Apache Spark Benchmark

Proje ile ilgili dökümananlara https://github.com/alperentahta/Mllib-Linear-Regression-Performance adresinden ulaşılabilir.

Özet

Mllib üzerindeki Linear Reggression fonksiyonu ile finans verisi üzerinde değişik core sayısı ve data büyüklükleri kullanılarak test edilmiş ve sonuçlar saniye cinsinden alınmıştır. Küçük verilerde çekirdek sayısının katkısı olmadığı, büyük verilerde ise tek çekirdek ile 8 çekirdek arasında yaklaşık 6 katlık (3698 saniye) fark olduğu gözlenerek, büyük verilerde Spark kullanmanın avantaj sağladığı gözlenmiştir.

Giriş

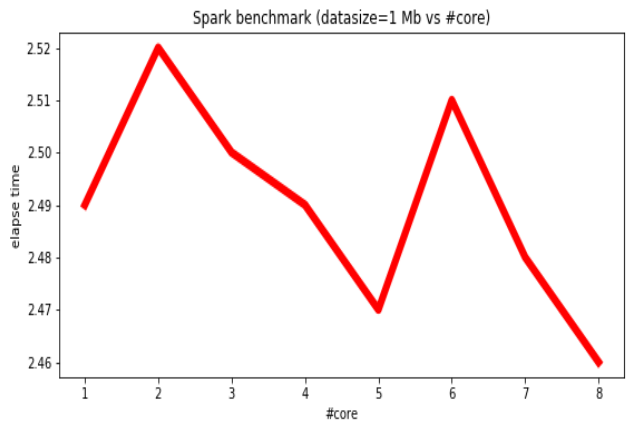
Bu çalışmada AAPL (Apple Inc. NASDAQ) verisi üzerinde Apache Spark Mllib kütüphanesi kullanılarak Linear Regression uygulanmıştır. Sistem değişik büyüklüklerde veriler ve değişik core sayıları ile denenmiş ve hız ölçümleri alınmıştır. Amacımız veri büyüdükçe Spark'ın sağladığı faydaları gözlemlemektir.

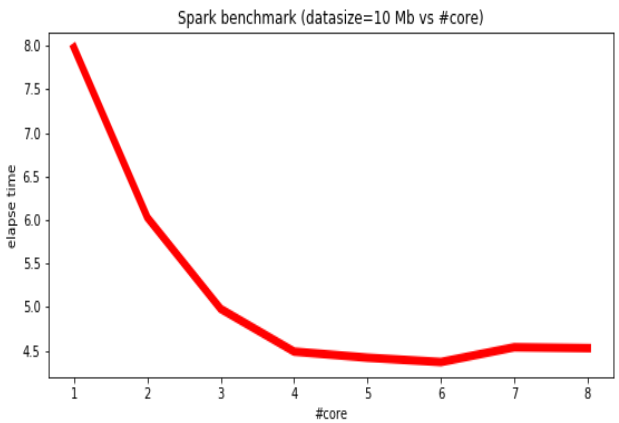
Dataset

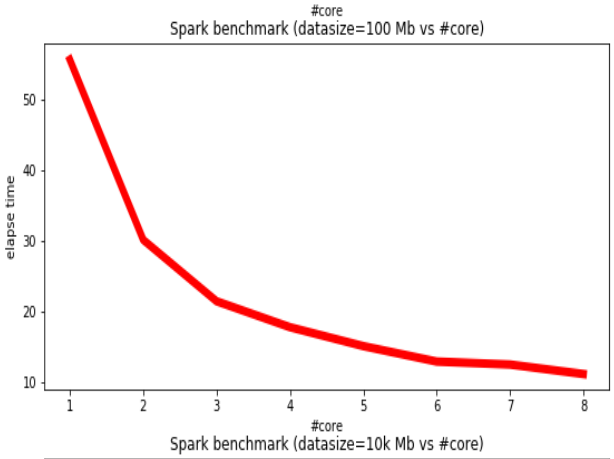
Open, High, Low, Close, Volume ve Adj Close vektörlerinden oluşan AAPL (Apple Inc. NASDAQ) verisi kullanılmıştır. Spark'ın büyük veri büyüklüğüne tepkisini gözlemleyebilmek amacıyla veri istenen boyuta gelene kadar arka arkaya eklenmiştir. Veri ile ilgili istatislikler Tablo 1'den görülebilir. Adj Close vektörü label olarak ayrılmış diğer vektörler ise feature olarak kullanılmıştır.

