ANALİZLERİN SONUÇLARI

Kullanılan veri kümesi Japon Havayollarına ait 25 adet özellikten (kolon) oluşan bir müşteri veri kümesidir. Bu araştırmada amaç Kuantum Bilgisayarlar ve Normal Bilgisayar arasındaki sınıflandırma probleminin tahmin süreleri ve tahmin sonucundaki doğruların analizidir.

NORMAL ML: Normal ML kümesi ile yapılan işlemler normal makine öğrenmesi işlemidir. *T*rain kümesinde 103903 adet veri ile öğrenme yapılmıştır. Test kümesinde ise 25975 adet veri ile işlem yapılmıştır. Özellik sayısı ise olduğu gibi 25 adet ile bırakılmıştır. Özelliklere gerekli biçimlendirme işlemi yapılmıştır ve gerekli dönüşümler uygulanmıştır.

LOCAL QUANTUM: Normal ML ile aynı adet test, train datası ve özellik kullanarak işlem süresini lokal Quantum simülatöründe bitiremedim. Bilgisayarı bir gece açık bırakmama rağmen hala öğrenme işlemi bitmemişti. Bunun sebebi ZZFeatures özelliği ile verinin çok katmanlı bir ortama geçirilip ansatz devresini çizerken devreler arası bağlantı sayısını arttırması ve bunun sonucunda daha karmaşık bir devre yaratması olduğunu yaptığım araştırmalar sonucunda buldum. Bu yüzden Lokal Quantum devresini küçülttüm. Özellik sayısı 5 adete indirildi test kümesi 200 adet ve train kümesi ise 2500 adete indirildi. Bu sayede daha optimal sürelere indirildi işlem süresi.

PARÇALANMIŞ ML: Yapılacak analizlerin daha adaletli olması için Lokal Quantumda yapılan veri adeti ile standart sınıflandırma problemi yaptım.

