1. Veri Hazırlama, Feature Extraction ve SoC Hesaplama Stratejisi

NASA veri setlerinde döngü bazında State of Health (SoH) hesaplaması, bataryanın her test döngüsündeki kapasite değeri üzerinden nominal kapasiteye oran alınarak yapıldı.
State of Charge (SoC) ise, veri setinde başlangıç (initial) SoC değeri paylaşılmadığından dolayı klasik yöntemlerle hesaplanamadı.
Bu eksiklik, projenin erken aşamasında veri setinin analizi sırasında tespit edildi. Ancak veri setinin kendi içindeki kapasite akışından yola çıkılarak, discharge döngüleri üzerinde ilerlemeye dayalı bir SoC gösterge metriği geliştirildi.
Discharge döngülerindeki kalan kapasite, döngüler arası kümülatif toplam (cumulative sum) ile normalize edilerek, SoC'nin döngüsel ilerlemesi yüzde (%) olarak elde edildi.
Bu sayede, başlangıç SoC verisinin yokluğunda da modellemede kullanılabilir bir SoC Progress özelliği üretildi.

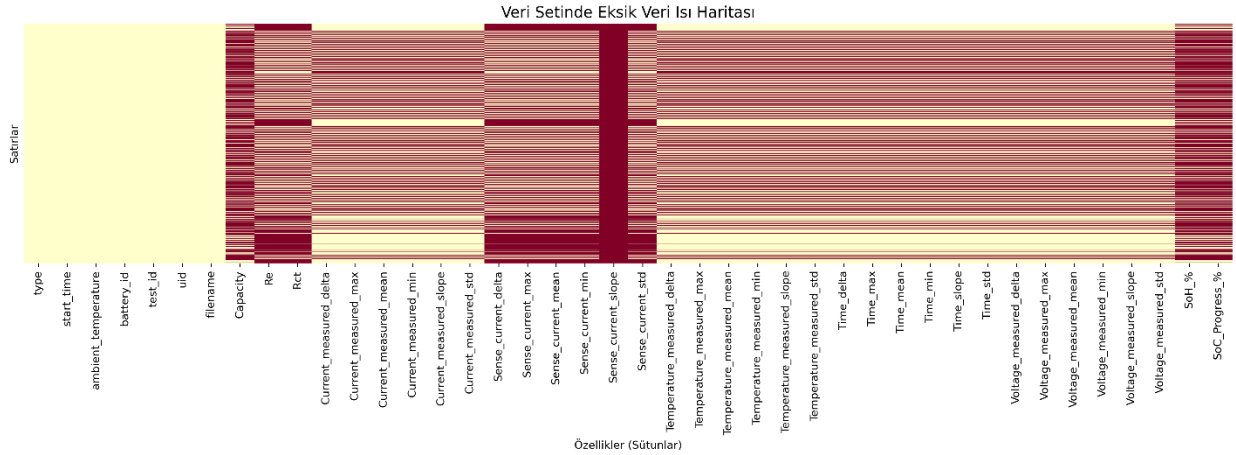
Veri hazırlama sürecinde, her bir .mat dosyasındaki tüm döngüler için, Voltage, Current, Temperature ve Time gibi temel sinyallerin ortalama, standart sapma, minimum, maksimum, delta ve eğim (slope) istatistikleri çıkarıldı.
Ayrıca cycle'a özel kapasite, iç direnç (Re), yük transfer direnci (Rct) gibi değerler de eklenerek veri setine dahil edildi.
Eksik veri oranı yüksek veya değeri sürekli sabit kalan sütunlar modellemeye ve analizlere alınmadı.
Sonuç olarak, nominal kapasiteye göre normalize edilmiş SoH ve yukarıda açıklanan alternatif yöntemle hesaplanmış SoC Progress özellikleri ile, modellemeye uygun, temiz ve bilgi yoğunluğu yüksek bir tabular veri seti oluşturuldu.

