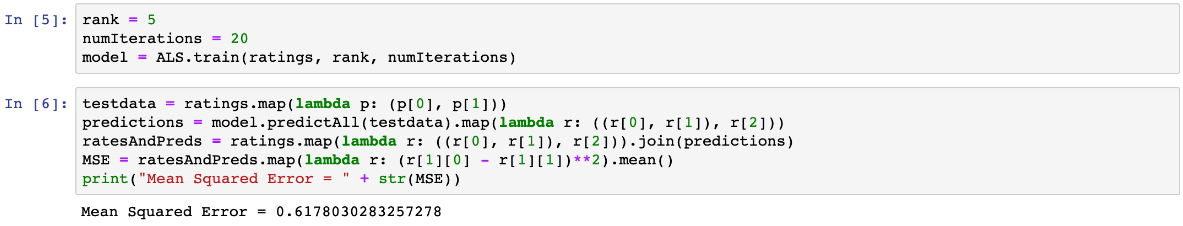
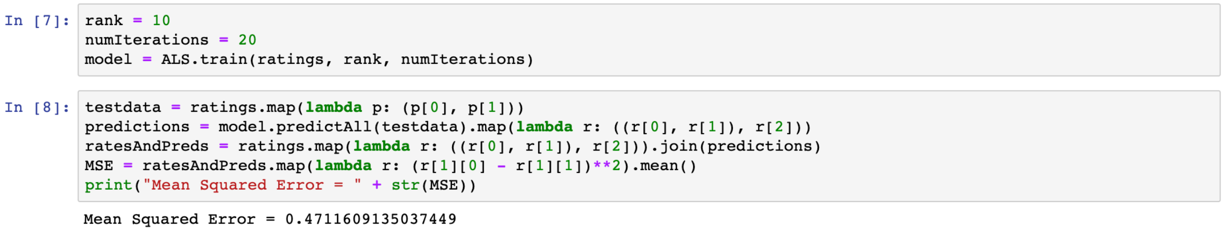
I. Test the MSE values with rank=5, 10, 20, 30



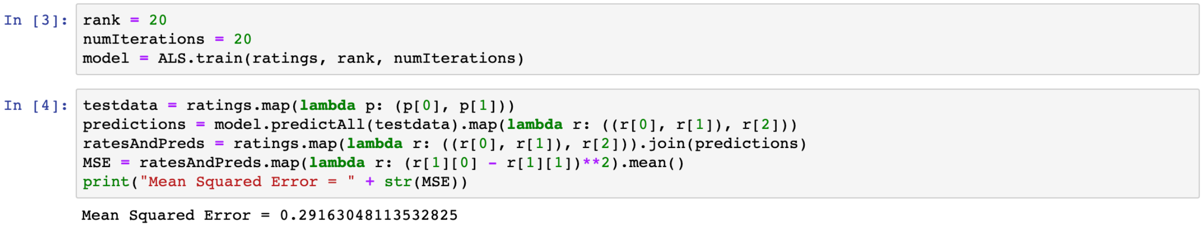
```
In [5]: rank = 5
        numIterations = 20
        model = ALS.train(ratings, rank, numIterations)
```

```
In [6]: testdata = ratings.map(lambda p: (p[0], p[1]))
        predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
        ratesAndPreds = ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
        MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
        print("Mean Squared Error = " + str(MSE))

        Mean Squared Error = 0.6178030283257278
```



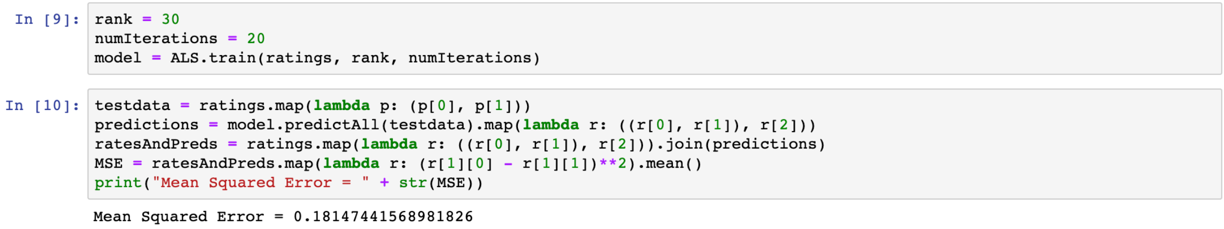
```
In [7]: rank = 10
        numIterations = 20
        model = ALS.train(ratings, rank, numIterations)
```

```
In [8]: testdata = ratings.map(lambda p: (p[0], p[1]))
        predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
        ratesAndPreds = ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
        MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
        print("Mean Squared Error = " + str(MSE))

        Mean Squared Error = 0.4711609135037449
```



```
In [3]: rank = 20
        numIterations = 20
        model = ALS.train(ratings, rank, numIterations)
```

```
In [4]: testdata = ratings.map(lambda p: (p[0], p[1]))
        predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
        ratesAndPreds = ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
        MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
        print("Mean Squared Error = " + str(MSE))

        Mean Squared Error = 0.29163048113532825
```

