*#import some Python packages that are required:*

**from** **pyspark.sql** **import** SQLContext

**from** **pyspark.sql.types** **import** \*

**from** **pyspark.sql** **import** Row

**from** **pyspark.mllib.regression** **import** LabeledPoint

**from** **pyspark.sql.functions** **import** udf

**from** **pyspark.mllib.linalg** **import** Vectors

**from** **pyspark.mllib.classification** **import** LogisticRegressionWithLBFGS, LogisticRegressionModel

**from** **pyspark.ml.classification** **import** LogisticRegression

**from** **pyspark.ml.param** **import** Param, Params

**from** **pyspark.ml.feature** **import** OneHotEncoder, StringIndexer

**from** **pyspark.mllib.linalg** **import** Vectors

**from** **pyspark.mllib.regression** **import** LabeledPoint

**from** **pyspark.mllib.stat** **import** Statistics

**from** **pyspark.ml.feature** **import** VectorAssembler

**from** **IPython.display** **import** display

**from** **ipywidgets** **import** interact

**import** **sys**

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**import** **time**

**import** **datetime**

**import** **matplotlib.pyplot** **as** **plt**

**import** **os.path**

%matplotlib inline

textFile = sc.textFile('2008.csv')

*#remove the header of file*

textFileRDD = textFile.map(**lambda** x: x.split(','))

header = textFileRDD.first()

textRDD = textFileRDD.filter(**lambda** r: r != header)

**def** parse(r):

**try**:

x=Row(Year=int(r[0]),\

Month=int(r[1]),\

DayofMonth=int(r[2]),\

DayOfWeek=int(r[3]),\

DepTime=int(float(r[4])), \

CRSDepTime=int(r[5]),\

ArrTime=int(float(r[6])),\

CRSArrTime=int(r[7]), \

UniqueCarrier=r[8],\

DepDelay=int(float(r[15])),\

Origin=r[16],\

Dest=r[17], \

Distance=int(float(r[18])),\

CarrierDelay=int(float(r[24])),\

WeatherDelay=int(float(r[25])),\

NASDelay= int(float(r[26])),\

SecurityDelay=int(float(r[27])),\

LateAircraftDelay=int(float(r[28])))

**except**:

x=None

**return** x

rowRDD = textRDD.map(**lambda** r: parse(r)).filter(**lambda** r:r != None)

airline\_df = sqlContext.createDataFrame(rowRDD)

airline\_df = airline\_df.withColumn('DepDelayed', airline\_df['DepDelay']>15)

**def** hour\_ex(x):

h = int(str(int(x)).zfill(4)[:2])

**return** h

*# register as a UDF*

f = udf(hour\_ex, IntegerType())

*#CRSDepTime: scheduled departure time (local, hhmm)*

airline\_df = airline\_df.withColumn('hour', f(airline\_df.CRSDepTime))

airline\_df.registerTempTable("airlineDF")

cause\_delay = sqlContext.sql("SELECT sum(WeatherDelay) Weather,sum(NASDelay) NAS,sum(SecurityDelay) Security,sum(LateAircraftDelay) lateAircraft,sum(CarrierDelay) Carrier**\**

FROM airlineDF ")

df\_cause\_delay = cause\_delay.toPandas()

df\_cause\_delay.head()

groupedDelay = sqlContext.sql("SELECT Origin, count(\*) conFlight,avg(DepDelay) delay **\**

FROM airlineDF **\**

GROUP BY Origin")

df\_origin = groupedDelay.toPandas()

df\_origin.sort('delay',ascending=0).head()

df = pd.read\_csv('airports.dat', index\_col=0,\

names = ['name', 'city', 'country','IATA','ICAO','lat','lng','alt','TZone','DST','Tz'], \

header=0)

df\_airports = pd.merge(df\_origin, df, left\_on = 'Origin', right\_on = 'IATA')

df\_airports.head()

df\_airports.sort('delay',ascending=0).head()