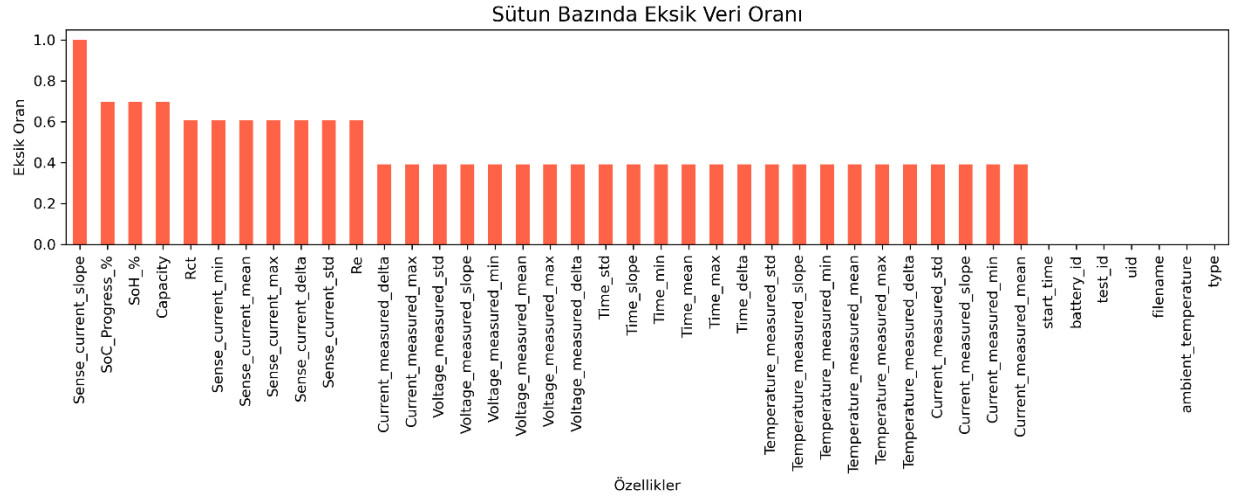
2. Feature Selection ve Analiz Mantığı

Veri setinde hangi feature'ların kullanılacağına, **eksik veri analizi**, **istatistiksel dağılım** ve **hedef değişkenlerle olan ilişki** temelinde karar verildi.
Aşağıda, çıkarılan her grafik için yapılan analiz ve seçime etki eden temel noktalar detaylıca açıklandı.

2.1 Eksik Veri Analizi (Isı Haritası ve Eksik Oran Grafiği)

İlk olarak, feature'ların veri setindeki eksik değer dağılımı, **ısı haritası** (eksik_veri_heatmap.png) ve **sütun bazlı eksik veri oranı grafiği** (eksik_veri_oranlari.png) ile incelendi.





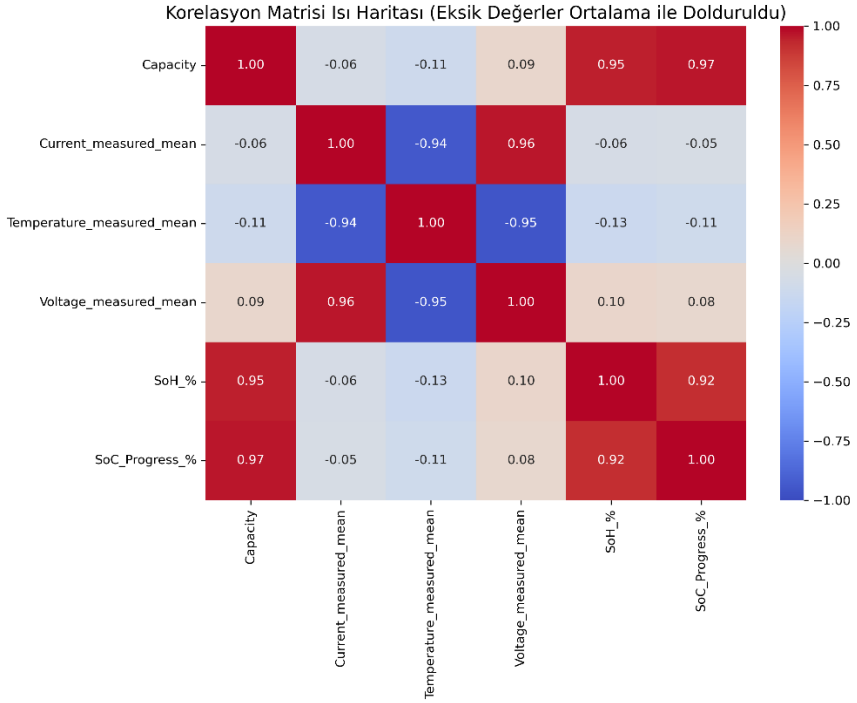
- Isı haritası, bazı feature'ların neredeyse tamamen eksik (ör: Sense_current_slope) olduğunu gösterdi.
- Eksik veri oranı grafiğinde, %50'nin üzerinde eksik olan feature'lar (ör: bazı slope, delta, min/max değerleri) modellemeye alınmadı.
- Kimlik veya meta bilgi taşıyan sütunlar (uid, test_id, battery_id, filename, type, start_time) zaten model dışı bırakıldı.