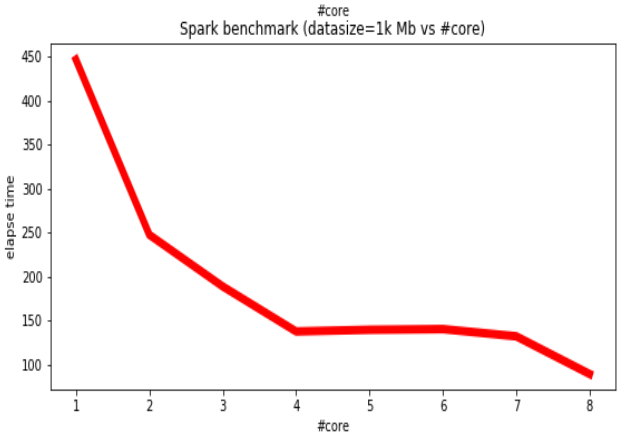
Tablo 1: Dataset İstatislikleri

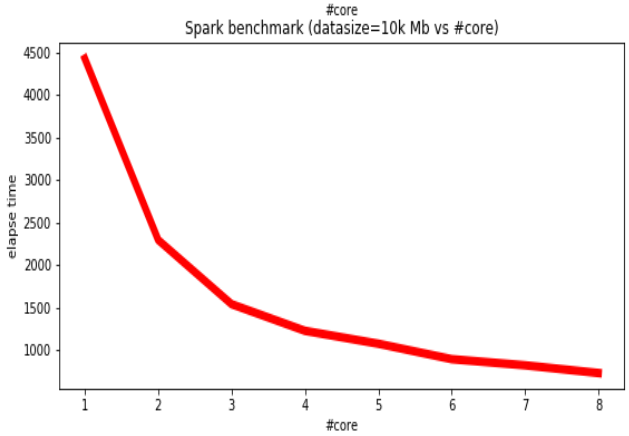
summary	count	mean	stddev	min	max
Date	9534	9.413206708621775E8	3.445064029577415E8	3.454164E8	1.5384276E9
Open	9533	25.227877389069594	45.035747904623484	0.198661	228.990005
High	9533	25.46629224934435	45.402302820535645	0.198661	230.0
Low	9533	24.976618823979813	44.669427190260855	0.196429	226.630005
Close	9533	25.227376554914493	45.04640919574818	0.196429	229.279999
Adj Close	9533	23.66256029403127	43.507792038119455	0.158904	229.279999
Volume	9533	8.787442552187139E7	8.695796711275896E7	347200.0	1.8554102E9











Benchmark

a) Sabit Veri Büyüklüğü İçin Değişen Core Miktarı

Denemenin ilk kısmında veri boyutu sabit tutularak core sayısı değiştirilmiştir. Veriler 1, 10 ,100, 1k, 10k Mb olarak sisteme verilmiş ve çekirdek sayısı 1'den e kadar arttırılmıştır. Figure 1 ve Figure 2' deki sonuclar incelendiğinde küçük verilerde core sayısının artışının hıza mantıklı şekilde etkisi yokken (1 mb için 1 core 2.49

sn, 8 core 2.46 sn) büyük verilerde (10k mb için 1 core 4426 sn, 8 core 728.79 sn) ciddi kazanımlar elde edilmiştir.

