**T.C.**

**SAKARYA ÜNİVERSİTESİ**

**FEN BİLİMLER ENSTİTÜSÜ**

**SAKARYA ÜNİVERSİTESİ**

**MEMNUNİYET ANALİZİ**

**YÜKSEK LİSANS PROJESİ**

**ABDURRAHMAN BURAK HALEFOĞLU**

**Enstitü Bilim Dal ı: Bilişim Teknolojileri Tezsiz Yüksek Lisans**

**Proje Danışmanı : Prof.Dr. Ümit KOCABIÇAK**

**OCAK- 2022**

**T.C.**

**SAKARYA ÜNİVERSİTESİ**

**FEN BİLİMLER ENSTİTÜSÜ**

**SAKARYA ÜNİVERSİTESİ**

**MEMNUNİYET ANALİZİ**

**YÜKSEK LİSANS PROJESİ**

**ABDURRAHMAN BURAK HALEFOĞLU**

**Enstitü Bilim Dal ı: Bilişim Teknolojileri Tezsiz Yüksek Lisans**

**İÇİNDEKİLER**

**ÖZET................................................................................................................... 4**

**GİRİŞ................................................................................................................... 5**

**BÖLÜM 1 – TEKNOLOJİLERİN TANIMI VE TARİHÇESİ.................. 7**

* 1. AÇIK KAYNAK FELSEFESİ VE TEKNOJİLERİ..................................... 7
  2. VERİ TABANLARI.................................................................................... 8
  3. FRAMEWORK’LER................................................................................... 9
  4. SANALLAŞTIRMA UYGULAMALARI VE BULUT BİLİŞİM ................ 9

**BÖLÜM 2 – TEKNOLOJİLERİN TANIMI VE TARİHÇESİ........................ 10**

**BÖLÜM 3 – VERİ BİLİMİ VE BÜYÜK VERİ................................................. 11**

**BÖLÜM 4 – SAKARYA ÜNİVERSİTESİ MEMNUNİYET ANALİZİ......... 12**

4.1 KULLANILAN TEKNOLOJİLERİN TANITIMI................................ 12

* + 1. .net 5.0 ve Devarchitecture ................................................................... 12
    2. Angular................................................................................................. 14
    3. Python................................................................................................... 14
    4. ELK-Stack............................................................................................. 14
    5. Mongodb............................................................................................. 15
    6. Apache Kafka...................................................................................... 15
    7. Apache Spark........................................................................................ 15
    8. Kubernetes .......................................................................................... 15

4.2 PROJE YOL HARİTASI.................................................................... 16

4.2.1 Proje Planı............................................................................................ 16

4.2.2 Back-end altpası.................................................................................. 16

4.2.3 Front-end altyapısı............................................................................. 18

4.2.4 Kubernetes ortamının oluşturulması.................................................... 19

* + 1. Bağımlı teknolojilerin kurulması.......................................................... 19
    2. Veri bilimi modelinin oluşturulması...................................................... 20
    3. Stream altyapısının kurulması ve entegrasyonu...................................... 21
    4. Uygulamaların kurulması ve testi........................................................ 22

**SONUÇ VE ÖNERİLER.................................................................................... 23**

**KAYNAKLAR................................................................................................... 24**

**EKLER.......................................................... .................................................... 26**

**ÖZGEÇMİŞ....................................................................................................... 28**

**SAÜ, Fen Bilimleri Enstitüsü Yüksek Lisans Proje Özeti**

|  |
| --- |
| **Projenin Başlığı:** “Sakarya Üniversitesi memnuniyet analizi |
| **Projenin Yazarı:** Abdurrahman Burak HALEFOĞLU **Danışman:** Prof. Dr. Ümit KOCABIÇAK |
| **Kabul Tarihi:** 11.09.2021 **Sayfa Sayısı:** VII (ön kısım) + 88 (proje) + 9 (ekler) |
| **Anabilim dalı**: Bilişim Teknolojileri **Bilim dalı:** Tezsiz Yüksek Lisans |
| Bu proje sosyal medya analizini ve çeşitli büyük veri teknolojilerini kullanarak Sakarya Üniversitesinin memnuniyet analizini ölçmektedir. Sosyal medya kullanımının artması ile bu kullanıcıların analizi ve ihtiyaç duyulabilecek çıktıların elde edilmesinin önemi günden güne artmaktadır. Fakat analiz için gereken teknolojilerin, güncel yazılım teknolojileri ile entegre edilmesi için birçok alternatif bulunmakta ve hangilerinin kullanılacağı ile ilgili kafa karışıklığı oluşturmaktadır. Bu proje, yapay zekâ analizi teknolojilerinin büyük veri ve güncel yazılım teknolojileri ile nasıl entegre edilebileceğine dair bir taban bilgi verilecektir. Ayrıca Pyspark’ın “twitter api” ile nasıl entegre edilebileceği, eğitim verisinin nasıl oluşturulabileceği ile ilgili bir fikir verecektir. |
| **Anahtar kelimeler**: Makine öğrenmesi, .Net 5, Angular, Kubernetes, elk stack, spark, kafka, twitter api, memnuniyet analizi |

**Giriş**

**Çalışmanın konusu:**

Memnuniyet analizleri üniversitelerin başarısını ölçmesi için önemli kriterlerden biridir. Bunu ölçmek için kullanılan yöntemlerden biri olan anket yöntemi sosyoloji ve psikoloji alanlarında en çok kullanılan yöntemlerden biridir (Kerlinger, 1986). Bu çalışmalarda çoktan seçmeli sorular öğrencilere yönetilerek üniversitenin başarısı ölçülmektedir.

Günümüz de sosyal Sosyal medya, herkesin ortaklaşa bir şekilde bilgi ve bilgiyi paylaşabileceği önemli platformalar haline gelmeye başladı (Bexheti vd., 2014). Bu nedenle memnuniyet analizleri için sosyal medya yeni kapılar açmış olmaktadır. Bu çalışmada Sakarya üniversitesinin twitter apisi ve twitter verileri kullanılarak memnuniyeti analiz edilmektedir.

**Çalışmanın önemi:**

Açık kaynak teknolojilerinin özellikle github gibi depoların gelmesiyle gitgide yaygınlaşmaktadır (Franco-Bedoya vd., 2017). Apache ve Cloud Native Computing Foundation gibi vakıflar ile teknolojiler sürekli gelişmekte ve yeniler gelmektedir. Bu çalışma ile sosyal medya analizi için gerekli olabilecek araçlar öğrenilmiş olacak. Kullanılabilecek güncel teknolojilerin nasıl bütünleştirildiği ve nasıl sunulduğuna dair bir fikir sahibi olunacaktır.

**Çalışmanın amacı:**

Bu çalışma ile çeşitli büyük veri teknolojileri hakkında fikir sahibi olunacak. Güncel soyutlama ve yönetim teknolojilerinden Docker ve Kubernetes hakkında bilgi sahibi olunacak. Ayrıca güncel yazılım teknolojilerinden olan Angular ve .Net ile ilgili kullanım örneklerine ayrıca veri bilimi ve büyük veri ile nasıl entegre edilebileceği hakkına bilgi sahibi olunacak.