Bu analizle, modelde kullanılacak feature'lar belirlenirken **veri doluluğu ve güvenilirliği** öncelikli tutuldu.

2.2 Korelasyon Analizi (Korelasyon Isı Haritası)

Kalan feature'lar arasında, hedef değişkenlerle ve birbirleriyle olan korelasyonlar iki farklı yöntemle analiz edildi:

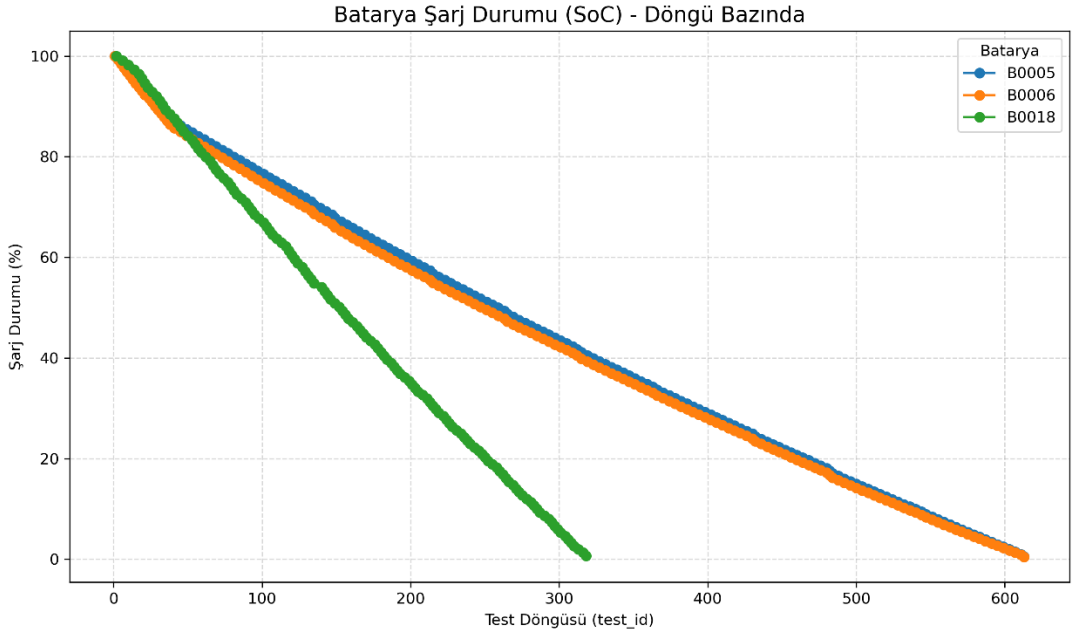
1. Eksik değerler **ortalama ile doldurulduktan sonra** korelasyon matrisi