Figure 1: Sabit veri boyutu için değişen core sayıları hız grafikleri

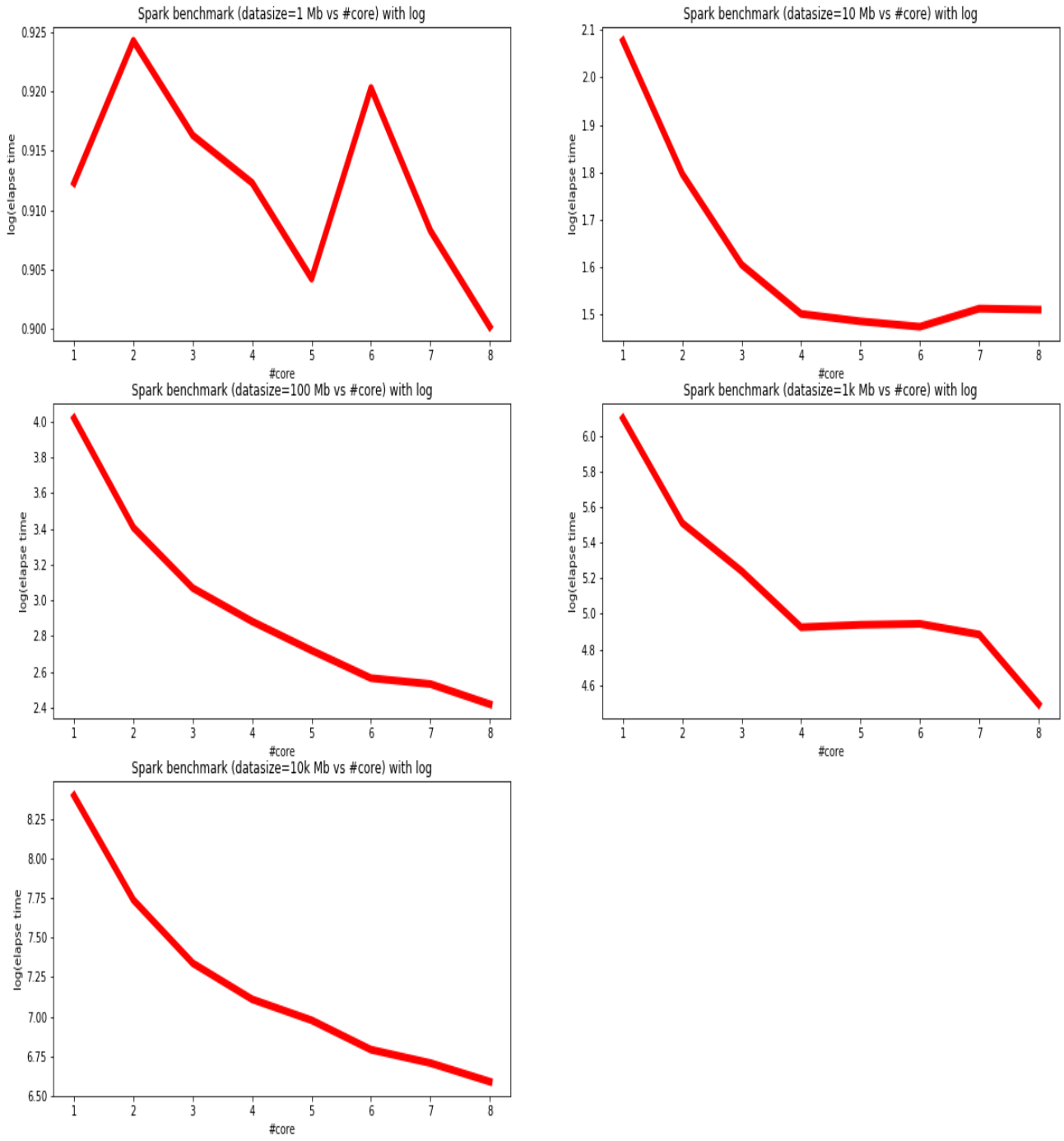
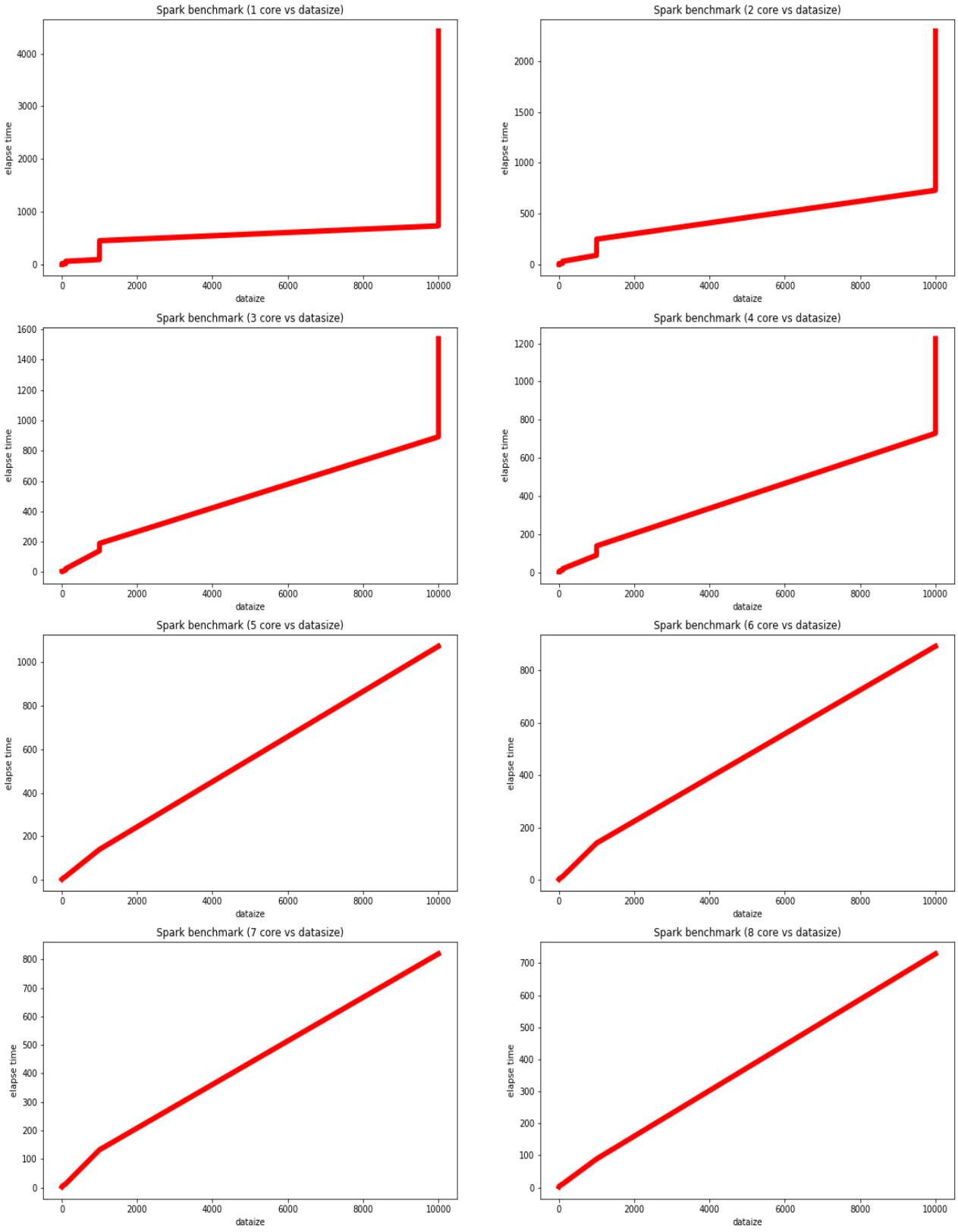


