Logistic Regression Kullanarak Diyabet Tahmini: Uçtan Uca Bir Çözüm

Batuhan Ayyıldız
Manisa Celal Bayar Üniversitesi

***Abstract*—Bu çalışmada, Kaggle'dan elde edilen diyabet veri setini kullanarak diyabet tahmini yapmak üzere bir Logistic Regression modeli geliştirilmiştir. Veri seti, eksik değerlerin temizlenmesi, yeni özelliklerin eklenmesi, çeşitli veri görselleştirme tekniklerinin uygulanması ve modellerin karşılaştırılması gibi bir dizi aşamadan geçirilmiştir. Elde edilen sonuçlara göre, Logistic Regression modeli, diyabet hastalarını en yüksek doğrulukla sınıflandırma yeteneği nedeniyle nihai model olarak seçilmiştir. Bu rapor, veri ön işleme aşamasından model dağıtımına kadar tüm süreçleri kapsamlı bir şekilde ele almaktadır.**

I. Giriş

Diyabet, dünya çapında giderek yaygınlaşan ve sağlık sistemleri üzerinde büyük bir yük oluşturan kronik bir hastalıktır. Bu hastalığın erken teşhisi, bireysel yaşam kalitesini artırmak ve sağlık hizmetleri maliyetlerini azaltmak için kritik öneme sahiptir. Makine öğrenimi ve veri bilimi alanındaki son gelişmeler, hastalıkların erken teşhisi ve yönetimi için devrim niteliğinde çözümler sunmaktadır.

Bu çalışmada, Kaggle'dan elde edilen bir veri seti kullanılarak diyabet tahmini için bir Logistic Regression modeli geliştirilmiştir. Model performansını artırmak amacıyla veri setindeki eksik değerler düzeltilmiş, yeni özellikler eklenmiş ve çeşitli veri görselleştirme teknikleri uygulanmıştır. Ayrıca, model performansını değerlendirmek için farklı makine öğrenmesi algoritmaları ile karşılaştırmalar yapılmıştır.

Logistic Regression modeli, yüksek doğruluk oranı ve yorumlanabilirliği nedeniyle bu çalışma için ana model olarak seçilmiştir. Modelin daha erişilebilir hale gelmesi için FastAPI ve Streamlit uygulamaları geliştirilmiş ve Docker konteynerleri içinde paketlenmiştir. Docker kullanımı, uygulamaların taşınabilirliğini ve uyumluluğunu artırmış, AWS üzerindeki dağıtım ise uygulamalara küresel erişim sağlamıştır. Bu rapor, veri ön işleme aşamasından model dağıtımına kadar tüm süreçleri ayrıntılı bir şekilde ele alarak etkili bir diyabet tahmini çözümü sunduğumuzu göstermektedir.

II. Veri Setinin Anlaşılması ve Yorumu

Veri analizi ve modelleme sürecinde kullanılan diyabet veri seti Kaggle platformundan elde edilmiştir. Bu veri seti, 768 gözlem ve 8 bağımsız değişken içermektedir. Özellikler ve veri setinin içeriği aşağıda detaylandırılmıştır:

- **Pregnancies (Gebelikler)**: Bir kadının geçirdiği gebelik sayısını temsil eder. Gebelik sayısının diyabet üzerindeki etkisini anlamak için kullanılır.
- **Glucose (Glukoz)**: Kan şekeri seviyelerini ölçer. Diyabet teşhisinde önemli bir biyomarker olarak kabul edilir.
- **BloodPressure (Kan Basıncı)**: Kan basıncını ölçer. Yüksek kan basıncı diyabetle ilişkili bir faktör olabilir.
- **SkinThickness (Cilt Kalınlığı)**: Cilt kalınlığını ölçer. Cilt kalınlığı, insülin direnci ile ilişkili olabilir ve dolayısıyla diyabet riskine bağlıdır.
- **Insulin (İnsülin)**: Kan insülin seviyelerini ölçer. İnsülin seviyeleri diyabet teşhisinde önemli bir rol oynar.
- **BMI (Vücut Kitle İndeksi)**: Kilo ve boy oranını ölçer. Obezite ile diyabet arasındaki ilişkiyi anlamada önemli bir göstergedir.
- **DiabetesPedigreeFunction (Diyabet Pedigree Fonksiyonu)**: Aile geçmişine dayalı diyabet riskini ölçer. Genetik faktörlerin etkisini anlamak için kullanılır.
- **Age (Yaş)**: Hastaların yaşını belirtir. Yaş, diyabet riskinde etkili bir faktör olabilir.
- **Outcome (Sonuç)**: Hedef değişken olarak, kişinin diyabet olup olmadığını belirtir (0: diyabet yok, 1: diyabet mevcut).

A. Özellik Mühendisliği

Veri setindeki mevcut özelliklere ek olarak, model performansını iyileştirmek ve sağlık durumunu daha iyi anlamak için bazı yeni özellikler türetilmiştir. Bu süreç, veri analizi ve modelleme sürecinin önemli bir parçasıdır ve modelin doğruluğunu artırmak için uygulanmıştır.

B. Glukoz Sınıflandırması

Glukoz seviyeleri, diyabet teşhisinde kritik bir rol oynar. Bu bağlamda, glukoz seviyeleri üç ana kategoriye ayrılmıştır:

- **Normal:** 100 mg/dL altındaki değerler bu kategoriye dahil edilir. Bu değerler genellikle sağlıklı bireylerin glukoz seviyelerini yansıtır.
- **Prediabetes (Prediyabet):** 100 mg/dL ile 125 mg/dL arasındaki değerler prediyabet olarak sınıflandırılır. Bu aralık, diyabet riskinin arttığı bir durumu temsil eder.
- **Diabetes (Diyabet):** 125 mg/dL ve üzerindeki değerler diyabet olarak sınıflandırılır. Bu, bireylerin diyabet teşhisi aldığı anlamına gelir.

Bu sınıflandırma, glukoz seviyeleri ile diyabet riski arasındaki ilişkiyi daha iyi anlamak ve modelin performansını artırmak için önemli bir adımdır.