Neden rbf ve ZZFeatureMaps/COBYLA sınıflandırmaları

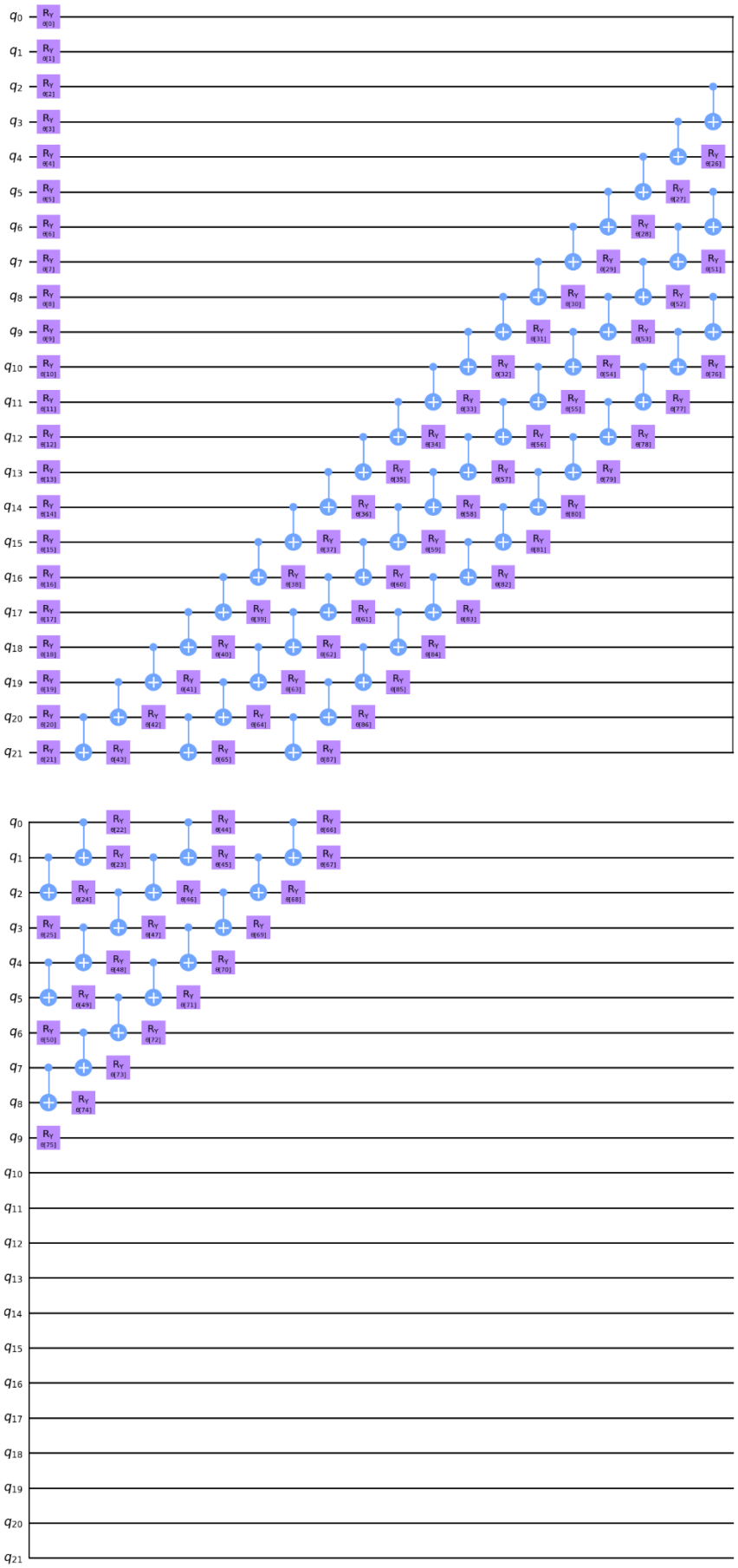
Makine öğrenmesi ve Quantum makine öğrenmesinde kullandığım kerneller aşağıdadır. Bunları kullanmamın sebebi ikisinin de doğrusal olmayan problemleri çözmek için kullanılmasından dolayıdır.

RBF: Doğrusal olmayan özellikler ekleyerek verileri daha yüksek boyutlu uzaylara dönüştürür, böylece doğrusal olmayan ilişkileri yakalayabilir.

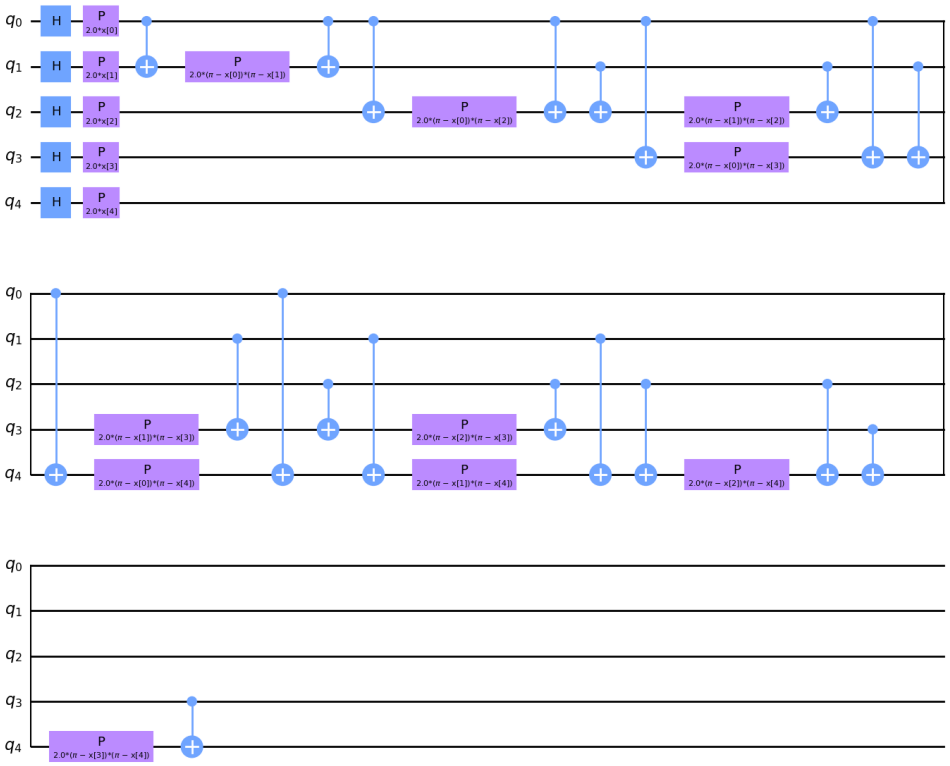
COBYLA: doğrusal olmayan kısıtlı problemleri çözmek için kullanılır. Her yinelemede, hedef fonksiyonun ve kısıtların doğrusal bir yaklaşıklığını kullanarak bir alt problem oluşturur ve çözer.