Figure 2: Sabit veri boyutu için değişen core sayıları hız grafikleri (logaritmik)

b) Sabit Core Sayısı İçin Değişen Değişken Veri Büyüklüğü

Bu kısımda sabit core sayısı için veri boyutu değiştirilmiştir. Çekirdek sayısı 1 den 8 e kadar verilmiş, veriler 1, 10 ,100, 1k, 10k Mb olarak arttırılmıştır. Figure 3 ve Figure 4' deki sonuclar incelendiğinde düşük core sayısı az



olduüunda elapse time veriye büyüklüğüne bağlı olarak çok fazla artış gösteriyor ancak çekirdek sayısı arttırıldığında veri büyümesine bağlı elapse time artışı lineere yaklaşıyor.

Figure 3 : Sabit core sayısı için değişen Veri Büyüklüğü grafikleri

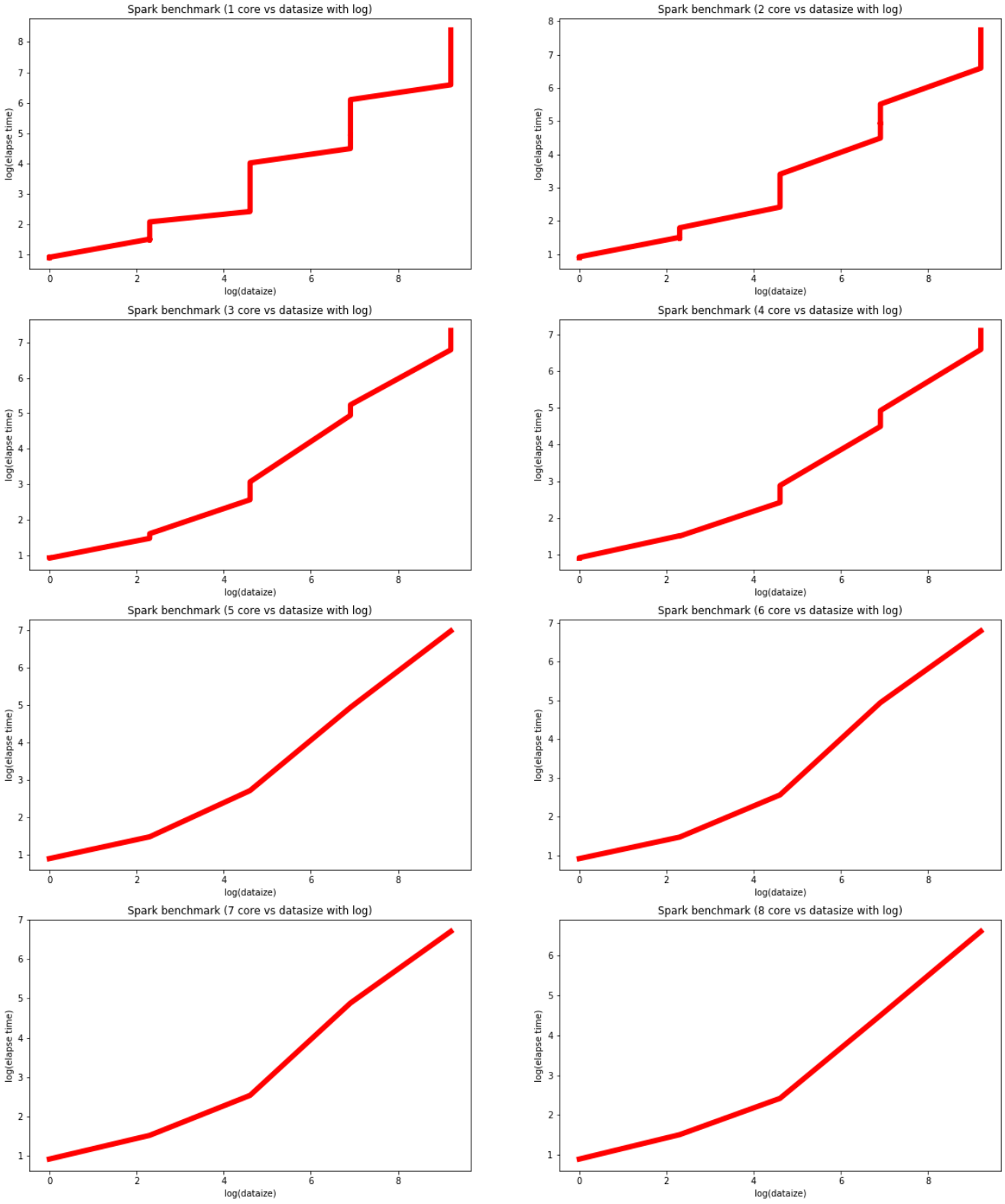


Figure 4 : Sabit core sayısı için değişen Veri Büyüklüğü grafikleri (logaritmik)