C. Vücut Kitle İndeksi (BMI) Kategorileri

Vücut Kitle İndeksi (BMI), kilo ve boy oranını gösterir ve sağlık durumunun değerlendirilmesinde önemli bir göstergedir. BMI değerleri dört kategoriye ayrılmıştır:

- **Underweight (Zayıf):** BMI değeri 18.5'in altında olan bireyler bu kategoriye dahildir. Bu, yetersiz beslenme veya sağlık sorunlarını gösterebilir.
- **Healthy (Sağlıklı):** BMI değeri 18.5 ile 24.9 arasında olan bireyler sağlıklı kilo aralığında kabul edilir.
- **Overweight (Fazla Kilo):** BMI değeri 25 ile 29.9 arasında olan bireyler fazla kilo olarak sınıflandırılır. Bu, obezite riskinin arttığını gösterebilir.
- **Obese (Obez):** BMI değeri 30 ve üzerinde olan bireyler obez olarak sınıflandırılır. Bu, yüksek diyabet riskini ifade eder.

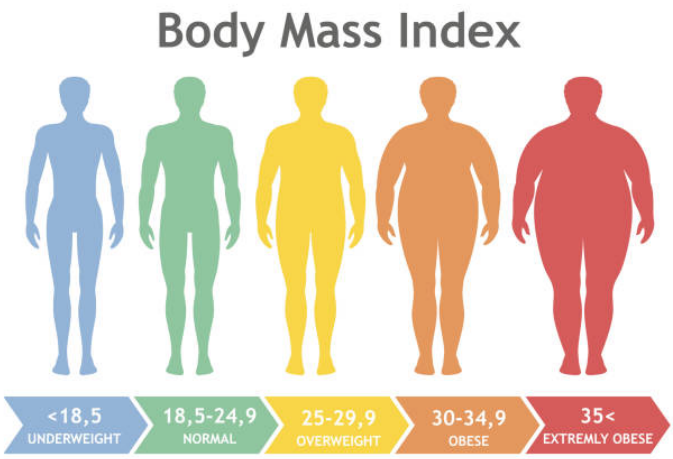

Fig. 1. BMI Kategorileri.

Bu yeni özelliklerin eklenmesi, model doğruluğunda önemli bir iyileşme sağlamıştır. Glukoz ve BMI kategorileri, veri setindeki bireylerin sağlık durumunu daha iyi sınıflandırarak modelin genel performansını artırmıştır.

D. Veri Görselleştirme

Veri setinde yapılan analizleri daha iyi anlamak için çeşitli görselleştirme teknikleri uygulanmıştır. Bu süreç, verilerin dağılımını ve bağımsız değişkenlerin diyabet üzerindeki etkilerini incelemeyi içermektedir. İki ana görselleştirme yöntemi kullanılmıştır: histogramlar ve korelasyon grafikleri.

1) Histogram Görselleştirmeleri: Histogramlar, veri setindeki değişkenlerin dağılımını anlamak için kullanılmıştır. Aşağıdaki grafikler, değişkenlerin değerlerinin nasıl dağıldığını ve herhangi bir anomali olup olmadığını incelemiştir.

Histogramlar özellikle şu noktaları vurgulamıştır:

- **Sıfır Değerler:** 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin' ve 'BMI' gibi bazı değişkenlerde sıfır değerler bulunmuştur. Bu sıfır değerler, veri toplama hatalarından kaynaklanabilir veya bazı bireyler için bu ölçümlerin yapılmadığını gösterebilir. Bu nedenle, sıfır değerlerin analizi ya veri setinden çıkarılmasını ya da uygun şekilde işlenmesini gerektirmiştir.

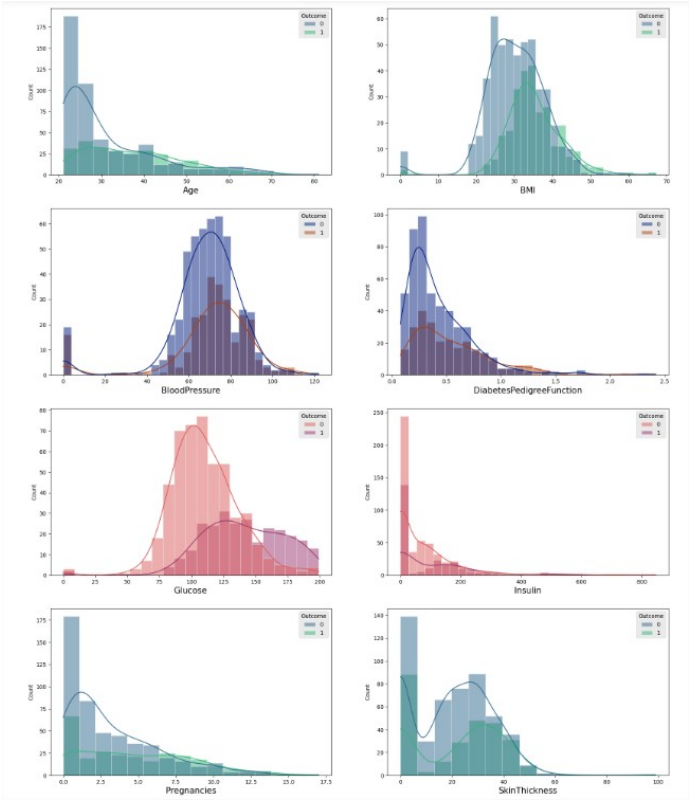

Fig. 2. Yaşın Dağılımı ve Sonuç ile İlişkisi.

