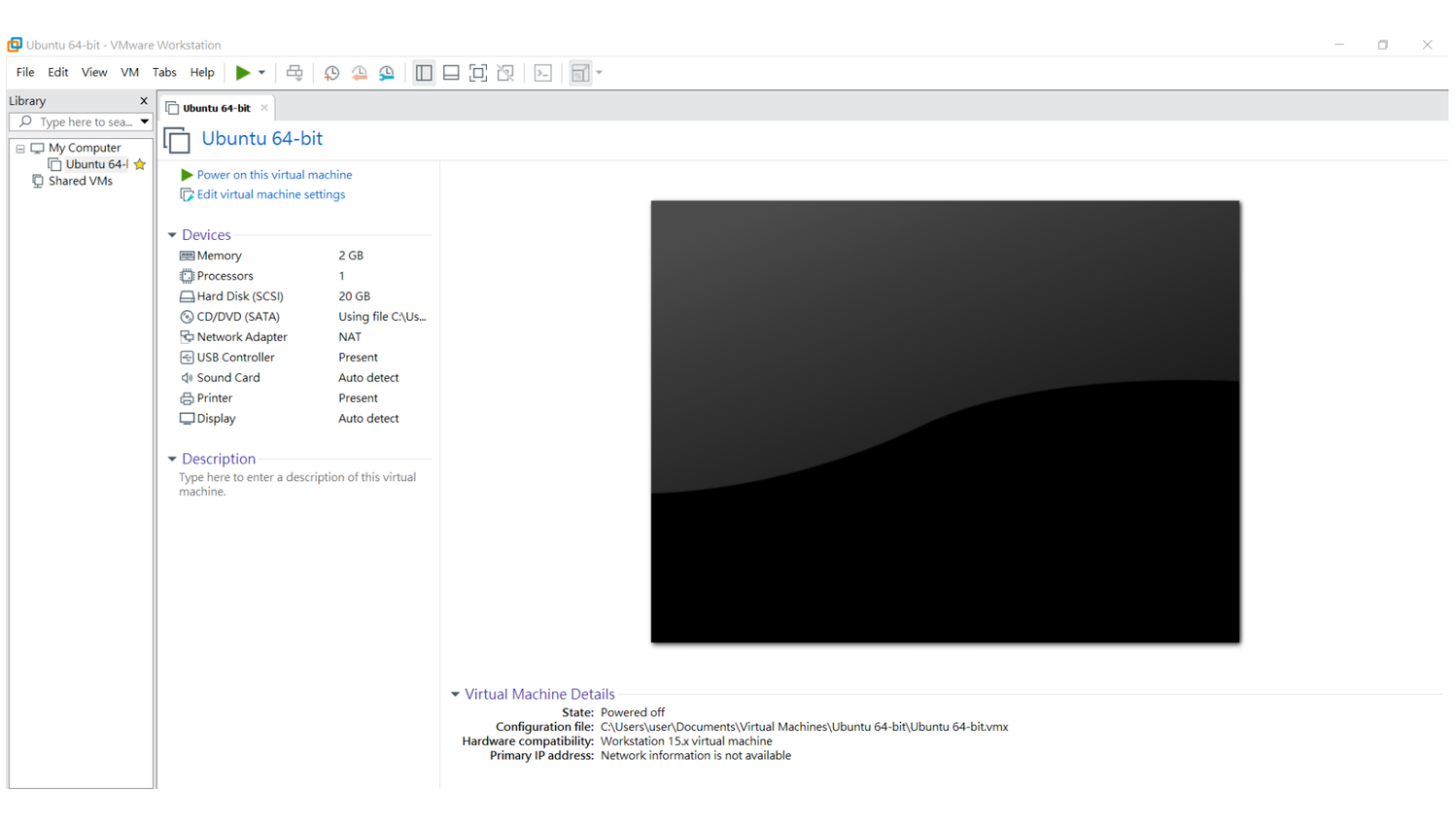