**Çalışmanın yöntemi:**

Kullanılan teknolojilere bakacak olursak, angular ara yüz sağlamak ve grafikler oluşturmak için kullanıldı. Dot net 5.0 verileri Ara yüze taşımak için kullanıldı. mongodb verileri barındırmak ve nosql’in gücünü kullanmak için, elk-stack (elastic search, logstash, kibana) yapısı loglama ve görselleştirme için kullanıldı. Apache Spark Stream verilerin gerçek zamanlı analizi ve Apache kafka veri akışı için kullanıldı. kubernetes ve docker uygulamaları soyutlama ve yönetimi için kullanıldı. Projede twitter api kullanılarak, Türkiye’de bulunan insanların Sakarya üniversitesi ile ilgili oluşturduğu tweet’ ler filtrelenerek, veriler toplandı. Bu veriler emojiler kullanılarak, memnun veya memnun değil olarak işaretlendi ve makine öğrenmesi modeli öğretildi. Bu bilgiler .net apisi kullanılarak angular üzerinde grafiğe döküldü. ELK-stack yapısı hataların loglanması için kullanıldı. kubernetes ve docker teknolojisi servisleri soyutlayarak sanal sunucularda barındırmamızı sağlandı. Makine öğrenmesi modeli çıkarıldıktan sonra apinin verdiği sınır kadar kullanıcı gerçek zamanlı analiz edilebilir hale getirildi. Bu oranlar gerçek zamanlı olarak grafiğe yansıtıldı. Ayrıca Sonuçlar veri tabanına kaydedileceği için memnuniyet artışları ve azalışları grafikte izlenebilir hale geldi.

**BÖLÜM 1 – TEKNOLOJİLERİN TANIMI VE TARİHÇESİ**

* 1. **AÇIK KAYNAK FELSEFESİ VE TEKNOJİLERİ**

Kaynak özellikle yazılım projeleri içerinde yazılımın kodları ve içeriğinde kullanılan her şey olarak tanımlayabiliriz.

Açık kaynak bu yazılımın derlenemeden önceki halini tamamının veya bir kısmının erişilebilir bir lisansla lisanslanması ve herkesin erişimine açılması olarak düşünülebilir. Bu sayede projede bulunan bir hatada kodlara direk müdahale mümkün olmaktadır (Bretthauer, 2001). Açık kaynak ürünlere örnek olarak Linux, wordpress, docker, kubernetes, mysql, mongodb ve apache spark gibi uygulamaları, javacript, Java ve python gibi yazılım dillerini, angular, spring boot ve .net core gibi frameworkleri örnek olarak gösterebiliriz. Bu teknolojilere github üzerinden ulaşabilir, kaynak kodlarını inceleyebilir ve istersek bizde geliştirici olarak yazarlar arasına katılabiliriz.

Günümüzde açık kaynak projelerine ve kullanım yaygınlığına bakacak olursak:

Wordpress hakkında sayfasında yapılan tanıma göre “wordPress, 2003 yılında Mike Little ve Matt Mullenweg'in bir b2/cafelog çatısı oluşturmasıyla başladı. Zarif, iyi tasarlanmış bir kişisel yayıncılık sistemine duyulan ihtiyaç o zamanlar fark edilmişti. Bugün wordpress, PHP ve mysql üzerine kuruludur ve GPLv2 kapsamında lisanslanmıştır. Aynı zamanda web'deki tüm sitelerin %43'ünden fazlasının tercih ettiği platformdur” (Wordpress About, 2021). Web sitelerinizi barındırmanızı sağlar. wordpress, woocommerce gibi açık kaynak projeleri mevcuttur.

Apache Software Foundation’ın hakkında sayfasında yapılan tanıma göre “1999 yılında kurulan ASF, bireysel bağışlar ve kurumsal sponsorlar tarafından finanse edilen bir ABD 501(c)(3) hayır kurumudur. Tamamen gönüllülerden oluşan kurulumuz, dünyanın en popüler Web sunucusu yazılımı olan Apache HTTP Sunucusu da dahil olmak üzere 350'den fazla önde gelen Açık Kaynak projesini yönetmektedir” (Apache about, 2021).

Cloud Native Computing Foundation’ın kendi web sayfasında yaptığı tanıma göre “Cloud Native Computing Foundation (CNCF), küresel teknoloji altyapısının kritik bileşenlerini barındırır. CNCF dünyanın en iyi geliştiricilerini, son kullanıcılarını ve satıcılarını bir araya getirir ve en büyük açık kaynak geliştirici konferanslarını yürütür. CNCF, kâr amacı gütmeyen Linux Vakfı'nın bir parçasıdır.” (CNCF about, 2021). Kubernetes teknolojisinde bu vakıftan mezun olmuş ve piyasada çokça kullanılan teknolojilerden biridir.

* 1. **VERİ TABANLARI**

Veri tabanlarının tarihçesi bilgisayarların icadından çok öncelerine, kütüphane, hükumet ve tıbbi kayıtlara kadar dayanmaktadır (Date, 2003). Kâğıt, parşömen vb. veri kayıt araçlarının yerini yazılımın ve bilgisayarların icadıyla bilgisayar ortamları almıştır. Bu sayede daha çok veriyi, daha kolay ve efektif şekilde saklamak mümkün olmuştur.

Veri tabanı fiziksel kayıtların temsili olan verilerin birbiriyle ilişkilerini içeren ayrıca verilerin kendisi hakkında bilgilerinde barındığı ortama verilen addır (Kroenke & Auer, 2007).

İlk nosql terimi Carlo Strozzi tarafından standart SQL arabirimini göstermeyen hafif, açık kaynaklı ilişkisel veritabanını adlandırmak için 1998'de kullanılmış fakat Eric Evans açık kaynaklı dağıtılmış veritabanı yaklaşımı için bu tanımı tekrar düzenlemiştir (Taylor, 2007).

Büyük Veri Analitiği, İş Zekâsı ve petabayt veri kümeleri üzerinden sosyal ağ gibi uygulamaların hesaplama ve depolama gereksinimleri, sql benzeri merkezi veritabanlarının sınırlarını zorlamıştır (Konstantinou. Vd., 2011). Bu nedenle dağıtık, yatay ölçeklenebilir ve ilişkisel olmayan veri tabanlarına bu alanlarda sıklıkla kullanılmaktadır (Moniruzzaman, 2013).

İlişkisel veri tabanlarından örnek verecek olursak: MsSql, Postgresql, MySql, MariaDb, OracleDb vb…

İlişkisel olmayan veri tabanlarından örnek verecek olursak: Mongodb, Cassandra, ElasticSearch, HBase, Amazon DynamoDB vb.

* 1. **FRAMEWORK’LER**

Yazılım framework, yazılım uygulamaları geliştirmek için birçok farklı kütüphanenin biraraya getirilerek oluşturulduğu yeniden kullanılabilir yazılım kodlarıdır (Software frameworks buffalo, 2021). Bu sayede yazılım geliştirme sürecini hızlandırır. Günümüzde aktif kullanılan birçok yazılım diline ait framework bulunmaktadır. Bunlardan bazıları:

* Python programlama dili ile yazılmış; Django, Flask,
* Java ile yazılmış; Spring, Hibernate, Apache Hadoop,
* Javascript ile yazılmış; Angular, Node. Js, Vue. Js, React,
* C# ile yazılmış; DevArchitecture, .Net Core, .Net 5’i örnek gösterebiliriz.
  1. **SANALLAŞTIRMA UYGULAMALARI VE BULUT BİLİŞİM**

Sanallaştırma, uygulamalar ile CPU, depolama, bellek gibi kaynaklarının soyutlanması olarak kullanılmaktadır. Sanallaştırma ile uygulamalar arasında donanım kaynaklarının paylaşması sağlanabilmektedir (Xing vd., 2012).

Bulut bilişim uygulamaları dağıtmak için internetle erişilebilen uzak sunucuların kullanıldığı teknolojidir. Bu teknoloji, depolamayı, belleği, işlemeyi ve bant genişliğini merkezileştirerek, donanım teknolojilerine erişimi ve kullanımı daha verimli kılar (Xing vd., 2012).

Sanallaştırma teknolojilerinin gelişmesiyle birlikte Amazon Web Service in 2002 yılında EC2 ürünü ise 2004 yılında piyasaya sürülerek bu teknolojilerin öncüsü haline gelmiştir (Narendula, 2012).