Aşağıdaki görsellerde özellik haritaları ve ansatz devrelerinin özelik sayısına göre değişimini göreceksiniz.

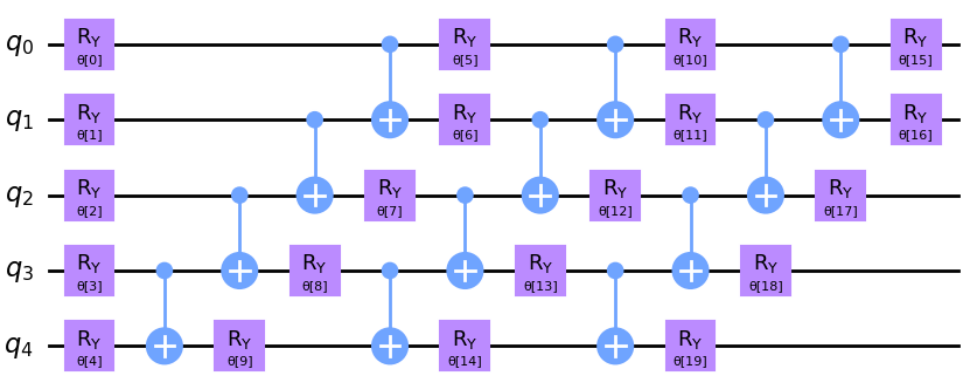
25 adet özellik ile feature maps yapınca maalesef görüntü çok büyük oluyor bu yüzden o görüntüyü buraya tıklayarak açabilirsiniz



25 adet özellik ile çıkartılmış ansatz devresi



5 adet özellik ile çıkartılan özellik haritası



5 adet özellik ile çıkartılan ansatz devresi

Özellikler değiştikçe meydana gelen devre ve özellik haritalarının katmanlaşmalarını görüyorsunuz. Ne kadar büyük devre olursa o kadar fazla uzun sürüyor işimiz.

TEST SONUÇLARI

2500 adet parçalanmış veri

	Normal ML	Parçalanmış ML	LOCAL QUANTUM	AWS BRAKET
execution time (dk)	3,904370	0,012375	20,385109	-
accuracy	0,954188	0,655000	0,545000	-
precision	0,968023	0,811321	0,542859	-
recall	0,951183	0,637037	0,545	-
f1 score	0,959529	0,713693	0,542826	-
specificity	0,959529	0,692308	0,542826	-
kernel	"rbf"	"rbf"	"zzfeaturemaps"	"zzfeaturemaps"

2000 adet parçalanmış veri

	Normal ML	Parçalanmış ML	LOCAL QUANTUM	AWS BRAKET
execution time (dk)	3,904370	0,008737	20,680000	-
accuracy	0,954188	0,600000	0,570000	-
precision	0,968023	0,759259	0,565407	-
recall	0,951183	0,602941	0,57	-
f1 score	0,959529	0,672131	0,560676	-
specificity	0,959529	0,593750	0,703704	-
kernel	"rbf"	"rbf"	"zzfeaturemaps"	"zzfeaturemaps"