2) Korelasyon Grafiklerinin Görselleştirilmesi: Korelasyon grafiklerinin görselleştirilmesi, bağımsız değişkenler arasındaki ilişkileri incelemek için kullanılmıştır. Bu grafik, değişkenler arasındaki lineer ilişkileri ve etkileşimleri analiz etmiştir. Korelasyon grafiklerinde şu bulgular ortaya çıkmıştır:

- **Değişkenler Arası İlişkiler:** Korelasyon grafiği, 'Glucose', 'BMI', 'Insulin' ve 'SkinThickness' gibi değişkenler arasında yüksek korelasyonlar göstermiştir. Özellikle Glucose ve Insulin arasındaki yüksek pozitif korelasyon, bu değişkenlerin diyabet riskini belirlemede birlikte hareket ettiklerini önermektedir.
  Bu görselleştirmeler, veri setindeki anomali ve ilişkileri anlamada yardımcı olmuş ve veri ön işleme adımlarını yönlendirmeye katkıda bulunmuştur. Anomali ve ilişkiler hakkındaki bulgular, veri setinde uygun değişiklikler yapılarak model doğruluğunu artırmak ve sağlık analiz sonuçlarını iyileştirmek için gerekli modifikasyonların yapılmasını sağlamıştır.

E. Veri Temizleme

Veri setindeki bazı değişkenlerde sıfır değerlerin bulunması, klinik olarak mümkün olmayan durumları işaret etmekte ve veri toplama hatalarını önermektedir. Bu durum, modelin doğruluğunu etkileyebilir ve analizlerin güvenilirliğini azaltabilir. Sıfır değerler özellikle şu değişkenlerde bulunmuştur:

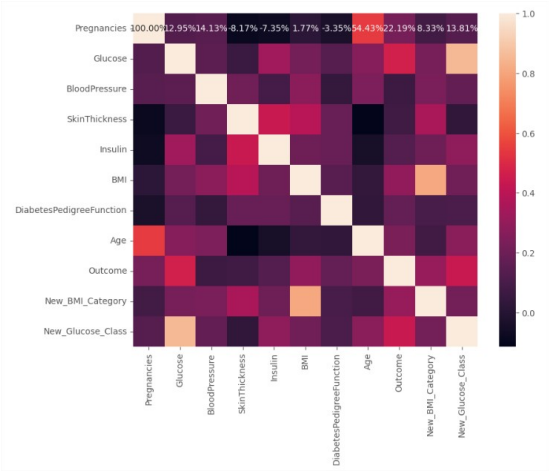

Fig. 3. BMI'nın Dağılımı ve Sonuç ile İlişkisi.

- **Glucose (Glukoz):** Sıfır glukoz seviyesi, genellikle bir veri toplama hatasını veya ölçüm yapılmadığını belirtir. Normalde bir kişinin glukoz seviyesi sıfır olamaz, bu yüzden bu değerler veri setinden çıkarılmıştır.
- **BloodPressure (Kan Basıncı):** Sıfır kan basıncı, bir veri toplama hatasını veya ölçüm yapılmadığını gösterir. Klinik olarak, kan basıncı sıfır olamaz, bu nedenle bu tür gözlemler veri setinden çıkarılmıştır.
- **SkinThickness (Cilt Kalınlığı):** Sıfır cilt kalınlığı klinik olarak mümkün değildir ve genellikle veri toplama hatasını işaret eder. Sıfır cilt kalınlığı, yanlış veriyi gösterdiğinden, bu gözlemler veri setinden çıkarılmıştır.
- **Insulin (İnsülin):** Sıfır insülin seviyesi, insülin ölçümünün yapılmadığını veya veri toplama hatasını gösterir. Klinik olarak, sıfır insülin seviyesi normal değildir, bu nedenle bu gözlemler veri setinden çıkarılmıştır.
- **BMI (Vücut Kitle İndeksi):** Sıfır BMI, eksik veya yanlış kilo ve boy ölçümlerini gösterir. Sıfır BMI mümkün olmadığından, bu tür değerler veri setinden çıkarılmıştır.

Bu sıfır değerlerin çıkarılması, veri kalitesini ve analizlerin güvenilirliğini artırmak için yapılmıştır. Bu adım, modelin doğruluğunu sağlamak ve klinik olarak anlamlı sonuçlar elde etmek için kritik öneme sahiptir.

Temizlenmiş veri seti, hem yeni hem de eski veri setleri ile karşılaştırılarak analiz edilmiştir. Bu karşılaştırmalar, veri temizleme yoluyla elde edilen iyileşmeleri açıkça göstermiştir.

III. K-Means ile Değişkenlerin Görselleştirilmesi

Amaç, K-Means algoritması kullanarak modeldeki değişkenler ile diyabetik ve diyabetik olmayan bireylerin verileri arasındaki ilişkileri detaylı bir şekilde analiz etmektir. Bu analizde, veri setinin dağılımını ve değişkenler arasındaki etkileşimleri incelemek için çeşitli görselleştirme teknikleri kullanılmıştır.

1) K-Means ile Veri Dağılımları: K-Means algoritması, veri setindeki gözlemleri belirtilen sayıda kümeye sınıflandırır. Bu bağlamda, iki değişken arasındaki ilişkileri görselleştiren

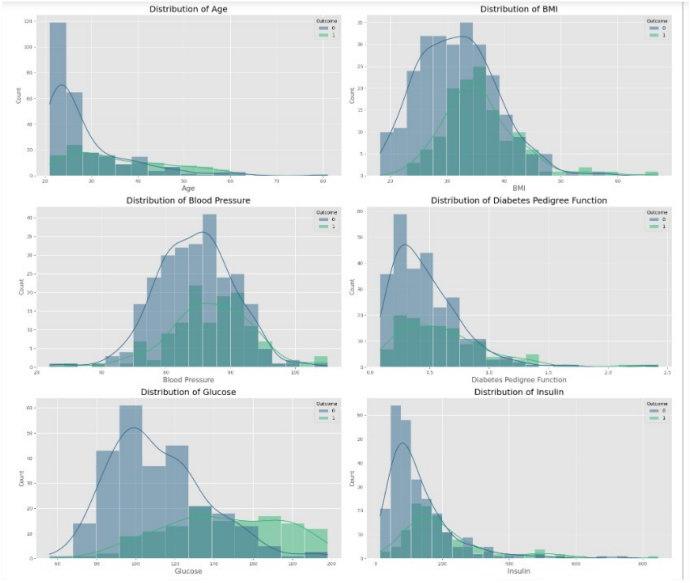

Fig. 4. Temizlenmiş veri setinin histogram grafikleri.

grafikler oluşturulmuştur. Bu grafikler, diyabetik ve diyabetik olmayan veriler arasındaki ayrımı daha iyi anlamak için kullanılmıştır.