**BÖLÜM 2 – SOSYAL MEDYA**

Sosyal medya internet üzerinden yapılan iletişim biçimidir. Bloglar, podcast’ler, fotoğraf paylaşım siteleri, video paylaşım siteleri, sınırlı metin paylaşım siteleri ve sanal dünyalar olarak çeşitlenebilen sosyal medya türleri bulunmaktadır (Introduction to Social Media, 2021). Tablo 1’de göründüğü üzere günümüzde sosyal medya platformlarına Facebook, youtube ve whatsapp’ı örnek gösterebiliriz. 2019 yılında Facebook ve youtube kullanımı 2 milyar’a ulaşmıştır. Kepios’un çalışmasından alındım tablo 2’deki verilere göre 2021 yılı itibariyle 4.55 milyar sosyal medya kullanıcısı bulunmakta, ortalama günde 2.5 saat sosyal medyada vakit geçirilmektedir. Tüm bu rakamlara göre her güm milyarlarca veriler üretilmekte ve içerikler tüketilmektedir. Bu nedenle büyük veri ve yapay zekâ için zengin bir veri kaynağı oluşturmaktadırlar.

**BÖLÜM 3 – VERİ BİLİMİ VE BÜYÜK VERİ**

Veri bilimi büyük miktarlara çıkan verilerin toplanması, temizlenmesi, analiz edilip görselleştirilmesine kadarki süreci kapsayan bir bilim alanıdır. Amacımız veriden anlamlı çıktılar elde edebilmektir.

Büyük veri ise sosyal medya şirketlerinin çok büyük miktarlarda veriyi analiz edebilme sorunları üzerine çıkmış bir kavramdır. Pete baytlarca verinin analiz edilmesi klasik veri bilimi yaklaşımlarıyla mümkün olamamaktadır. Bunların ışığında büyük verinin 5V si doğmuştur.

Büyüklük (volume), hız (velocity), çeşitlilik (variety), değer (value) ve doğruluk (veracity) olarak sınıflandırılan veriler, büyük veri sınıfına girip girmediği anlaşılabilmektedir. Mesela çok hızlı değişen bir veri yapınız varsa bu büyük veri tanımına girebilmektedir. Fakat çeşitliği yok ya da veri yapısı sabit ise büyük veriye girmemektedir. Büyük veri esasında analiz için verileri dağıtık bir şekilde sunucularda paylaştırarak analiz edilmekte ve bu sayede zamandan tasarruf edilebilmektedir. Bunun için farklı teknolojilerin doğması kaçınılmaz olmuştur. Bu teknolojilerden en bilineni ve ilki Hadoop’tur. Disklerde saklanan verilerin dağıtık sunucularda analiz edilmesine yaramaktadır. batch yani gerçek zamanlı olmayan analizler için uygundur. Hadoop’tan sonra piyasaya çıkan spark RAM gücünü kullanarak gerçek zamanlı analizi mümkün kılmıştır. Disk yerine ram kullanıldığı için daha maliyetlidir. Yazılım dili olarak veri biliminde birçok dil kullanılmaktadır. Bunlardan en meşruları python ve java ve scala dilidir. Scala javaya yakın fakat fonksiyonellik desteği ile öne çıkmaktadır. Python da hem nesnellik hem de fonksiyonellik barındırmakta, veri bilimi camiasında en çok tercih edilen dillerden biridir. Biz de projemizde python kullanacağız. Son olarak projemizde kullanacağımız Duygu analizinin ne olduğunu bir tanımlayalım;

Duygu analizi doğal dil işleme teknikleri ve diğer teknikler ile metinler üzerinden özel kelimelerin belirlenmesi, pozitif, negatif ve nötr şeklinde kategoriler çıkarılıp analizin yapılaması işlemi şeklinde tanımlayabiliriz (MANYIKA vd., 2011).

**BÖLÜM 4 – SAKARYA ÜNİVERSİTESİ MEMNUNİYET ANALİZİ**

* 1. **KULLANILAN TEKNOLOJİLERİN TANITIMI**

4.1.1 .net 5.0 ve Devarchitecture

.net 5.0 Microsoft tarafından geliştirilip desteklenen c# dili ile yazılan bir framework’tür. .net core 3.0 ve .net framework 4.x teknolojilerinin birleşimi olarak düşünülebilir. .net framework Windows ortamlarına odaklanırken, .net core teknolojisi Linux gibi platformlara yazılım üretilebilen, açık kaynak bir frameworktür. Microsoft artık iki ayrı proje yürütme zahmetinden kurtulmak ve açık kaynak camiasına destek olabilmek adına 5.0 sürümüne geçiş yapmıştır.

Devarchitecture ise .net 5.0’ın bir parçası olan web api alt yapısını kullanan bir açık kaynak teknolojidir. Hedefi solid prensiplerine odaklanarak, onion architecture ve aspect orianted altapısını entegre edebilmek adına geliştirilmiştir ve amacı kod geliştirmeyi hızlandırmaktır. Projede Web api olarak kullanılmıştır. Kullanılan kavramları tek tek açıklayalım. SOLID prensiplerindeki her bir harfi açıklayacak olursak;

S harfi, Tek sorumluluk ilkesini temsil etmektedir. Bir sınıfı veya fonksiyonu değiştirmek için birden çok neden varsa bu yapı birden çok sorumluluğa sahip demektir (Singh & Hassan, 2015). Solid’in “S” si gereği tek bir sorumluluğa sahip olması gerekir. Bu nedenle iş parçaları bölünmelidir.

O harfi, açık kapalı ilkesini temsil eder. Bir sınıf değişime dirençli fakat genişletilmeye müsait olması gerekir. Eğer bir değişiklik yapılacak ise bunu gerçekleştirmek için halihazırda bir sınıfı etkilemeden yenilerinin eklenebiliyor olması gerekir (Singh & Hassan, 2015). Bu prensibe tak çıkar prensibi de denilmektedir.

L harfi, Liskov ikame prensibi olarak geçmektedir. Türetilen tüm sınıflar, ara yüzden gelen tüm özellikleri desteklemelidir (Haoyu & Haili, 2012). Bu prensibi açıklamak için ördek örneği çokça kullanılmaktadır. Fakat gerçek hayattan örnek vermek daha doğru olacaktır. Bir şirket düşünelim. İlişkide olduğu çalışanları, müşterileri ve paydaşları bulunmakta. Bunların ayrı ayrı sınıfılar şeklinde tanımladığımızı düşünelim. Bunlar da “IPerson” adında bir ara yüz sınıfından türetilmiş olsun. Bu sınıflardan “Employee” sınıfının “calculateSalary()” şeklinde bir fonksiyona sahip olsun. Eğer bu fonksiyon “Person” içinde tanımlanırsa, diğer sınıflar da “calculateSalary()” adında bir fonksiyona sahip olmuş olacak. Fakat paydaşlar maaş almadığı için ve ara yüz tarafından zorunlu eklendiği için kafa karışıklığına ve projede karmaşaya sebep olacaktır. Liskov ikame prensibi buna karşı çıkmaktadır.

I harfi ara yüz ayırımı prensibi olarak adlandırılmaktadır. Ara yüzlerin kategorilere ayırılması prensibine dayanır (Jaiswal, 2019). Liskov prensibine çok benzemektedir. Fakat Liskov’dan farklı olarak ihtiyaç olabilecek tüm fonksiyonları temel bir ara yüz içinde toplamaktansa bunları bölmemiz gerektiğini söyler. Liskovda örnek verdiğimiz çalışan, müşteri ve paydaşların tüm fonksiyonlarının ortak bir ara yüzde toplanmasına karşı çıkmaktadır diyebiliriz.

D harfi bağımlılık ters çevirme ilkesi olarak geçmektedir. Bu prensip somut işlevler ve sınıflar yerine arabirimler ile soyutlanmasına dayanır (Martin, 2000). Bu prensip ile somut sınıflar arası bağımlılık azaltılmış olur. Bu sayede yeni bir işlev ekleneceği zaman var olan kodların değiştirilmesine gerek kalmaz.

Onion yani soğan mimarisi, solid prensiplerini mimari düzeyde uygulayabilmek için tasarlanmıştır. Tablo 4’te de görüldüğü üzere, core, domain, model, domain service, application service katmanları görünmektedir. Test, user interface ve infrasctracture katmanlarının etrafında sarılmış şekilde görünmektedir. Bunları sırası ile açıklayacak olursak, core katmanı, tüm projelerde ortak kullanılabilecek kodları içerir. Domain modeli veri tabanı tablolarının karşılıklarıdır. Domain servisi veri erişim katmanı olarak geçer ve burada yalnızca veri tabanına erişimi ilgilendiren sınıfları barındırır. Application servisi, iş katmanı olarak da adlandırılır. Bu katmanda iş kodları yer almaktadır. Bu proje test, veri tabanları ve kullanıcı ara yüzü ile iletişim halindedir.

Aspect oriented programlama ise bağımlılıkları ve proje karmaşıklığını azaltmak için biçilmiş kaftandır. Fonksiyonlar başlamadan önce hata atma durumunda ve fonksiyon bitince tetiklenebilen yapılar inşa edilir. Bu yapılar sayesinde hata loglama, performans takibi, veri işlemlerini izleme, ön belleğe alma gibi birçok fonksiyonu çok kolay bir şekilde gerçekleştirmemizi sağlar.

4.1.2 Angular

Angular Google tarafından geliştirilen açık kaynaklı bir javascript – typescript frameworküdür. Typescript ise Microsoft tarafından geliştirilen, javascript tabanlı bir üst seviye dildir. Javascript’e nesnel özellikler kazandırılarak meydana getirilmiştir. Derleyici vasıtasıyla javascripte dönüştürülerek kullanılır. Angularda’ki amaç saf javascript projelerine nazaran oluşacak karmaşayı önleyerek, daha sürdürülebilir projeler oluşturmaktır. Projede grafik ara yüzü için kullanılmıştır.

4.1.3 Python

1990 yılında hayatımıza giren python programlama dili, yorumlamalı, nesne yönelimli ve fonksiyonel programlama özelliklerini bir arada destekleyebilen üst seviye bir dildir. C dili üzerine yazılarak oluşturulmuştur ve derlemeye ihtiyaç yoktur. Kullanımı oldukça kolay olan bu dil veri bilimi projelerinde yoğun bir şekilde kullanılmakla birlikte büyük bir geliştirici ağına sahiptir. Projede veri biliminde ve kafka producer’ı yazmak için kullanılmıştır.

4.1.4 ELK-Stack

ELK Stack: Elasticsearch, Logstash, Kibana uygulamalarının bir arada kullanılmasına verilen bir isimdir. Bu yapı log takibi için sıklıkla kullanılmaktadır. Elastic NV firmasının açık kaynak ürünüdür. Projede log takibi ve en çok kullanılan kelimelerin görselleştirilmesi için kullanılmıştır.

4.1.5 Mongodb

Mongodb belge tabanlı, json dilini kullanarak veri depolamayı sağlayan açık kaynak bir veri tabanıdır. İlişkisel modeli desteklemektedir. Fakat ilişkisel olmayan özellikleri ile çok güçlü ve tutarlı bir veri tabanıdır. Yatay ölçeklenebilir. Büyük veri alanında oldukça popülerdir. Bu nedenle projede veri tabanı olarak kullanılmıştır.

4.1.6 Apache Kafka

İlk adımlarını linkedin firmasının attığı ve sonradan Apache vakfına devrettiği bir mesaj kuyruk projesidir. Veri tutarlılığı ve yüksek veri işleme kapasitesine sahiptir. Stream veri akışında (gerçek zamanlı) oldukça yoğun olarak kullanılmaktadır. Projede de bu amaç için kullanılmıştır.

4.1.7 Apache Spark

Hadoop’un eksiğini tamamlamak amacıyla piyasaya çıkan spark, gerçek zamanlı makine öğrenmesinden, filtrelemeye birçok işi hızlı bir şekilde yapabilmektedir. Hadoop’un aksine ram gücünden faydalanan spark neredeyse 100 kat daha hızlı iş yapabilmektedir. Projede gerçek zamanlı veri analizi için kullanılmıştır.

4.1.8 Kubernetes

Kubernetes, soyutlama teknolojilerinin popülerleşmesinin ardından ortaya çıkan docker gibi konteyner teknolojisinin yönetim zorluğunu ortadan kaldırmak için geliştirilmiş Cloud Native Computing vakfının desteklediği ve bu vakıftan mezun olan bir projedir. Uygulamaların yönetimini sağlar. Veri bilimi için hayati öneme sahiptir diyebiliriz. Projede uygulama yönetimi için kullanılmıştır.

* 1. **PROJE YOL HARİTASI**
     1. **Proje Planı**

Tablo 3’te görüldüğü üzere Grafik ara yüzü için Angular teknolojisinden yararlanılmıştır. Angular nginx aracılığı ile .net backendine bağlanarak veri tabanından veriler talep edebilecek. Bu sayede verileri alıp görselleştirecek ve veriler bu sayede anlam kazanacaktır. Elk stack yapısı kodlama hatalarını yakalayabilmek ve kelime tekrarlarını görebilmek için kullanılmış ve hem .net apisinden, hem de spark’tan veri alabilmektedir. Spark kullanılarak daha önceden oluşturulmuş makine öğrenmesi modeli streamin for prediction uygulamasında gerçek zamanlı analiz edilerek veriler güncellenmektedir. Apache kafka aracılığı ile analiz edilecek yeni veriler spark tarafına gerçek zamanlı olarak gönderilmektedir. Kafka ile Twitter api arasında ise pyton ile yazılmış kafka producer’ı vardır. Tahminlenmiş veriler kafkaya gönderilerek, consumer’lar aracılığı ile mongodb’ye kaydedilmektedir.

* + 1. **Back-end altpası**

Back-end, arkaplan ya da server tarafı devarchitecture açık kaynak proje altyapısı kullanılarak tasarlanmıştır. Visual studio 2019 ide’si üzerinde geliştirilmiştir. Yardımcı eklenti olarak rest-sharp kullanılmıştır. Projede genel klasör yapısı; Core, Entities, DataAccess, Business, WebApi ve Test şeklindedir.

Core içerisinde Autofac kütüphanesi kullanılarak oluşturulan aspect yapısı bulunmaktadır. Bu yapı içerisinde cache, expection, logging, performance, transaction, ve validation yapılarını görmekteyiz. Bu yapılar bize aspect mimarisinin getirisi olarak, önbelleklemeyi, hata yakalamayı, loglamayı, performans takibini, işlemleri ger alabilmeyi ve doğrulamayı merkezileştirmeyi ve defalarca kullanılabilmeyi sağlamaktadır. Bir diğer önemli yapı ise CrossCuttingConcerns olarak geçmektedir. Bu yapı katmanlar arası benzer işleri yapmamız gereken yapıları barındırır. Data Access yapısı temel veri erişim kodlarını, DependencyResolvers, bağımlılık çözümleyici olarak geçen, bağımlılıkları bir merkezde toplamamızı ve yönetmemizi sağlayan yapıyı barındırır. Entities temel veri tabanı tabloları sınıf karşılıklarını kapsamaktadır. Bunlar, Concrete içerisinde tanımlanmıştır ve sırasıyla “OperationClaim”, “User” ve “Log” dur. OperationClaim yetkilerin, User üyelerin, Log üretilen verinin temsilinin tanımlandığı sınıflardır. Extentisons var olan sınıflara eklediğimiz ekstra özellikleri, Utility ise Temel kullanılan birtakım özellikleri barındırır. Bunlardan bazıları mesaj kuyruk sistemi, güvenlik, mail altyapısıdır.

