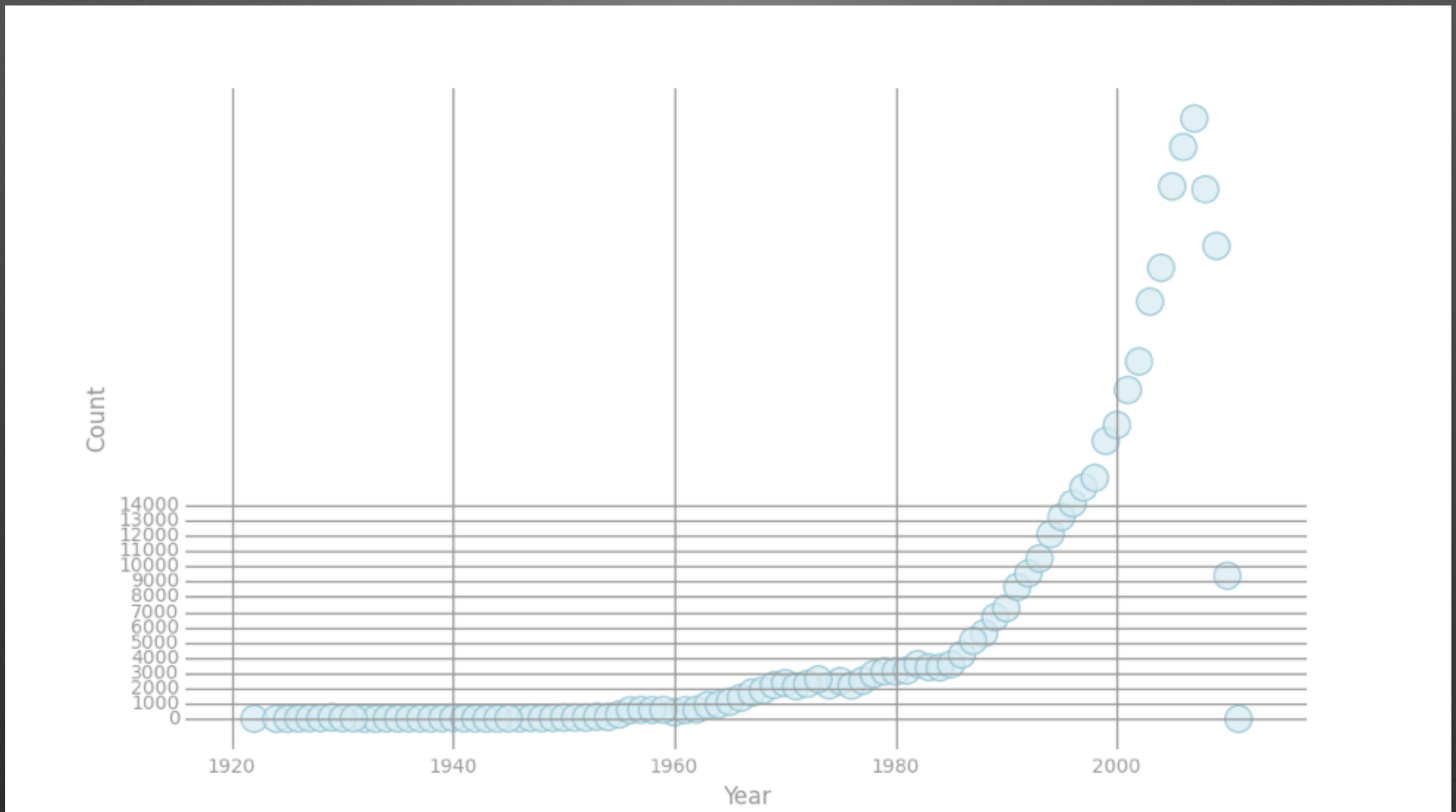
Big Data And Data Analysis

case study.

CASE 1

- Data Visualization
- Data Splitting
- Data Scaling
- Feature Reduction
- Model Selection
- Parameter Tuning
- Evaluation

Data Visualization:



Split the Dataset

```
weights = [.8, .1, .1]
seed = 42
parsedTrainData, parsedValData, parsedTestData =
parsedData.randomSplit(weights, seed)
parsedTrainData.cache()
parsedValData.cache()
parsedTestData.cache()
nTrain = parsedTrainData.count ()
nVal = parsedValData.count()
nTest = parsedTestData.count()
```

Data Scaling

```
label = data.map(lambda x: x.label)
features = data.map(lambda x: x.features)
 scaler1 = StandardScaler().fit(features)
 scaler2 = StandardScaler(withMean=True,
withStd=True).fit(features)
```

Feature Reduction

PCA Correlation

```
Feature Correlation
15/07/11 10:47:05 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
15/07/11 10:47:05 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
0       0.223058
1       0.283844
2       -0.238186
3       0.0517251
4       0.274186
5       -0.270314
6       0.11157
7       -0.199703
8       0.125478
9       0.0688658
10      0.0856964
11      0.0839962
12      -0.158992
13      0.0309892
14      -0.213706
15      -0.0652509
16      0.0289546
17      -0.0875178
18      -0.089637
19      0.155469
20      -0.179663
21      0.100631
22      -0.0365329
23      -0.15303
24      -0.102902
25      -0.217903
26      -0.0801444
```