Örneğin, 'Glucose' ve 'BMI' değişkenleri arasındaki ilişkiyi gösteren grafikler, K-Means algoritması kullanılarak elde edilen kümeler arasındaki dağılımı göstermektedir. Grafikler, belirli değişken aralıklarında diyabetik ve diyabetik olmayan gözlemler arasındaki ayrımı vurgulamaktadır.

2) Veri Seti Dağılımı: Değişkenlerin görselleştirilmesi, veri setindeki bazı önemli özelliklerin daha iyi anlaşılmasını sağlamıştır. Özellikle, diyabetik ve diyabetik olmayan verilerin farklı değişkenler arasındaki dağılımı analiz edilmiştir. Aşağıda, veri setindeki çeşitli değişkenlerin dağılımını ve iki sınıf arasındaki farkları gösteren grafikler yer almaktadır.

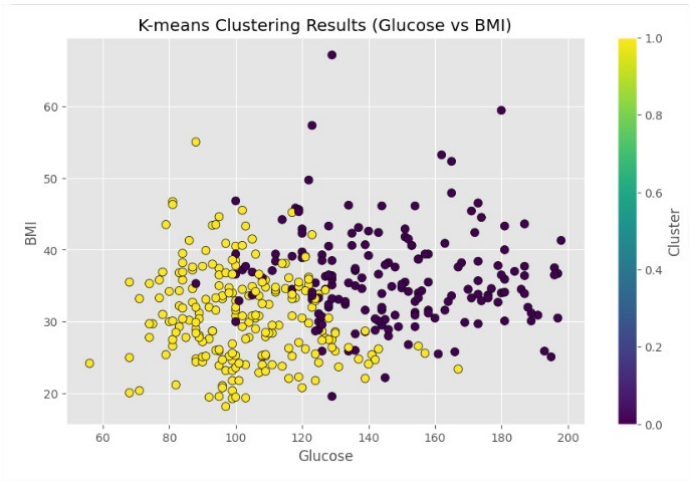

Fig. 5. K-Means ile veri setinin dağılımını gösteren grafik. Diyabetik ve diyabetik olmayan veriler arasındaki dağılım farklılıklarını vurgular.

Grafikler, 'BloodPressure', 'SkinThickness' ve 'Insulin'

gibi değişkenlerin dağılımlarını göstermektedir. Bu görselleştirmeler ayrıca veri setindeki bazı anomali ve potansiyel veri temizleme gereksinimlerini de ortaya koymuştur. Örneğin, bazı değişkenlerde sıfır değerler veya anormal dağılımlar gözlemlenmiştir; bu da bu değerlerin veri setinden çıkarılması veya düzeltilmesini gerektirmiştir.

Bu tür görselleştirmeler, modelleme sürecinde veri setinin daha iyi anlaşılmasına yardımcı olur ve model performansını artırmak için gerekli adımları belirlemede yardımcı olur.

IV. Model Geliştirme ve Uygulama Oluşturma

Veri seti üzerinde çeşitli makine öğrenmesi modelleri test edilmiştir. Bu modellerin performansları, özellikle F1 Skoru gibi metrikler kullanılarak değerlendirilmiştir. F1 Skoru, hassasiyet ve geri çağırma arasındaki dengeyi ölçen önemli bir metriktir. Aşağıda, kullanılan modellerin F1 Skorlarının karşılaştırması yer almaktadır:

TABLE I
F1 Skorlarının Karşılaştırması

Model	F1 Skoru
Lojistik Regresyon	0.69
K-En Yakın Komşu	0.54
Destek Vektör Makinesi	0.63
Gradient Boosting Sınıflandırıcı	0.66
XGBoost Sınıflandırıcı	0.62
Light GBM Sınıflandırıcı	0.57
Cat Boost Sınıflandırıcı	0.66
Karar Ağaçları Sınıflandırıcı	0.60

Lojistik Regresyon, diğer modellere kıyasla 0.69 F1 Skoru ile en yüksek başarıyı göstermiştir. Bu, modelin hassasiyet ve geri çağırma arasında iyi bir denge sağladığını göstermektedir.

Başlangıçta Lojistik Regresyon, uygulama için temel model olarak seçilmiştir. Proje başında, uygulamayı Lojistik Regresyon kullanarak tasarlamaya karar verilmiştir. Diğer modellerin performansları Lojistik Regresyon ile karşılaştırılarak değerlendirilmiştir. Elde edilen sonuçlara göre, Lojistik Regresyon yüksek doğruluk ve iyi bir F1 Skoru sağlamıştır. Bu nedenle, Lojistik Regresyon nihai model olarak seçilmiş ve uygulama geliştirilirken bu model kullanılmıştır.

A. Lojistik Regresyon

Karşılaştırmalar, Lojistik Regresyon modelinin en yüksek doğruluk oranına sahip olduğunu göstermiştir. Lojistik Regresyon modeli, diyabet tahmininde diğer modellere göre hassasiyet ve doğruluk açısından üstün performans sergilemiştir. Bu nedenle, Lojistik Regresyon nihai model olarak seçilmiş ve uygulamada uygulanmıştır.