Entities katmanı içerisinde veri tabanı tablo karşılıklarını barındırır. Bu projede kullanılanlar Concrete klasörü çerisinde tanımlanmıştır. Bunlar “ApiInfoModel” ve “ClientModel” dir. ApiInfoModel makine öğrenmesinin çıktılarını da içinde barındırır. ClientModel analiz edilen verinin tablo karşılığıdır.

DataAccess katmanında veri tabanı tabloları üzerinde işlemlerin yapılabilmesi için tanımlanan sınıflar ve fonksiyonlar bulunur. Veri ekleme, güncelleme, silme, filtreleme gibi birçok işlev burada tanımlanmıştır. Bu tanımlar için Entity Framework ten yararlanılmıştır.

Business katmanında iş kodlarının bulunduğu sınıflar yer almaktadır. Mediator paterni uygulanmıştır. Bu sayede verinin değiştirildiği ve sorgulandığı sınıflar birbirinden soyutlanmıştır. Bunlar Command ve Query şeklinde iki ayrı yapı içinde bulunmaktadır. Handler içerisinde muhafaza edilmektedir. Aspect yapısı içerisindeki tüm işlevler handler içerisinde yer alan fonksiyonalar üzerinde tanımlanır. Çünkü tüm bu yapılar iş kodudur!

Web api katmanı görevi gereği yalnızca veri alışverişi sağlar. Business katmanı fonksiyonları burada controller içerisinde yer alan sınıflardan çağrılır. Controller içerisinde hiçbir iş kodu yer almaz.

Test içerisinde business katmanında yer alan handler fonksiyonları test edilmiştir. Tüm testlerin geçmesi kodların doğru çalıştığını göstermez. Bu nedenle bir api test aracı olan POSTMAN üzerinde test edilmiş ve testleri geçtiği doğrulanmıştır. Ardından ara yüz tasarımına geçilmiştir.

* + 1. **Front-end altyapısı**

Ön yüz projesi Angular ile tamamlanmıştır. Klasör yapısına bakacak olursak components, directives, guards, interceptors, models, services ve temel app comonenti ve modül dosyalarından oluşmaktadır. Directives klasörünün içinde yer alan must-match.ts dosyasında yer alan fonksiyon ile parolaların uyuşmasının kontrol edildiği fonksiyon yer almaktadır. Şifreler uyuşmaz ise hata dönecektir. Guards içinde yer alan login-guard dosyası izinleri kontrol ederek, izin verilen sayfalara girişi sağlar. Bu izinler yetki bazlı olabileceği gibi, yalnızca giriş yapılmasının yeterli olması durumları da kullanılabilir. Models klasöründe veri tabanı tabloları karşılığı yer almaktadır. Apiden gelen veriler bu sınıflarla karşılanır. Services klasöründe yardımcı fonsiyonlar bulunur. Bu fonksiyonlar arayüzün manipüle edilmesi dışında kalan tüm fonksiyonları içerir. Components kalsörü içerisinde yapılar arayüzü temsil eder ve arayüzü ilgilendiren fonsiyonlar bulunur. Bunlar, html css, ve ts dosyalarıdır. Her bir component kendine özgü bir alanı temsil eder. Bizim projede, login, register, dashboard, footer, layouts ve navbar şeklinde bölümlendirilmişidir. Dashboard içeriği örnek görseli tablo 5’te yer almaktadır.

* + 1. **Kubernetes ortamının oluşturulması**

Kubernetes ortamının kurulmasının birçok yolu vardır. Kendi bilgisayar ortamınıza test için, minikube, k3s, kind gibi kubernetes’in daha hafif sürümleri kurulabilir ya da k8s i direk kurabilirsiniz. Kendi şirketinizde fiziksel sunucularınız var ise Kubernetes’i direk buralara kurarak kullanılmaya başlanabilir. Fakat böyle bir imkânınız yok ise; Bulut servis sağlayıcılarından, digitalocean, aws, azure, Google, ibm gibi firmaların kubernetes hizmetleri kullanılabilir. Biz projemizde “Digitalocean” firmasının sunduğu k8s hizmetini kullandık.

* + 1. **Bağımlı teknolojilerin kurulması**

**Mongodb**: Mongodb uygulamasını kubernetes’e yüklemek için bitnami firmasının artifact hub üzerindeki reposu kullanıldı. Uygulama adımları şu şekildedir;

$ helm repo add bitnami <https://charts.bitnami.com/bitnami>

$ helm install <my-release> \

--set auth.rootPassword=secretpassword,auth.username=my-user,auth.password=my-password,auth.database=my-database \

bitnami/mongodb

“repo add” komutu helm kütüphanesinin bitnami paketlerinin yolunu tanıması sağlanır. Bu yolu kullanarak bitnamiye ait herhangi bir uygulamayı kubernetes’e sorunsuz ekleyebiliriz.

<my-release> uygulamanın adını temsil eder. –set ile tanımlanan parametreler ise dokümantasyonunda gösterilen ve projenin içinde yer alan varsayılan parametreleri değiştirmemizi sağlar. Biz projede yalnızca şifreleri değiştirdik.

**Apache kafka**: Apache kafka kurulumu ve ayarlaması zor bir uygulama olmasına karşın “strimzi” imdadımıza yetişmektedir. Strimzi açık kaynak kodlu kubernteste özel yazılan bir kafka operatörüdür. Kafkanın yönetimini ve kurulumunu kolaylaştırmak için tasarlanmıştır. Kurulum adımları şu şekildedir.

$ kubectl create namespace kafka

$ kubectl create -f 'https://strimzi.io/install/latest?namespace=kafka' -n kafka

$ kubectl apply -f https://strimzi.io/examples/latest/kafka/kafka-persistent-single.yaml -n kafka

Kubernetes de tanımlanabilen namespaceler yazılımdakine benzer bir mantıkta soyutlama işlemleri için kullanılmaktadır. Bu sayede yetkisiz kişiler bu uygulamaya erişemez.

“strimzi.io/install/latest” bu alanda strimzinin kafka yönetimini sağlayan operatörü bulunmaktadır. Bunun kurulumunu bu şekilde yapabiliyoruz.

kafka-persistent-single.yaml ise 1 replica’ya sahip kafka ve Zookeeper uygulaması ayağa kaldırmaktadır.

**Elk stack**: elk stack için docker hub üzerinde bulunan image’lerden faydalanılarak manuel olarak kurulum gerçekleştirilmiştir.

**Apache Spark**: Apache spark uygulaması için Google cloud platform’unun açık kaynak olarak yayınladığı “spark-on-k8s-operator” kullanılmıştır. Bunun aws üzerindeki muadili “EMR” dır. Ekleme adımları aşağıdaki gibidir.