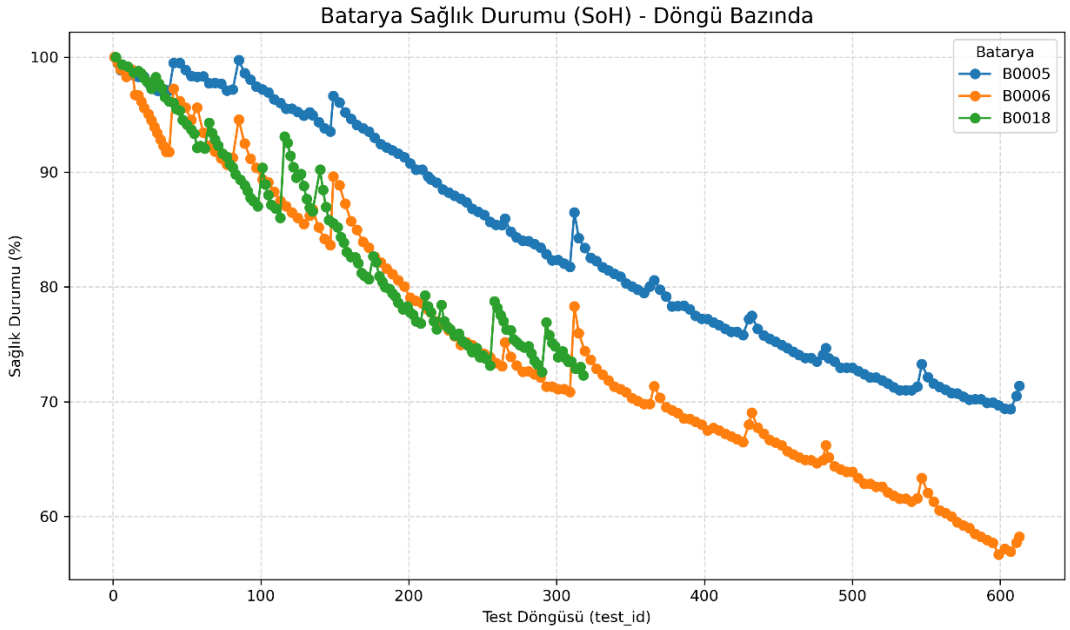
- Bu analizlerde, özellikle **Capacity, Current_measured_mean, Temperature_measured_mean, Voltage_measured_mean** gibi feature'ların SoH ve SoC_Progress ile yüksek korelasyon gösterdiği tespit edildi.
- Aralarındaki yüksek korelasyonlar, modelde multikolineeriteye yol açmaması için dikkate alındı ancak veri setinin doğası gereği domain bilgisinin önde geldiği kararlaştırıldı.

2.3 Dağılım ve Scatter Plot Analizleri

Öne çıkan feature'ların SoH ve SoC gibi hedef değişkenlerle ilişkisi, scatter plot'larla görselleştirildi.