c) Speed Up

Figure 5'deki speed up grafikleri incelendiğinde Figure 3 ve Figure 4'ü doğrulayan sonuçlar görülecektir. Küçük verilerde neredeyse hiç kazanç sağlanamamış, veri büyüdükçe ideale yakınsamaya başlamıştır.

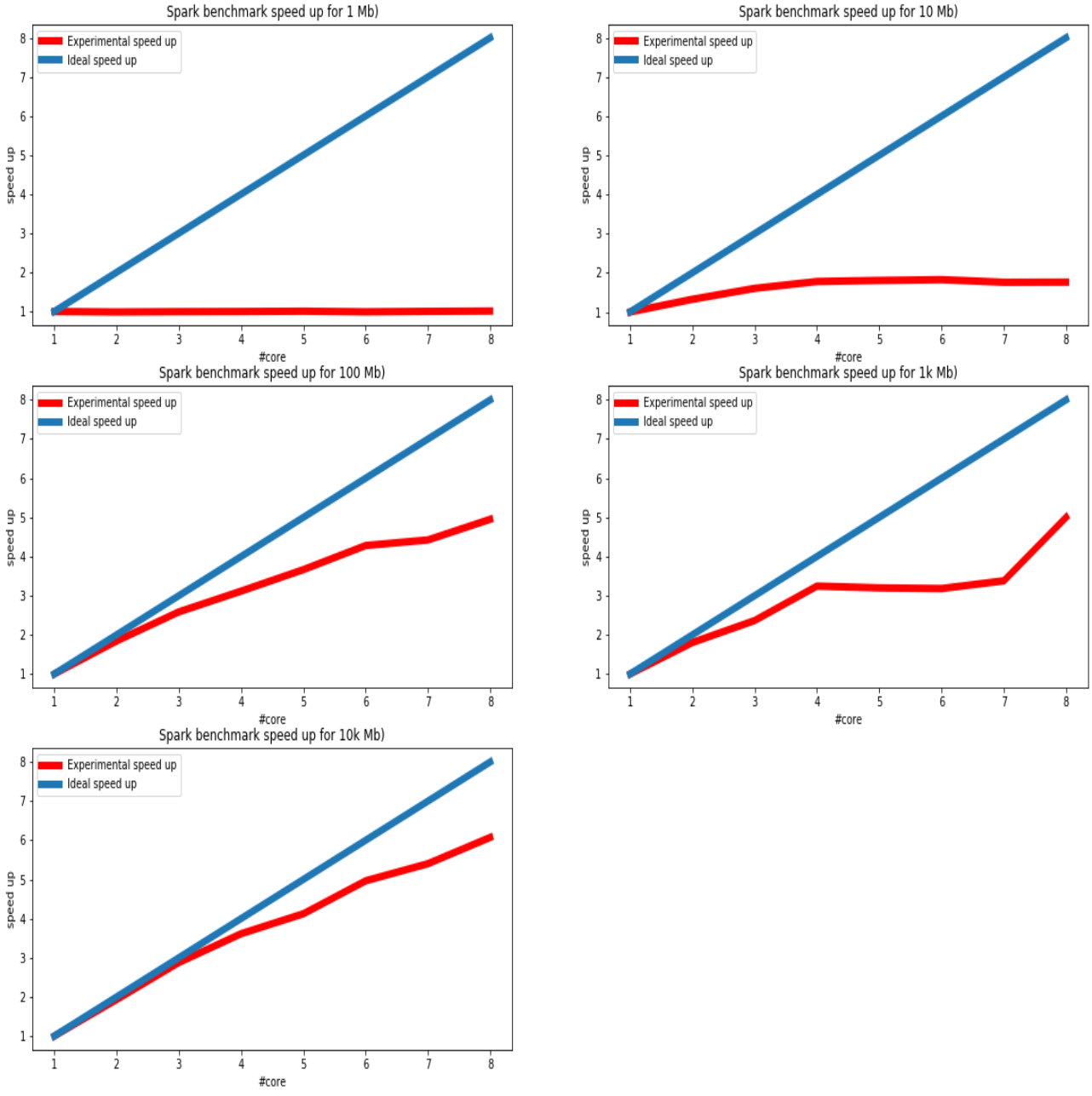


Figure 5 : Farklı Veri Büyüklükleri İçin Speed Up

Sonuçlar

Tablo 2: Benchmark(#core vs veri buyuklugu) sonuclari(saniye)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1 Mb	2.49	2.52	2.5	2.49	2.47	2.51	2.48	2.46
10 Mb	7.98	6.03	4.98	4.49	4.42	4.37	4.54	4.53
100 Mb	55.6	30.17	21.5	17.84	15.17	12.99	12.57	11.23
1k Mb	446.68	247.61	188.86	137.69	139.61	140.37	132.16	89.28
10k Mb	4426.33	2293.38	1538.47	1225.01	1073.47	891.45	819.45	728.79

Benchmark sonuçları Tablo 2'de verilmiştir. Bu verilerle Figure 6 ve Figure 7 de görselleştirilmiştir. Figure 7 'de görülebileceği gibi küçük verilerde kazanç sağlanamazken veri büyüdükçe ciddi kazançlar elde edilmiştir.

Küçük verilerde Spark'ın verimsiz olmasının sebebi Spark'ın sahip olduğu overhead diğer araçlara göre daha fazladır. Veri küçük olduğunda overhead normal çalışma süresinin önüne geçeceğinden negatif performans olarak etki gözlenir. Veri büyüdüğünde overhead oransansal olarak çalışma süresine göre küçük kalacağından yoksayılabilir, bu durumda Spark'ın avantajları görülebilir.

