Introduction to Data Formats and S3

In [1]:

```
import pandas as pd
import numpy as np

import boto3
import sagemaker.amazon.common as smac
```

In [2]:

```
np.random.seed(5)
```

In [3]:

```
# NOTE: Specify your bucket
s3_bucket_name = 'dwb-ml-sagemaker'
```

Sample DataSet

Three features x1,x2,x3 and a target variable y

In [4]:

```
n = 10

x1 = np.random.random_sample(n)       # n floating point numbers between 0 and 1
x2 = np.random.randint(100,200,n)     # n integers
x3 = np.random.random_sample(n) * 10  # n floating point numbers between 0 and 10
y = np.random.randint(0,2,n)          # Response variable 0 or 1
```

In [5]:

```
y
```

Out[5]:

```
array([0, 0, 1, 1, 1, 1, 0, 0, 0, 1])
```

In [6]:

```
df = pd.DataFrame({'x1':x1,
              'x2':x2,
              'x3':x3,
              'y':y})
```

In [7]:

```
df
```

Out[7]:

	x1	x2	x3	y
0	0.221993	153	2.041547	0
1	0.870732	180	1.190954	0
2	0.206719	127	8.779031	1
3	0.918611	144	5.236753	1
4	0.488411	177	4.921360	1
5	0.611744	175	7.318711	1
6	0.765908	165	0.145808	0
7	0.518418	147	0.933630	0
8	0.296801	130	8.265542	0
9	0.187721	184	8.334927	1

In [8]:
```
# Write to SageMaker Notebook Instance
df.to_csv('demo_file.csv',index=False)
```

In [9]:
```
# Write and Reading from S3 is just as easy
# files are referred as objects in S3.
# file name is referred as key name in S3
# Files stored in S3 are automatically replicated across 3 different availability z
# in the region where the bucket was created.

# http://boto3.readthedocs.io/en/latest/guide/s3.html
def write_to_s3(filename, bucket, key):
    with open(filename,'rb') as f: # Read in binary mode
        return boto3.Session().resource('s3').Bucket(bucket).Object(key).upload_fil
```

In [10]:
```
# http://boto3.readthedocs.io/en/latest/guide/s3.html
def download_from_s3(filename, bucket, key):
    with open(filename,'wb') as f:
        return boto3.Session().resource('s3').Bucket(bucket).Object(key).download_f
```

In [11]:
```
write_to_s3('demo_file.csv', s3_bucket_name, 'data_format/demo_file.csv')
```

In [12]:
```
download_from_s3('demo_file_from_s3.csv',s3_bucket_name,'data_format/demo_file.csv'
```

RecordIO Format

We will use SageMaker SDK write_numpy_to_dense_tensor() method to create RecordIO files

Data Types: Int32, Float32, Float64

Reference: https://github.com/aws/sagemaker-python-sdk/blob/master/src/sagemaker/amazon/common.py

In [13]:

```
df.head()
```

Out[13]:

	x1	x2	x3	y
0	0.221993	153	2.041547	0
1	0.870732	180	1.190954	0
2	0.206719	127	8.779031	1
3	0.918611	144	5.236753	1
4	0.488411	177	4.921360	1

In [14]:

```
# X must be an array
X = df[['x1','x2','x3']].to_numpy()
```

In [15]:

```
X
```

Out[15]:

```
array([[2.21993171e-01, 1.53000000e+02, 2.04154748e+00],
       [8.70732306e-01, 1.80000000e+02, 1.19095357e+00],
       [2.06719155e-01, 1.27000000e+02, 8.77903071e+00],
       [9.18610908e-01, 1.44000000e+02, 5.23675290e+00],
       [4.88411189e-01, 1.77000000e+02, 4.92135999e+00],
       [6.11743863e-01, 1.75000000e+02, 7.31871100e+00],
       [7.65907856e-01, 1.65000000e+02, 1.45807511e-01],
       [5.18417988e-01, 1.47000000e+02, 9.33630336e-01],
       [2.96800502e-01, 1.30000000e+02, 8.26554249e+00],
       [1.87721229e-01, 1.84000000e+02, 8.33492742e+00]])
```

In [16]:

```
type(X)
```

Out[16]:

```
numpy.ndarray
```

In [17]:

```
# Response/Target variable needs to a vector
# y must be a vector
y = df[['y']].to_numpy()
```

In [18]:

```
# it is right now a array of dimensions 10x1
y.shape
```

Out[18]:

```
(10, 1)
```

In [19]:

```
y
```

Out[19]:

```
array([[0],
       [0],
       [1],
       [1],
       [1],
       [1],
       [0],
       [0],
       [0],
       [1]])
```

In [20]:
```
# Flatten to a single dimension array of 10 elements
y = y.ravel()
```

In [26]:
```
y
```

Out[26]:
```
array([0, 0, 1, 1, 1, 1, 0, 0, 0, 1])
```

In [23]:
```
def write_recordio_file (filename, x, y=None):
    with open(filename, 'wb') as f:
        smac.write_numpy_to_dense_tensor(f, x, y)
```

In [24]:
```
def read_recordio_file (filename, recordsToPrint = 10):
    with open(filename, 'rb') as f:
        record = smac.read_records(f)
        for i, r in enumerate(record):
            if i >= recordsToPrint:
                break
            print ("record: {}".format(i))
            print(r)
```

In [25]:
```
write_recordio_file('demo_file.recordio',X,y)
```

In [27]:
```
df.head(3)
```

Out[27]:

	x1	x2	x3	y
0	0.221993	153	2.041547	0
1	0.870732	180	1.190954	0
2	0.206719	127	8.779031	1

In [28]:
```
read_recordio_file('demo_file.recordio',3)
```

```
record: 0
features {
  key: "values"
  value {
    float64_tensor {
      values: 0.22199317108973948
      values: 153.0
      values: 2.0415474783059215
    }
  }
}
label {
  key: "values"
  value {
    int32_tensor {
      values: 0
    }
  }
}

record: 1
features {
  key: "values"
  value {
    float64_tensor {
      values: 0.8707323061773764
      values: 180.0
      values: 1.1909535747826039
    }
  }
}
label {
  key: "values"
  value {
    int32_tensor {
      values: 0
    }
  }
}

record: 2
features {
  key: "values"
  value {
    float64_tensor {
      values: 0.20671915533942642
      values: 127.0
      values: 8.779030712603621
    }
  }
}
label {
  key: "values"
  value {
    int32_tensor {
      values: 1
```

```
        }
    }
}
```

In [29]:

```
write_to_s3('demo_file.recordio', s3_bucket_name, 'data_format/demo_file.recordio')
```

In [30]:

```
download_from_s3('demo_file_from_s3.recordio',s3_bucket_name,'data_format/demo_file
```

In [ ]: