PROJE SON RAPOR

APACHE FLİNK İLE STREAMİNG DATA ÜZERİNDE BİTCOİN DEĞER TAHMİNİ

Danışman:

Prof. Dr. Suat ÖZDEMİR

Grup Üyeleri:

141180035 SİMGE NUR KABATAŞ

131180031 ZEYNEP FENERCİ

BM495 BİLGİSAYAR PROJESİ II

Haziran 2018

İÇİNDEKİLER

ÖZET

Apache Flink frameworku üzerinde bitcoin fiyatının tahmini gerçekleştirip, analizini yapmaktır. Önce 2012 ile düne kadarki geçmiş bitcoin verilerini batch data olarak tutup, daha sonra verilerin modelini oluşturup, ön işlemesini yapıp daha sonra temizlenen data ile modelimizin eğitimini gerçekleştirip, temel amacı olan stream olarak alınacak bitcoin verisi üzerinde tahminin gerçekleştirilmesi ve daha sonra farklı algoritmalar üzerinde çalıştırılıp karşılaştırması ve analizinin yapılmasıdır.

1. GİRİŞ

Bitcoin, dünyanın herhangi bir yerinden herhangi bir kişiye online ödeme yapmayı sağlayan herhangi bir devlet, şirket veya otorite tarafından kontrol edilemeyen, merkezi olmayan para sistemi ve para birimidir. Bire bir elektronik nakit sistem olarak dizayn edilen Bitcoin'in finansal politikalara tepki olarak doğması ilgi uyandırmış ve kısa sürede yayılmasına sebep olmuştur. Bizim de bu projedeki amacımız, Bitcoin'in fiyatını Apache Flink frameworkü ile tahmin edip analizini yapmaktır. Araştırmamızın ilk aşaması apache flinkin ne olduğu, kullanım alanları, makine öğrenmesi algoritmaları için kullanıldığı kütüphaneler vs. Bunlara dair çalışmalar yapıldı. Araştırmamızın bizim amacımız olan durum analizi için gerekli verisetleri vs. araştırmalar yapıldı.

2. SİSTEM TASARIMI

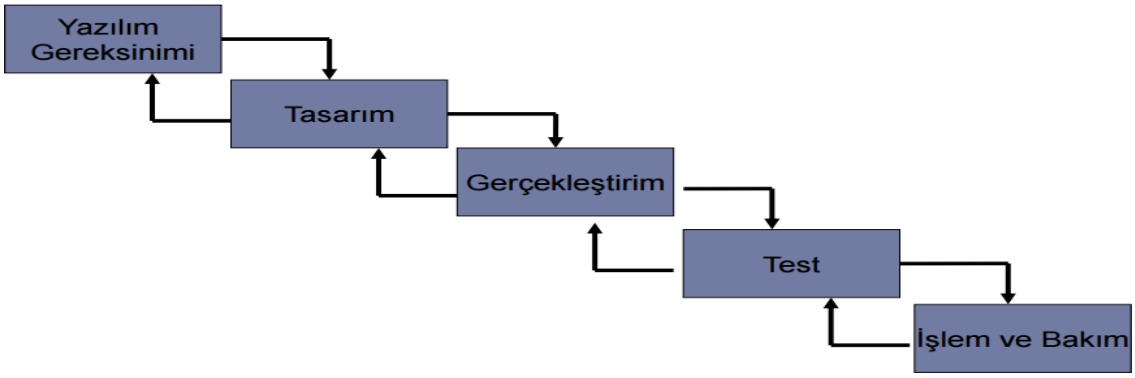
2.1. Tanım

Belgenin bu bölümünde proje geliştirme süreci boyunca uygulanacak olan yaşam döngüsü modeli ve bu modelin neden seçildiği, proje ekibinin görev ve sorumlulukları ile proje aşamalarında kullanılacak araç ve tekniklerden bahsedilecektir.

2.1.1. Yazılım Süreç Modeli

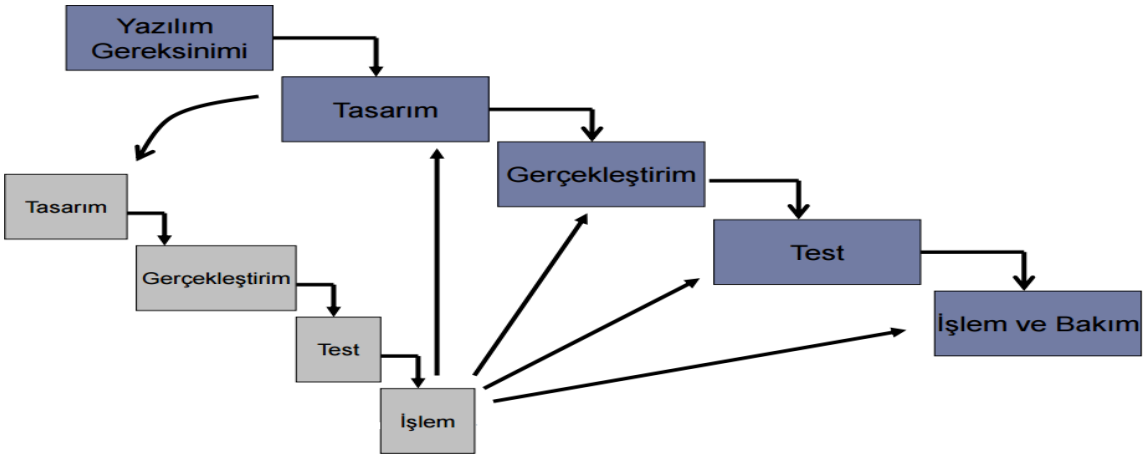
Bu proje de yazılım süreç modeli olarak Temel Çağlayan Modeli(Waterfall) kullanılacaktır. Bu modeli tercih etmemizin sebebi gereksinimlerin kesin olarak belirlenmiş ve iyi bilinen anlaşılabilen adımlardan oluşmasıdır. Proje aşamalardan oluşur. Şelale modeli proje içerisindeki dokümantasyonu ayrı bir süreç olarak değil üretimin doğal bir parçası olarak ele alır. Çağlayan modelinde bir aşama tamamlanmadan diğer bir aşamaya geçilemez. Bu durum, proje bütünlüğü ve takip edilebilirliği açısından kolaylık sağlar. Ayrıca, bu modelde aşamalar arasında bir sonraki adımda bir önceki adımda eksik ya da hatalı bir durum ile karşılaşıldığında geri dönüşlerin olabileceği bir modeldir. Ancak analiz aşamasında tasarımın bütün detayların

gerçekleştirilebilmesi için müşteri ve sistem gereksinimlerinin en ince ayrıntısına kadar belirlenmesi gerekir. İşleyiş şekli aşağıda gösterilmiştir(Şekil 2.1.).



Şekil 2.1. Temel Çağlayan Modeli(Waterfall)

Tasarım aşaması da, yazılımın tüm gereksinimlerini karşılayacak şekilde detaylı bir çalışma gerektirmektedir. Kodlama veya test aşamalarında olabilecek bu değişikliklerin sisteme/yazılıma yansıtılması maliyeti ise çok yüksektir. Bu sorunu ortadan kaldırmak için en iyi çok çözüm önerisi çözümleme aşamasının önüne bir ön-tasarım aşaması ekleme olacaktır. Böylece hazırlanacak programın kısıtları önceden belirlenip anlaşılabilir. Yine bu soruna ikinci bir alternatif çözüm olarak her adımın sonunda geniş ve ayrıntılı belgelendirme yapmak olacaktır. Burada çıkabilecek diğer problem ise eğer tasarlanacak program orijinal ise sistemi yapmadan önce deneysel testler yapılmasına gerek duyulacaktır. Bu deneysel testlerin yapılması içinde hipotezleri sınamak için prototip modeli gerçekleştirilecektir. Bu prototip modeli yapıldıktan sonra yazılım süreç modelinin işleyişi aşağıda gösterilmiştir(Şekil 2.1.2.) [1].



Şekil 2.2. Prototip yaptıktan sonra Çağlayan Modeli

Tasarlanacak proje daha önce gerçekleştirilen modeller ile benzer çıktılara ve sonuçlara sahip olacağı için çok fazla deneysel test gerektirmeyecektir. Test aşamasında ortaya çıkabilecek sorun çıkacak olursa çözüm olarak çözümleme aşamasının önüne bir ön tasarım aşaması eklenerek çözeceğiz. Şelale Modelindeki bu risklerin göz ardı edilebilirliğinden dolayı bu yazılım süreç modeli ile gerçekleştirmeye karar verdik. Yazılım mühendisliğinde kullanılan bir yazılım projesi yönetim modelidir. Bu model aşağıdaki 4 temel merhaleden oluşmaktadır:

- tahlil (analiz, analysis)
- tasmim (tasarım, design)
- tatbik (uygulama, implementation)
- tecrübe (test, test)

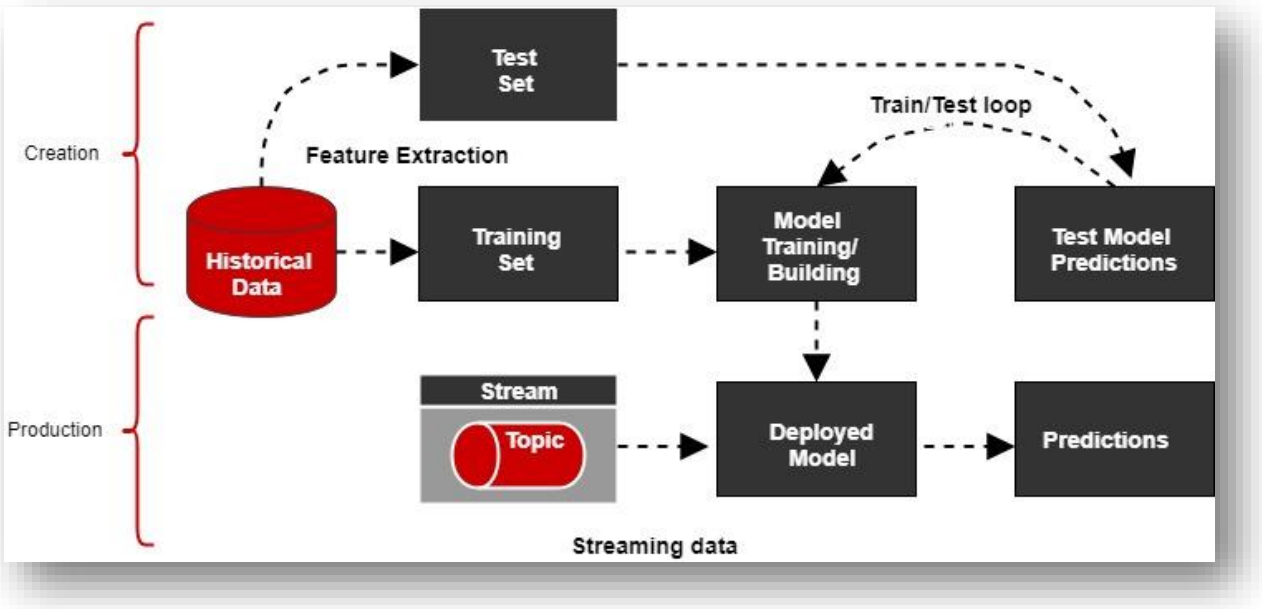
Yazılım mühendisliğindeki diğer modellere temel teşkil eden bu modelde yukarıdaki aşamalar sırasıyla izlenir. Aşamalar arası geçişleri oldukça sıkı olan bu modelde geri dönüşler oldukça maliyetli olmaktadır. Basit ve kolay yazılım programları için uygundur**.** Yazılım şirketleri bu modeli kullanarak uygulama geliştirdiklerinde, her bir bölüm ardışık olarak yapılır, her bölümden sonra gerçekleştirilen bölümün sonuçları gösterilir. Fakat waterfall model uzun süreli projeler için uygun olmamakla beraber, esneklik sağlamamaktadır.

1	Tarih	Şimdi	Açılış	Yüksek	Düşük
2	03.02.2012	6,0	6,0	6,0	6,0
3	04.02.2012	5,9	5,9	5,9	5,9
4	05.02.2012	5,7	5,7	5,7	5,7
5	06.02.2012	5,4	5,4	5,4	5,4
6	07.02.2012	5,7	5,7	5,7	5,7
7	08.02.2012	5,6	5,6	5,6	5,6
8	09.02.2012	5,8	5,8	5,8	5,8
9	10.02.2012	5,9	5,9	5,9	5,9
10	11.02.2012	5,6	5,6	5,6	5,6
11	12.02.2012	5,5	5,5	5,5	5,5
12	13.02.2012	5,3	5,3	5,3	5,3
13	14.02.2012	4,5	4,5	4,5	4,5
14	15.02.2012	4,3	4,3	4,3	4,3
15	16.02.2012	4,3	4,3	4,3	4,3
16	17.02.2012	4,4	4,4	4,4	4,4
17	18.02.2012	4,2	4,2	4,2	4,2
18	19.02.2012	4,4	4,4	4,4	4,4
19	20.02.2012	4,4	4,4	4,4	4,4
20	21.02.2012	4,3	4,3	4,3	4,3

Şekil 2.3. Bitcoin verilerinin İşlenmemiş Hali [2]

2.1.2. Tasarım

Bu aşamada projede uygulanacak adımlar modellenmiştir. Bitcoinin 2012 ile 2018 arasındaki geçmiş verileri önce önişleme aşamalarından geçirilmiş ve eğitim veri olarak hazır hale getirilmiştir. Hazır hale gelen eğitim verisinden bir veri modeli oluşturulmuştur. Bu veri modeli artık verilerin eğitiminde kullanılmaya hazır hale getirildikten sonra modelimiz eğitilmiş ve test datamız olarak streaming data olan bitcoin verisi alındıktan sonra test aşaması gerçekleştirilmiştir [3].