Feature Reduction

```
[0, 1, 2, 4, 5, 7, 12, 14, 19, 20, 23, 25, 28, 35, 37, 39, 41, 45, 46, 47, 49, 52, 53, 54, 56, 59, 65, 66, 67, 68, 72, 76, 78, 80, 81, 84]
```

Model Selection(Pipeline API)

Regularizer: [1e-10, 1e-5, 1]

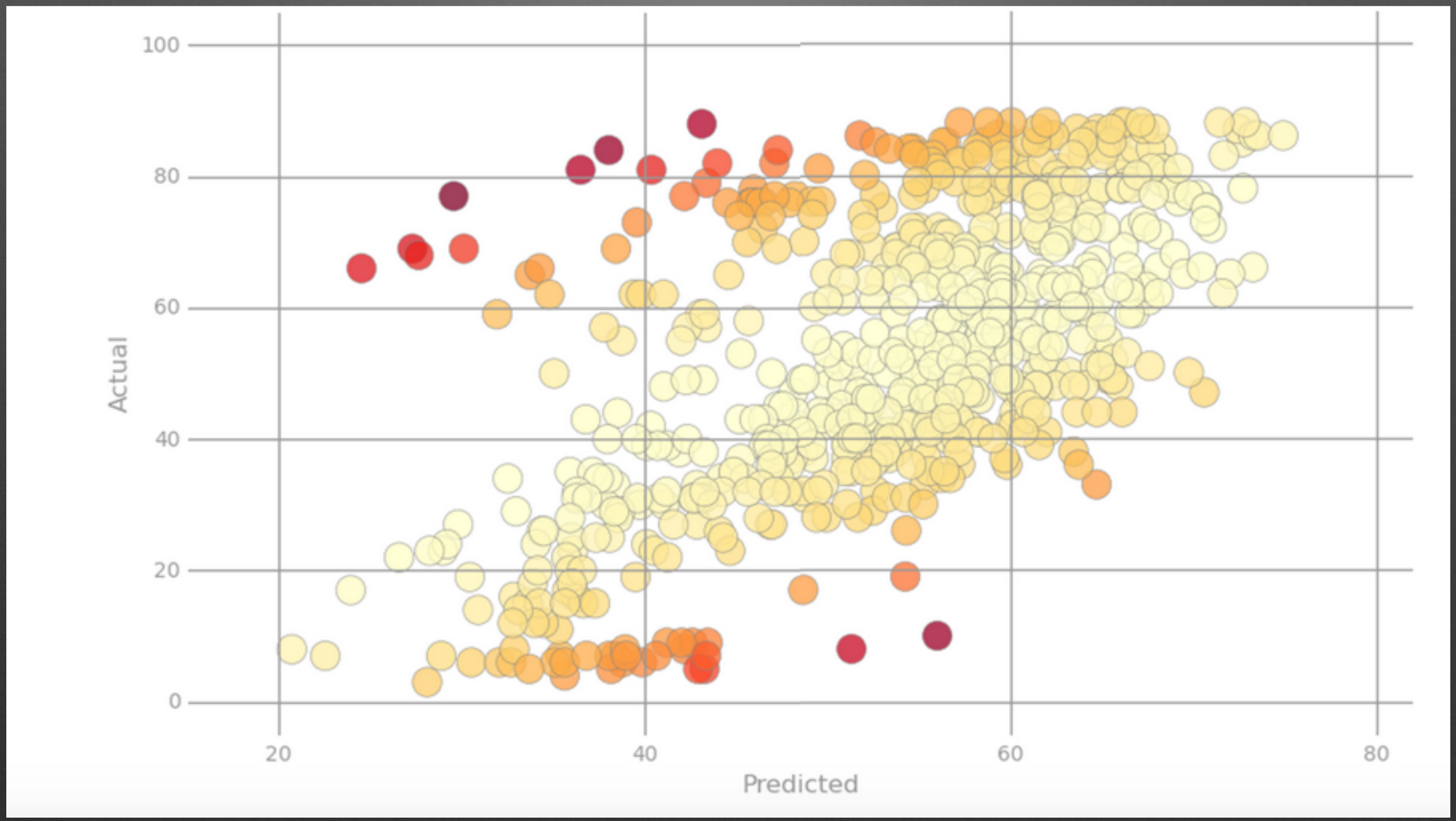
Alph: [1e-5, 10]

Iterations: [5,100,500]

Evaluation

```
alpha = 1e-05, numIters = 5, RMSE = 56.970
alpha = 1e-05, numIters = 500, RMSE = 56.893
alpha = 1e+01, numIters = 5, RMSE = 355124769.961
alpha = 1e+01, numIters = 500, RMSE = 331122904139598097689562654420421957284401080954235831
38219793939693394522086794593018370837166847970517683562822919782 40.000
```