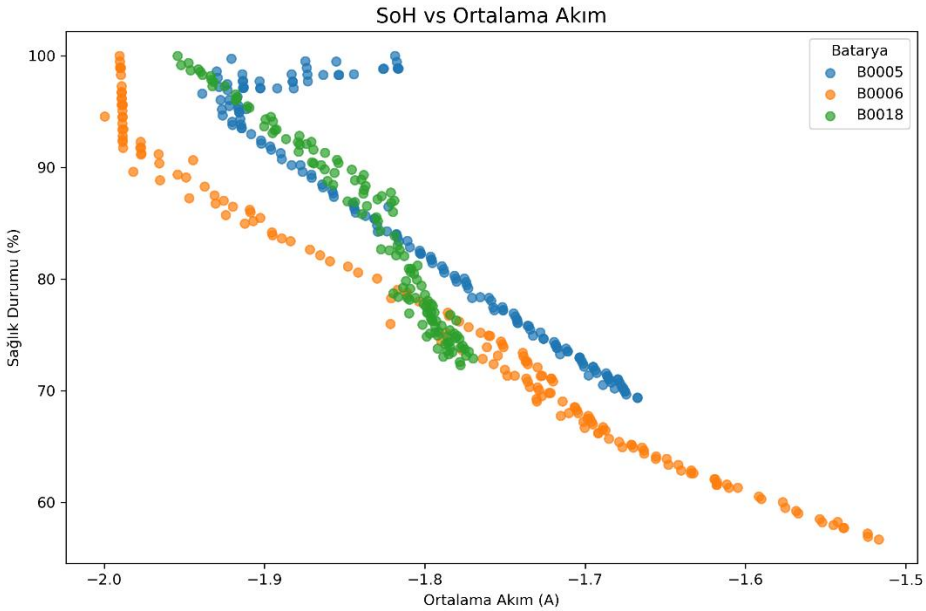


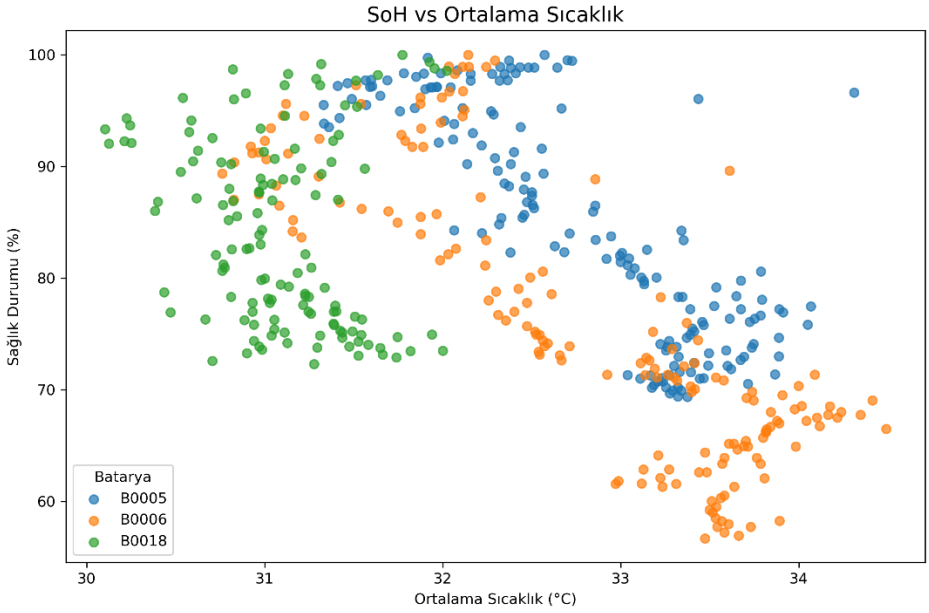
- SoC Progress'in test döngüsüne göre düşüş eğilimi, batarya kapasitesinin zamanla azaldığını ve discharge döngüsünde belirgin bir pattern olduğunu gösterdi.
- Bu, SoC Progress'in model hedefi veya ek feature olarak kullanımını destekledi.



- SoH'nin döngü sayısı arttıkça düştüğü ve bataryaların sağlık durumunun zamana bağlı olarak bozulduğu gözlemlendi.

- Bu pattern, kapasite ve SoH'nin model için anlamlı hedefler olduğunu doğruladı.





- Ortalama akım ile SoH arasında negatif, ortalama sıcaklık ile SoH arasında ise veri setine göre değişen ilişki görüldü.
- Özellikle yüksek akımın ve sıcaklığın, batarya sağlığına olumsuz etkisi grafiklerle de doğrulandı.

2.4 Özellik Seçimi Sonucunda Modelde Kullanılanlar

Yukarıdaki analizler sonucunda, aşağıdaki değişkenler modelde kullanılmaya uygun bulundu:

- **Capacity**
- **Current_measured_mean**
- **Temperature_measured_mean**
- **Voltage_measured_mean**
- **Re**
- **Rct**
- **SoC_Progress_%** (alternatif yöntemle elde edilen, yukarıda açıklandığı şekilde hesaplandı)

Eksik veri oranı yüksek veya düşük varyansa sahip diğer feature'lar modellemeye alınmadı.

2.5 Teknik ve Domain Yorum

Her grafik çıkarıldıktan sonra yapılan analizler,

- Feature selection sürecinde sadece sayısal kriterlerin değil, **domain bilgisinin** de öne çıktığını gösterdi.
- Özellikle batarya sağlığı ve şarj durumu üzerinde etkili olduğu bilinen temel fiziksel parametreler (kapasite, akım, sıcaklık, voltaj, iç direnç) **modelde öncelikli olarak yer aldı**.
- Ayrıca korelasyon ve scatter plot'lar, seçilen feature'ların hedef değişkenlerle istatistiksel olarak da ilişkili olduğunu ortaya koydu.

3. Model Seçimi, ML-DL Karşılaştırması ve Sonuçların Değerlendirilmesi

Tabular veri üzerinde model seçimi yaparken, örnek sayısının azlığı, veri yapısının karmaşıklık düzeyi ve elde edilmek istenen sonuçların deterministik mi yoksa stochastic mı olacağı belirleyici olur.
Karmaşık olmayan, örnek sayısı sınırlı, ancak çoklu özniteliğe sahip veri setlerinde, klasik makine öğrenmesi yöntemleri (örneğin XGBoost, LightGBM) yüksek başarı, hızlı eğitim ve tutarlı çıktı sunar. Bu algoritmalar, boosting mekanizması sayesinde, fazla örnek olmayan ve yüksek boyutlu ortamlarda bile overfitting'i minimalde tutar, eğitim-tahmin süreçlerinde deterministik davranır.