$ helm repo add spark-operator <https://googlecloudplatform.github.io/spark-on-k8s-operator>

$ helm install my-release spark-operator/spark-operator --namespace spark-operator --create-namespace

**Nginx kurulumu ve Ingres controller:** Ingress controller http seviyesinde api gateway görevi gören kubernestes objesidir. Nginx ise ıngress görevinin yanında cache, auth gibi görevleri de yapabilen açık kaynak yazılımdır. Controller’ın yüklenmesi aşağıdaki komut ile mümkündür:

$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.1.0/deploy/static/provider/do/deploy.yaml

* + 1. **Veri bilimi modelinin oluşturulması**

Model eğitilmesi için belirli bir miktarda eğitim verisinin toplanması gerekiyordu. Bunun için python dilini kullanarak twitter batch kütüphanelerinden “snscrape” kütüphanesi kullanıldı. Bu kütüphane aracılığı ile 2,129,034 veri toplandı. Bu veriler toplanırken, memnuniyet belirten emojiler ile filtrelendi ve bu veriler pozitif olarak işaretlendi. Memnuniyetsizlik belirten emojiler ile filtrelenerek, veriler negatif olarak etiketlendi ve mongodb veri tabanına kaydedildi. Mongodb ye kaydedildikten sonra sıra veri temizliği devreye girdi. spark aracılığı ile tweetlerin tamamı içeriğinden, hastagler, linkler, anlam ifade etmeyen kelimeler, etiket ifadeleri, noktalama işaretleri, emojiler ve rakamlar temizlendi. Kalan kelimelerin tamamı küçük harfe çevrildi. Boşluklar silindi. Daha sonra bu kelimler array’a çevirildi. En son HashingTF ve tf–idf yöntemi kullanılarak, kelimeler sayısal temsile dönüştürüldü. Bu işlemlerden sonra her bir text birer sayı dizisine dönüşmüş oldu. Bu kolon feature kolonu olarak değerlendirildi. Emojiler kullanılarak oluşturulan **ikili sınıflandırma** bilgileri Label kolonu olarak eklendi. 0 bilgisi memnuniyetsizliği, 1 bilgisi de memnuniyeti temsil etmektedir. Bu verilerin tamamını rastsal bir şekilde, %90 eğitim, %10 test, %10 doğrulama verisi olarak 3’e bölündü. Logistic regression makine öğrenmesi yöntemi ile eğitim verisi kullanılarak, model eğitildi. Eğitilen modele doğrulama ve test verilerinin tahmini yaptırıldı. Makine öğrenmesi modelinin doğrulanması ve testi sonucu oluşan çıktılar aşağıdaki gibidir.

* MlResultAccuracyRate: 75.84606999722793
* MlResultF1Rate: 75.8491134267859
* MlResultROCRate: 75.83797787585628
* true\_positive: 166969
* true\_negative: 155888
* false\_negative: 50379
* false\_positive: 52438

Eğitilmiş model dışarı çıkarılarak locale kaydedildi ve gerçek zamanlı analizi için kubernetes ortama eklenip kullanıldı.

* + 1. **Stream altyapısının kurulması ve entegrasyonu**

Bu işlemlere başlamadan önce twitter geliştirici hesabı açıp bu platform üzerinden yapılacak işletimin anlatılması ve hedefinize uygun bir kategori tanımlatmanız gerekmektedir. Bu onay sonrasında bize belli bir miktar istek hakkı tanımlanacak ve belli bir güvenlik şifreleri atanacak. Bu sayede twitter’dan gerçek zamanlı veriler almak mümkün olacaktır. Python aracılığı ile Tweepy kütüphanesi kullanılarak twitter api üzerinden gerçek zamanlı bir şekilde getirilen, Sakarya üniversitesi ile ilgili tweetler, anonimleştirilerek kafka’ya gönderildi. Spark connector kafkaya bağlanarak spark structured stream yapısı ile öğrenmiş model kulanılarak gerçek zamanlı duygu analizi yapılan veriler tekrar kafkaya gönderildi. Kafkayı dinleyen başka bir uygulama ise mongodb’ye ve elk stack yapısına kaydetti.

* + 1. **Uygulamaların kurulması ve testi**

Uygulamaların kurulumu için kullanılabilecek çeşitli yöntemler vardır. Bu yöntemler arasında CI/CD prduction ortalar için uygundur. Bu yöntem sürekli entegrasyon, sürekli teslim olarak çevirilebilir. Bu yöntem test ve canlı ortamlarına eklemelerin otomatikleştirilmesi için kullanılır. Bu projede manuel yükleme yapılmıştır. Yükleme adımları şı şekilde gerçekleşmiştir.

1. Öncelikle tüm uygulamalrın “Dockerfile” dosyaları hazırlanmıştır.
2. Docker uygulaması indirilerek bu uygulamalar Docker üzerine Docker build komutuyla eklenmiştir. Bu uygulamalara “docker tag” komutu ile erişilebilmesi için isimlendirme yapılmıştır. Daha sonra uygulamaların image olarak barınacağı registry hesabına cli ile giriş yapılmış ve buraya “docker push” komutu ile gönderilmiştir.
3. Sanal uygulamalarınızın barınacağı yer olarak digital ocean firmasının Container Registry ürünü kullanılmıştır. 500 mb’ye kadar ücretsizdir.
4. Son olarak bu image’lerin yolları kullanılarak “Deployment” objesi oluşturulup kubernetes ortamında uygulamalar ayağa kaldırılmıştır.

**Sonuç ve Öneriler**

Modelimiz her 4 veriden 3’ünü doğru tahmin etmektedir. Modelde tip 1 ve tip 2 hatasının birbirine yakın olması, modelin herhangi bir tarafa kaymadığını, dengeli olduğunu çıkarabiliriz.

Sakarya üniversitesi memnuniyet analizi 228.393 verinin batch analizi sonucu çıkan memnuniyet oranı %44 tür.

Canlı sisteme <https://ngapp-j2bif.ondigitalocean.app> adresinden;

Kullanıcı adı: [test@email.com](mailto:test@email.com)

Şifre: Test1234.

ile ulaşabilirsiniz.

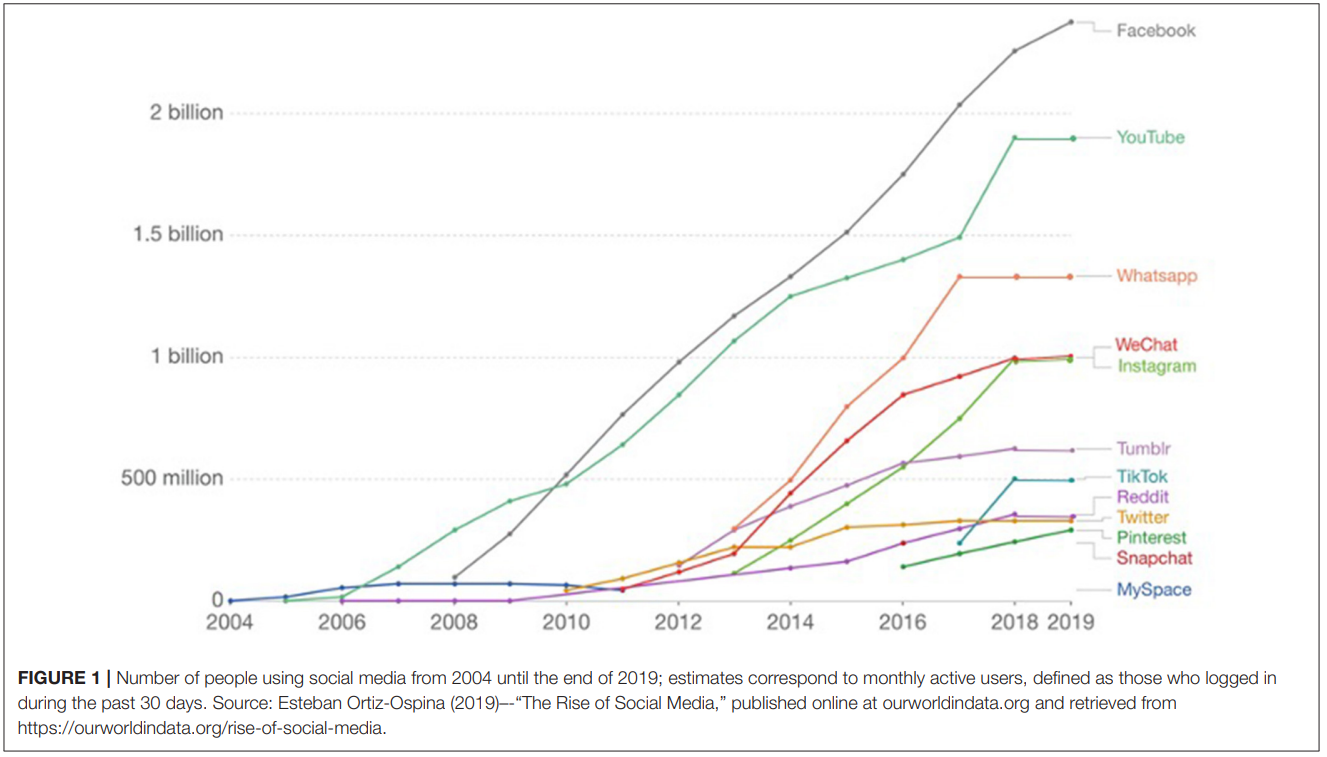
Donanımsal kısıtlılıklardan dolayı Cross-validation (çapraz doğrulama) ve hiper parametre optimizasyonu yapılamadı. Bunun yapılması modelin tahmin doğruluğunu arttıracaktır. Derin öğrenmenin veya topluluk öğrenmesi algoritmalarının kullanılması ve model optimizasyonu modelin doğruluğunu arttırabilir. Modelde ikili sınıflandırma yerine 3. bir sınıflandırma olan nötr sınıflandırmanın devreye girmesi doğruluk oranını arttırabilir. Eğitim verisi hazırlanırken uygulanan filtreleme yöntemleri daha ayrıntılı, hassas ve mecaz anlamı yakalayabilecek filtreler ile desteklenirse modelin güvenirliği artacaktır. Daha büyük bir eğitim verisinin kullanılması modelin tutarlılığını arttırır. Kubernetes sistemi gerçek kullanıcı ortamı için kullanılacaksa mutlaka sertifikalar (TLS) eklenmeli ve haberleşmeler şifrelenmelidir. Veri tabanı bağlantıları uzun ve karmaşık şifreler ile sağlanmalı veya sertifikalar kullanılmalıdır.