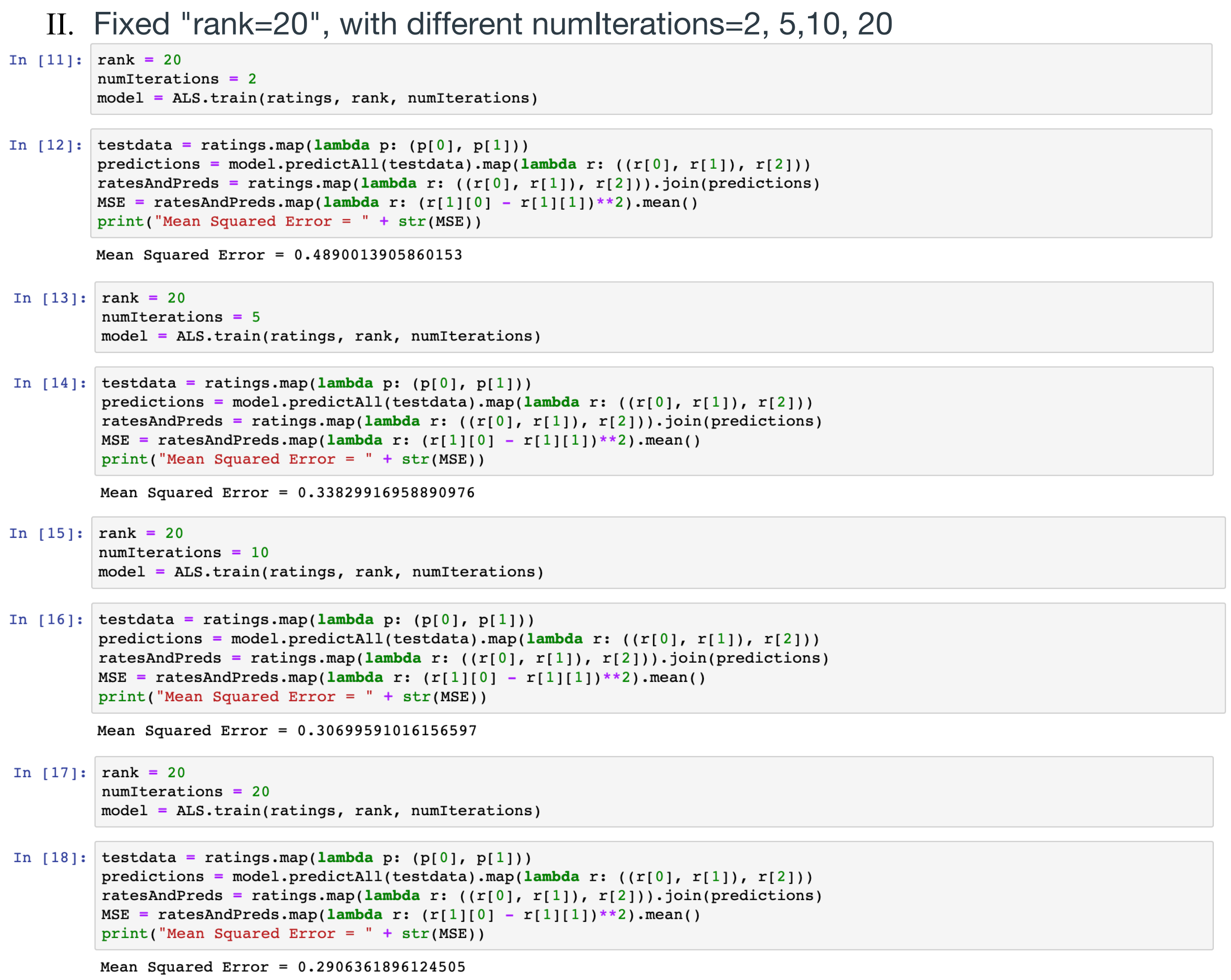


```
In [9]: rank = 30
        numIterations = 20
        model = ALS.train(ratings, rank, numIterations)
```

```
In [10]: testdata = ratings.map(lambda p: (p[0], p[1]))
         predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
         ratesAndPreds = ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
         MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
         print("Mean Squared Error = " + str(MSE))

         Mean Squared Error = 0.18147441568981826
```

Rank	
5	0.6178030283257278
10	0.4711609135037449
20	0.29163048113532825
30	0.18147441568981826

As it shows above, the MSE decreased as rank increased.



II. Fixed "rank=20", with different numIterations=2, 5,10, 20

```
In [11]: rank = 20
         numIterations = 2
         model = ALS.train(ratings, rank, numIterations)
```

```
In [12]: testdata = ratings.map(lambda p: (p[0], p[1]))
         predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
         ratesAndPreds = ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
         MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
         print("Mean Squared Error = " + str(MSE))

         Mean Squared Error = 0.4890013905860153
```

```
In [13]: rank = 20
         numIterations = 5
         model = ALS.train(ratings, rank, numIterations)
```

```
In [14]: testdata = ratings.map(lambda p: (p[0], p[1]))
         predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
         ratesAndPreds = ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
         MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
         print("Mean Squared Error = " + str(MSE))

         Mean Squared Error = 0.33829916958890976
```

```
In [15]: rank = 20
         numIterations = 10
         model = ALS.train(ratings, rank, numIterations)
```

```
In [16]: testdata = ratings.map(lambda p: (p[0], p[1]))
         predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
         ratesAndPreds = ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
         MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
         print("Mean Squared Error = " + str(MSE))

         Mean Squared Error = 0.30699591016156597
```

```
In [17]: rank = 20
         numIterations = 20
         model = ALS.train(ratings, rank, numIterations)
```

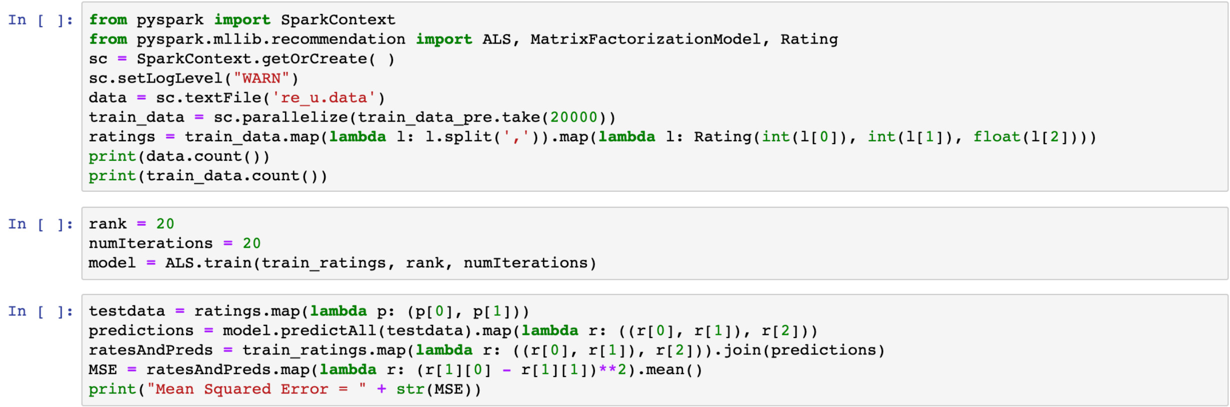
```
In [18]: testdata = ratings.map(lambda p: (p[0], p[1]))
         predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
         ratesAndPreds = ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
         MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
         print("Mean Squared Error = " + str(MSE))

         Mean Squared Error = 0.2906361896124505
```

Rank = 20

Numiterations	MSE
2	0.4890013905860153
5	0.33829916958890976
10	0.3069959101615697
20	0.2906361896124505

As it shows above, the MSE decreased as numIteration increased.

III. Fixed "rank=20" and "numIterations=20", with = 2000, 5000, 10000, 20000, 50000, 100000



```
In [ ]: from pyspark import SparkContext
        from pyspark.mllib.recommendation import ALS, MatrixFactorizationModel, Rating
        sc = SparkContext.getOrCreate( )
        sc.setLogLevel("WARN")
        data = sc.textFile('re_u.data')
        train_data = sc.parallelize(train_data_pre.take(20000))
        ratings = train_data.map(lambda l: l.split(',')).map(lambda l: Rating(int(l[0]), int(l[1]), float(l[2])))
        print(data.count())
        print(train_data.count())
```

```
In [ ]: rank = 20
        numIterations = 20
        model = ALS.train(train_ratings, rank, numIterations)
```

```
In [ ]: testdata = ratings.map(lambda p: (p[0], p[1]))
        predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
        ratesAndPreds = train_ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
        MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
        print("Mean Squared Error = " + str(MSE))
```

rank=20" and "numIterations=20

size	MSE
2000	9.9191004400280498 e-05
5000	0.000659682781938134
10000	0.002988928947876204
20000	0.025387303979932457
50000	0.151221837001804823
100000	0.291294923919096340

As the data shows, MSE increase with data-size increase.