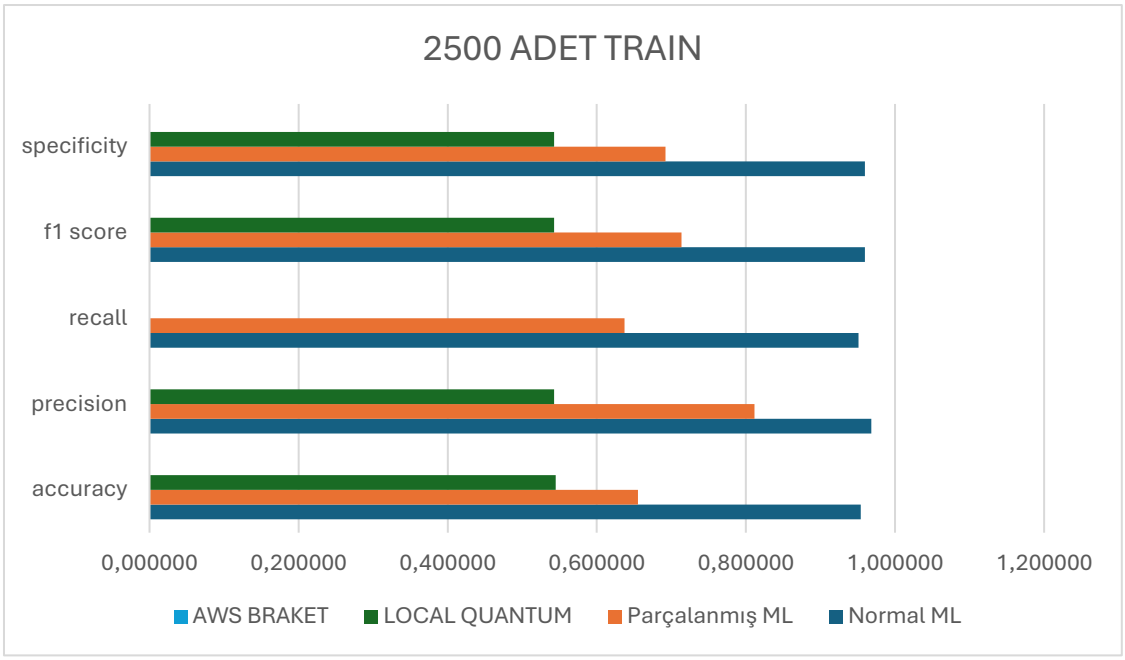
Doğruluk (Accuracy): Modelin doğru tahminlerinin oranıdır. Tüm doğru tahminlerin, toplam tahmin sayısına bölünmesiyle bulunur.