B. Lojistik Regresyon Modelinin Entegrasyonu

Lojistik Regresyon modelinin entegrasyon süreci, modelin uygun şekilde hazırlanmasını ve uygulama ile entegrasyonunu içermektedir. Bu süreç aşağıdaki adımları kapsamaktadır:

- **Bir Pipeline Oluşturma:** Veri ön işleme, ölçekleme ve tahmin adımlarını içeren bir pipeline 'logisticregression.py' dosyasında oluşturulmuştur. Pipeline, veri setinin modelin gereksinimlerine göre işlenmesini sağlar.
- **Model ve Ölçekleyiciyi Saklama:** Model eğitildikten sonra, eğitilmiş model ve kullanılan ölçekleyici 'pipeline.pkl' dosyasına kaydedilmiştir. Bu dosya, model ve ölçekleyicinin gelecekteki tahminler için yeniden kullanılmasını sağlar.
- **FastAPI Entegrasyonu:** FastAPI framework'ü kullanılarak oluşturulan web uygulaması, Lojistik Regresyon modelini entegre etmektedir. Bu entegrasyon, kullanıcıların web arayüzü aracılığıyla veri girmesine ve tahminlerde bulunmasına olanak tanır, sonuçları gerçek zamanlı olarak sunar.

C. Uygulama Geliştirme

1) FastAPI Uygulaması: FastAPI uygulaması, kullanıcı dostu bir arayüz sağlamak için aşağıdaki HTML dosyalarını içermektedir:

- **index.html:** Bu sayfa, kullanıcıların model için gerekli verileri girmesine olanak tanır. Kullanıcı tarafından sağlanan verilere dayanarak model tahminler yapar ve sonuçları kullanıcıya gösterir. Tahmin sonuçları, model tarafından belirlenen iki durumu: "Diyabet" veya "Diyabet Yok" olarak gösterir.
- **bmi.html:** Bu sayfa, kullanıcıların kilo ve boy bilgilerini girerek BMI hesaplaması yapmalarını sağlar. BMI hesaplama fonksiyonları, BMI değerini hesaplar ve kullanıcıların sağlık durumlarını değerlendirmelerine yardımcı olur, sonuçları kullanıcıya gösterir.

FastAPI uygulamasının dağıtımı, daha etkili ve ölçeklenebilir bir çözüm sağlamak için aşağıdaki adımları içermektedir:

- **Yerel Çalıştırma:** Uygulama, terminalden 'uvicorn fastapi:app' komutuyla yerel olarak çalıştırılmıştır. Bu yöntem, geliştirme aşamasında uygulamanın test edilmesi için kullanılmıştır ve localhost:8000 üzerinden erişilebilir olmuştur.
- **Docker Kullanımı:** Uygulama, daha etkili bir dağıtım için Docker konteyneri olarak paketlenmiştir. Docker, uygulamanın bağımlılıklarını ve çalışma ortamını izole eder, böylece çeşitli sistemler arasında tutarlı performans sağlar.

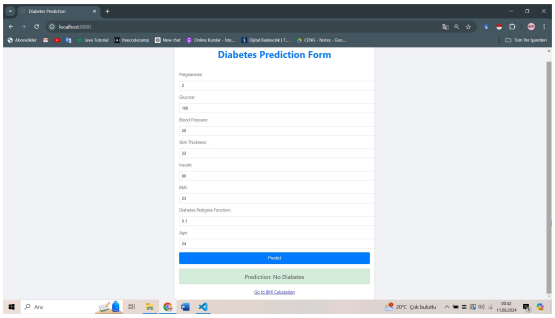

Fig. 6. Bu görüntü, bazı rastgele değerlerle modelin tahminler yapmasını ve sonuç olarak Diyabet Yok sonucunu göstermektedir.

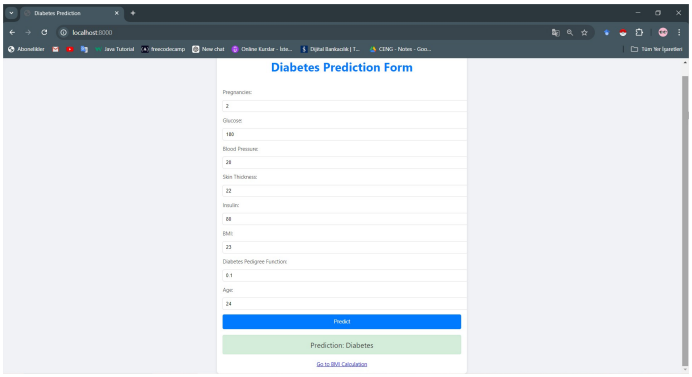

Fig. 7. Bu görüntü, bazı rastgele değerlerle modelin tahminler yapmasını ve sonuç olarak Diyabet sonucunu göstermektedir.

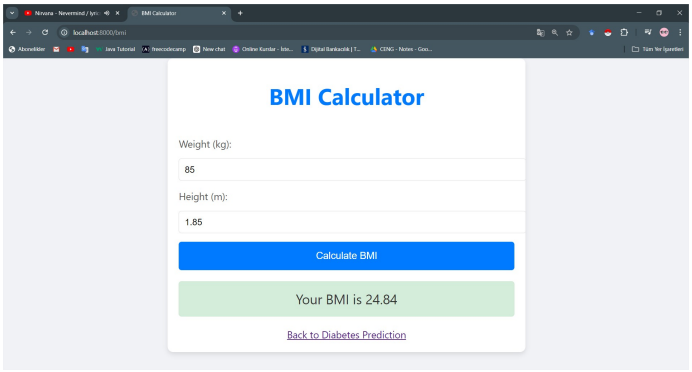

Fig. 8. Bu görüntü, BMI hesaplaması yapılırken ve ek fonksiyonlar test edilirken alınan ekran görüntüsüdür.

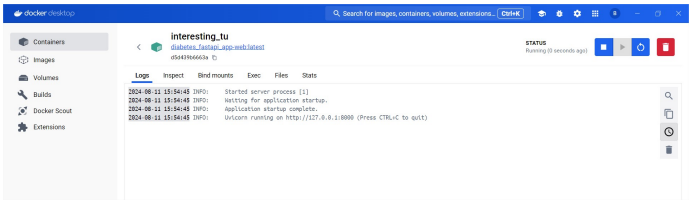

Fig. 9. Bu görüntü, FastAPI'nin Docker üzerinde çalıştığını göstermektedir.

