion.
        trainingSet.drop(['end_week', 'end_minute','end_dayOfWeek_id0','end_dayOfWeek_id1','end_
                          'end_dayOfWeek_id3','end_dayOfWeek_id4','end_dayOfWeek_id5','end_dayOf
                          'end_year_2014','end_year_2015','end_month_1','end_month_2','end_month
                          'end_month_5','end_month_6','end_month_7','end_month_8','end_month_9',
                          'end_month_11','end_month_12','end_quarter_1','end_quarter_2','end_qua
                          'end_hour1', 'end_hour2','end_hour3','end_hour4','end_hour5','end_hour
                          'end_hour10','end_hour11','end_hour12','end_hour13','end_hour14','end_
                          'end_hour17','end_hour18','end_hour19','end_hour20','end_hour21','end_
                         1,inplace=True)

        testingSet.drop(['end_week', 'end_minute','end_dayOfWeek_id0','end_dayOfWeek_id1','end_d
                          'end_dayOfWeek_id3','end_dayOfWeek_id4','end_dayOfWeek_id5','end_dayOf
                          'end_year_2014','end_year_2015','end_month_1','end_month_2','end_month
                          'end_month_5','end_month_6','end_month_7','end_month_8','end_month_9',
                          'end_month_11','end_month_12','end_quarter_1','end_quarter_2','end_qua
                          'end_hour1', 'end_hour2','end_hour3','end_hour4','end_hour5','end_hour
                          'end_hour10','end_hour11','end_hour12','end_hour13','end_hour14','end_
                          'end_hour17','end_hour18','end_hour19','end_hour20','end_hour21','end_
                         1,inplace=True)
```

```
In [ ]: # !!!!!!!! U L T I M A   C E L D A  !!!!!!!!
        print "Saving to new csvs..."
        print "Saving trainingSet to ../CSVs/finalTraining.csv..."
        trainingSet.to_csv('../CSVs/finalTraining.csv')
        print "Saving testingSet to ../CSVs/finalTesting.csv..."
        testingSet.to_csv('../CSVs/finalTesting.csv')
```