**def** sigmoid(x):

**return** 1 / (1 + np.exp(-x))

**def** zscore(x):

**return** (x-np.average(x))/np.std(x)

**from** **mpl\_toolkits.basemap** **import** Basemap

**import** **matplotlib.pyplot** **as** **plt**

**from** **pylab** **import** rcParams

%matplotlib inline

rcParams['figure.figsize'] = (14,10)

my\_map = Basemap(projection='merc',

resolution = 'l', area\_thresh = 1000.0,

llcrnrlon=-130, llcrnrlat=22, *#min longitude (llcrnrlon) and latitude (llcrnrlat)*

urcrnrlon=-60, urcrnrlat=50) *#max longitude (urcrnrlon) and latitude (urcrnrlat)*

my\_map.drawcoastlines()

my\_map.drawcountries()

my\_map.drawmapboundary()

my\_map.fillcontinents(color = 'white', alpha = 0.3)

my\_map.shadedrelief()

*# To create a color map*

colors = plt.get\_cmap('hot')(np.linspace(0.0, 1.0, 30))

colors=np.flipud(colors)

*#----- Scatter -------*

countrange=max(df\_airports['conFlight'])-min(df\_airports['conFlight'])

al=np.array([sigmoid(x) **for** x **in** zscore(df\_airports['delay'])])

xs,ys = my\_map(np.asarray(df\_airports['lng']), np.asarray(df\_airports['lat']))

val=df\_airports['conFlight']\*4000.0/countrange

my\_map.scatter(xs, ys, marker='o', s= val, alpha = 0.8,color=colors[(al\*20).astype(int)])

*#----- Text -------*

df\_text=df\_airports[(df\_airports['conFlight']>60000) & (df\_airports['IATA'] != 'HNL')]

xt,yt = my\_map(np.asarray(df\_text['lng']), np.asarray(df\_text['lat']))

txt=np.asarray(df\_text['IATA'])

zp=zip(xt,yt,txt)

**for** row **in** zp:

*#print zp[2]*

plt.text(row[0],row[1],row[2], fontsize=10, color='blue',)

**print**("Every marker is an airport.")

**print**("dimension of markers: Airport Traffic (larger circle means higher number of flights in year)")

**print**("Color of markers: Average Flight Delay (More red means longer delays)")

plt.show()

grp\_rout\_Delay = sqlContext.sql("SELECT Origin, Dest, count(\*) traffic,avg(Distance) avgDist,**\**

avg(DepDelay) avgDelay**\**

FROM airlineDF **\**

GROUP BY Origin,Dest")

rout\_Delay = grp\_rout\_Delay.toPandas()

df\_airport\_rout1 = pd.merge(rout\_Delay, df, left\_on = 'Origin', right\_on = 'IATA')

df\_airport\_rout2 = pd.merge(df\_airport\_rout1, df, left\_on = 'Dest', right\_on = 'IATA')

df\_airport\_rout = df\_airport\_rout2[["Origin","lat\_x","lng\_x","Dest","lat\_y","lng\_y",\

"avgDelay", "traffic"]]

df\_airport\_rout.sort('avgDelay',ascending=0).head()

rcParams['figure.figsize'] = (14,10)

my\_map = Basemap(projection='merc',

resolution = 'l', area\_thresh = 1000.0,

llcrnrlon=-130, llcrnrlat=22, *#min longitude (llcrnrlon) and latitude (llcrnrlat)*

urcrnrlon=-60, urcrnrlat=50) *#max longitude (urcrnrlon) and latitude (urcrnrlat)*

my\_map.drawcoastlines()

my\_map.drawcountries()

my\_map.drawmapboundary()

my\_map.fillcontinents(color = 'white', alpha = 0.3)

my\_map.shadedrelief()

delay=np.array([sigmoid(x) **for** x **in** zscore(df\_airports["delay"])])

colors = plt.get\_cmap('hot')(np.linspace(0.0, 1.0, 40))

colors=np.flipud(colors)

xs,ys = my\_map(np.asarray(df\_airports['lng']), np.asarray(df\_airports['lat']))

xo,yo = my\_map(np.asarray(df\_airport\_rout['lng\_x']), np.asarray(df\_airport\_rout['lat\_x']))

xd,yd = my\_map(np.asarray(df\_airport\_rout['lng\_y']), np.asarray(df\_airport\_rout['lat\_y']))

my\_map.scatter(xs, ys, marker='o', alpha = 0.8,color=colors[(delay\*20).astype(int)])

al=np.array([sigmoid(x) **for** x **in** zscore(df\_airport\_rout["avgDelay"])])

f=zip(xo,yo,xd,yd,df\_airport\_rout['avgDelay'],al)