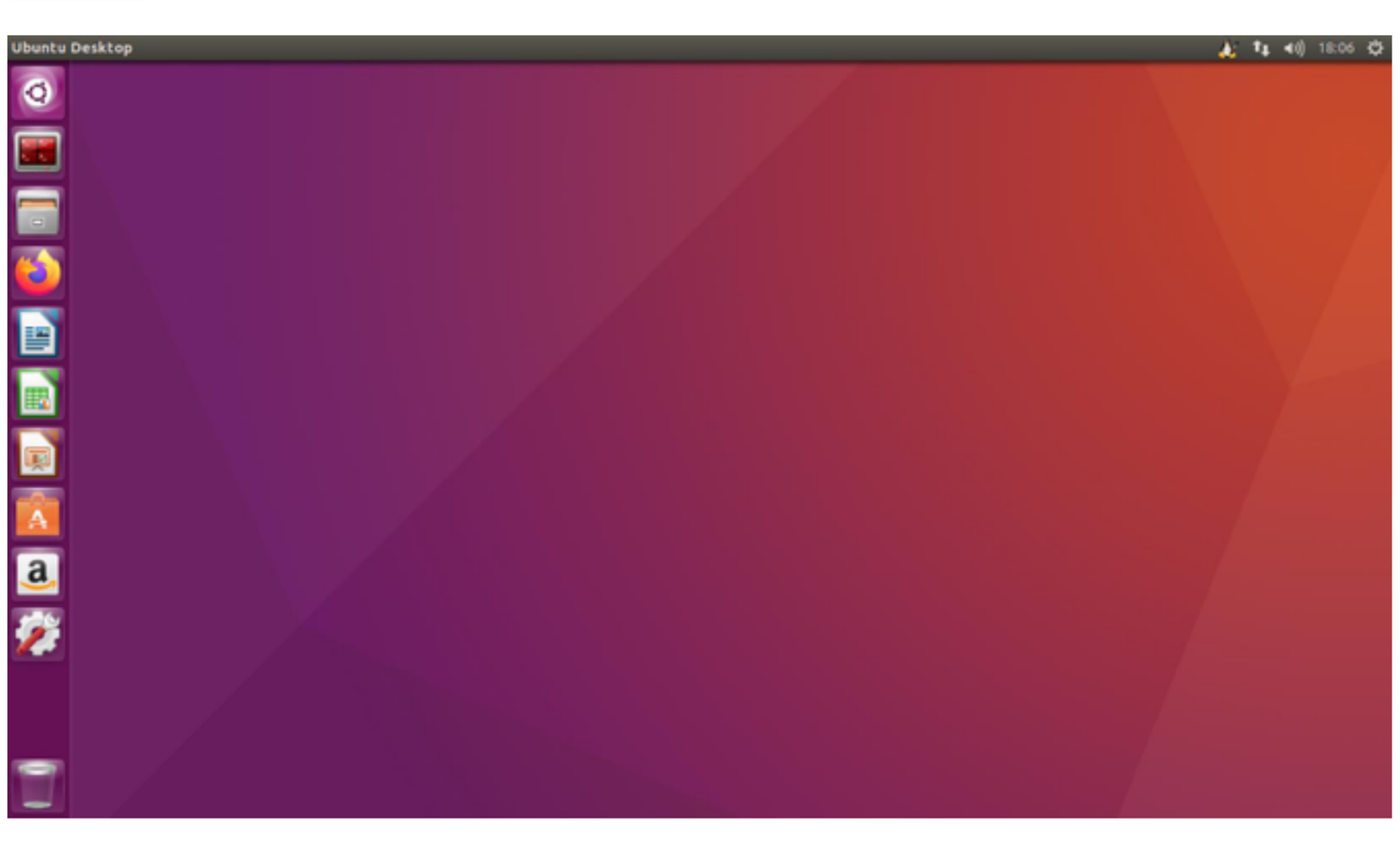
HW6: Spark