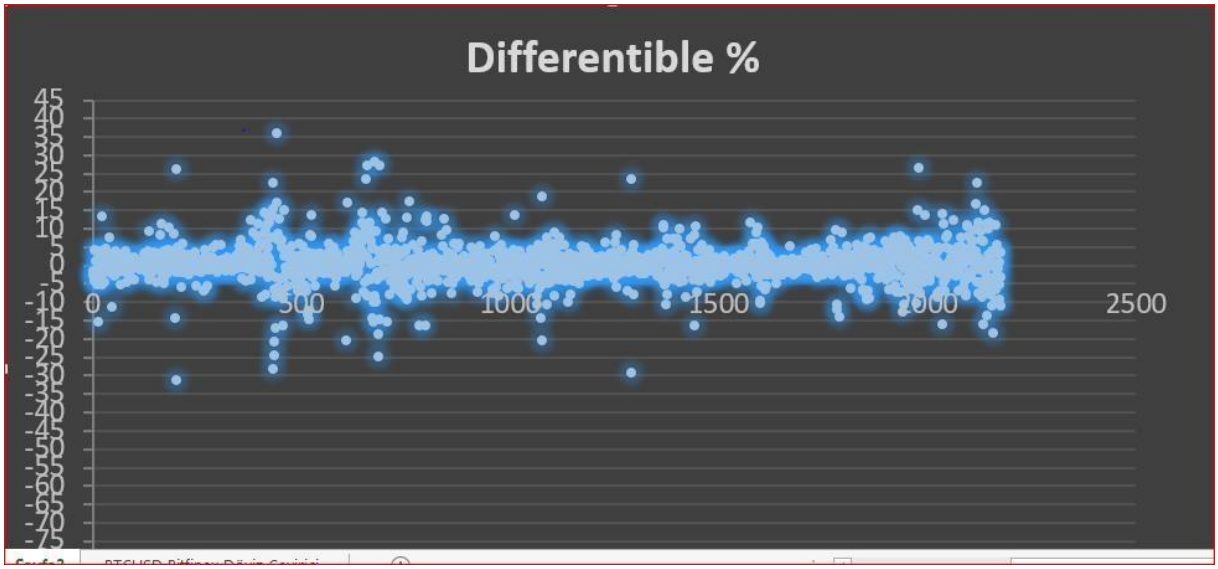
Şekil 2.5. Proje Tasarım Modeli

2.1.3. Projede Kullanılan Veriseti Özellikleri

Verisetimiz 2012 yılından bugüne kadarki değerleri tutmaktadır. Verisetinde bulunan featurelar: Bitcoinin gün içindeki açılış değeri, gün içindeki en yüksek ve en düşük değeri, gün içindeki kapanış değeri, hacmi ve fark. Fark sütunu, 2012 ile 2018 arasındaki değer aralığı çok geniş olduğu için tahminin daha iyi yapılmasını sağlamak adına o güne ait (kapanış-açılış)/100 ile artış-azalış yüzdelerini içeren bizim tarafımızdan eklenen bir feature' dur. Eğitim verilerimizi hazırladıktan sonra Flink' in Machine Learning kütüphanesi olan FlinkML kütüphanesi scala dilinde yazıldığı için ve biz de scala dilini tercih etmek istemediğimiz için Java' nın kendi Machine Learning kütüphanelerinden olan Weka kütüphanesini kullanmayı tercih ettik.

```
DataModel{ simdi=16917.0, acilis=15159.0, yuksek=15390.0, dusuk=14064.0, hacim=4647000.0, fark=0.03, sinif=8}
DataModel{ simdi=17161.0, acilis=16911.0, yuksek=17101.0, dusuk=14769.0, hacim=5463000.0, fark=11.59, sinif=6}
DataModel{ simdi=16196.0, acilis=17163.0, yuksek=17252.0, dusuk=16251.0, hacim=2961000.0, fark=1.44, sinif=4}
DataModel{ simdi=14930.0, acilis=16216.0, yuksek=17176.0, dusuk=15726.0, hacim=3084000.0, fark=-5.62, sinif=4}
DataModel{ simdi=14423.0, acilis=14902.0, yuksek=16279.0, dusuk=13760.0, hacim=6495000.0, fark=-7.82, sinif=5}
DataModel{ simdi=14896.0, acilis=14426.0, yuksek=15355.0, dusuk=14122.0, hacim=4344000.0, fark=-3.4, sinif=6}
DataModel{ simdi=13266.0, acilis=14895.0, yuksek=14896.0, dusuk=13338.0, hacim=5733000.0, fark=3.28, sinif=3}
DataModel{ simdi=13783.0, acilis=13248.0, yuksek=14949.8, dusuk=12639.0, hacim=7170000.0, fark=-10.94, sinif=6}
DataModel{ simdi=14191.0, acilis=13794.0, yuksek=14095.0, dusuk=12778.0, hacim=3742000.0, fark=3.9, sinif=6}
DataModel{ simdi=13558.0, acilis=14190.0, yuksek=14580.0, dusuk=13760.0, hacim=2901000.0, fark=2.96, sinif=4}
DataModel{ simdi=13575.0, acilis=13558.0, yuksek=14391.0, dusuk=12874.3, hacim=3560000.0, fark=-4.46, sinif=6}
DataModel{ simdi=11072.0, acilis=13594.0, yuksek=14350.0, dusuk=13307.0, hacim=3346000.0, fark=0.13, sinif=1}
DataModel{ simdi=11082.0, acilis=11059.0, yuksek=13604.0, dusuk=9949.4, hacim=1.3186E7, fark=-18.44, sinif=6}
DataModel{ simdi=11045.0, acilis=11101.0, yuksek=11490.0, dusuk=9231.1, hacim=1.3223E7, fark=0.09, sinif=5}
DataModel{ simdi=11476.0, acilis=11036.0, yuksek=11881.0, dusuk=10515.0, hacim=8524000.0, fark=-0.33, sinif=6}
DataModel{ simdi=12728.0, acilis=11462.0, yuksek=11879.0, dusuk=10649.0, hacim=4855000.0, fark=3.9, sinif=8}
DataModel{ simdi=11514.0, acilis=12732.0, yuksek=13002.0, dusuk=11425.0, hacim=4630000.0, fark=10.91, sinif=3}
DataModel{ simdi=10771.0, acilis=11519.0, yuksek=12732.0, dusuk=11020.0, hacim=5362000.0, fark=-9.54, sinif=4}
DataModel{ simdi=10819.0, acilis=10770.0, yuksek=11886.0, dusuk=10009.0, hacim=6635000.0, fark=-6.45, sinif=6}
DataModel{ simdi=11414.0, acilis=10811.0, yuksek=11383.0, dusuk=9901.1, hacim=6385000.0, fark=0.45, sinif=7}
DataModel{ simdi=11146.0, acilis=11402.0, yuksek=11529.0, dusuk=10454.0, hacim=4382000.0, fark=5.5, sinif=5}
DataModel{ simdi=11070.0, acilis=11144.0, yuksek=11723.0, dusuk=10857.0, hacim=3444000.0, fark=-2.35, sinif=5}
DataModel{ simdi=11461.0, acilis=11068.0, yuksek=11647.0, dusuk=10298.0, hacim=5675000.0, fark=-0.68, sinif=6}
DataModel{ simdi=11839.0, acilis=11455.1, yuksek=11683.0, dusuk=10822.0, hacim=3440000.0, fark=3.53, sinif=6}
DataModel{ simdi=11212.0, acilis=11836.0, yuksek=12181.0, dusuk=11398.0, hacim=2962000.0, fark=3.3, sinif=4}
DataModel{ simdi=10175.0, acilis=11215.0, yuksek=11957.0, dusuk=11074.0, hacim=2565000.0, fark=-5.3, sinif=3}
DataModel{ simdi=10284.0, acilis=10170.0, yuksek=11250.0, dusuk=9864.9, hacim=6128000.0, fark=-9.25, sinif=6}
DataModel{ simdi=9181.1, acilis=10278.0, yuksek=10411.0, dusuk=9761.0, hacim=4323000.0, fark=1.07, sinif=3}
DataModel{ simdi=8895.8, acilis=9181.0, yuksek=10311.0, dusuk=8941.0, hacim=7878000.0, fark=-10.72, sinif=5}
```

Şekil 2.6. Bitcoin Verilerinin Modellenmiş Hali



Şekil 2.7. Bitcoin Verilerinin Modellenmiş Halinin Grafiği

2.2. Rol ve Sorumluluklar

Tarih	Tanım	Yetkili
26/02/2018	Yazılım gereksinimlerinin belirlenmesi	✓ Simge Nur Kabataş ✓ Zeynep Fenerci
10/03/2018	Mevcut risklerin tanımlanması	✓ Simge Nur Kabataş ✓ Zeynep Fenerci

13/03/2018	Veri Setlerinin Analiz Edilmesi	✓ Simge Nur Kabataş ✓ Zeynep Fenerci
17/04/2018	Veri Setlerinin Algoritmaya İşlenmesi	✓ Simge Nur Kabataş ✓ Zeynep Fenerci

2.3. Araç ve Teknikler

Projeyi oluşturacak alt parçalar aşağıda belirtilmiştir.

- Windows 10
- Intellij IDEA
- Bitcoin verisetleri
- Stream olarak alınan bitcoin verisi

Kaynak kod Java programlama diliyle yazılacaktır. Kodlamada Intellij IDEA Ide'si kullanılacak olup Windows 10 işletim sistemi üzerinde Apache Flink çalıştırılıp , önce batch data için model oluşturup modelin eğitimi gerçekleştirilip daha sonra stream olarak alınan datanın Java ' nın machine learning kütüphanesi olan Weka kütüphanesi kullanılarak farklı algoritmaların çalıştırlmasıyla proje gerçekleştirilecektir.

Apache Flink

Apache Flink, veri akışları üzerinden dağıtılmış hesaplamalar için iletişim, hata toleransı ve veri dağıtımı sağlayan bir veri akış motoru olan açık kaynaklı bir platformdur. Ayrıca Flink, Hadoop ile tamamen uyumlu ölçeklenebilir bir veri analizi çerçevesi ve çok yüksek hızda üretilen verileri işleyebilen büyük ölçekli veri işleme çerçevesidir. Apache Flink, yeni nesil Büyük Veri aracıdır. Flink, Apache' nin üst düzey bir projesidir. Flink hem akış işlemeyi hem de toplu işlemeyi kolayca çalıştırabilir. Apache Flink'in pipeline mimarisi, akışlı verilerin mikro toplu mimarilere (Spark) göre daha düşük gecikme ile daha hızlı işlenmesini sağlar [4].

Apache Camel

Apache Camel, bilinen Kurumsal Entegrasyon Kalıplarına dayanan çok yönlü bir açık kaynak entegrasyon çerçevesidir. Camel, Java tabanlı Akıcı API veya Blueprint XML Yapılandırma dosyaları ve bir Scala DSL dahil olmak üzere, alana özgü çeşitli dillerde

yönlendirme ve birleştirme kurallarını tanımlamanızı sağlar . Java, Scala veya XML editöründe ya da IDE'nizde yönlendirme kurallarının akıllıca tamamlanmasını sağlar. Apache Camel, herhangi bir Java uygulamasında kolayca gömülmek için minimum bağımlılığa sahip küçük bir kütüphanedir. Apache Camel, hangi tür Aktarım kullanıldığına bakılmaksızın aynı API ile çalışmanıza izin verir - bu nedenle API'yi bir kez öğrenin ve kutudan sağlanan tüm Bileşenler ile etkileşim kurabilirsiniz. Apache Camel, CDI ve Spring, Blueprint ve Guice gibi popüler çerçevelerle sorunsuz entegrasyon için destek sağlar . Camel ayrıca rotalarınızı test etmek için kapsamlı bir desteğe de sahiptir. Streaming datayı kuyruğa ve veritabanına atmka için kullanılmıştır [5].