Result



CASE 3

- Load LIBSVM File
- Data Scaling
- KMeans Model
- GaussianMixture Model

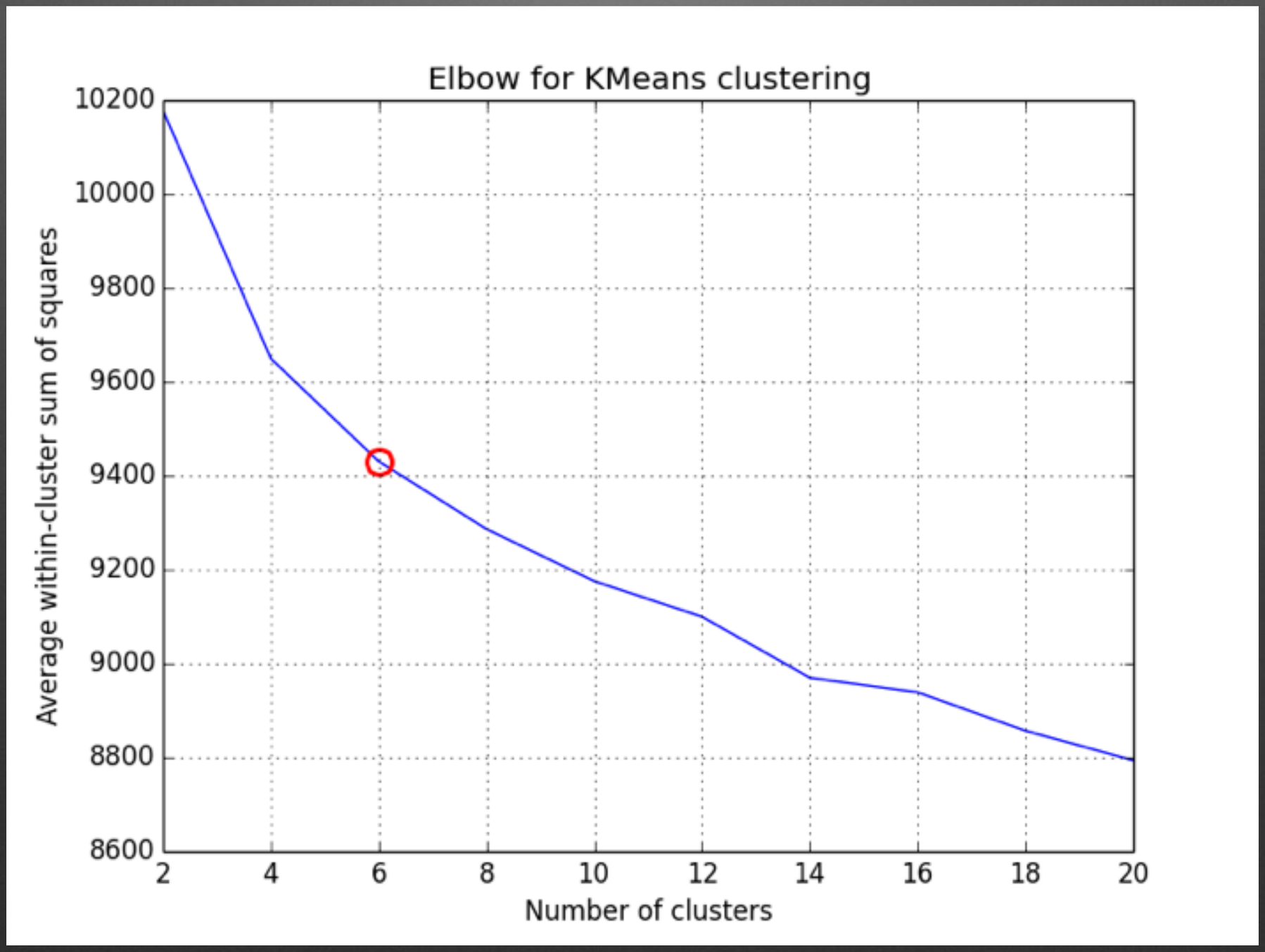
Load Data

```
data = MLUtils.loadLibSVMFile(sc, sys.argv[1])
```

Data Scaling

```
label = data.map(lambda x: x.label)
features = data.map(lambda x: x.features)
scaler1 = StandardScaler().fit(features)
 scaler2 = StandardScaler(withMean=True,
withStd=True).fit(features)
 data2 = label.zip(scaler1.transform(features.map(lambda
x: Vectors.dense(x.toArray()))))
```

KMeans


Elbow for KMeans clustering
Average within-cluster sum of squares
Number of clusters
10200
10000
9800
9600
9400
9200
9000
8800
8600
2
4
6
8
10
12
14
16
18
20

- GaussianMixture Model

- ('weight = ', 0.13252511046711482, 'mu = ', DenseVector([-0.6232, 0.557, 0.6411, -0.1042, -0.1961, 0.282, -0.2823, 0.2458]),

- ('weight = ', 0.30912004194106 49, 'mu = ', DenseVector([1.4278, 0.1049, -0.2541, 0.4918, 0.6913, -0.2621, 0.7239, 0.0676])

Result

- It seems 6 cluster could get the best result
- GaussianMixture consume more memory