**GAZİ ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



**BMT-436 YAPAY ZEKAYA GİRİŞ**

**2021-2022 BAHAR DÖNEMİ**

**PROJE RAPORU**

**KESTİRİMCİ BAKIMDA ÇOKLU SINIF SINIFLANDIRMA YÖNTEMİNİN UYGULANMASI**

**181816076 – MEERİM KUBATOVA**

**İÇİNDEKİLER LİSTESİ**

[1. GİRİŞ 1](#_Toc105492177)

[2. PROJE İÇERİĞİ 2](#_Toc105492178)

[2.1. PROJE TANITIMI 2](#_Toc105492179)

[2.2. YÖNTEM 8](#_Toc105492180)

[2.2.1. PROBLEMİN TANIMI 8](#_Toc105492181)

[2.2.2. PROBLEMİ ÇÖZME AŞAMALARI 8](#_Toc105492182)

[2.3. KULLANILAN TEKNOLOJİLER 13](#_Toc105492183)

[3. SONUÇ 14](#_Toc105492184)

[4. EK 17](#_Toc105492185)

[5. KAYNAKÇA 18](#_Toc105492186)

**ŞEKİLLER LİSTESİ**

[Şekil 1. Motor üzerinde kestirimci bakım sistemi. 3](file:///D:\school\last%20semester\yapay%20zeka\project\yapayzeka_projeraporu.docx#_Toc105492187)

[Şekil 2. Kestirimci Bakımda kullanılan yapay zeka yöntemleri. 4](file:///D:\school\last%20semester\yapay%20zeka\project\yapayzeka_projeraporu.docx#_Toc105492188)

[Şekil 3. Model Eğitimi Süreci. 5](file:///D:\school\last%20semester\yapay%20zeka\project\yapayzeka_projeraporu.docx#_Toc105492189)

[Şekil 4. NASA Turbofan motoru veri seti içeriği 8](file:///D:\school\last%20semester\yapay%20zeka\project\yapayzeka_projeraporu.docx#_Toc105492190)

[Şekil 5. Bağımlı Değişkenin Etikelenme Kodları. 10](file:///D:\school\last%20semester\yapay%20zeka\project\yapayzeka_projeraporu.docx#_Toc105492191)

[Şekil 6. Özellik Ölçeklendirme Normalizasyon Formülü. 10](#_Toc105492192)

[Şekil 7. LSTM Modeli Mimarisi. 12](file:///D:\school\last%20semester\yapay%20zeka\project\yapayzeka_projeraporu.docx#_Toc105492193)

[Şekil 8. Label sütununun sınıf frekansı. 12](file:///D:\school\last%20semester\yapay%20zeka\project\yapayzeka_projeraporu.docx#_Toc105492194)

[Şekil 9. Class\_weight fonksiyonu. 13](file:///D:\school\last%20semester\yapay%20zeka\project\yapayzeka_projeraporu.docx#_Toc105492195)

[Şekil 10. Bağımlı değiken ile Bağımsız değişken arası korelasyon hesabı. 14](#_Toc105492196)

[Şekil 11. Hata Matrisi 15](file:///D:\school\last%20semester\yapay%20zeka\project\yapayzeka_projeraporu.docx#_Toc105492197)

[Şekil 12. Doğruluk grafiği 15](#_Toc105492198)

[Şekil 13. Loss Grafiği 16](#_Toc105492199)

[Şekil 14. Tahmin Edilen ve Olması Gereken değerlerin kesişme grafiği. 16](#_Toc105492200)

**TABLOLAR LİSTESİ**

[Tablo 1. Kestirimci Bakımda kullanılan problem çözme yaklaşımları. 4](#_Toc105492201)

[Tablo 2. Makine Öğreniminde kullnaılan kütüphaneler. 4](#_Toc105492202)

[Tablo 3. Bağımlı Değişkenin Etiketlendirilmesi. 9](#_Toc105492203)

[Tablo 4. Bağımlı Değişkenin Sınıfları. 10](#_Toc105492204)

[Tablo 5. Veriseti Kalan Kullanışlı Ömür (RUL), Bağımlı Değiken (Label) sütunları. 11](#_Toc105492205)

[Tablo 6. Verinin Ardışık Dizi'ye Dönüştürülmesi. 12](#_Toc105492206)

[Tablo 7. Model eğitimi parametreleri. 13](#_Toc105492207)

1. GİRİŞ

Proje, NASA Turbofan motorların kalan kullanışlı ömrününü (Remaining Useful Life) kullanarak motorun ‘iyi’, ‘uyarı’, ‘alarm’ durumlarında olup olmadığını tahmin eden makine öğrenimi modelinin eğitilmesi üzerinde konu almıştır. Model eğitimi çalışmaları esnasında makine öğrenimi süreçleri eksiksiz uygulanmıştır. Makine öğrenimi için Keras kütüphanesi kullanılmıştır.

İlk başta hedef değişkeninin etiketlenmesi işlemi ekipmanın plansız durma bilgilerine göre manuel olarak yapılmıştır. Keşifsel Veri analizi esnasında öznitelikler arasındaki ilişkiyi incelemek için Pearson korelasyonu hesabı yapılmıştır ve görselleştirme için Matplotlib, Seaborn kütüphaneleri kullanılmıştır. Veri görselleştirilmesi yardımıyla gürültülü veriler tespit edilerek veri setinden kaldırılmıştır.

Boyutsallık azaltma yöntemi olarak Korelasyon Eşiği ve Gömülü Yöntemler kullanılmıştır. Gömülü Yöntemlerden Random Forest algoritmasında gömülü olan Feature Importance tekniği kullanılmıştır. Ölçeklendirme çeşidi olarak normalizasyon uygulanmıştır. Makine öğrenimi algoritması olarak RNN (Tekrarlayan Sinir Ağları) ailesinden LSTM algoritması kullanılmıştır. Performans metriği olarak sınıflandırma problemlerinde doğruluğu ölçmek için kullanılan ‘accuracy’ metriğine ek olarak ‘precision’, ‘recall’ ve ‘F1-score’ metrikleri de göz önüne alınmıştır. Modelin değerlendirilmesi sonrasında, modeli iyileştirmek için farklı hiper parametreleri denenmiştir.

Proje boyunca yapılan her işlem rapora kaydedilmiştir. En uygun ve en başarılı model elde edildikten sonra proje tamamlanmıştır ve hazırlanan raporlar bir bütün haline getirilerek proje raporu elde edilmiştir.

İleride eğitilen model Yapay Zeka destekli kestirimci bakım sistemine entegre edilebilmektedir. Uygulanan her bir makine öğrenimi sürecinin farklı yöntemleri uygulanarak deneyim kazanılmıştır.

1. PROJE İÇERİĞİ
   1. PROJE TANITIMI

Kestirimci Bakım

Bir hastalığın önlenmesi, tedavi edilmesini gerektirecek duruma bırakmadan onun ortaya çıkmasını engeller. Bu yaklaşım yeni değildir ve ilk olarak Antik Yunanistan’da başlamıştır, zaman geçtikçe popülerliği artmıştır. 20.Yüzyıl boyunca, hastalığın ortaya çıkmasını önlemek için sadece insan vücudu üzerine olan yoğunlaşma milyonlarca insanın hayatını kurtarmıştır. Önlenebilir arızalar için geliştirilen “Kestirici Bakım” teknolojisi benzer fayda sağlayabilir. Gelecekte, bu tür bakım prosedürlerinin kullanılması zaman ve kaynaklarda büyük tasarruf sağlamanın yanı sıra şirketlerin en büyük sorunu olan makinanın arızalanma riskini azaltma kapasitesine sahiptir.

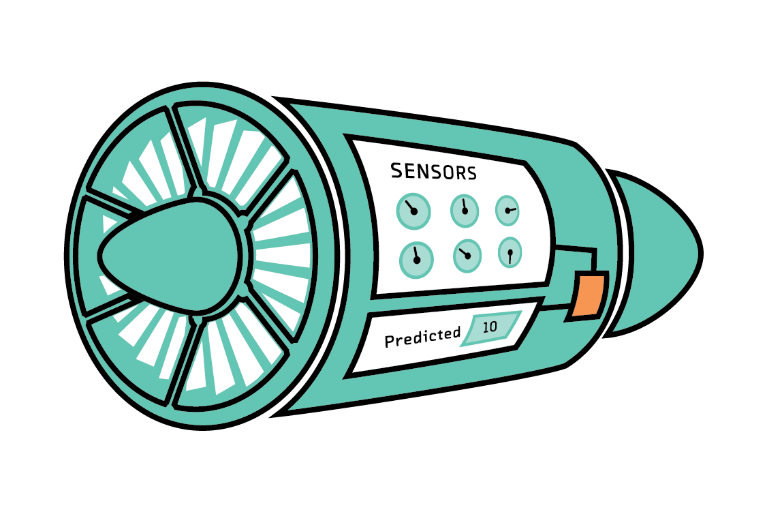
Günümüzde, endüstriyel üretimden tarıma kadar sürekli bir otomasyon süreciyle ilişkilendirilmeyen sektör yoktur. Dolayısıyla, şirketlerdeki endüstriyel makinaların kullanımı epeyce artmıştır. Bu ekipmaların plansız arızalanma olayları şirkete çok büyük zarar getirebilmektedir, ayrıca şirketlerin kapanmasına neden olabilmektedir.

Yapay zekanın ekipmanın etkili bir şekilde çalışmasını nasıl sağlayabileceğine ayrıntılı olarak bakmadan önce, kestirici bakım ve mevcut sistemin farklılıklarını anlamak önemlidir.

Her iki yöntemin de amacı, ekipmanların mümkün olduğu kadar uzun süre içerisinde, doğru çalıştırma koşulları altında fonksiyonelliğinden maksimum olarak faydalanabilmesini sağlamaktır. Ancak, metodolojileri farklıdır.

Geleneksel önleyici bakım, ekipmanın arıza riskini önleyen ve çalışma sırasında yapılan düzenli bakım süreci olarak tanımlanmaktadır. Proaktif olan bu yaklaşım, genellikle ekipmanın performans kontrolleri, dijital veya analog bileşenlerin ayarları, temizleme, yağlama, onarım ve parçaların değiştirilmesi veya iyileştirilmesi eylemlerini içerir. Bu yaklaşımın temel amacı, ekipmanın ömrünü uzatmak, zamanından erken devreden çıkmasını önlemek. Yapay Zekâ ve Nesnelerin İnterneti tarafından üretilen çözümler sayesinde yeni yaklaşım olan Kestirimci Bakım, koyulan hedefe ulaşılmasını kolaylaştırmıştır.

Kestirimci Bakım, ekipmanın durumunun yansıtıldığı sensörlerden veri toplar, analiz eder ve kestirim işlemi için Yapay Zekâyı kullanır. Bu Yapay Zekâ algoritmaları yardımıyla hata veya arızanın oluşmasından önce tespit edilip gerekli eylemlerin yapılması için fırsat tanırlar. Bu sebepten, Kestirimci Bakım ’da Yapay Zekânın kullanımı işletmelerin bakım prosedürlerini katı bir programa uymaya zorlamak yerine her ekipmanın bakımını ihtiyaç olup olmadığına göre uygulayarak para ve kaynak tasarrufu yapmalarını sağlar.

Şekil 1’de bir motor ve bu motorun durumunu tahmin eden yapay zeka tabanlı sistem gösterilmiştir.

Şekil 1. Motor üzerinde kestirimci bakım sistemi.

Yapay Zeka tabanlı Kestirimci Bakım nihayetinde önleyici bir metodoloji olsa da kapsamı, ekipmanın bir yıllık manuel muayenesinin çok ötesine geçebilir.

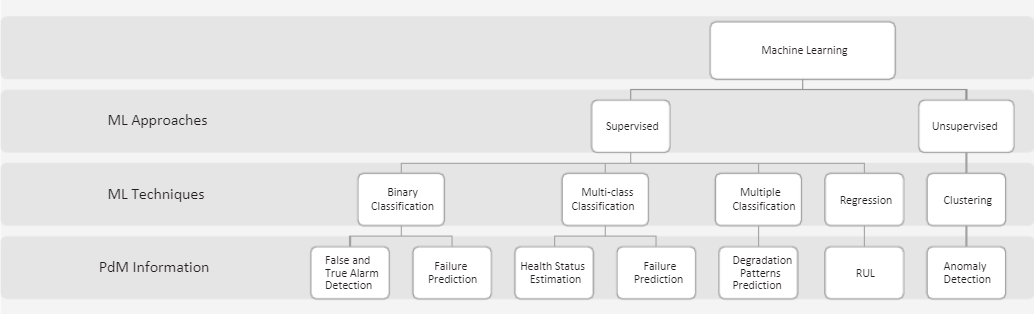
Vanson Bourne Research tarafından yapılan bir araştırmaya göre, planlanmamış ekipman arıza olayları ortalama dört saat içerisinde ve arıza başına toplam 1 milyon doların üzerindebir maliyetle sonuçlanmıştır. Bu duruma ek olarak, planlanmamış ekipman arızaları, şirketlerin %21’inin kapanmasına sebep olmaktadır. Bunca sebepleri saydıktan sonra hangi metodolojinin seçilmesi daha mantıklıdır sorusunun cevabı açıktır diye düşünüyorum.

NASA'nın Turbofan Motor Bozulma Simülasyonu veri setini kullanarak Kestirimci Bakım uygulanmasının Sınıflandırma Yaklaşımı ile gerçekleştirilmesi

Motorun kurulu olan belirli sayıda ayarları ve çalışma sayısına dair bilgiler vardır, bu bilgilere göre motorun durumu değişmektedir. Bu değişimler sensörlere yansıtılır, o sensörlerden toplanan bilgiler, ayar bilgileri, zaman bilgisi (bu projede cycle sayısı), arızasız gerçekleştirdiği toplam çalışma sayısı ve arızalanma durumundaki toplanan bilgiler kullanılarak bir sınıflandırma modeli eğitilir. Eğitilen model ile motorların durumları hakkında tahminler elde edilir, bu tahminlere göre elde olan gerçek bilgiler ile karşılaştırılarak modelin performansı ölçülür. Elde edilen doğruluk makül değerin üzerinde ise, bu model tahmin için kullanılabilmesi kabul edilir.

Ekipmanın bozulma sebebini bulmak için öznitelikler arasındaki ilişkiler incelenmesi gerekmektedir. Ayrıca kalan kullanışlı ömrünün (gün, saat, dk, vs. ölçüsünde) tahmin edilmesi gerekmektedir. Yukarıda geçen problemler ile uğraşan Yapay Zeka yöntemleri Regresyon ve Sınıflandırma yöntemleridir.

Şekil 2’de, kestirimci bakım probleminde kullanılan makine öğrenme teknikleri gösterilmiştir.



Şekil 2. Kestirimci Bakımda kullanılan yapay zeka yöntemleri.

Literatürdeki çalışmalarda Regresyon ve İkili Sınıflandırma yöntemleri mevcuttur [1]. Projenin yenilikçi yanı, Çoklu Sınıf Sınıflandırma (Multi-Class-Classification) yöntemini uygulayarak problemi çözmektir.

|  |  |  |
| --- | --- | --- |
| No: | Yöntem: | Tahmin Ettiği Değerler |
| 1 | Regresyon | Sürekli |
| 2 | İkili Sınıflandırma | {0,1} |
| 3 | Çoklu Sınıf Sınıflandırma | {0,1,2,…} |

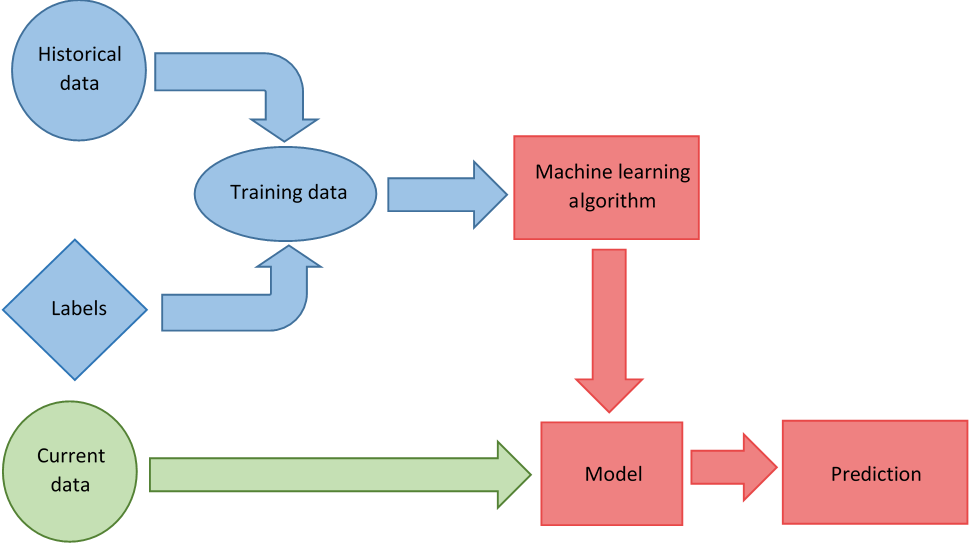
Tablo 1. Kestirimci Bakımda kullanılan problem çözme yaklaşımları.

Mekine Öğreniminde Kullanılan Kütüphaneler

|  |  |  |
| --- | --- | --- |
| No: | Kütüphaneler: | Kullanım Alanı: |
| 1 | Scikit-Learn | Scikit-Learn, açık kaynak kodlu temel makine öğrenmesi kütüphanesidir. Kullanımı ve öğrenmesi kolay bir arayüze (API ) sahip olması sebebiyle, günümüzde hala birçok kişi kullanmakta ve bu kütüphaneye katkıda bulunmaktadır. Scikit-Learn kütüphanesi kolay kullanımlı arayüzünün arkasında devasa ve çok iyi bir şekilde tasarlanmış, bir yapı bulunmaktadır. |
| 2 | Tensorflow | Tensorflow, derin öğrenme alanında en çok kullanılan kütüphanelerden biridir. Google tarafından geliştirilen açık kaynaklı TensorFlow ile derin öğrenme destekli yapay zekâ uygulamaları geliştirebiliyoruz. |
| 3 | Keras | Keras, neredeyse her tür derin öğrenme modelini tanımlamak ve eğitmek için uygun bir yol sağlayan Python için bir derin öğrenme kütüphanesidir. Keras, Tensorflow , Theano ve CNTK üzerinde çalışabilen Python ile yazılmış bir üst düzey sinir ağları API’sıdır. Keras ile derin öğrenme modeli oluşturmadan önce derin öğrenme literatüründe geçen bazı terimleri burada açıklamak gerekmekte. Bu sayede model oluşturduğumuz kısımda yeni başlayanlar çok fazla sorun yaşamayabilirler. |

Tablo 2. Makine Öğreniminde kullnaılan kütüphaneler.

Sınıflandıma Problemlerinde Kullanılan Makine Öğrenimi Algoritması

Geleneksel yapay sinir ağları bağlantı kurarak anlama özelliğine sahip değildir ve bu onun en önemli dezavantajıdır. Tekrarlayan Yapay Sinir Ağları (RNN) ise bu yeteneğe sahiptir.

Şekil 3. Model Eğitimi Süreci.

“Ardışık Bellek”, beynimizin bir dizi modelini tanıması için işi kolaylaştıran bir mekanizmadır. RNN bu mekanizmayı kullanır ve genel olarak tahmin için verileri ardışık olarak işler. Bu ardışık verileri döngü şeklinde, önceki düğümün çıkışını bir sonraki düğümün girişi olarak kullanır. Bu şekilde, her adımda yeni bilgi ile eski bilgilerimiz birlikte işlenir ve yeni çıktı oluşturarak önceki bilgileri hatırlama yeteneğine sahip olur.

RNN’lerin hatırlayabilme yeteneği sayesinde son yıllar içerisinde bir dizi alanlarda inanılmaz gelişmeler yaşanmıştır. Bu gelişmelerin birçoğu da RNN’nin türü olan Uzun Kısa-Vadeli Bellek (LSTM) yapısına ait olduğu söylenmektedir.

RNN’lerin önemli özelliği, hatırlayabilmeleridir. Diğer Yapay Sinir Ağlarından farkı ise, diğer sinir ağlarında her girdi birbirinden bağımsızdır, RNN’lerde girdiler birbiri ile ilişkilidir. Bu, RNN’lerin bir sonraki adımı takip edebilmeleri için girdiler arasında ilişki kurarlar ve eğitilirken tüm ilişkilerini hatırlarlar.

Avantajları:

* Geçmiş bilgileri dikkate alarak hesaplama
* Girdi büyüklüğüyle artmayan model boyutu
* Herhangi bir uzunluktaki girdilerin imkânı
* Zaman içinde paylaşılan ağırlıklar.

Dezavantajları:

* Yavaş hesaplama
* Mevcut durum için gelecekteki herhangi bir girdinin düşünülmemesi
* Uzun zaman önceki bilgiye erişme zorluğu.

Uzun Kusa-Vadeli bellek (LSTM)

LSTM yapısı RNN’den farklı olarak kapılar (gates) içerir. Bu kapılar hangi bilgilerin hatırlanması gerektiğini, hangilerinin unutulması gerektiğini belirler. Eğer girdi önemsizse unutulur, önemliyse bir sonraki aşamaya aktarılır.

Bazen elimizdeki görevi tamamlamak için yalnızca en son bilgilere ihtiyaç duyarız. Örneğin, bir sonraki kelimeyi önceki kelimelere dayanarak tahmin etmeye çalışan bir dil modelini düşünelim. “Bıçakla elimi kestim” cümlesindeki son sözcüğü tahmin etmek istiyorsak, daha geniş bir bağlama ihtiyacımız yok. Bu durumda son kelimenin “kestim” olacağı çok açıktır. Bu durumda, gerçek bilgi ile ihtiyaç duyulan yer arasındaki mesafe küçük olduğundan, RNN'ler geçmişten gelen bilgileri kullanmayı öğrenebilir. Ancak daha fazla bağlama ihtiyaç duyduğumuz zamanlar da vardır. Diyelim ki, "fransa'da büyüdüm... Ana dilim Türkçedir, akıcı bir şekilde konuştuğum bir diğer dil ise Fransızcadır" metnindeki son kelimeyi tahmin etmek istiyoruz. Doğrudan bağlam, son kelimenin dilin adı olacağını gösteriyor, ancak hangi dil olduğunu belirlemek için daha uzakta bulunan “fransa” bağlamına ihtiyacımız var. Böylece, gerçek bilgi ile uygulama noktası arasındaki boşluk çok büyük olabilir. Ne yazık ki, bu mesafe arttıkça, RNN'ler bilgi bağlama yeteneğini kaybederler.

RNN’ler kısa cümleler üzerinde gayet iyi bir sonuç veriyor olsa bile, sıra uzun cümleler veya paragraflara geldiğinde pek çok bekleneni karşılamamaktadır. Buna “Vanishing Gradient” problemi denir.

Aktivasyon fonksiyonları (girdi veya girdiler setinin ağırlıklı toplamının çıktıya nasıl dönüştürüleceğini belirler) sayesinde girdimizi belirli bir aralığa indirgeyebiliriz. Bu aralık genelde 0 ve 1 aralığındadır. Ufak bir alana indirgediğimiz için girdimizdeki büyük bir değişim, aktivasyon fonksiyonunda o kadar büyük bir değişime yol açmayabilir. Dolayısıyla türevi de küçük olur. Eğer türevi çok küçükse, o katman yeterli kadar öğrenemez.

RNN’de çok erken aşamalarda bile bu durum gerçekleşebilir. Bu katmanlar öğrenmediği için, RNN’ler daha uzun metinlerde gördüklerini unutabilir ve böylece kısa süreli bir hafızaya sahip olurlar. Bu problemin çözümlerinden biri de LSTM algoritmasıdır.

LSTM, uzun süreli bağımlılıkları öğrenebilen özel bir tür tekrarlayan sinir ağı mimarisine sahiptir. Bir dizi farklı görevi mükemmel bir şekilde çözmektedir ve günümüzde yaygın olarak kullanılmaktadır. LSTM'ler, uzun vadeli bağımlılık probleminden kaçınmak için özel olarak tasarlanmıştır. Bilgileri uzun süre ezberlemek onların normal davranışıdır, öğrenmekte zorlandıkları bir şey değildir.

LSTM yapısı RNN’den farklı olarak kapılar (gates) içerir. Bu kapılar hangi bilgilerin hatırlanması gerektiğini, hangilerinin unutulması gerektiğini belirler. Eğer girdi önemsizse unutulur, önemliyse bir sonraki aşamaya aktarılır. Bu işlem “gate” ve “cell rtate” yardımıyla yapılır. Toplamda dört tane kapı mevcuttur:

1. Forget Gate
2. Input Gate
3. Cell Gate
4. Output Gate

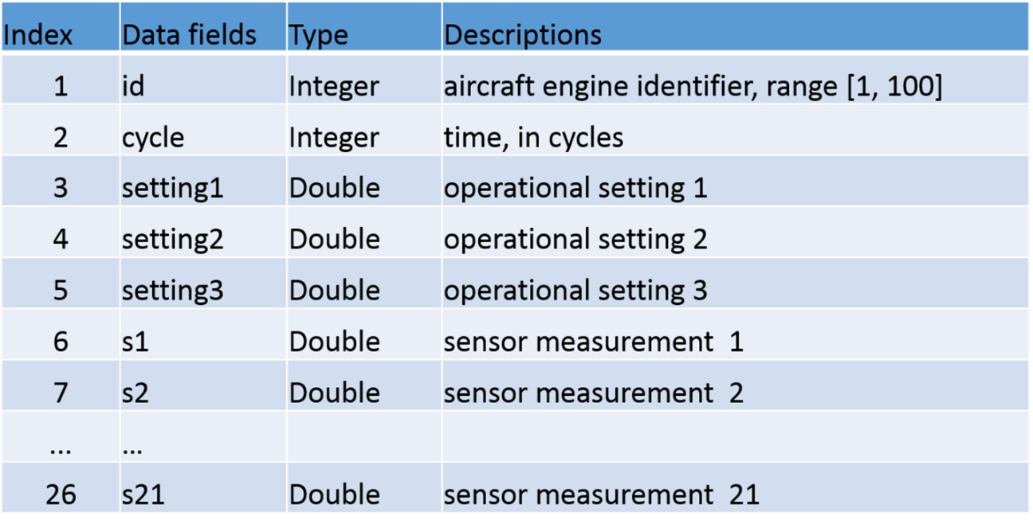
Forget Gate, bir önceki katmandan gelen bilgileri ve güncel bilgileri Sigmoid aktivasyon fonksiyonundan geçirir. (Elde edilen sayı 0’a ne kadar yakınsa, unutması o kadar muhtemeldir demek)

Input Gate, Cell Gate’i güncellemek için kullanılır. Burada da ilk önce Sigmoid fonksiyonu uygulanır ve hangi bilginin tutulacağına karar verilir. Daha sonra, ağı düzenlemek için (aktivasyon fonksiyonu yardımıyla (-1,1) arasına indirgenir ve çıkan iki sonuç çarpılır. (( hiperbolik tanjant)

Cell Gate, hücre içerisindeki en önemli bilgiyi tutar. Ağ üzerindeki veri akışını Cell State yardımıyla sağlanır. İlk olarak Forget Gate’den gelen sonuç ile bir önceki katmanın sonucu çarpılır sonrasında da Input Gate’den gelen değer ile toplanır.

Output Gate, bir sonraki katmana gidecek olan değere karar verir. Bu değer tahmin için kullanılır. Öncellikle şu anki girdi ile bir öneki değer Sigmoid fonksiyonundan geçer, sonra Cell Gate’den gelen değer ( aktivasyon fonksiyonundan geçtikten sonra iki değer çarpılır ve bir sonraki katmana “bir önceki değer” olarak gider. Cell State ile paralel ilerler.

* 1. YÖNTEM
     1. PROBLEMİN TANIMI

Projenin ana odak noktası makinanın ne zaman arızalanacağını mümkün olduğunca doğru tahmin etmek ve arızalanmasına sebep olan etkenleri bulmaktır. Bu işlem için makinanın durumunu gösteren sensör veriler kullanılır. Modelin girdi değişkenleri olarak hem durum verileri hem de zaman verisi kullanılmaktadır.

Şekil 4. NASA Turbofan motoru veri seti içeriği

Sınıflandırma

Önceden gözlemlenen değerleri kullanarak gelecekteki değerleri tahmin etme eylemine zaman serisi tahmini denilmektedir.

Projede yapay zeka çalışmaları için sensörlerden alınan tarihsel veriler kullanılmıştır. Önceki veriler ile eğitim yapılıp gelecekteki değerleri tahmin etmek için zaman serisi sınıflandırma işlemi gerçekleştirilmiştir.

Literatürde çok kaynak bulunmamaktadır. Ayrıca, her problem benzersiz olduğu için alınan probleme yeni teknikler ve yöntemler uygulanarak modelin performansı arttırılmaya çalışılmıştır. Bu teknikler ve yöntemler bir sonraki bölümde detaylı olarak açıklanmıştır. Mevcut olan çalışmalarda, regresyon analizi, ikili sınıflandırma yöntemleri kullanılmıştır. Bunlar:

* + *Motorun kalan kullanışlı ömrünü (RUL)* 🡪 *Linear Regression*
  + *Motorun kalan kullanışlı ömrünü (RUL)’ne göre arızadan önce motorun alarm(1) ve normal(0) zaman diliminde olup olmadığını* 🡪 *Binary Classification*

Projenin yenilikçi yanı, yeni yaklaşımın kullanılmasdır:

* + *Motorun kalan kullanışlı ömrünü (RUL)’ne göre arızadan* *önce alarm(2), uyarı(1) ve normal(0) zaman diliminde olup olmadığını 🡪 Multiclass Classification*
    1. PROBLEMİ ÇÖZME AŞAMALARI

1. Veri setinin hazırlanması
   1. Veri Setinin Temizlenmesi
      1. Uygun olmayan veri okumalarının kaldırılması
   2. Etikenlendirme
2. Özellik Mühendisliği
   1. Korelasyon Hesabı
   2. Özellik Ölçeklendirme
      1. Normalizasyon
      2. Standardizasyon
3. Modelin Eğitilmesi
4. Tahminlerin Elde Edilmesi
5. Modelin Değerlendirilmesi

İlk olarak, veri setini inceledim kaç satır, kaç sütundan oluşur. Veri setinde olan veriler ne verileridir. Neden bu veriler toplandı ve makine öğrenimi sürecinde bizim işimize yarayacak mı sorularını yanıtlamaya çalıştım. Bu soruları çözdükten sonra, zaman serisi değikenleri sürekli veriden oluşuyor mu, tamamen boş değerlerden oluşan değiken sayısı ve onlar nelerdir, sadece sabit değerlerden oluşan değişkenler nelerdir ve sayısı ne kadardır bilgisi kontrol edilmiştir. Boş ve sabit değişkenlerin modele hiçbir etkisi olmayacağından veri setinden kaldırılmasına karar verilmiştir.

Bu sütünlar:

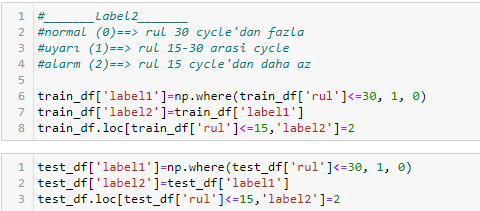
['sensor\_22', 'sensor\_23', 'settings\_3', 'sensor\_1', 'sensor\_5', 'sensor\_6', 'sensor\_10', 'sensor\_16', 'sensor\_18', 'sensor\_19', 'sensor\_22', 'sensor\_23']

Sonrasında, Hedef değişkeninin etiketlenmesi işlemi için senaryolar oluşturulmuştur. Regresyon Analizinde, tahmin edeceğimiz değişkene “hedef değişken” adı verilmektedir. Modelin girdi değişkenlerine ise, bağımsız değişken denmektedir. Tahmin değişkeninin etiketleri mevcut değil ise, etiketleme işlemi manuel olarak yapılmaktadır.

|  |  |
| --- | --- |
| No: | Etiketleme Aşaması: LABEL={0:normal,1:uyarı,2:alarm} |
| 1 | Arızalanma durumuna kadar toplam çalıştığı sayısının her gözlemle azalarak giden ve arızalandığı durunundaki gözlemin 0 olacak şekilde sürekli değerler ile etiketlenmiştir. (Örneğin, id’si 1 olan 142’den 0’a kadarki sayılar) |
| 2 | Gözlemler her saniye toplandığını varsayarsak, id’si 1 olan motorun son 5 satır(göszlem) için kalan kullanışlı ömrü {4,3,2,1,0} olup, motorun bozulmasına 4, 3, 2, 1, 0 saniye kaldığını ifade etektedir. |
| 3 | LABEL sütunununa, RUL sütünunun 30 dan az olan satırlarına 1 etiketi, 15 ten az olan satırlarına 2 etiketi atanır. |

Tablo 3. Bağımlı Değişkenin Etiketlendirilmesi.

|  |  |  |
| --- | --- | --- |
| Normal | Uyarı | Alarm |
| 30 < RUL | 15 < 30 <= RUL | 0 <= RUL <=15 |
| 0 | 1 | 2 |

Tablo 4. Bağımlı Değişkenin Sınıfları.

Şekil 5. Bağımlı Değişkenin Etikelenme Kodları.

Sonrasına, Keşifsel Veri Analizi işlemine geçtim. Bu süreçte ilk önce veri görselleştirilmesi işlemi yapılmıştır. Görselleştirmede gürültülü veriler tespit edilmiştir. Hemen ardından, Pearson korelasyon hesabı yapılmıştır ve matrisi çizdirilmiştir. Veri seti çok boyutlu olduğu için korelasyon katsayıları gözle görülemeyecek kadar küçük hücrelerden oluşan matrisi üzerinde birakaç işlem yapılmıştır. Keşifsel Veri Analizi, veri setine derin olarak bakmak, keşfetmek, analiz etmek olarak tanımlanabilir. Analiz ile veri setinin optimizasyonu yapabiliriz. Görselleştirme ve korelasyon gibi işlemler veri setindeki bozuk verilerin tespitinde büyük rol oynamanktadır. Bozuk veri modeli de bozar, bunun için verileri detaylara kadar incelemek lazım. Örneğin, A sensöründen bir noktaya kadar doğru veriler toplanıyor olup hirhangi bir sebepten dolayı yanlış veri (şüphelendirici şekilde hep aynı veri geliyor olabilir, sensör değişkeninin sadece pozitif değerlerden oluşması gerekirken negatif değerlerin alınması, vs.) alınmaya başlayabilir. Bunun gibi durumları tespit etmek gerekmektedir.

Veriyi tam olarak görebilmek için tablolar yazdırmak yetersizdir. Bunun için Matplotlib, Seaborn kütüphanelerin yardımı ile verilerin grafikleri çizdirilmiştir.

Keşigsel Veri Analizi Sonucunda tespit edilen gereksiz değişkenler model eğitimi sırasında, modeli tam tersine bozduğu için veri setinden çıkarılması işlemi gerçekleştirilmiştir.

Sonrasında, Özellik Ölçeklendirme işlemi türü olan normalizasyon yapılmıştır.

Normalizasyon, değerleri 0 ile 1 arasında olacak şelikde ölçekleyen bir özellik ölçeklendirme tekniğidir. Özelliğin normalleştirilmiş olan her bir elemanı aşağıdaki formül kullanılarak hesaplanır.



Şekil 6. Özellik Ölçeklendirme Normalizasyon Formülü.

x’ — özelliğin normalleştirilmiş elemanı,

x(min) — özelliğin en küçük elemanı,

x — normalleştirilecek olan özellik elemanı,

x(max) — özelliğin en büyük elemanı.

Bu tür ölçeklendirmenin amacı, veri setindeki sayısal sütunların değerlerini değiştirerek aralıklarındaki farklılıkların korunmasıdır. Makine Öğrenmesinde bir veri seti, özellikler farklı aralıklara sahip olduğunda ve bu nedenle Tahmin Değikenleri (Predictor Variables) ile Hedef Değişkeni (Target Variable) arasındaki ilişkinin algısını bozduğu için normalleştirme gerektirir.

Modelin eğitilmesine başlamadan önce, verimizi (satır, time steps, özellik=sütun) olarak 3 boyutlu diziye dönüştürmek gerekmektedir. Bunun sebebi, zaman serisi sınıflandırma yapacağımız için, önceki n tane gözlem ardışık olarak aynı anda öğrenilir. Öğrenilen önceki verilere göre tahminler edilir. RNN mimarisi de tam olarak bu özelliği sayesinde performanslıdır.

RNN:

* Geleneksel yapay sinir ağları bağlantı kurarak anlama özelliğine sahip değildir ve bu onun en önemli dezavantajıdır.
* “Sequence memory”, beynimizin bir dizi modelini tanıması için işi kolaylaştıran bir mekanizmadır. RNN de bu mekanizmayı kullanır ve genel olarak tahmin için kullandığı ardışık verileri (ya da dizi verilerini) işlemede iyidir.
* RNN’lerin hatırlayabilme yeteneği sayesinde son yıllar içerisinde bir dizi alanlarda inanılmaz gelişmeler yaşanmıştır. Bu gelişmelerin birçoğu da RNN’nin türü olan LSTM (Uzun Kısa-Vadeli Bellek) yapısına ait olduğu söylenmektedir.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | cycle | … | Label | RUL | Toplam çalışma sayısı: |
| **1** | **1** | **…** | **0** | **192** | **193** |
| **1** | **…** | **…** | **0** | **191** |
| **1** | **…** | **…** | **0** | **…** |
| **1** | **…** | **…** | **0** | **…** |
| **1** | **…** | **…** | **0** | **1** |
| **1** | **193** | **…** | **1** | **0** |
| **2** | **1** | **…** | **0** | **142** | **143** |
| **2** | **…** | **…** | **0** | **141** |
| **2** | **…** | **…** | **0** | **140** |
| **2** | **…** | **…** | **…** | **…** |
| **2** | **143** | **…** | **…** | **0** |
| **3** | **…** | **…** | **…** | **…** | **…** |
| **…** | **…** | **…** | **…** | **…** | **…** |
| **100** | **…** | **…** | **…** | **…** | **…** |

Tablo 5. Veriseti Kalan Kullanışlı Ömür (RUL), Bağımlı Değiken (Label) sütunları.

# id=1 satır sayısı=193

# 192-50=142

# (0,142)-->(50,192)

# 0 50 => 1 51 => 2 52 =>...=> 141 191 => 142 192

# toplam = (192-50)/1=142 tane 50'şerlik ardışık dizi elde edilir.

toplam eğitim satır sayısı== 20631.

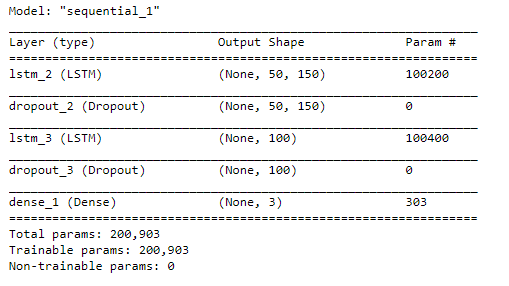
Her bir motor için ayrı ayrı ardışık dizi oluşturulacağı için satır sayı 20631’den 15631’e düşecektir.

seq\_lenth==50 --> 50\*100(motor)=5000 --> 20631-5000=15631 (satır sayısı)

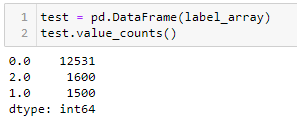
|  |  |  |
| --- | --- | --- |
| Veri: | Ardışık diziye dönüştürülmeden önce: | Ardışık diziye dönüştürüldüğünde: |
| Eğitim | (20631,16) | (15631,50,16) |
| Test | (13096,16) | (93,50,16) |

Tablo 6. Verinin Ardışık Dizi'ye Dönüştürülmesi.

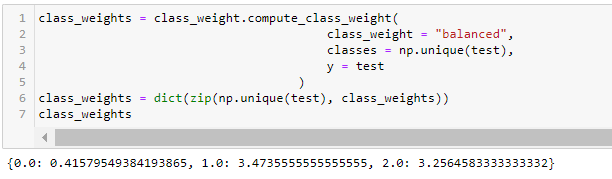
Sonrasında, LSTM modeli oluşturulur.



Şekil 7. LSTM Modeli Mimarisi.

Verideki bağımlı değişkenimizin etiketleri dengeli olmadığından, şöyle bir işlem gerçekleştirildi.

Şekil 8. Label sütununun sınıf frekansı.

 Yukarıdaki, class\_weight fonksiyonu, modelin her bir etiketi ne kadar öğrenmesini belirlemektedir.

Şekil 9. Class\_weight fonksiyonu.

En son olarak, model eğitilir. Eğitim için aşağıdaki parametreler seçilmiştir.

|  |
| --- |
| history=model.fit(seq\_array, label\_array, epochs=40, batch\_size=200, validation\_split=0.05, verbose=2, class\_weight=class\_weights,  callbacks=[keras.callbacks.EarlyStopping(monitor='val\_loss', min\_delta=0, patience=10, verbose=0, mode='min'),  keras.callbacks.ModelCheckpoint(model\_path, monitor='val\_loss', save\_best\_only=True, mode='min', verbose=0)]  ) |

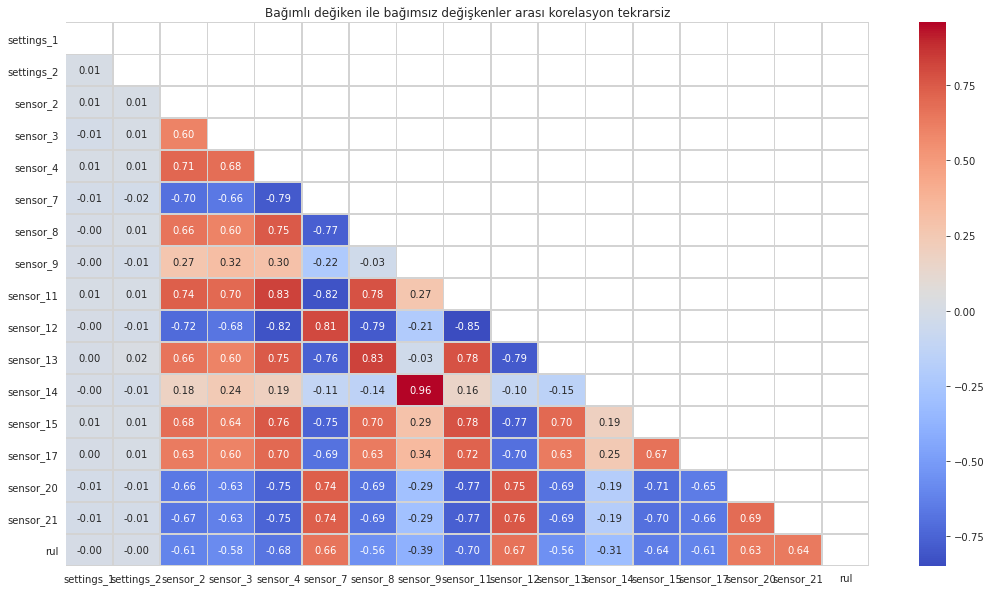
Tablo 7. Model eğitimi parametreleri.