- Yaygın olarak kullanılan tüm Kurumsal Entegrasyon Kalıplarının (EIP) somut uygulamaları
- Çok çeşitli taşımacılık ve API'lara bağlantı
- EIP'leri bağlamak ve birlikte taşımak için Etki Alanı'na Özel Diller (DSL) kullanımı kolay

MicroSerce Architech

Aynı zamanda mikro servis mimarisi olarak da bilinir - bir uygulamayı, işletme yeteneklerini uygulayan gevşek bir şekilde birleştirilmiş hizmetler topluluğu olarak yapılandıran bir mimari stildir. Mikroservis mimarisi, büyük, karmaşık uygulamaların sürekli dağıtım / dağıtımını sağlar. Ayrıca bir organizasyonun teknoloji kümesini geliştirmesini sağlar [6].

Spring

Stream datanın kaydeilmesi içn gerekli olan frameworktür. Güvenlik için kullanılır [9].

Özellikler

- Bağımsız Spring uygulamaları oluşturma
- Tomcat, Jetty veya Undertow'u doğrudan gömün (WAR dosyalarını dağıtmaya gerek yoktur)
- Yapı yapılandırmanızı basitleştirmek için görüş bildiren 'başlangıç' bağımlılıkları sağlayın
- Mümkün olduğunda Yay ve 3. taraf kitaplıklarını otomatik olarak yapılandırın
- Metrikler, sağlık kontrolleri ve harici yapılandırma gibi üretime hazır özellikler sağlayın
- Kesinlikle kod oluşturma ve XML yapılandırması için gereklilik yok

Apache Maven

Apache Maven bir yazılım proje yönetimi ve anlama aracıdır. Bir proje nesnesi modeli (POM) kavramına dayanarak bir projenin oluşturulmasını, raporlanmasını ve dokümantasyonunu merkezi bir bilgi parçasından yönetebilir.

Bilginin biriktirilmesi anlamına gelen Yidce bir sözcük olan Maven, başlangıçta Jakarta Türbin projesinde inşa süreçlerini basitleştirmeye yönelik bir girişim olarak başlatıldı. Her biri biraz farklı olan kendi Ant build dosyalarını içeren birkaç proje vardı ve JAR'ler CVS'ye kontrol edildi. Projeleri oluşturmanın standart bir yolunu, projenin nelerden oluştuğunu açık bir şekilde tanımlamak, proje bilgilerini yayınlamanın kolay bir yolunu ve JAR'ları birçok projede paylaşmanın bir yolunu istedik.

Sonuç, artık herhangi bir Java tabanlı projeyi oluşturmak ve yönetmek için kullanılabilecek bir araçtır. Java geliştiricilerinin günlük çalışmalarını kolaylaştıracak ve genellikle Java tabanlı bir projenin anlaşılmasına yardımcı olacak bir şey yarattığımızı umuyoruz.

Maven'in birincil hedefi, bir geliştiricinin en kısa sürede bir geliştirme çabasının tam durumunu kavramasına izin vermektir. Bu hedefe ulaşmak için, Maven'in uğraşmak istediği bazı endişeler vardır:

- Yapım sürecini kolaylaştırmak
- Tekdüzen bir yapı sistemi sağlanması
- Kaliteli proje bilgisi sağlamak
- En iyi uygulama geliştirme için kurallar sağlamak
- Yeni özelliklere şeffaf geçişe izin verme

Maven kısmen POM'ınızdan alınan ve kısmen projenizin kaynaklarından elde edilen yararlı proje bilgileri sağlar. Örneğin Maven şunları sağlayabilir:

- Doğrudan kaynak kontrolünden oluşturulan günlük dokümanı değiştir
- Çapraz referanslı kaynaklar
- Posta listeleri
- Bağımlılık listesi
- Kapsam dahil olmak üzere birim test raporları

Maven'in iyileştirdiği gibi, sağlanan bilgiler iyileştirilecek ve bunların hepsi Maven kullanıcısına şeffaf olacaktır. Diğer ürünler Maven eklentileri, Maven tarafından verilen bazı standart bilgilerin yanısıra yine POM'a dayanarak proje bilgileri setine izin verir [7].

Hibernate

Hibernate, Java platformunda yazılmış bir ORM (Object/Relational Mapping) aracıdır. ORM, nesne odaklı (object oriented) dillerdeki nesnelerin, ilişkisel veri tabanlarındaki (relational databases) kayıtlara nasıl karşılık geldiğini yürüten bir teknolojidir. NHibernate adında .NET çatısı için yeniden yazılmış bir türevi bulunur. Hibernate gibi ORM araçlarıyla, bir nesneyi veri tabanına kaydetmek, yeni halini güncellemek ve sorgulama yapmak düz SQLbağlantılarına göre çok kolaydır [8].

Spring Secuty

Spring Security, Java uygulamalarına hem kimlik doğrulamayı hem de yetkilendirmeyi sağlamaya odaklanan bir çerçevedir. Tüm Spring projelerinde olduğu gibi, Spring Security'nin gerçek gücü de özel gereksinimleri karşılamak için ne kadar kolay genişletilebileceği bulundu [10].

Özellikler

- Kimlik Doğrulama ve Yetkilendirme için kapsamlı ve genişletilebilir destek
- Oturum sabitleme, tıklama, çapraz site talebi sahteciliği gibi saldırılara karşı koruma
- Servlet API entegrasyonu
- Spring Web MVC ile isteğe bağlı entegrasyon

```
<dependencies>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-web</artifactId>
        <version>5.0.5.RELEASE</version>
    </dependency>
</dependencies>
```

JPA (Java Persistence API)

Java Persistence API, büyük miktarlardaki verileri bir veritabanına kalıcı olarak depolamak için kullanılan sınıflar ve yöntemlerden oluşan bir koleksiyondur. Bu öğretici, Kalıcılık temel

anlayışını (veritabanı nesnesinin kopyasını geçici belleğe saklamak) sağlar ve JAVA Persistence API'sinin (JPA) anlaşılmasını öğreniriz.

Herhangi bir kurumsal uygulama, büyük miktarlarda veri depolayarak ve geri getirerek veritabanı işlemlerini gerçekleştirir. Depolama yönetimi için mevcut tüm teknolojilere rağmen, uygulama geliştiricileri normal olarak verimli veritabanı işlemleri gerçekleştirmek için mücadele ediyoruz.

Genellikle, Java geliştiricileri kod birçok kullanın veya JPA kullanarak oysa, veritabanı ile etkileşim için özel bir çerçeve kullanın veritabanı ile etkileşim yükü önemli ölçüde azaltır. Nesne modelleri (Java programı) ve ilişkisel modeller (veritabanı programı) arasında bir köprü oluşturur [10].

Spring Data

Spring Data'nın görevi, temel veri deposunun özel özelliklerini korurken, veri erişimi için bilinen ve tutarlı bir Spring tabanlı programlama modeli sağlamaktır.

Veri erişim teknolojilerini, ilişkisel ve ilişkisel olmayan veritabanlarını, harita küçültme çerçevelerini ve bulut tabanlı veri servislerini kullanmayı kolaylaştırır. Bu, belirli bir veritabanına özgü birçok alt proje içeren bir şemsiye projesidir. Projeler, bu heyecan verici teknolojilerin arkasındaki pek çok şirket ve geliştirici ile birlikte çalışarak geliştirilmiştir [9].

Özellikler

- Güçlü depo ve özel nesne eşleme soyutlamaları
- Depo yöntem adlarından dinamik sorgu türevi
- Temel özellikler sağlayan uygulama alanı temel sınıfları
- Şeffaf denetim desteği (oluşturuldu, en son değiştirildi)
- Özel depo kodunu entegre etme olanağı
- JavaConfig ve özel XML ad alanları aracılığıyla Easy Spring entegrasyonu
- Bahar MVC kontrolörleri ile gelişmiş entegrasyon
- Mağazalar arası kalıcılık için deneysel destek

Intellij IDEA: Kaynak kodun yazılacağı IDE' dir.

MS Word: Projenin dokümanlarını yazmak için kullanılacak olan programdır.

Uygulama yaparken kullanılacak olan araç ve teknikler aşağıdaki kriterler doğrultusunda

kararlaştırıldı;

- Stabilite
- Performans
- Görünüm ve etkileşim
- Anlaşılabilirdik ve kullanılabilirlik
- İleri Geliştirme ve Bakım Süreçleri
- Kullanacağımız platform, araçların eklenti desteği
- Kişisel bilgisayarımızın destekleyip desteklemediği

2.4. Araç ve Teknolojilerin Temini

Apache Flink teknolojisi kullanılmıştır. Apache Flink teknolojisinin temini ise kendi sitesinde gerekli dosyaların indirilmesiyle kurulumu gerçekleştirilmiştir.

Tahmin için gerekli olan makine öğrenmesi algoritmaları için Java'nın weka kütüphanesi kullanılmıştır.