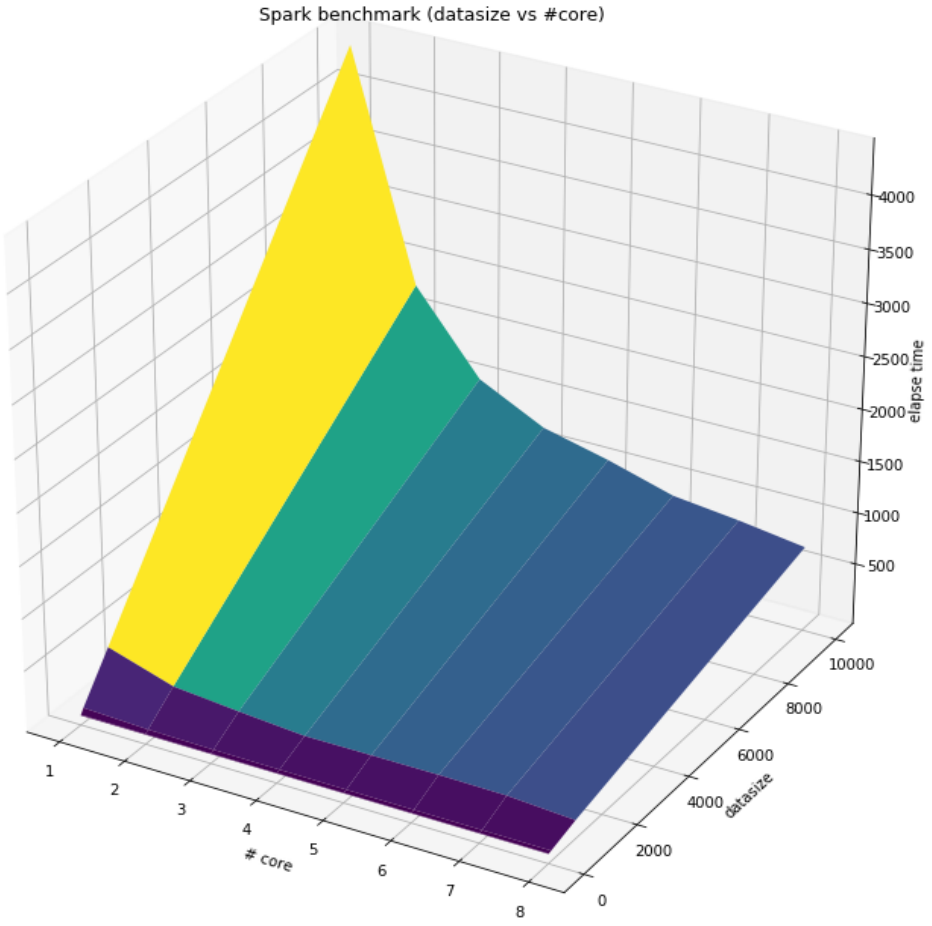


Figure 6 : Benchmark Sonuçları