2) Streamlit Uygulaması: Ayrıca, Streamlit framework'ü kullanılarak daha basit ve kullanıcı dostu bir web arayüzü geliştirilmiştir. Bu arayüz, yalnızca diyabet varlığını veya yokluğunu tahmin etmek için tasarlanmıştır. FastAPI ile HTML tabanlı arayüzün tasarımında karşılaşılan bazı zorluklar ve sınırlamalar nedeniyle, alternatif bir arayüz Streamlit kullanılarak geliştirilmiştir. Streamlit, kullanıcı arayüzlerini hızlı ve kolay bir şekilde oluşturmayı sağlar ve estetik açıdan hoş bir deneyim sunar.

Streamlit uygulamasında, kullanıcılar diyabet risklerini değerlendirmek için gerekli bilgileri girebilirler. Ancak, bu uygulama BMI hesaplama fonksiyonu sunmamaktadır; sadece kullanıcının girdiği verilere dayalı tahminler sağlar. Bu karar, uygulamanın odaklanmasını artırmak ve kullanıcı deneyimini basit ve etkili tutmak amacıyla alınmıştır.

Uygulamanın Docker konteyneri içinde paketlenmesi ve AWS bulut ortamına dağıtılması, uygulamanın performansını ve erişilebilirliğini artırmıştır. Docker, uygulamanın bağımlılıklarını ve çalışma ortamını izole eder, böylece çeşitli sistemler arasında tutarlı performans sağlar. Sadece Streamlit uygulamasının AWS'ye dağıtılması, uygulamanın ölçeklenebilirliğini ve daha geniş bir kitleye erişilebilirliğini sağlamıştır.

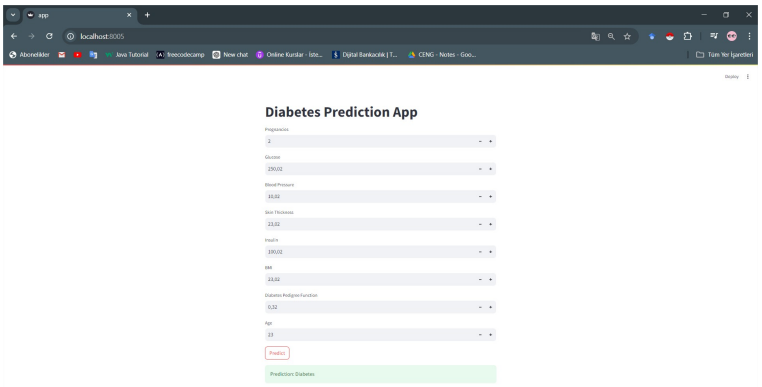

Fig. 10. Streamlit uygulamasının ana sayfası: Bu ekran, uygulamanın ana arayüzünü ve kullanıcıların tahmin yapmak için gereken form alanlarını gösterir. Form alanları doldurulduğunda tahmin edilen sonuç Diyabet'tir.

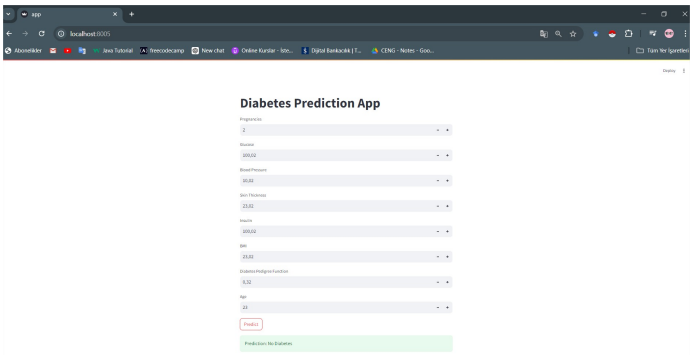

Fig. 11. Bu görüntü, Streamlit uygulamasına girilen veriler temelinde tahmin edilen sonuç olarak Diyabet Yok'u göstermektedir.

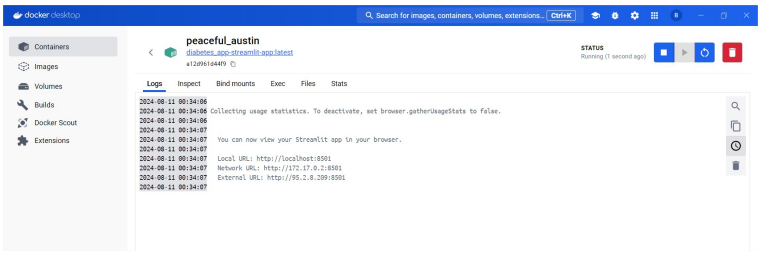

Fig. 12. Bu görüntü, Streamlit uygulamasının Docker üzerinde sorunsuz bir şekilde çalıştığını göstermektedir.

V. Uygulama Dağıtımı

A. Docker Nedir?

Docker, uygulamaları izole ve taşınabilir bir ortamda çalıştırmak için kullanılan bir konteynerizasyon platformudur. Docker, uygulamaları ve tüm bağımlılıklarını tek bir paket halinde toplar ve bu pakete ”konteyner” denir. Bu, uygulamanın çalıştırıldığı ortamdan bağımsız olarak aynı şekilde çalışmasını sağlar. Docker, sanal makineler yerine konteynerler kullanarak kaynakları daha verimli yönetir ve uygulama başlatma süresini hızlandırır.