3. VERİTABANI TASARIMI

3.1. Tanım

Projenin ikinci kısmında streaming datayı alıp bir tane veritabanına kaydedip daha sonra o veriyi projemizin başarısını artırmak için kullanılması amaçlanmıştır. Bu amaçla light base bir tane veri tabanı kullanılması tercih edilmiştir. Bu amaçla popüler veri tabanlarından h2 veri tabanı kullanılmıştır. Bunu seçme sebebimiz ise okuma ve yazma performansının kullanılan teknolojiler ile uyumlu çalışabilmesidir.

3.2. Kaynak İhtiyacı

Veritabanına kaydedilmek üzere her 40 sn bir değişen streaming dataya ihtiyaç vardır. İnternete çıkışı olan bir tane cihaza ihtiyacımız vardır. Bunun için gerekli ayarların firewallda verilmesi gerekmektedir.

3.3. Sonuçlar ve Temeltaşlar

3.4. Bağımlılıklar ve Kısıtlar

Güvenli anlık işleyen verisetine ihtiyaç vardır.

3.5. Riskler ve Arızalar

Risk olarak gerekli donanım cihazlarının olmaması ya da arIza vermesi durumları öngörülmüştür. Bunun için proje bakım aşamasında verilerin yedeklenmesi yapılacaktır.

4. ARAYÜZ TASARIMI

Arayüz tasarımı yapılmamıştır. Jar olarak terminal ekranından java –jar komutu ile çalıştırılması gerekmektedir.

5. KODLAMA

5.1. Tanım

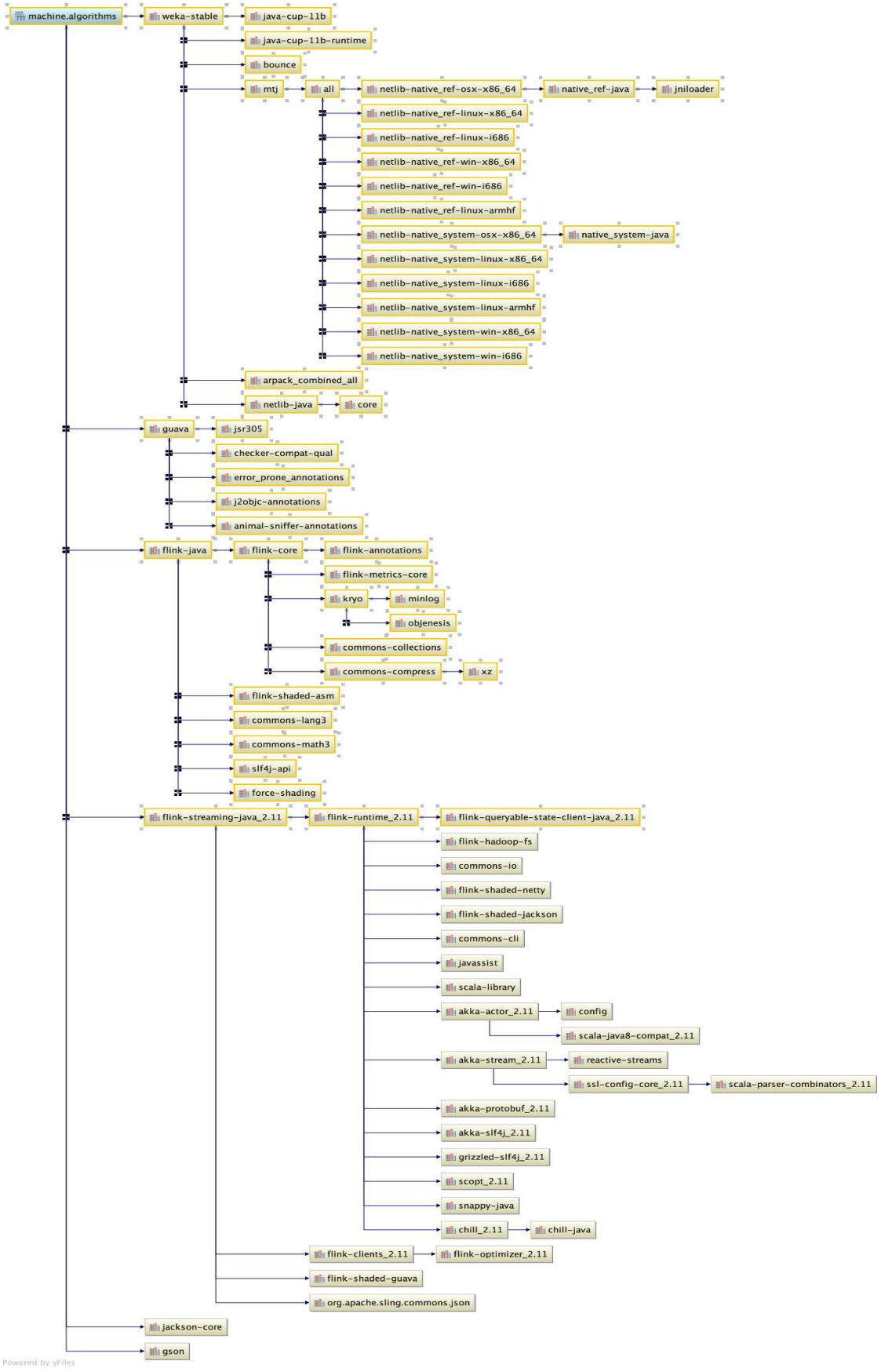
5.2. Sonuçlar ve Temeltaşlar

Projeyle ilgili gerekli rapor ve literatür taraması işlemleri gerçekleştirilmiştir. Doğru ve güvenilir veri setlerine ihtiyaç vardır. Bu veri setleri gerçek tarihleri içermelidir ve her gün kendini güncellemelidir.

5.3. Bağımlılıklar ve Kısıtlar

Kod Dependency

Genel olarak, ne kadar çok veri seti o kadar iyidir. Genel kanının aksine daha fazla veri seti manuel iş miktarını önemli ölçüde artırmaz. Hesaplama süreleri üzerinde, genellikle doğru veri analizi BT altyapısı ile ihmal edilebilir olan küçük bir etkisi vardır. Kod bağımlılık haritası aşağıda vardır. Uygulamada java dili ile bir çeşit web crawler (Yukarıda bahsedilecek) yazılmış olup apache flink streaming kütüphanesi kullanılarak real time sonuçlar üreten isterleri karşılayan bir çeşit öneri motoru yazılmıştır.


machine.algorithms
weka-stable
java-cup-11b
java-cup-11b-runtime
bounce
mtj
all
netlib-native_ref-osx-x86_64
native_ref-java
jniloader
netlib-native_ref-linux-x86_64
netlib-native_ref-linux-i686
netlib-native_ref-win-x86_64
netlib-native_ref-win-i686
netlib-native_ref-linux-armhf
netlib-native_system-osx-x86_64
native_system-java
netlib-native_system-linux-x86_64
netlib-native_system-linux-i686
netlib-native_system-linux-armhf
netlib-native_system-win-x86_64
netlib-native_system-win-i686
arpack_combined_all
netlib-java
core
guava
jsr305
checker-compat-qual
error_prone_annotations
j2objc-annotations
animal-sniffer-annotations
flink-java
flink-core
flink-annotations
flink-metrics-core
kryo
minlog
objenesis
commons-collections
commons-compress
xz
flink-shaded-asm
commons-lang3
commons-math3
slf4j-api
force-shading
flink-streaming-java_2.11
flink-runtime_2.11
flink-queryable-state-client-java_2.11
flink-hadoop-fs
commons-io
flink-shaded-netty
flink-shaded-jackson
commons-cli
javassist
scala-library
akka-actor_2.11
config
scala-java8-compat_2.11
akka-stream_2.11
reactive-streams
ssl-config-core_2.11
scala-parser-combinators_2.11
akka-protobuf_2.11
akka-slf4j_2.11
grizzled-slf4j_2.11
scopt_2.11
snappy-java
chill_2.11
chill-java
flink-clients_2.11
flink-optimizer_2.11
flink-shaded-guava
org.apache.sling.commons.json
jackson-core
gson
Powered by yFiles

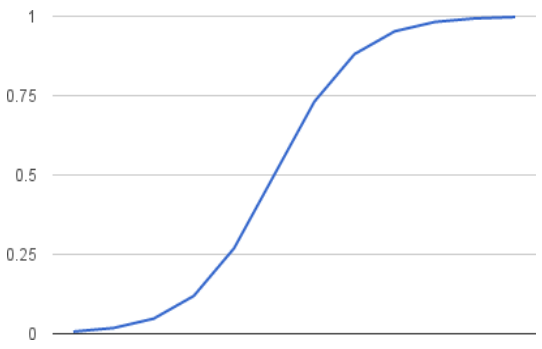
6. TEST AŞAMASI

6.1. Sonuçlar ve Temeltaşlar

Weka'nın kendi içindeki algoritmalarında eğitim verimiz eğitilip, daha sonra stream olarak aldığı veriye göre de tahmin gerçekleştirip daha sonra accuracylerin ölçümü yapılmıştır. Kullanılan her bir algoritma için accuracy aşağıda gösterilmektedir.