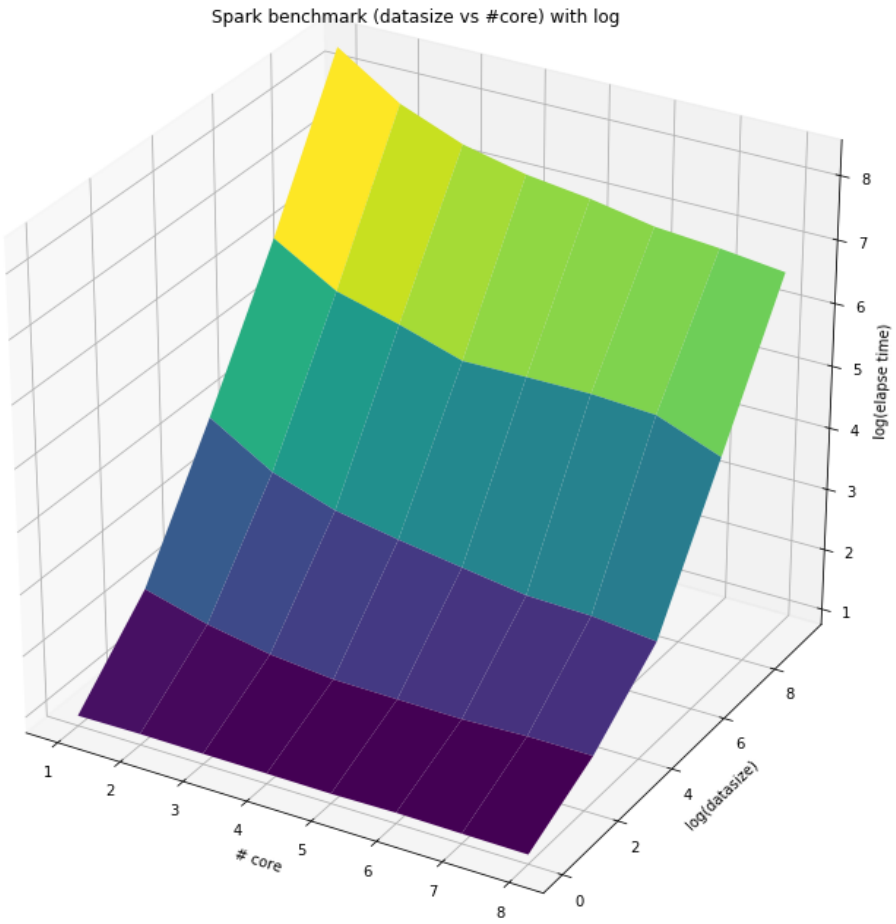


Figure 7 : Benchmark Sonuçları (Logaritmik)