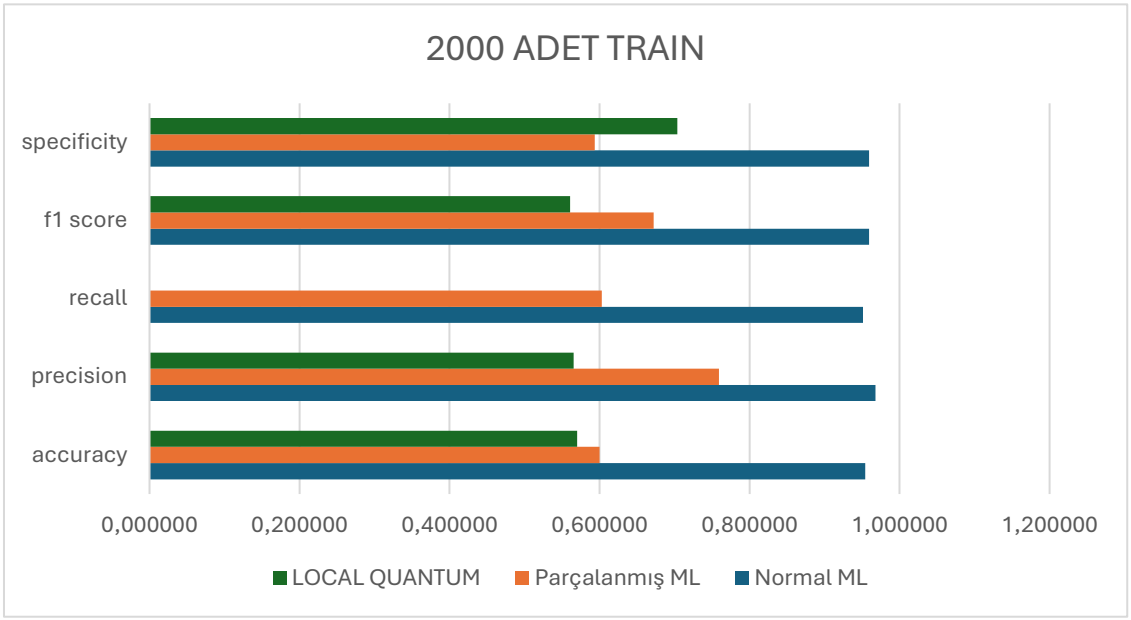
Kesinlik (Precision): Modelin pozitif tahminlerinin ne kadarının doğru olduğunu gösterir. Doğru pozitiflerin, tüm pozitif tahminlere bölünmesiyle hesaplanır.

Geri Çağırma (Recall): Modelin gerçek pozitifleri ne kadar doğru tespit ettiğini gösterir. Doğru pozitiflerin, gerçek pozitiflerin toplamına bölünmesiyle hesaplanır.

F1 Skoru (F1 Score): Kesinlik ve geri çağırmanın harmonik ortalamasıdır. Hem kesinlik hem de geri çağırmanın dikkate alındığı bir metriktir.

Özgüllük (Specificity): Modelin negatif örnekleri doğru bir şekilde tanımlama yeteneğini gösterir. Doğru negatiflerin, gerçek negatiflerin toplamına bölünmesiyle hesaplanır.





YAPTIĞIM DEĞERLENDİRMELER

Normal ML veri kümesi doğal olarak 150 binden fazla veriyle çalıştığı için en doğru sonuçları veriyor ve en optimum sürelerde bu çıktıyı sağlıyor. Fakat Quantum Computing simülasyonunda derinlik faktörü işin içine girdiği için basit veri kümelerinde afallama olasılığı artıyor ve bunu bu çıktılarda görüyoruz. Quantum hesaplama nedir adlı yazımda bahsettiğim Lityum ION bataryaların içsel yapısı daha komplike olduğu için ve bunu en doğru şekilde modellemek istediğimiz zaman normal makine öğrenmesi yetersiz kalacaktır. Şu da unutulmaz bir gerçek ki Kuantum hesaplama yöntemleri hala gelişmekte olan bir teknolojidir. Şu an belki küçük işler için verimli görünmese bile ileride yeni algoritmalar ile daha verimli hale gelebilir.

KİŞİSEL NOTLARIM

AWS Brakette real-time development yapılabiliyor ve bu geliştirme AWS'nin kendi oluşturduğu jupyter notebook üzerinden yapılıyor. Ancak bu notebook'un açık olduğu her süre ücretlendirme yapılıyor, makinenin çalıştığı süre haricinde böyle bir ücret olması saçma. Bunun dışında

aws-braket-sdk lokalde istediğimiz verimlilikte çalışmıyor. Buradaki ücretlendirme politikasından kaynaklı tam efektif kod yazılamıyor.

Ayrıca veri kümesini birden fazla uzaya ayırmasından dolayı basit veri kümeleri için gereksiz zaman kaybı. Bunun dışında veri kümesi komplike olmadığı sürece doğru sonuca yaklaşamıyor.