- Group 7
  - 309554001 劉雨恩
  - 309706013 黃柏元
  - 0613144 葉之晴
  - 309707007 黃如君

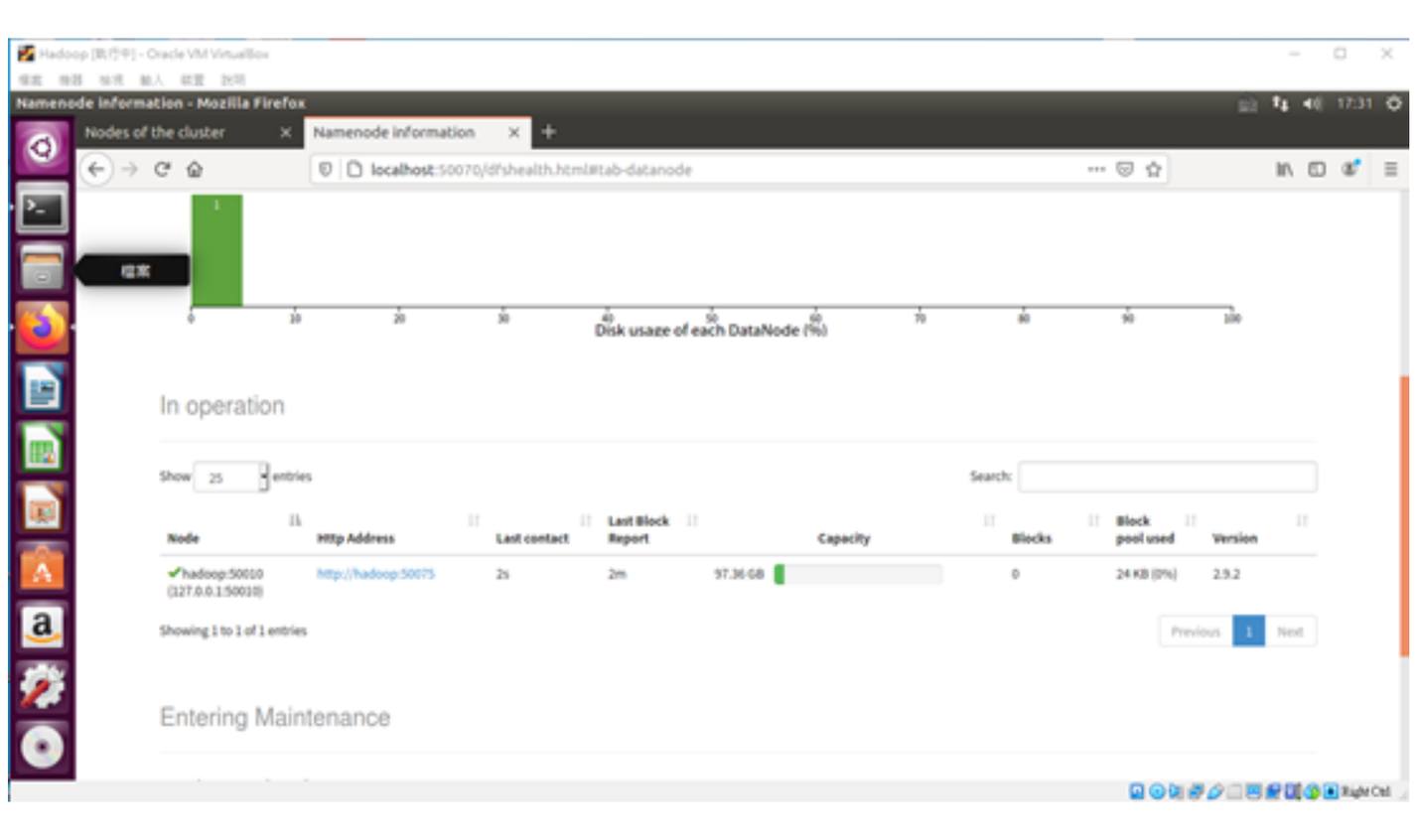
1. Hadoop & Spark 安裝