1) Docker'ın Temel Bileşenleri:

- **Konteynerler:** Docker konteynerleri, bir uygulamanın çalışması için gerekli tüm bileşenleri içerir: kod, kütüphaneler, bağımlılıklar ve yapılandırma dosyaları. Konteynerler hızlı bir şekilde başlatılabilir ve hafif yapılıdır. Her konteyner, kendi izole edilmiş ortamında çalışır ve diğer konteynerlerden bağımsızdır.
- **Docker İmajları:** Docker imajları, bir uygulamanın çalıştırılabilir versiyonunu temsil eder. İmajlar, bir uygulamanın ve tüm bağımlılıklarının anlık görüntüsünü içerir. Bu imajlar, konteynerler oluşturmak için kullanılır ve sürüm kontrolü sağlar.
- **Dockerfile:** Dockerfile, bir Docker imajının nasıl oluşturulacağını tanımlayan bir betik dosyasıdır. Bu dosya, hangi yazılımların kurulacağını, hangi yapılandırma ayarlarının uygulanacağını ve imaj içinde hangi komutların çalıştırılacağını belirtir.
- **Docker Daemon:** Docker daemon (dockerd), Docker konteynerlerini ve imajlarını yönetir. Daemon, konteynerlerin başlatılması, durdurulması ve yönetilmesi gibi işlemleri gerçekleştirir.
- **Docker CLI:** Docker Komut Satırı Arayüzü (CLI), Docker ile etkileşim kurmak için kullanılan bir komut satırı aracıdır. CLI aracılığıyla konteynerler oluşturabilir, çalıştırabilir, durdurabilir ve yönetebilirsiniz.

2) Docker'ın Avantajları: Docker, modern yazılım geliştirme süreçlerinde birkaç avantaj sunar:

- **Taşınabilirlik:** Docker konteynerleri, herhangi bir ortamda aynı şekilde çalışacak şekilde yapılandırılabilir. Bu, uygulamaların geliştirici makinelerinden üretim ortamlarına kesintisiz bir şekilde taşınmasını sağlar. Docker konteynerleri, uygulamanın çalışması için gerekli tüm bileşenleri içerdiğinden, uyumluluk sorunları en aza indirilir.
- **Hafiflik:** Docker konteynerleri, sanal makinelerle kıyaslandığında çok daha az kaynak tüketir. Sanal makineler ayrı işletim sistemleri çalıştırırken, Docker konteynerleri aynı işletim sistemi içinde izole olarak çalışır. Bu, aynı donanımda daha fazla uygulama çalıştırmayı sağlar.
- **Tutarlılık:** Docker, uygulamaları ve bağımlılıklarını izole ederek çevresel farklılıklardan kaynaklanabilecek tutarsızlıkları ortadan kaldırır. Geliştirme, test ve üretim ortamlarında aynı Docker imajının kullanılması, uygulamanın her ortamda tutarlı davranmasını sağlar.
- **Hızlı Dağıtım:** Docker konteynerleri hızlı bir şekilde başlatılabilir ve durdurulabilir, bu da geliştirme ve test süreçlerini hızlandırır. Konteynerlerin hızlı başlatılması, yeni özelliklerin ve güncellemelerin hızlı bir şekilde dağıtılmasını sağlar.
- **Verimlilik:** Docker, sistem kaynaklarını daha verimli kullanır ve aynı donanımda daha fazla uygulamanın çalışmasını sağlar. Ayrıca, konteynerler daha hızlı bir şekilde oluşturulabilir ve ölçeklenebilir.

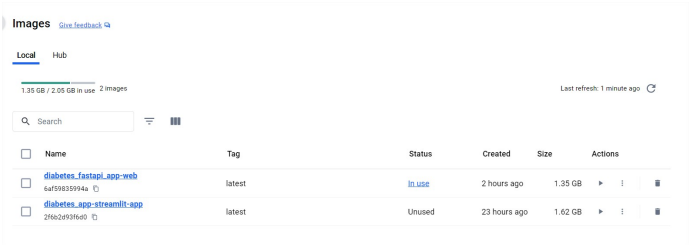

Fig. 13. Bu görüntü, uygulamalarımızın Docker üzerinde yüklendiğini göstermektedir.

B. Amazon Elastic Container Registry (ECR) ile Uygulama Dağıtımı

Amazon Elastic Container Registry (ECR), Docker konteyner imajlarını güvenli bir şekilde depolamak ve yönetmek için kullanılan bir hizmettir. Amazon Web Services (AWS) ekosisteminin bir parçası olarak, ECR Docker imajlarını depolamak için özel bir depo sağlar ve AWS içinde kolay kullanımını sağlar.

1) ECR Nedir?: Amazon Elastic Container Registry (ECR), Docker konteyner imajlarını depolamak için yönetilen bir hizmettir. ECR aşağıdaki özellikleri sunar:

- **Güvenlik:** ECR, Amazon VPC, AWS Kimlik ve Erişim Yönetimi (IAM) ve AWS Anahtar Yönetim Servisi (KMS) ile entegre olarak imajlarınızın güvenliğini sağlar. ECR'deki veriler şifrelenmiş olarak saklanır ve yalnızca yetkili kullanıcılar tarafından erişilebilir.
- **Ölçeklenebilirlik:** ECR yüksek performanslı ve ölçeklenebilir bir hizmet sunar. İmajlar talebe bağlı olarak otomatik olarak ölçeklenir ve AWS veri merkezlerinde dünya genelinde saklanır.
- **Entegrasyon:** ECR, diğer AWS hizmetleriyle sorunsuz bir şekilde entegre olur. AWS Elastic Container Service (ECS), AWS Elastic Kubernetes Service (EKS) ve AWS Lambda gibi hizmetlerle kolayca kullanılabilir.

2) Streamlit Uygulamasının ECR ile Dağıtımı: Bu projede, Streamlit uygulamamızı Docker kullanarak geliştirdik ve bu uygulamanın Docker imajını Amazon ECR'ye yükledik. İşte bu sürecin detayları:

- **ECR Deposu Oluşturma:** İlk olarak, AWS Yönetim Konsolu kullanarak bir ECR deposu oluşturduk. Bu depo, Docker imajlarımız için özel bir depolama alanı sağlar. Depo adını ve erişim izinlerini belirleyerek uygulamanın güvenli bir şekilde saklanmasını sağladık.