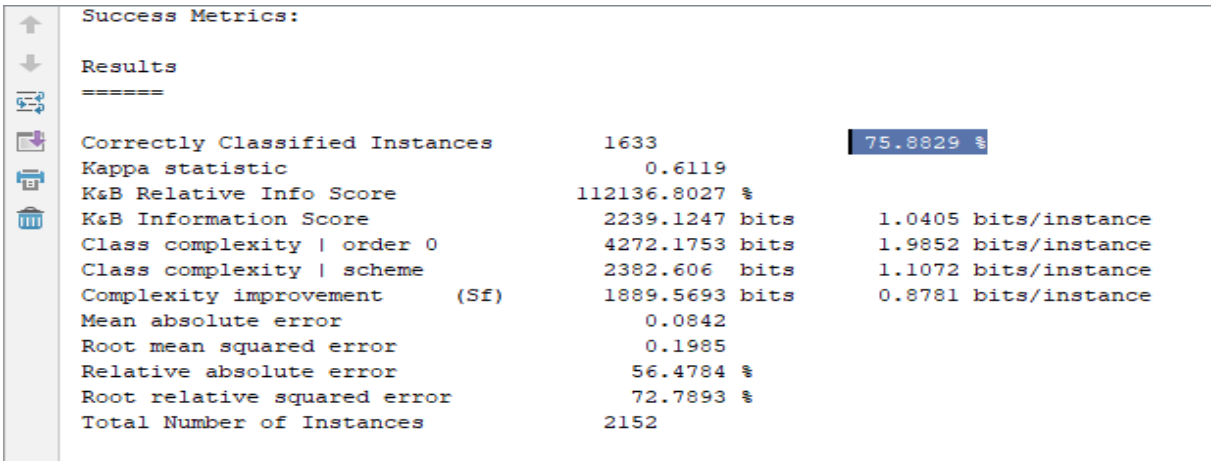
Logistic Regression Algoritması:

Lojistik Regresyon, düzlem üzerinde verileri yakalamaya çalışır. Ancak bu algoritma bunları doğrusal değil de Logaritmik olarak eğri üzerinde yakalamaktadır. Bu da bize inişli çıkışlı verilerde daha yüksek tahmin başarısı getirmektedir.



Şekil 6.1. Logistic Regression Algoritması

Formül olarak **1 / (1 + e^-x)** formülü kullanarak eğriyi oluşturulmaktadır. Elimizdeki verisetini algoritmaya uyguladığımızda %76'lık bir başarı elde edilmiştir.



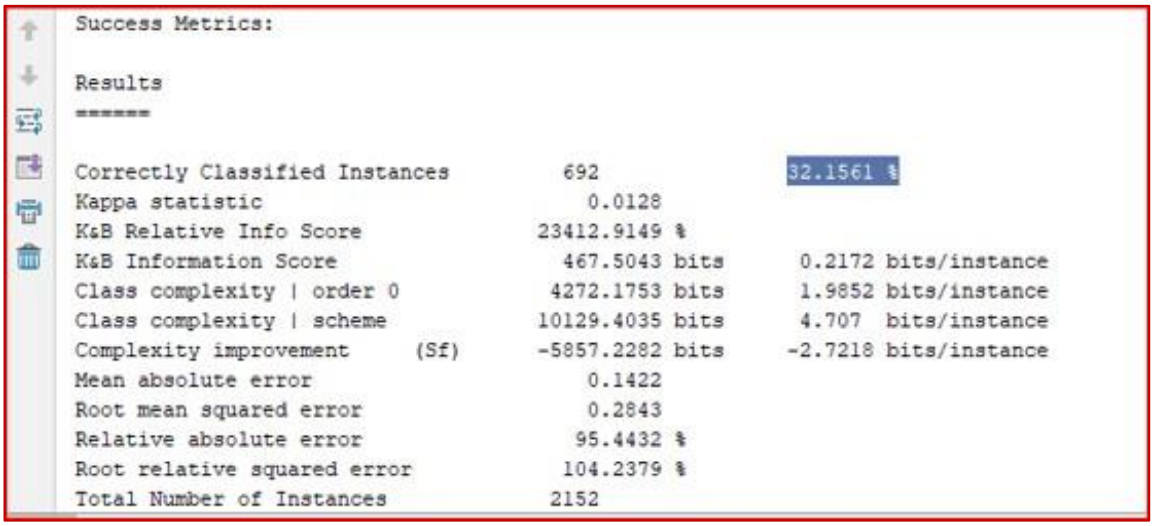
```
Success Metrics:

Results
=======

Correctly Classified Instances        1633               75.8829 %
Kappa statistic                          0.6119
K&B Relative Info Score             112136.8027 %
K&B Information Score                 2239.1247 bits      1.0405 bits/instance
Class complexity | order 0            4272.1753 bits      1.9852 bits/instance
Class complexity | scheme             2382.606  bits      1.1072 bits/instance
Complexity improvement     (Sf)       1889.5693 bits      0.8781 bits/instance
Mean absolute error                      0.0842
Root mean squared error                  0.1985
Relative absolute error                 56.4784 %
Root relative squared error             72.7893 %
Total Number of Instances             2152
```

Şekil 6.2. Logistic Regression Algoritması Başarı Oranı

Naive Bayes Algoritması:

Naive Bayes sınıflandırma algoritması, adını Matematikçi Thomas Bayes'den alan bir sınıflandırma/ kategorilendirme algoritmasıdır. Naive Bayes sınıflandırması olasılık ilkelerine göre tanımlanmış bir dizi hesaplama ile, sisteme sunulan verilerin sınıfını yani kategorisini tespit etmeyi amaçlar.

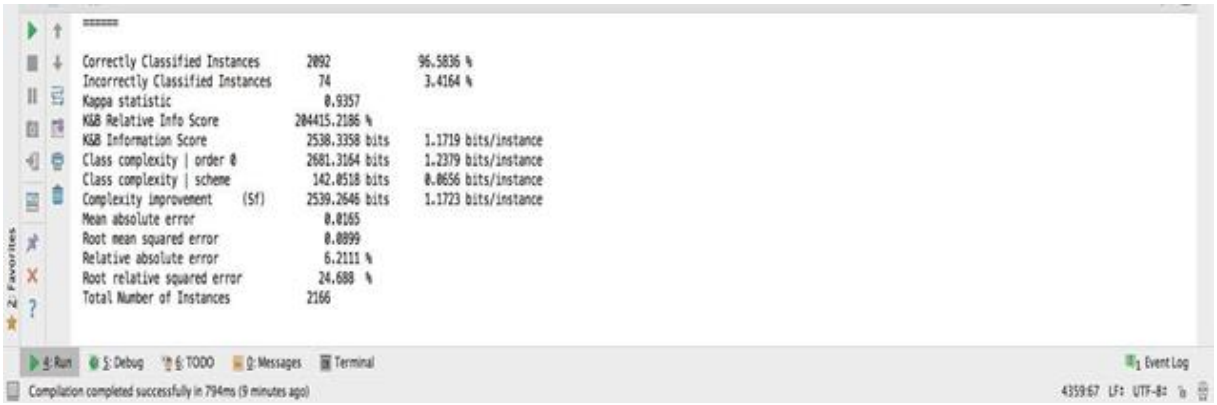
Naive Bayes sınıflandırmasında sisteme belirli bir oranda öğretilmiş veri sunulur (Örn: 100 adet). Öğretim için sunulan verilerin mutlaka bir kategorisi bulunmalıdır. Öğretilmiş veriler üzerinde yapılan olasılık işlemleri ile sisteme sunulan yeni test verileri, daha önce elde edilmiş olasılık değerlerine göre işletilir ve verilen test verisinin hangi kategoride olduğu tespit edilmeye çalışılır. Elbette öğretilmiş veri sayısı ne kadar çok ise, test verisinin gerçek kategorisini tespit etmek o kadar kesin olabilmektedir. Elimizdeki verisetini algoritmaya uyguladığımızda %32'lık bir başarı elde edilmiştir.



Şekil 6.3. Naive Bayes Algoritması Başarı Oranı

Random Forest:

Random Forest algoritması, hiper parametreli ayar yapmadan bile, çoğu zaman büyük bir sonuç üreten, esnek, kullanımı kolay bir makine öğrenme algoritmasıdır. Aynı zamanda en çok kullanılan algoritmalardan biridir, çünkü hem basitlik hem de sınıflandırma ve regresyon görevleri için kullanılabilir.