Step1. 虛擬機安裝

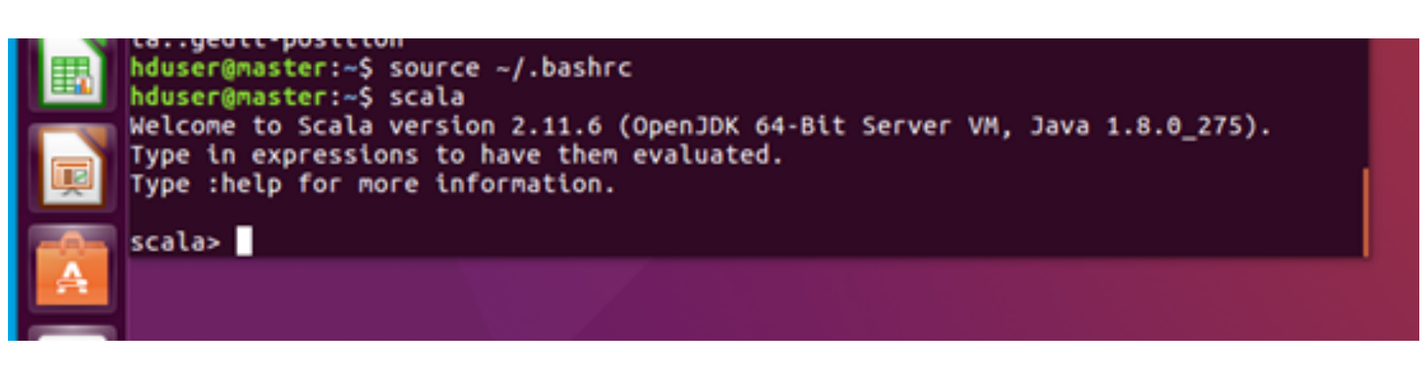




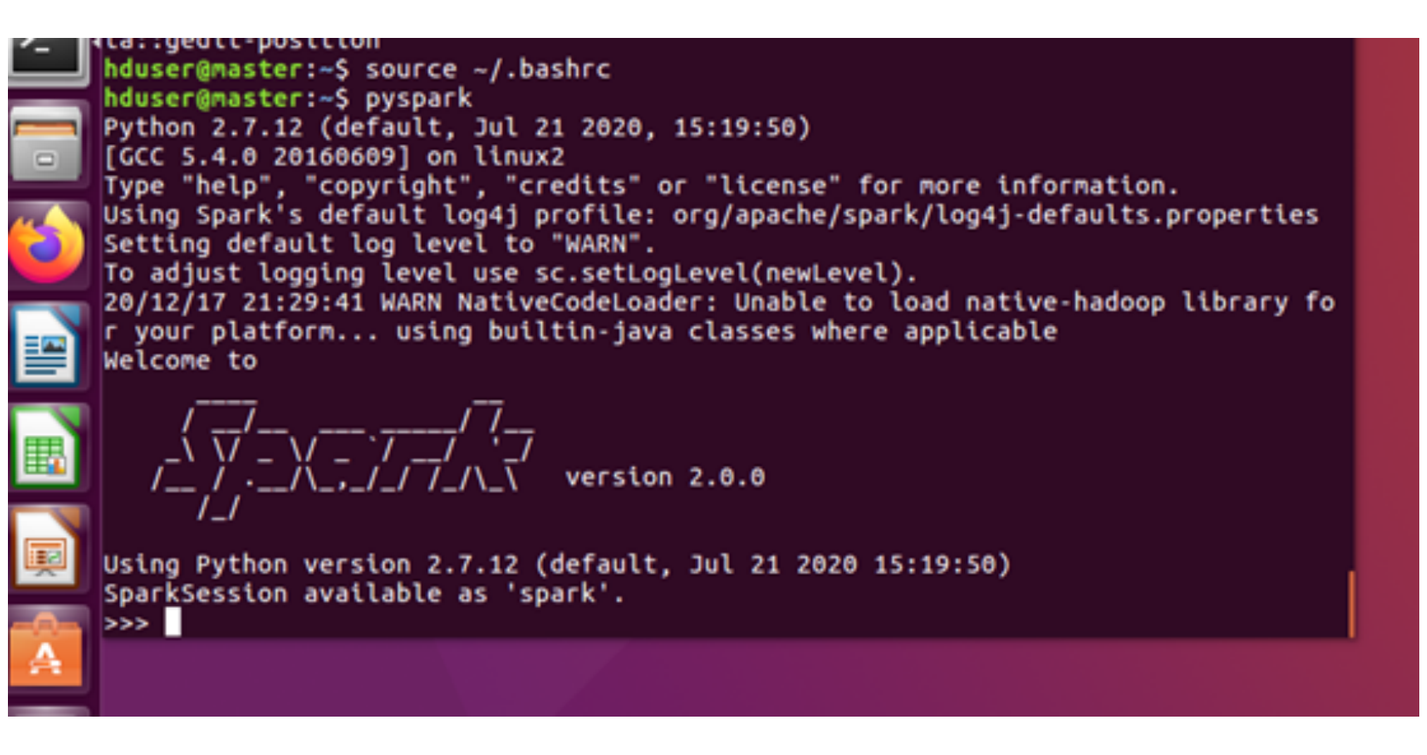
Step2. 安裝 Hadoop Single Node Cluster 並啟動 Datanode