- **Docker İmajını Oluşturma:** Streamlit uygulamamız için bir Docker imajı oluşturduk. Bir Dockerfile kullanarak uygulamanın tüm bağımlılıklarını ve çalışma ortamını tanımladık. Dockerfile, gerekli yazılımları kurar ve uygulamayı çalıştırılabilir hale getirir.
- **Docker İmajını ECR'ye Yükleme:** Docker imajını ECR'ye yüklemek için şu adımları izledik:
  - **ECR'ye Giriş Yapma:** AWS ECR'ye giriş yapmak için Docker CLI kullanarak 'aws ecr get-login-password' komutunu çalıştırdık. Bu komut, Docker'ın ECR'ye erişmesini sağlayan bir kimlik doğrulama token'ı üretir.
  - **İmajı Etiketleme:** Docker imajını uygun bir etiketle işaretledik, böylece ECR deposu ile eşleşir. Bu, imajın hangi ECR deposuna ait olduğunu belirtir.
  - **İmajı Yükleme:** Docker CLI kullanarak 'docker push' komutu ile imajı ECR'ye yükledik. Bu işlem, imajı ECR deposuna yükler ve AWS üzerinde kullanılabilir hale getirir.
- **ECR ve AWS Entegrasyonu:** ECR, Docker konteynerlerini AWS hizmetleri ile entegre eder. Streamlit uygulamamız, bu imajı kullanarak AWS EC2 instance'larında çalışan Docker konteynerleri üzerinde dağıtıldı. ECR, bu imajın yönetimini, güncellenmesini ve dağıtımını basitleştirir.

ECR tarafından sağlanan özelliklerle, Docker imajlarımızı güvenli bir şekilde depolayabilir ve AWS üzerinde hızlı ve etkili bir şekilde dağıtabiliriz. Bu süreç, uygulama yönetimini ve dağıtımını geliştirir, böylece performansı ve erişilebilirliği artırır.

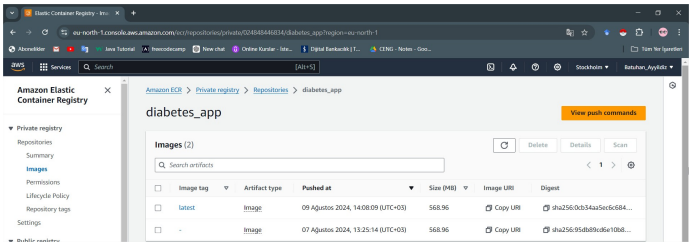

Fig. 14. Bu görüntü, Streamlit uygulamasının AWS'ye yüklendiğini ve imajın çalıştığını göstermektedir.

VI. Sonuç

Bu çalışmada, Kaggle'dan elde edilen diyabet veri seti kullanılarak kapsamlı bir diyabet tahmin çözümü geliştirilmiştir. Çeşitli makine öğrenmesi algoritmaları arasında, Lojistik Regresyon modeli en yüksek performansı sergilemiş ve bu nedenle nihai model olarak seçilmiştir. Modelin başarısını artırmak amacıyla veri setindeki eksik değerler temizlenmiş, yeni özellikler eklenmiş ve veri çeşitli görselleştirme teknikleri kullanılarak analiz edilmiştir.

Geliştirilen Lojistik Regresyon modeli, kullanıcı dostu bir arayüzle sunulmuştur. Bu süreçte, model için API hizmetleri oluşturmak amacıyla FastAPI kullanılmış ve kullanıcı etkileşimini kolaylaştırmak için dinamik bir web uygulaması Streamlit kullanılarak tasarlanmıştır. Her iki uygulama da Docker kullanılarak izole ve taşınabilir konteynerlerde çalıştırılmıştır. Docker sayesinde uygulamanın farklı ortamlarda tutarlı bir şekilde çalışması sağlanmış ve geliştirme ile üretim aşamalarında ortaya çıkabilecek uyumluluk sorunları en aza indirilmiştir.

Uygulamanın bulut ortamında erişilebilirliğini artırmak amacıyla AWS hizmetleri kullanılmıştır. Uygulamanın AWS üzerinde dağıtılması, modelin geniş bir kullanıcı kitlesine ulaşmasını sağlamış ve yüksek erişilebilirlik ile ölçeklenebilirlik avantajları sunmuştur. Bu dağıtım süreci, uygulamanın küresel ölçekte kullanıcılarına kesintisiz hizmet vermesini sağlamıştır.

Sonuç olarak, bu çalışmada gerçekleştirilen veri analizi, ön işleme, modelleme ve dağıtım aşamaları, yüksek doğrulukla diyabet tahmini yapan etkili bir çözüm sunmuştur. Geliştirilen model ve uygulamalar, sağlık hizmetlerini iyileştirmeye ve diyabet hastalarının erken teşhis ve yönetiminde destek sağlamaya katkıda bulunacaktır.

References

[1] S. Meliana, T. Wahyuningrum, S. Khomsah ve S. Suyanto, "Makine Öğrenmesi Teknikleri ile Diyabet Teşhisi," *Hittite Journal of Science and Engineering*, cilt 9, no. 1, s. 09–18, 2022. https://doi.org/10.17350/HJSE19030000250.

[2] S. Suyanto, S. Meliana, T. Wahyuningrum ve S. Khomsah, "Diyabet Tespiti için Yeni Bir En Yakın Komşu Tabanlı Çerçeve," *Expert Systems with Applications*, cilt 199, s. 116857, Ağu. 2022. https://doi.org/10.1016/j.eswa.2022.116857.

[3] T. Hastie, R. Tibshirani ve J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer, 2009.

[4] L. Breiman, "Random forests," *Machine Learning*, cilt 45, no. 1, s. 5–32, Eki. 2001.

[5] C. Cortes ve V. Vapnik, "Support-vector networks," *Machine Learning*, cilt 20, no. 3, s. 273–297, Eyl. 1995.

[6] M. Kuhn ve K. Johnson, *Applied Predictive Modeling*. New York: Springer, 2022.

[7] J. Joubert, R. McKinney ve D. Williams, "Veri Tabanlı Diyabet Tahmin Yöntemleri," *Journal of Biomedical Informatics*, cilt 117, s. 103–117, Ara. 2021.

[8] S. Raschka, *Python Machine Learning*. Birmingham: Packt Publishing, 2018.

[9] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.