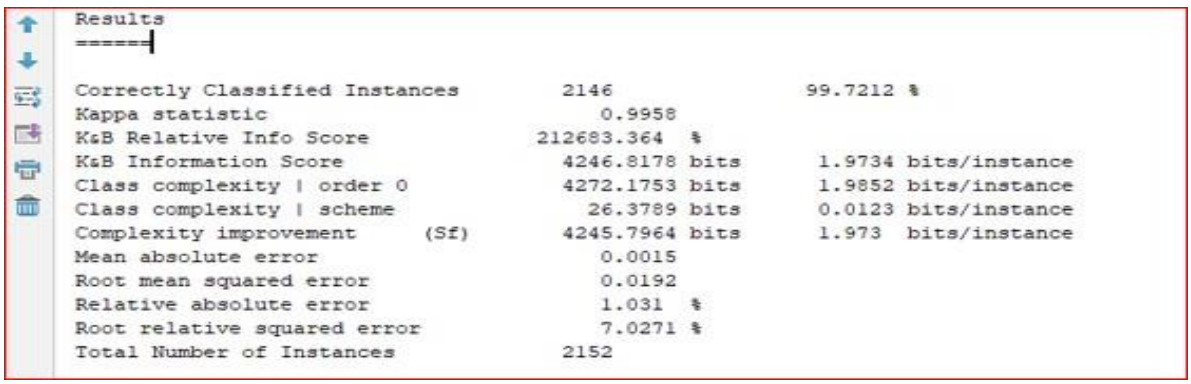
Şekil 6.4. Random Forest Başarı Oranı

KNN Algoritması:

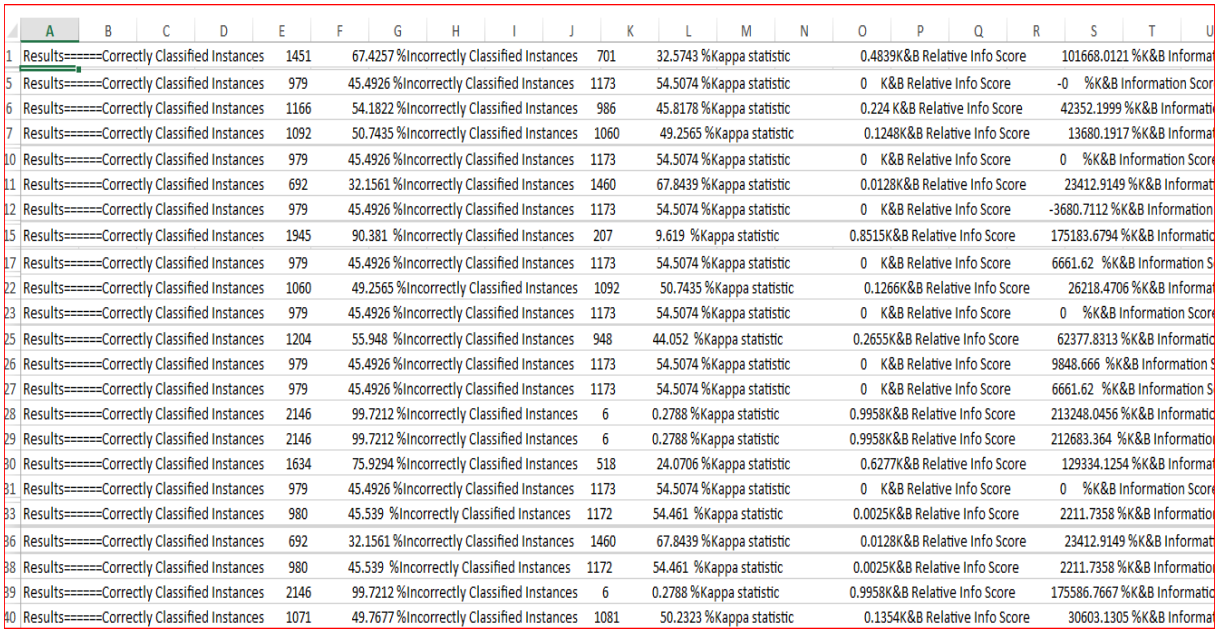
Bu algoritmanın adı K komşu algoritmasıdır. Çalışma mantığı nesnelerin bir biri arasında yakınlık ilişkilerine göre kümeleme işlemi yapar. Doğrusal ayrıştırma yöntemi ile koordinat düzleminde çalışır. Bu algoritma beş adımdan oluşur.

1. Öncelikle K değeri belirlenir.
2. Diğer nesnelerden hedef nesneye olan öklid uzaklıkları hesaplanır.
3. Uzaklıklar sıralanır ve en minimum uzaklığa bağlı olarak en yakın komşular bulunur.
4. En yakın komşu kategorileri toplanır.
5. En uygun komşu kategorisi seçilir.

Bunlardan Weka Kütüphanesi kullanıldı. Daha sonra KNN algoritmasını verisetine uygulandı. Çıktı sonucu Şekil 5'te görüldüğü üzere %99 olarak görülmektedir. Bu başarı oldukça iyidir.