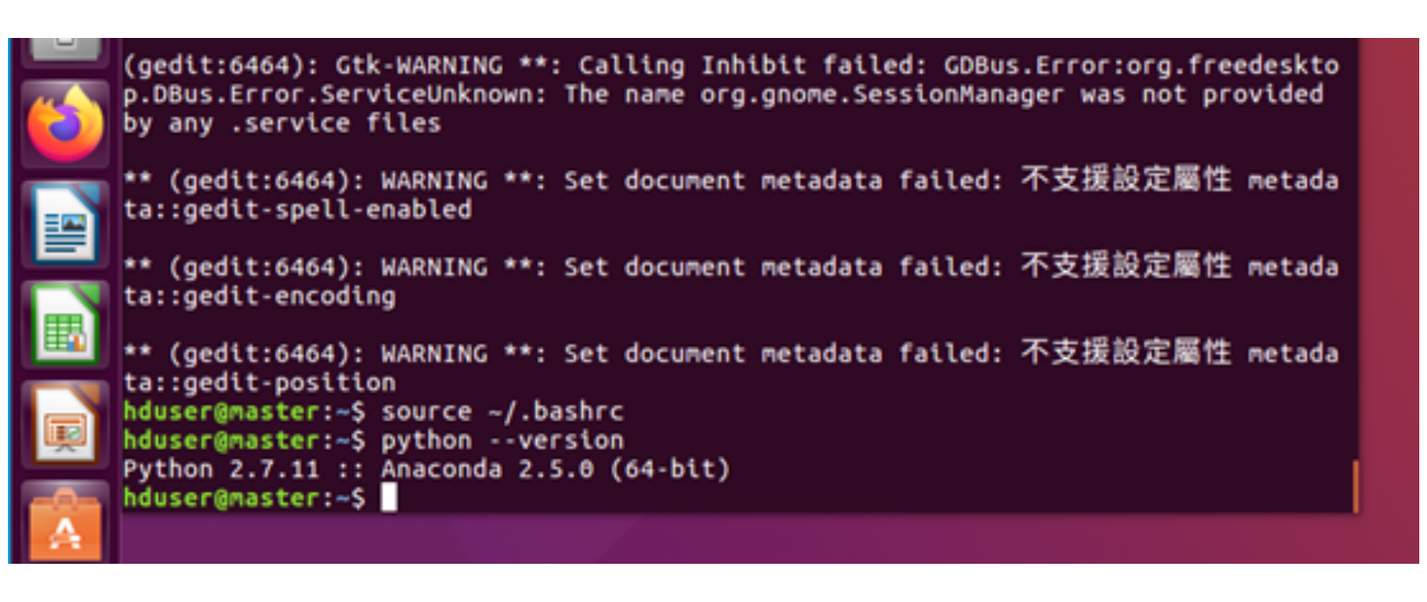
Step3. 安裝 Scala



Step4. 安裝 Spark



Step5. 安裝 Anaconda



2. Data Preprocessing

2.1 Read Data, drop unused columns, and fill nan value

```python
import pandas as pd

data = pd.read_csv("train.csv", sep = ',')
data = data.drop(columns = ['Descript', 'Resolution', 'Address'])
data.fillna(0, inplace=True)
data.info()
data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 878049 entries, 0 to 878048
Data columns (total 6 columns):
Dates         878049 non-null object
Category      878049 non-null object
DayOfWeek     878049 non-null object
PdDistrict    878049 non-null object
X             878049 non-null float64
Y             878049 non-null float64
dtypes: float64(2), object(4)
memory usage: 40.2+ MB
```

	Dates	Category	DayOfWeek	PdDistrict	X	Y
0	2015-05-13 23:53:00	WARRANTS	Wednesday	NORTHERN	-122.425892	37.774599
1	2015-05-13 23:53:00	OTHER OFFENSES	Wednesday	NORTHERN	-122.425892	37.774599
2	2015-05-13 23:33:00	OTHER OFFENSES	Wednesday	NORTHERN	-122.424363	37.800414
3	2015-05-13 23:30:00	LARCENY/THEFT	Wednesday	NORTHERN	-122.426995	37.800873
4	2015-05-13 23:30:00	LARCENY/THEFT	Wednesday	PARK	-122.438738	37.771541
...	...	...	...	...	...	...
878044	2003-01-06 00:15:00	ROBBERY	Monday	TARAVAL	-122.459033	37.714056
878045	2003-01-06 00:01:00	LARCENY/THEFT	Monday	INGLESIDE	-122.447364	37.731948
878046	2003-01-06 00:01:00	LARCENY/THEFT	Monday	SOUTHERN	-122.403390	37.780266
878047	2003-01-06 00:01:00	VANDALISM	Monday	SOUTHERN	-122.390531	37.780607
878048	2003-01-06 00:01:00	FORGERY/COUNTERFEITING	Monday	BAYVIEW	-122.394926	37.738212