**Kaynakça**

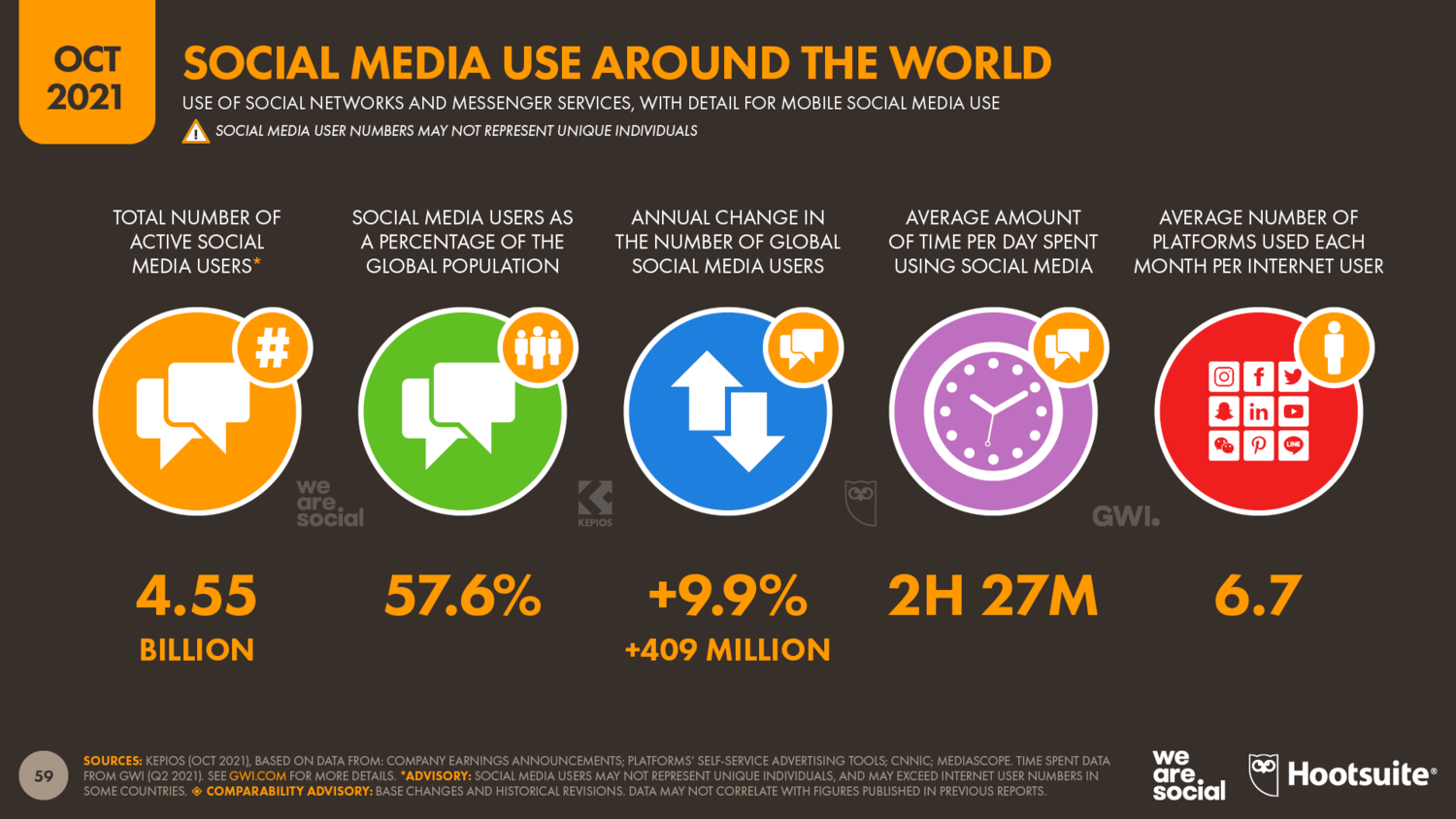
1. Kerlinger, F.N., 1986, Foundations of behavioral research (CBS Publishing Japan Ltd, New York).
2. Bexheti, L. A., Ismaili, B. E., & Cico, B. H. (2014, March). An analysis of social media usage in teaching and learning: The case of SEEU. In Proceedings of the 2014 International Conference on Circuits, Systems, Signal Processing, Communications and Computers (pp. 90-94).
3. Franco-Bedoya, O., Ameller, D., Costal, D., & Franch, X. (2017). Open source software ecosystems: A Systematic mapping. Information and software technology, 91, 160-185.
4. Bretthauer, D. (2001). Open source software: A history.
5. Wordpress About 2021, https://wordpress.org/about/ 20.10.2021.
6. Apache About 2021, https://www.apache.org/foundation/ 20.10.2021.
7. CNCF About 2021, https://www.cncf.io/about/who-we-are/ 20.10.2021.
8. Date, C. J. (2003). An Introduction to Database Systems, Fifth Edition. Boston, MA: Addison Wesley.
9. Kroenke, D. & Auer, D. (2007). Database Concepts.3rd ed. New York, NY: Prentice.
10. Taylor, A. (2007): SQL for Dummies. New York, NY: John Wiley & Sons, Inc.
11. Konstantinou, I., Angelou, E., Boumpouka, C., Tsoumakos, D., & Koziris, N. (2011, October). On the elasticity of nosql databases over cloud management platforms. In Proceedings of the 20th ACM international conference on Information and knowledge management (pp. 2385-2388). ACM.
12. Moniruzzaman, A. B. M., & Hossain, S. A. (2013). Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. arXiv preprint arXiv:1307.0191.
13. Software Frameworks buffalo, [https://wiki.cse.buffalo.edu/services/content/software-frameworks / 25.10.2021](https://wiki.cse.buffalo.edu/services/content/software-frameworks%20/%2025.10.2021).
14. Xing, Y., & Zhan, Y. (2012). Virtualization and Cloud Computing. Future Wireless Networks and Information Systems, 305–312.
15. Narendula, R. (2012). Amazon web services-a case study (No. REP\_WORK).
16. Introduction to Social Media (2021) https://www.usf.edu/ucm/marketing/intro-social-media.aspx 01.11.2021
17. MANYIKA James – Michael CHUI – Brad BROWN – Jacques BUGHIN – Richard DOBBS – Charles
18. ROXBURGH – Angela Hung BYERS, “Big Data: The next frontier for innovation, compettition, and productivity”, Report McKinsey Global Institute, JUNE 2011.
19. Singh, H., & Hassan, S. I. (2015). Effect of solid design principles on quality of software: An empirical assessment. International Journal of Scientific & Engineering Research, 6(4).
20. Haoyu, W., & Haili, Z. (2012, August). Basic design principles in software engineering. In 2012 Fourth International Conference on Computational and Information Sciences (pp. 1251-1254). IEEE.
21. Jaiswal, M. (2019). Software Architecture and Software Design. International Research Journal of Engineering and Technology (IRJET) e-ISSN, 2395-0056.
22. Martin, R. C. (2000). Design principles and design patterns. Object Mentor, 1(34), 597.

