

Large Scale NYX Taxi Data Analysis

Yue Wen

Step1:

Download the data from:

https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2016-01.csv

We use 'wget' to download the data and store it into the cluster.

```
wget https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2016-01.csv
gcloud compute copy-files yellow_tripdata_2016-01.csv cluster-1-m:~/
hadoop fs -put yellow_tripdata_2016-01.csv
```

Step2:

Construct a bounding box for New York City (<http://boundingbox.klokantech.com/>) and remove any data points that have coordinates outside this box. Use the box: westlimit=-74.2635; southlimit=40.4856; eastlimit=-73.7526; northlimit=40.9596

We simply used a filter function to realize the purpose.

Step3:

Divide the drop off and pickup zones into bins.

Here, we use a trick to create the bucket.

Firstly, perform a linear projection such that westlimit=-74.2635 is mapped to 0 and southlimit=40.4856 is mapped to 20, then all the datapoints' longitude falls between interval (0,20), then we ceil the double to integer. Then all the data points' longitude falls into set {0,1,2,3,..19}.

Similarly, we can perform another linear transformation to map all the data points' latitude to integer {0,1,2,3,4,5,6,7,8,9}.

Then we use the $\text{map}(\text{longitude}) * 10 + (\text{latitude})$ to get the 200 bucket from 0 to 199.

The overall formula is displayed as following:

```
((p.long.toDouble*39.1466+2907.16353).toInt)*10 + (p.lai.toDouble*21.09705-854.1267).toInt
```

Then we apply this function to both pick up coordinate and drop off coordinate to get the zone.

Step4: Data analysis

- Top five drop off zones ordered by high average tip.

```
res12: scala.collection.immutable.ListMap[Int,Double] = Map(2 -> 28.725882352941174, 17 -> 20.692857142857147, 7 -> 20.325, 9 -> 18.0, 79 -> 16.742619047619048, 3 -> 16.557499999999997, 6 -> 16.33238095238095, 0 -> 14.472424242424243, 199 -> 13.341833333333334, 49 -> 13.0425, 10 -> 12.9648, 5 -> 12.752307692307692, 109 -> 12.633846153846154, 37 -> 12.42153846153846, 77 -> 11.95566037735849, 24 -> 11.909313725490197, 170 -> 11.67, 11 -> 11.376666666666665, 68 -> 11.218181818181819, 30 -> 11.098571428571429, 38 -> 10.961666666666666, 21 -> 10.788166666666666, 65, 19 -> 10.751428571428571, 34 -> 10.54260983564623, 55 -> 10.349722222222221, 44 -> 10.316764705882353, 179 -> 10.180473933649289, 31 -> 10.101368421052632, 197 -> 10.095, 99 -> 9.99735294117647, 27 -> 9.982750000000001, 78 -> 9.81328...
```

Then the top five zone is:2, 17, 7,9,79

- Most frequent pickup zone

```
res15: scala.collection.immutable.ListMap[Int,Long] = Map(105 -> 3974957, 115 -> 2699747, 116 -> 1271914, 126 -> 756294, 104 -> 523064, 94 -> 328651, 183 -> 191346, 125 -> 186073, 156 -> 161758, 106 -> 140851, 155 -> 96918, 114 -> 69603, 95 -> 48477, 193 -> 47116, 135 -> 45207, 124 -> 34781, 127 -> 32629, 145 -> 17093, 136 -> 11866, 146 -> 8464, 113 -> 8273, 103 -> 7379, 134 -> 6276, 174 -> 3770, 137 -> 3695, 123 -> 3270, 173 -> 1914, 165 -> 1876, 164 -> 1831, 138 -> 1500, 117 -> 1490, 133 -> 1295, 93 -> 990, 147 -> 970, 112 -> 810, 184 -> 768, 144 -> 751, 143 -> 676, 85 -> 673, 157 -> 654, 34 -> 618, 148 -> 543, 154 -> 538, 194 -> 537, 175 -> 510, 64 -> 437, 122 -> 352, 158 -> 332, 92 -> 321, 102 -> 319, 84 -> 314, 167 -> 307, 153 -> 251, 163 -> 204, 96 -> 154, 166 -> 134, 75 -> 128, 1...
```

105,115,116,126,104

- Most frequent drop-off zone

```
scala> ListMap(drop_off_result.toSeq.sortWith(_.2>_.2):_*)
res17: scala.collection.immutable.ListMap[Int,Long] = Map(105 -> 3448303, 115 -> 2577581, 116 -> 1258961, 126 -> 912419, 104 -> 540144, 94 -> 331245, 125 -> 236819, 106 -> 175751, 114 -> 161313, 127 -> 107269, 124 -> 106405, 135 -> 98136, 156 -> 85930, 95 -> 74480, 183 -> 74252, 155 -> 52171, 113 -> 49678, 145 -> 40302, 134 -> 39509, 136 -> 38069, 103 -> 26808, 123 -> 24501, 137 -> 24317, 34 -> 15637, 138 -> 14651, 193 -> 13132, 164 -> 10918, 165 -> 10254, 174 -> 8925, 146 -> 8888, 93 -> 8755, 147 -> 8557, 144 -> 8217, 133 -> 7069, 148 -> 6781, 157 -> 6590, 112 -> 6104, 175 -> 6005, 154 -> 5401, 122 -> 5204, 92 -> 4857, 184 -> 4402, 102 -> 4153, 143 -> 3874, 173 -> 3754, 158 -> 3659, 167 -> 3414, 185 -> 2984, 117 -> 2671, 194 -> 2519, 85 -> 2084, 195 -> 1974, 82 -> 1697, 176 -> 1572, 16...
```

105,115,116,126,104

- Histogram

The most frequent one is 105

After we filter the data, the result is as below.

```
scala> tip_rate.histogram( Array(0.0, 0.05,0.1, 0.15,0.2, 0.25,0.3,0.35,0.4,0.45,0.5))
res27: Array[Long] = Array(1229851, 78581, 209879, 190464, 1146568, 358862, 154030, 46920, 13301, 5926)

scala> tip_rate.count()
res28: Long = 3446725

scala> val test = 1229851+78581+209879+190464+1146568+358862+154030+46920+13301+5926
test: Int = 3434382
```

Note here, we only check the tip rate arrange from 0 to 0.5, and the last two results shows that we do not lose too much data points by doing so.

Step5: Logistic Regression

Intercept term: 0.0

```
scala> model.intercept
res45: Double = 0.0
```

The coefficient of (pick up zone, drop off zone, trip distance, passenger count and payment type) is ([0.06369797187134343,

0.053406770692665737,

-1.3580632628969216E-8,

0.08314692711037643,

-9.952407913717822])

```
scala> model.weights  
res46: org.apache.spark.mllib.linalg.Vector = [0.06369797187134343,0.053406770692665737,-1.3580632628969216E-8,0.08314692711037643,-9.952407913717822]
```

We might also be interested in the other two performance measures.

See the results below.

```
scala> precision.collectAsMap.maxBy(_._2)  
res50: (Double, Double) = (0.9139650697297554,0.9652097885856893)
```

```
scala> val auROC = metrics.areaUnderROC  
auROC: Double = 0.9661462061881627
```