878049 rows × 6 columns

2.2 Change `DayofWeek` & `PdDistrict` to dummy features

```python
# Change 'DayofWeek' to dummy features
all_dum = pd.get_dummies(pd.Series(data['DayOfWeek']))
new_data = pd.concat([data, all_dum], axis=1).drop(columns=['DayOfWeek'])

# Change 'PdDistrict' to dummy features
all_dum = pd.get_dummies(pd.Series(data['PdDistrict']))
new_data = pd.concat([new_data, all_dum], axis=1) \
            .drop(columns=['PdDistrict'])
new_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 878049 entries, 0 to 878048
Data columns (total 21 columns):
Dates         878049 non-null object
Category      878049 non-null object
X             878049 non-null float64
Y             878049 non-null float64
Friday        878049 non-null uint8
Monday        878049 non-null uint8
Saturday      878049 non-null uint8
Sunday        878049 non-null uint8
Thursday      878049 non-null uint8
Tuesday       878049 non-null uint8
Wednesday     878049 non-null uint8
BAYVIEW       878049 non-null uint8
CENTRAL       878049 non-null uint8
INGLESIDE     878049 non-null uint8
MISSION       878049 non-null uint8
NORTHERN      878049 non-null uint8
PARK          878049 non-null uint8
RICHMOND      878049 non-null uint8
SOUTHERN      878049 non-null uint8
TARAVAL       878049 non-null uint8
TENDERLOIN    878049 non-null uint8
dtypes: float64(2), object(2), uint8(17)
memory usage: 41.0+ MB
```

2.3 Take hour from `Dates` & change to dummy features

```python
new_data['Dates'] = new_data['Dates'].str[11:-6]
all_dum = pd.get_dummies(pd.Series(new_data['Dates']))
new_data = pd.concat([new_data, all_dum], axis=1).drop(columns=['Dates'])
new_data.info()

# output preprocessed data csv
new_data.to_csv('data/train.csv')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 878049 entries, 0 to 878048
Data columns (total 44 columns):
Category      878049 non-null object
X             878049 non-null float64
Y             878049 non-null float64
Friday        878049 non-null uint8
Monday        878049 non-null uint8
Saturday      878049 non-null uint8
Sunday        878049 non-null uint8
Thursday      878049 non-null uint8
Tuesday       878049 non-null uint8
Wednesday     878049 non-null uint8
BAYVIEW       878049 non-null uint8
CENTRAL       878049 non-null uint8
INGLESIDE     878049 non-null uint8
MISSION       878049 non-null uint8
NORTHERN      878049 non-null uint8
PARK          878049 non-null uint8
RICHMOND      878049 non-null uint8
SOUTHERN      878049 non-null uint8
TARAVAL       878049 non-null uint8
TENDERLOIN    878049 non-null uint8
00            878049 non-null uint8
01            878049 non-null uint8
02            878049 non-null uint8
03            878049 non-null uint8
04            878049 non-null uint8
05            878049 non-null uint8
06            878049 non-null uint8
07            878049 non-null uint8
08            878049 non-null uint8
09            878049 non-null uint8
10            878049 non-null uint8
11            878049 non-null uint8
12            878049 non-null uint8
13            878049 non-null uint8
14            878049 non-null uint8
15            878049 non-null uint8
16            878049 non-null uint8
17            878049 non-null uint8
18            878049 non-null uint8
19            878049 non-null uint8
20            878049 non-null uint8
21            878049 non-null uint8
22            878049 non-null uint8
23            878049 non-null uint8
dtypes: float64(2), object(1), uint8(41)
memory usage: 54.4+ MB
```