**for** row **in** f:

plt.plot([row[0],row[2]], [row[1],row[3]],'-',alpha=0.07, \

color=colors[(row[5]\*30).astype(int)] )

**for** row **in** zp:

plt.text(row[0],row[1],row[2], fontsize=10, color='blue',)

**print**("Each line shows a route from the starting to reaching airport.")

**print**("As red color increases probablity of delay increases.")

plt.show()

grp\_rout\_Delay = sqlContext.sql("SELECT Origin, Dest, count(\*) traffic,avg(Distance) avgDist,**\**

avg(DepDelay) avgDelay**\**

FROM airlineDF **\**

GROUP BY Origin,Dest")

rout\_Delay = grp\_rout\_Delay.toPandas()

df\_airport\_rout1 = pd.merge(rout\_Delay, df, left\_on = 'Origin', right\_on = 'IATA')

df\_airport\_rout2 = pd.merge(df\_airport\_rout1, df, left\_on = 'Dest', right\_on = 'IATA')

df\_airport\_rout = df\_airport\_rout2[["Origin","lat\_x","lng\_x","Dest","lat\_y","lng\_y",\

"avgDelay", "traffic"]]

df\_airport\_rout.sort('avgDelay',ascending=0).head()

rcParams['figure.figsize'] = (14,10)

my\_map = Basemap(projection='merc',

resolution = 'l', area\_thresh = 1000.0,

llcrnrlon=-130, llcrnrlat=22, *#min longitude (llcrnrlon) and latitude (llcrnrlat)*

urcrnrlon=-60, urcrnrlat=50) *#max longitude (urcrnrlon) and latitude (urcrnrlat)*

my\_map.drawcoastlines()

my\_map.drawcountries()

my\_map.drawmapboundary()

my\_map.fillcontinents(color = 'white', alpha = 0.3)

my\_map.shadedrelief()

delay=np.array([sigmoid(x) **for** x **in** zscore(df\_airports["delay"])])

colors = plt.get\_cmap('hot')(np.linspace(0.0, 1.0, 40))

colors=np.flipud(colors)

xs,ys = my\_map(np.asarray(df\_airports['lng']), np.asarray(df\_airports['lat']))

xo,yo = my\_map(np.asarray(df\_airport\_rout['lng\_x']), np.asarray(df\_airport\_rout['lat\_x']))

xd,yd = my\_map(np.asarray(df\_airport\_rout['lng\_y']), np.asarray(df\_airport\_rout['lat\_y']))

my\_map.scatter(xs, ys, marker='o', alpha = 0.8,color=colors[(delay\*20).astype(int)])

al=np.array([sigmoid(x) **for** x **in** zscore(df\_airport\_rout["avgDelay"])])

f=zip(xo,yo,xd,yd,df\_airport\_rout['avgDelay'],al)

**for** row **in** f:

plt.plot([row[0],row[2]], [row[1],row[3]],'-',alpha=0.07, \

color=colors[(row[5]\*30).astype(int)] )

**for** row **in** zp:

plt.text(row[0],row[1],row[2], fontsize=10, color='blue',)

**print**("Each line represents a route from the Origin to Destination airport.")

**print**("The redder line, the higher probablity of delay.")

plt.show()

Origin\_Airport="SJC"

df\_ORG = sqlContext.sql("SELECT \* from airlineDF WHERE origin='"+ Origin\_Airport+"'")

df\_ORG.registerTempTable("df\_ORG")

df\_ORG.select('ArrTime','CRSArrTime','CRSDepTime',\

'DayOfWeek','DayofMonth','DepDelay','DepTime','Dest').show(2)

**print** "total flights from this ariport: " + str(df\_ORG.count())

grp\_carr = sqlContext.sql("SELECT UniqueCarrier,month, avg(DepDelay) avgDelay from df\_ORG **\**

WHERE DepDelayed=True **\**

GROUP BY UniqueCarrier,month")

s = grp\_carr.toPandas()

ps = s.pivot(index='month', columns='UniqueCarrier', values='avgDelay')[['AA','UA','US']]

rcParams['figure.figsize'] = (8,5)

ps.plot(kind='bar', colormap='Greens');

plt.xlabel('Average delay')

plt.ylabel('Month')

plt.title('How much delay does each carrier has in each month?')