Modeli iyileştirmek için, katman sayısı ve nöron sayısı, yukarıdaki model parametreleri üzerinde değişiklikler yapılarak, en iyisi elde edilmiştir. Sonuçlar bölümünde sonuçlar detaylı olarak açıklanmıştır.

* 1. KULLANILAN TEKNOLOJİLER

Proje Jupyter Notebook üzerinde Python dilinde, Tensorflow, Keras, Sklearn kütüphaneleri kullanılarak yazılmıştır. RNN ailesinden LSTM algoritması kullanılmıştır. Ortam olarak Anaconda kullanılmıştır.

1. SONUÇ



Şekil 10. Bağımlı değiken ile Bağımsız değişken arası korelasyon hesabı.

Seçilen Özellikler:

1. 'settings\_1',
2. 'settings\_2',
3. 'cycle\_norm'
4. 'sensor\_2',
5. 'sensor\_3',
6. 'sensor\_4',
7. 'sensor\_7',
8. 'sensor\_8',
9. 'sensor\_9',
10. 'sensor\_11',
11. 'sensor\_12',
12. 'sensor\_13',
13. 'sensor\_14',
14. 'sensor\_15',
15. 'sensor\_17',
16. 'sensor\_20'

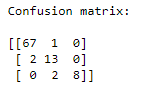
Test verisi üzerinde model değerlendirildi ve aşağıdaki sonuçlar elde edildi.

**accuracy**: 0.9677

**Precision**: 0.946236559139785

**Becall**: 0.946236559139785

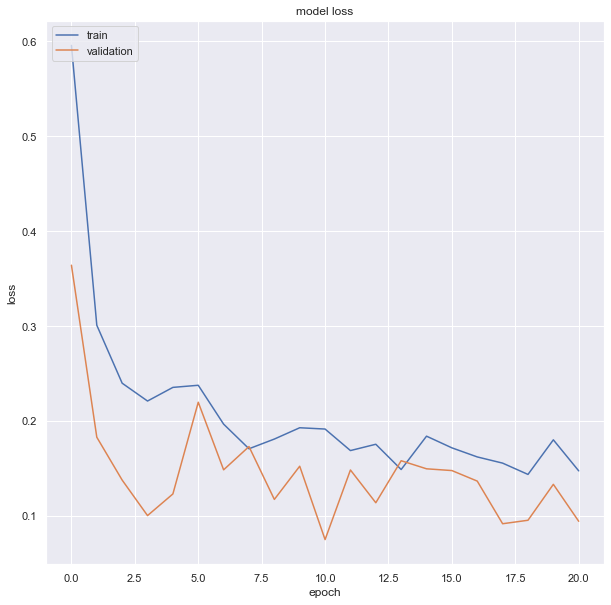
**F1-score**: 0.946236559139785



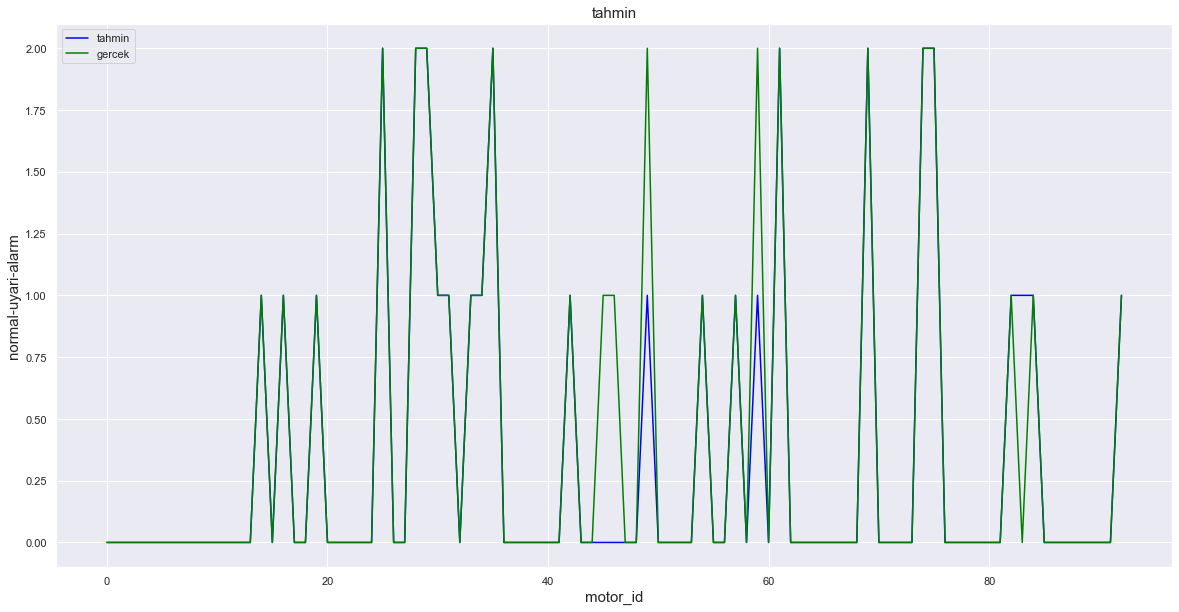
Şekil 11. Hata Matrisi



Şekil 12. Doğruluk grafiği



Şekil 13. Loss Grafiği



Şekil 14. Tahmin Edilen ve Olması Gereken değerlerin kesişme grafiği.

Şekil 14’te gözüktüğü gibi 4 yerde sapma gözükmektedir. Bu sapmalar hakkında açıklama olarak kabul edilebilir bir uyuşmazlık diyebilirim. İkinci ve üçüncü sapmalar için, durumu ‘alarm’ olan motora ‘uyarı’ durumunda olduğunu tahmin etmiştir. Eğer ‘alarm’ durumunda olan motora ‘normal’ durumunda olduğunu söyleseydi, hiç kontrol bile etmezdi. Bu durumda uyarı verildiğinde ekipmanın durumunu kontroleden çalışan motoru gözden geçirir, eğer kritik bir problem tespit ederse, yapılacak bakımları önceden yapmış olur. Oluşabilecek sorunu kestirerek, daha büyük problemleri önlemiş olur.

1. EK

**Proje Github linki: https:** [**https://github.com/meerimKubatova/BMT-436\_proje\_181816076**](https://github.com/meerimKubatova/BMT-436_proje_181816076)

1. KAYNAKÇA

[1] - <https://github.com/umbertogriffo/Predictive-Maintenance-using-LSTM/tree/master/src/lstm>