3. Model Training

Spark

訓練模型 `DecisionTree.trainClassifier`，並計算 `Accuracy`、`Precision` 和 `Recall`。

1. 將處理好的資料 (`data/train.csv`) load 進來

```python
print(" Load  Data... ")
rawDataWithHeader = sc.textFile(Path+"data/train.csv")
header = rawDataWithHeader.first()
rawData = rawDataWithHeader.filter(lambda x:x != header)
rData = rawData.map(lambda x: x.replace("\"",""))
lines = rData.map(lambda x: x.split(","))
print("共計：" + str(lines.count()) + "筆")
lines.take(10)
```

```
Load  Data...
共計：878049筆
```

```
lines.take(10)
```

```
[[u'23', u'6', u'4', u'-122.4258917', u'37.7745986', u'37'],
 [u'23', u'6', u'4', u'-122.4258917', u'37.7745986', u'21'],
 [u'23', u'6', u'4', u'-122.424363', u'37.80041432', u'21'],
 [u'23', u'6', u'4', u'-122.4269953', u'37.80087263', u'16'],
 [u'23', u'6', u'5', u'-122.4387376', u'37.77154117', u'16'],
 [u'23', u'6', u'2', u'-122.4032524', u'37.7134307', u'16'],
 [u'23', u'6', u'2', u'-122.423327', u'37.72513804', u'36'],
 [u'23', u'6', u'0', u'-122.3712743', u'37.72756407', u'36'],
 [u'23', u'6', u'6', u'-122.508194', u'37.77660126', u'16'],
 [u'23', u'6', u'1', u'-122.4190877', u'37.80780155', u'16']]
```

2. Construct RDD[LabeledPoint]

```python
labelpointRDD = lines.map(lambda r:LabeledPoint(extract_label(r), \
                                    extract_features(r,len(r) - 1)))
print ("labelpointRDD = ",labelpointRDD.first(),"\n")
```

```
('labelpointRDD = ', LabeledPoint(37.0, [23.0,6.0,4.0,-122.4258917,37.7745986]), '\n')
```

3. Randomly divide the data into 3 parts: train (80%), validation (10%), and test (10%)

```python
(trainData, validationData, testData) = labelpointRDD.randomSplit([8, 1, 1])
print(" trainData : " + str(trainData.count()) +
      " validationData : " + str(validationData.count()) +
      " testData : " + str(testData.count()))
```

```
trainData : 702525   validationData : 87520   testData : 88004
```

4. 訓練模型 `pyspark.mllib.tree.DecisionTree.trainClassifier`，並計算 `Accuracy`、`Precision` 和 `Recall`

```python
# train model
from pyspark.mllib.tree import DecisionTree
from pyspark.mllib.evaluation import MulticlassMetrics
model = DecisionTree.trainClassifier(trainData, numClasses=39,
                                     categoricalFeaturesInfo={},
                                     impurity="entropy",
                                     maxDepth=10, maxBins=10)

# evaluate model
def evaluateModel(model, validationData):
    score = model.predict(validationData.map(lambda p: p.features))
    scoreAndLabels = score.zip(validationData.map(lambda p: p.label))
    metrics = MulticlassMetrics(scoreAndLabels)
    accuracy = metrics.accuracy
    recall = metrics.recall()
    precision = metrics.precision()
    print "Accuracy = " , str(accuracy)
    print "Recall = ", str(recall)
    print "Precision = ", str(precision)

evaluateModel(model, validationData)
```

One computer

1. Randomly split to traing (75%) and test (25%) dataset

```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
                                    new_data.drop(columns=['Category']),
                                    new_data['Category'], test_size=0.25,
                                    random_state=42)
```

2. 訓練模型 `sklearn.tree.DecisionTreeClassifier`，並計算 `Accuracy`、`Precision` 和 `Recall`

```python
from sklearn import tree

# train model & predict
clf = tree.DecisionTreeClassifier()
clf = clf.fit(x_train, y_train)
y_test_pred = clf.predict(x_test)

from sklearn.metrics import accuracy_score, precision_score, recall_score

# Calculate & plot the Confusion Matrix
acc = accuracy_score(y_test, y_test_pred)
prec = precision_score(y_test, y_test_pred, average='macro')
recall = recall_score(y_test, y_test_pred, average='macro')
print('Accuracy: {}, Precision: {}, Recall: {}'.format(acc, prec, recall))
```

4. Results

Value	Spark	One computer
Accuracy	0.229124771481	0.24598543138675158
Precision	0.229124771481	0.10776022682136717
Recall	0.229124771481	0.10627600568659873