hour\_grouped = df\_ORG.filter(df\_ORG['DepDelayed']).select('DayOfWeek','hour','DepDelay').groupby('DayOfWeek','hour').mean('DepDelay')

rcParams['figure.figsize'] = (10,5)

dh = hour\_grouped.toPandas()

c = dh.pivot('DayOfWeek','hour')

X = c.columns.levels[1].values

Y = c.index.values

Z = c.values

plt.xticks(range(0,24), X)

plt.yticks(range(0,7), Y)

plt.xlabel('Hour of Day')

plt.ylabel('Day of Week')

plt.title('Average delay per hours and day?')

plt.imshow(Z)

*# Feature selection*

df\_model=df\_ORG

stringIndexer1 = StringIndexer(inputCol="Origin", outputCol="originIndex")

model\_stringIndexer = stringIndexer1.fit(df\_model)

indexedOrigin = model\_stringIndexer.transform(df\_model)

encoder1 = OneHotEncoder(dropLast=False, inputCol="originIndex", outputCol="originVec")

df\_model = encoder1.transform(indexedOrigin)

assembler = VectorAssembler(

inputCols = ['Year','Month','DayofMonth','DayOfWeek','hour','Distance','originVec'],

outputCol = "features")

output = assembler.transform(df\_model)

airlineRDD=output.map(**lambda** row: LabeledPoint([0,1][row['DepDelayed']],row['features']))

*# Spliting dataset into train and test dtasets*

trainRDD,testRDD=airlineRDD.randomSplit([0.7,0.3])

*# Build the model*

model = LogisticRegressionWithLBFGS.train(trainRDD)

*# Evaluating the model on testing data*

labelsAndPreds = testRDD.map(**lambda** p: (p.label, model.predict(p.features)))

**def** conf(r):

**if** r[0] == r[1] ==1: x= 'TP'

**if** r[0] == r[1] ==0: x= 'TN'

**if** r[0] == 1 **and** r[1] ==0: x= 'FN'

**if** r[0] == 0 **and** r[1] ==1: x= 'FP'

**return** (x)

acc1 = labelsAndPreds.map(**lambda** (v, p): ((v, p),1)).reduceByKey(**lambda** a, b: a + b).take(5)

acc = [(conf(x[0]),x[1]) **for** x **in** acc1]

TP=TN=FP=FN=0.0

**for** x **in** acc:

**if** x[0]=='TP': TP= x[1]

**if** x[0]=='TN': TN= x[1]

**if** x[0]=='FP': FP= x[1]

**if** x[0]=='FN': FN= x[1]

eps = sys.float\_info.epsilon

Accuracy = (TP+TN) / (TP + TN+ FP+FN+eps)

**print** "Model Accuracy for JFK: **%1.2f** **%%**" % (Accuracy\*100)

Destin = rout\_Delay[rout\_Delay['Origin']=='JFK'].Dest.unique()

@interact(Destination=tuple(Destin),Month=(1,12),DayOfWeek=(0,7),Hour=(0,23))

**def** g(Destination,Month,DayOfWeek,Hour):

Distance=int(rout\_Delay[(rout\_Delay['Origin']=='JFK') & (rout\_Delay['Dest']==Destination)]\

.avgDist.tolist()[0])

testcase=Row(Year=2007.0,Month=Month,DayofMonth=2.0,DayOfWeek=DayOfWeek,Hour=Hour,\

Origin='JFK',\

Dest=Destination,Distance=Distance)

TestCase\_df = sqlContext.createDataFrame(sc.parallelize([testcase]))

t1= model\_stringIndexer.transform(TestCase\_df)

t2=encoder2.transform(t1)

p=model.predict(assembler.transform(t2).take(1)[0]['features'])

**print** "Flight from JFK to "+Destination + ", Distance:" + str(Distance)

**if** p==0:

**print** "You flight doesnt have a delay, Accuracy= **%1.2f** **%%**" % (Accuracy\*100)

**else**:

**print** "You flight may be delayed, Accuracy= **%1.2f** **%%**" % (Accuracy\*100)