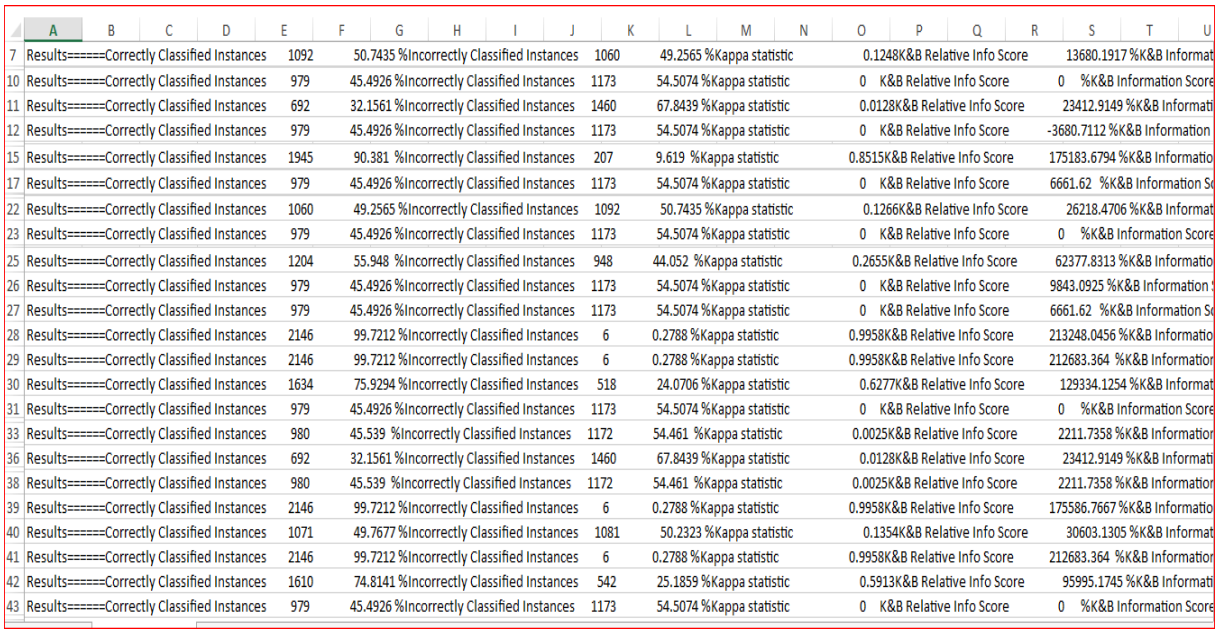


Şekil 6.5. KNN Algoritması Başarı Oranı



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Results======	Correctly Classified Instances	1451	67.4257	%Incorrectly Classified Instances	701	32.5743	%Kappa statistic							0.4839	K&B Relative Info Score			101668.0121	%K&B Informat	
5	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			-0	%K&B Information Scor	
6	Results======	Correctly Classified Instances	1166	54.1822	%Incorrectly Classified Instances	986	45.8178	%Kappa statistic							0.224	K&B Relative Info Score			42352.1999	%K&B Informati	
7	Results======	Correctly Classified Instances	1092	50.7435	%Incorrectly Classified Instances	1060	49.2565	%Kappa statistic							0.1248	K&B Relative Info Score			13680.1917	%K&B Informat	
10	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			0	%K&B Information Score	
11	Results======	Correctly Classified Instances	692	32.1561	%Incorrectly Classified Instances	1460	67.8439	%Kappa statistic							0.0128	K&B Relative Info Score			23412.9149	%K&B Informat	
12	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			-3680.7112	%K&B Information	
15	Results======	Correctly Classified Instances	1945	90.381	%Incorrectly Classified Instances	207	9.619	%Kappa statistic							0.8515	K&B Relative Info Score			175183.6794	%K&B Informatic	
17	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			6661.62	%K&B Information S	
22	Results======	Correctly Classified Instances	1060	49.2565	%Incorrectly Classified Instances	1092	50.7435	%Kappa statistic							0.1266	K&B Relative Info Score			26218.4706	%K&B Informat	
23	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			0	%K&B Information Score	
25	Results======	Correctly Classified Instances	1204	55.948	%Incorrectly Classified Instances	948	44.052	%Kappa statistic							0.2655	K&B Relative Info Score			62377.8313	%K&B Informatic	
26	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			9848.666	%K&B Information S	
27	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			6661.62	%K&B Information S	
28	Results======	Correctly Classified Instances	2146	99.7212	%Incorrectly Classified Instances	6	0.2788	%Kappa statistic							0.9958	K&B Relative Info Score			213248.0456	%K&B Informatic	
29	Results======	Correctly Classified Instances	2146	99.7212	%Incorrectly Classified Instances	6	0.2788	%Kappa statistic							0.9958	K&B Relative Info Score			212683.364	%K&B Informatio	
30	Results======	Correctly Classified Instances	1634	75.9294	%Incorrectly Classified Instances	518	24.0706	%Kappa statistic							0.6277	K&B Relative Info Score			129334.1254	%K&B Informat	
31	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			0	%K&B Information Score	
33	Results======	Correctly Classified Instances	980	45.539	%Incorrectly Classified Instances	1172	54.461	%Kappa statistic							0.0025	K&B Relative Info Score			2211.7358	%K&B Informatio	
36	Results======	Correctly Classified Instances	692	32.1561	%Incorrectly Classified Instances	1460	67.8439	%Kappa statistic							0.0128	K&B Relative Info Score			23412.9149	%K&B Informat	
38	Results======	Correctly Classified Instances	980	45.539	%Incorrectly Classified Instances	1172	54.461	%Kappa statistic							0.0025	K&B Relative Info Score			2211.7358	%K&B Informatio	
39	Results======	Correctly Classified Instances	2146	99.7212	%Incorrectly Classified Instances	6	0.2788	%Kappa statistic							0.9958	K&B Relative Info Score			175586.7667	%K&B Informatic	
40	Results======	Correctly Classified Instances	1071	49.7677	%Incorrectly Classified Instances	1081	50.2323	%Kappa statistic							0.1354	K&B Relative Info Score			30603.1305	%K&B Informat	

Şekil 6.6. Exele Aktarılan algoritmalardaki Başarı



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
7	Results======	Correctly Classified Instances	1092	50.7435	%Incorrectly Classified Instances	1060	49.2565	%Kappa statistic							0.1248	K&B Relative Info Score			13680.1917	%K&B Informat	
10	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			0	%K&B Information Score	
11	Results======	Correctly Classified Instances	692	32.1561	%Incorrectly Classified Instances	1460	67.8439	%Kappa statistic							0.0128	K&B Relative Info Score			23412.9149	%K&B Informati	
12	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			-3680.7112	%K&B Information	
15	Results======	Correctly Classified Instances	1945	90.381	%Incorrectly Classified Instances	207	9.619	%Kappa statistic							0.8515	K&B Relative Info Score			175183.6794	%K&B Informatio	
17	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			6661.62	%K&B Information Sc	
22	Results======	Correctly Classified Instances	1060	49.2565	%Incorrectly Classified Instances	1092	50.7435	%Kappa statistic							0.1266	K&B Relative Info Score			26218.4706	%K&B Informat	
23	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			0	%K&B Information Score	
25	Results======	Correctly Classified Instances	1204	55.948	%Incorrectly Classified Instances	948	44.052	%Kappa statistic							0.2655	K&B Relative Info Score			62377.8313	%K&B Informatio	
26	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			9843.0925	%K&B Information	
27	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			6661.62	%K&B Information Sc	
28	Results======	Correctly Classified Instances	2146	99.7212	%Incorrectly Classified Instances	6	0.2788	%Kappa statistic							0.9958	K&B Relative Info Score			213248.0456	%K&B Informatio	
29	Results======	Correctly Classified Instances	2146	99.7212	%Incorrectly Classified Instances	6	0.2788	%Kappa statistic							0.9958	K&B Relative Info Score			212683.364	%K&B Informatio	
30	Results======	Correctly Classified Instances	1634	75.9294	%Incorrectly Classified Instances	518	24.0706	%Kappa statistic							0.6277	K&B Relative Info Score			129334.1254	%K&B Informat	
31	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			0	%K&B Information Score	
33	Results======	Correctly Classified Instances	980	45.539	%Incorrectly Classified Instances	1172	54.461	%Kappa statistic							0.0025	K&B Relative Info Score			2211.7358	%K&B Informatio	
36	Results======	Correctly Classified Instances	692	32.1561	%Incorrectly Classified Instances	1460	67.8439	%Kappa statistic							0.0128	K&B Relative Info Score			23412.9149	%K&B Informati	
38	Results======	Correctly Classified Instances	980	45.539	%Incorrectly Classified Instances	1172	54.461	%Kappa statistic							0.0025	K&B Relative Info Score			2211.7358	%K&B Informatio	
39	Results======	Correctly Classified Instances	2146	99.7212	%Incorrectly Classified Instances	6	0.2788	%Kappa statistic							0.9958	K&B Relative Info Score			175586.7667	%K&B Informatio	
40	Results======	Correctly Classified Instances	1071	49.7677	%Incorrectly Classified Instances	1081	50.2323	%Kappa statistic							0.1354	K&B Relative Info Score			30603.1305	%K&B Informat	
41	Results======	Correctly Classified Instances	2146	99.7212	%Incorrectly Classified Instances	6	0.2788	%Kappa statistic							0.9958	K&B Relative Info Score			212683.364	%K&B Informatio	
42	Results======	Correctly Classified Instances	1610	74.8141	%Incorrectly Classified Instances	542	25.1859	%Kappa statistic							0.5913	K&B Relative Info Score			95995.1745	%K&B Informati	
43	Results======	Correctly Classified Instances	979	45.4926	%Incorrectly Classified Instances	1173	54.5074	%Kappa statistic							0	K&B Relative Info Score			0	%K&B Information Score	

Şekil 6.7. Exele Aktarılan algoritmalardaki Başarı

8. SONUÇ

Sonuç olarak, toplanmış batch bitcoin verilerini eğiterek stream bitcoin versieti üzerinde tahminde bulunmuş olduk. Bu projede IDE üzerinden Apache Flink kullanılarak machine learnig farklı farklı algoritmalar üzerindeki başarılar sonuçlandırılmıştır.

9. KAYNAKLAR

[1] **İnternet:***Codex*. (2014). Mayıs 29, 2018 tarihinde Yazılım Geliştirme Süreç Modelleri: https://www.codex.com.tr/yazilim-gelistirme-modelleri adresinden alındı

[2]İnternet: *İnvesting.com*. (tarih yok). Şubat 12, 2018 tarihinde Kripto: https://www.investing.com/crypto/bitcoin/btc-usd-historical-data adresinden alındı

[3] **İnternet:** (2010, Ocak 09). Mayıs 29, 2018 tarihinde Real Time Credit Card Fraud Detection with Apache Spark and Event Streaming: https://www.cnblogs.com/bnuvincent/p/8253797.html adresinden alındı

[4] **İnternet:** (2014-2017). Ocak 20, 2018 tarihinde Apache Flink: https://flink.apache.org/ adresinden alındı

[5] **İnternet: :** (2004-2015). Mayıs 09, 2018 tarihinde Apache Camel: http://camel.apache.org/what-is-camel.html adresinden alındı

[6] **İnternet:** Richardson, C. (tarih yok). *Microservice Architecture*. Mayıs 20, 2018 tarihinde http://microservices.io/ adresinden alındı

[7] **İnternet:** *Apache Maven Project*. (2002–2018). Mart 12, 2018 tarihinde What is Maven ?: https://maven.apache.org/ adresinden alındı

[8] **İnternet:** *tutorialspoint*. (2018). Mayıs 10, 2018 tarihinde Hibernate: https://www.tutorialspoint.com/hibernate/hibernate_overview.htm adresinden alındı

[9] **İnternet:** *Spring*. (2018). Nisan 20, 2018 tarihinde Spring Security: https://projects.spring.io/spring-security/ adresinden alındı

[10] **İnternet:** *tutorialpoint*. (2018). Mayıs 09, 2018 tarihinde JPA: https://projects.spring.io/spring-security/ adresinden alındı