Derin öğrenme tabanlı yaklaşımlar (ör: Tabular MLP) ise, veri seti yeterince büyük ve heterojen olmadıkça, çoğunlukla yavaş eğitim ve yüksek stochastic karakter sunar. Özellikle bu projede, DL modelinin düşük örnek sayısı nedeniyle hem yavaş, hem de tutarsız ve değişken sonuçlar verdiği açıkça görüldü.
DL'in stochastic doğası, aynı veri ve aynı parametrelerle bile, tekrarlanamayan ve stabil olmayan skorlar üretti.
Bu nedenle, derin öğrenme modeli burada "benchmark" veya "alternatif" nitelikte test edildi, fakat öngörücü ve güvenilir model olarak kullanılmadı.

Veri setinin yapısı ve örnek sayısının kısıtlı olması nedeniyle, model seçiminde istatistiksel olarak anlamlı ve güvenilir sonuçlar verecek yöntemler tercih edildi.
Küçük veri setlerinde, çoklu model karşılaştırmalarının anlamlı bir genel başarı ölçüsü sunmadığı teknik literatürde de bilinmektedir.

Bu nedenle, model geliştirme sürecinde, domain bilgisinin ve istatistiksel güvenilirliğin ön planda tutulduğu, doğrulanabilir ve açıklanabilir modellerin uygulanmasına odaklanıldı.

Sonuç olarak, **beklendiği gibi klasik ML modelleri (özellikle LightGBM ve XGBoost), düşük hata ve yüksek tutarlılıkla öne çıktı; DL modeli ise bu özel veri yapısında daha yavaş ve daha az stabil skorlar üretti**.

Hedef	Model	MAE	RMSE	Eğitim Süresi	Tahmin Süresi	Model Dosyası
SoH_%	**XGBoost**	**3.8628**	**4.1571**	**0.37s**	**0.013s**	**xgboost_model_SoH_%_discharge.pkl**
SoH_%	**LightGBM**	**3.3747**	**3.9044**	**0.13s**	**0.003s**	**lightgbm_model_SoH_%_discharge.pkl**
SoC_Progress_%	**XGBoost**	**4.7944**	**6.0708**	**0.36s**	**0.010s**	**xgboost_model_SoC_Progress_%_discharge.pkl**
SoC_Progress_%	**LightGBM**	**4.9706**	**6.0901**	**0.10s**	**0.004s**	**lightgbm_model_SoC_Progress_%_discharge.pkl**

Hedef	MAE	RMSE	Eğitim Süresi	Tahmin Süresi	Model Dosyası	Scaler Dosyası
SoH_%	3.0119	3.7162	5.03s	0.000s	mlp_regressor_SoH_%_discharge.pt	scaler_SoH_%_discharge.pkl
SoC_Progress_%	5.4924	7.4903	5.67s	0.000s	mlp_regressor_SoC_Progress_%_discharge.pt	scaler_SoC_Progress_%_discharge.pkl

4. Dağıtım, API ve Arayüz (Docker, FastAPI, Streamlit)

Projede, farklı servislerin uyumlu ve güvenli şekilde aynı ortamda çalışmasını sağlamak için, Docker tabanlı bir dağıtım altyapısı kullanıldı.
Her bir servis (API backend ve Streamlit arayüz) ayrı Docker container olarak yapılandırıldı. docker-compose ile servisler tek komutla ayağa kaldırıldı.
API backend, FastAPI framework'üyle, eğitilmiş model dosyasını yükleyip /predict endpoint'i üzerinden tahmin servis etti.
Streamlit arayüzü ise, kullanıcıdan alınan CSV dosyasını API'ya göndererek tahmin sürecini başlattı ve sonuçları görsel olarak sundu.
Bu teknik yapı, taşınabilirlik, tekrar üretilebilirlik ve hızlı entegrasyon ihtiyaçlarına tam olarak karşılık verdi.