**Ekler**

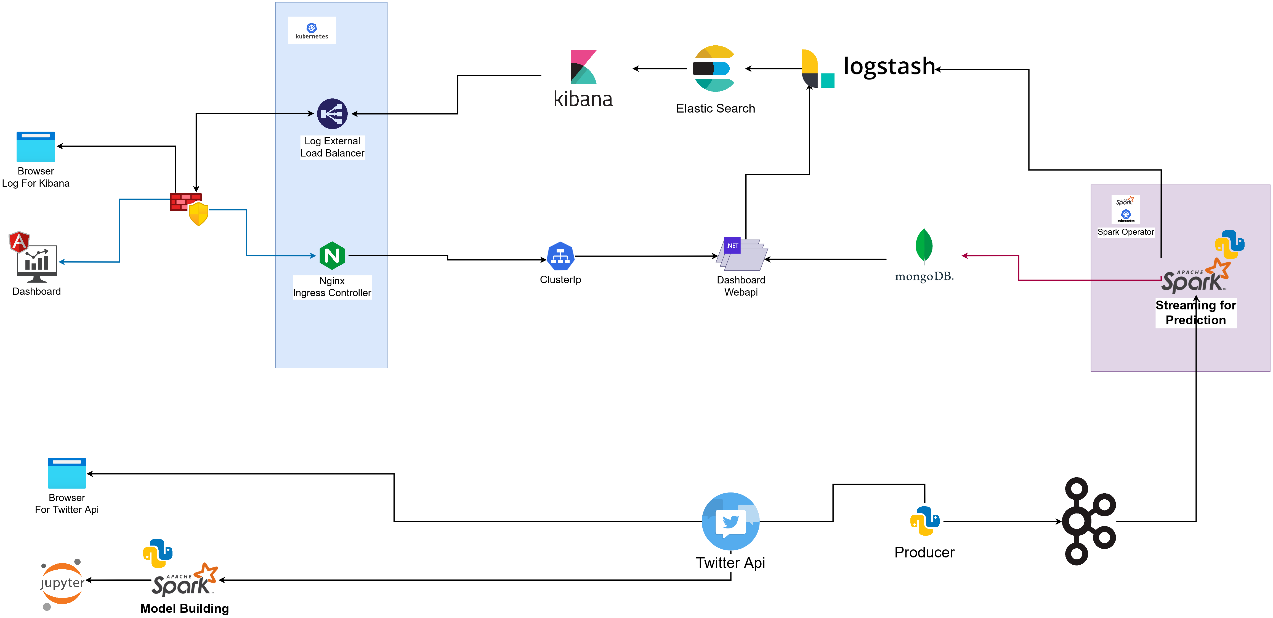
**Tablo 1:**



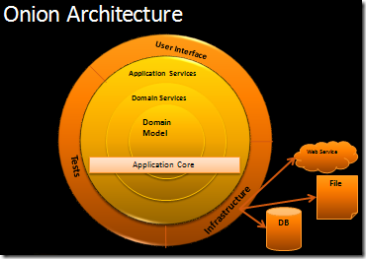
**Tablo 2**



**Tablo 3**

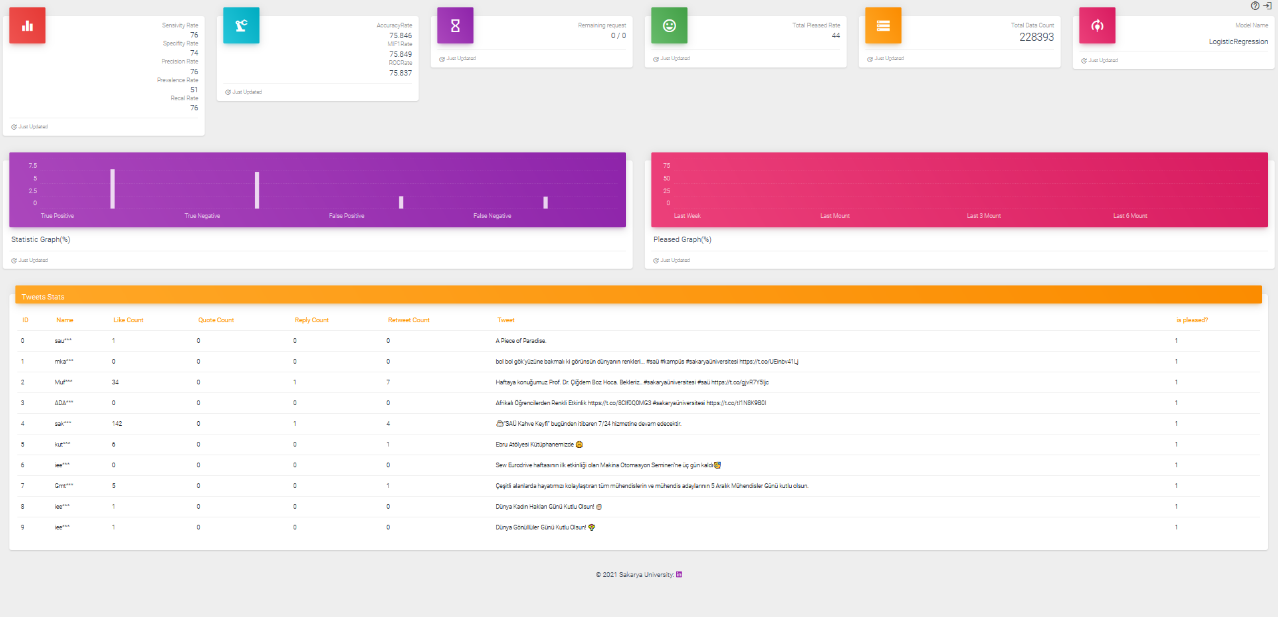
****

**Tablo 4**



J. Palermo, “The onion architecture: part 1. jeffrey palermo (.com). lipiec 29, 2008,” online. [Online]. Available: URL:http://jeffreypalermo.com/blog/the-onion-architecture-part-1/

**Tablo 5**

****

**Özgeçmiş**

Liseyi Osman ötken Anadolu lisesinde okudu. Ön lisansını Anadolu üniversitesi İşletme yönetiminde, lisansını Hacettepe üniversitesi Hemşirelik bölümünde tamamladı. 5 yıldır oyun geliştirme teknolojileriyle uğraştı ve şu an oyun teknolojilerine yapay zekâ çözümleri üreten bir erken aşama startup’ta kurucu ortak olarak görev yapmaktadır.