PROJE RAPORU

İnovasyon Mühendislik Ltd. Şti.

Proje Adı: Lithium-Ion Batarya SoH Analizi

Danışman

Okan Ulu

Hazırlayan

Hasan Taşkın

29.07.2024

Eskişehir

İçindekiler

1. Giriş

Bu proje, lityum-iyon bataryaların sağlık durumu (SoH) tahmin modelinin geliştirilmesini ve performanslarının değerlendirilmesini hedeflemektedir. Bataryaların etkili ve verimli bir şekilde kullanılmasını destekleyerek enerji depolama teknolojilerinin gelişimine katkıda bulunmayı amaçlamaktadır. Proje süreci, veri ön işleme, keşifsel veri analizi (EDA), model geliştirme ve performans değerlendirme aşamalarını içermektedir.

2. Veri Seti ve Ön İşleme

2.1 Veri Seti

Bu projede kullanılan veri setleri, üç farklı lithium-ion bataryaya ait performans verilerini içermektedir: B0005, B0006, ve B0018. Her bir veri seti, bataryaların kapasite, döngü sayısı, akım, voltaj ve diğer ilgili parametrelerini içermektedir.

2.2 Veri Yükleme ve İnceleme

Veri setleri, pandas kütüphanesi kullanılarak CSV dosyalarından okunur ve incelenir. Bu adımda, veri setlerinin ilk ve son birkaç satırı gösterilerek verilerin yapısı hakkında bilgi edinilir.

[107]:
```
#Gerekli kutuphanelerin yuklenmesi
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
```

[108]:
```
# Veri setlerini okuma
b0005_data = pd.read_csv("C:\\Users\\hasan\\OneDrive\\Masaüstü\\staj\\orjinal_veri\\B0005.csv")
b0006_data = pd.read_csv("C:\\Users\\hasan\\OneDrive\\Masaüstü\\staj\\orjinal_veri\\B0006.csv")
b0018_data = pd.read_csv("C:\\Users\\hasan\\OneDrive\\Masaüstü\\staj\\orjinal_veri\\B0018.csv")

# İlk ve son 5 satiri gosterme
b0005_head = b0005_data.head(-5)
b0006_head = b0006_data.head(-5)
b0018_head = b0018_data.head(-5)

b0005_head, b0006_head, b0018_head
```

```
(       cycle  ambient_temperature            datetime  capacity  \
0          1                   24 2008-04-02 15:25:41  1.856487
1          1                   24 2008-04-02 15:25:41  1.856487
2          1                   24 2008-04-02 15:25:41  1.856487
3          1                   24 2008-04-02 15:25:41  1.856487
4          1                   24 2008-04-02 15:25:41  1.856487
...      ...                  ...                 ...       ...
50275    168                   24 2008-05-27 20:45:42  1.325079
50276    168                   24 2008-05-27 20:45:42  1.325079
50277    168                   24 2008-05-27 20:45:42  1.325079
50278    168                   24 2008-05-27 20:45:42  1.325079
50279    168                   24 2008-05-27 20:45:42  1.325079

       voltage_measured  current_measured  temperature_measured  current_load  \
0              4.191492         -0.004902             24.330034       -0.0006
1              4.190749         -0.001478             24.325993       -0.0006
2              3.974871         -2.012528             24.389085       -1.9982
3              3.951717         -2.013979             24.544752       -1.9982
4              3.934352         -2.011144             24.731385       -1.9982
...                 ...               ...                   ...           ...
50275          3.563350         -0.000948             35.623242        0.0006
50276          3.566589          0.000416             35.479866        0.0006
50277          3.570132         -0.000338             35.345455        0.0006
50278          3.573139          0.001471             35.171253        0.0006
50279          3.576159          0.001138             34.966434        0.0006

       voltage_load      time
0             0.000     0.000
1             4.206    16.781
2             3.062    35.703
3             3.030    53.781
4             3.011    71.922
...             ...       ...
50275         0.000  2732.359
50276         0.000  2742.093
50277         0.000  2751.843
50278         0.000  2761.687
50279         0.000  2771.500

[50280 rows x 10 columns],
       cycle  ambient_temperature            datetime  capacity  \
0          1                   24 2008-04-02 15:25:41  2.035338
1          1                   24 2008-04-02 15:25:41  2.035338
2          1                   24 2008-04-02 15:25:41  2.035338
3          1                   24 2008-04-02 15:25:41  2.035338
4          1                   24 2008-04-02 15:25:41  2.035338
...      ...                  ...                 ...       ...
50275    168                   24 2008-05-27 20:45:42  1.185675
50276    168                   24 2008-05-27 20:45:42  1.185675
50277    168                   24 2008-05-27 20:45:42  1.185675
50278    168                   24 2008-05-27 20:45:42  1.185675
50279    168                   24 2008-05-27 20:45:42  1.185675

       voltage_measured  current_measured  temperature_measured  current_load  \
0              4.179800         -0.002366             24.277568       -0.0006
1              4.179823          0.000434             24.277073       -0.0006
2              3.966528         -2.014242             24.366226       -1.9990
3              3.945886         -2.008730             24.515123       -1.9990
4              3.930354         -2.013381             24.676053       -1.9990
...                 ...               ...                   ...           ...
50275          3.685923         -0.003033             32.912466        0.0006
50276          3.686512         -0.003900             32.820743        0.0006
50277          3.687451         -0.004003             32.682428        0.0006
50278          3.688081         -0.004603             32.507369        0.0006
50279          3.689038         -0.001839             32.595132        0.0006

       voltage_load      time
0             0.000     0.000
1             4.195    16.781
2             3.070    35.703
3             3.045    53.781
4             3.026    71.922
...             ...       ...
50275         0.000  2732.359
50276         0.000  2742.093
50277         0.000  2751.843
50278         0.000  2761.687
50279         0.000  2771.500

[50280 rows x 10 columns],
       cycle  ambient_temperature            datetime  capacity  \
0          1                   24 2008-07-07 15:15:28  1.855005
1          1                   24 2008-07-07 15:15:28  1.855005
2          1                   24 2008-07-07 15:15:28  1.855005
3          1                   24 2008-07-07 15:15:28  1.855005
4          1                   24 2008-07-07 15:15:28  1.855005
...      ...                  ...                 ...       ...
34856    132                   24 2008-08-20 08:37:19  1.341051
34857    132                   24 2008-08-20 08:37:19  1.341051
34858    132                   24 2008-08-20 08:37:19  1.341051
34859    132                   24 2008-08-20 08:37:19  1.341051
34860    132                   24 2008-08-20 08:37:19  1.341051

       voltage_measured  current_measured  temperature_measured  current_load  \
0              4.188109          0.000131             23.819520        0.0006
1              4.188196          0.001459             23.828807        0.0006
2              3.977432         -2.005672             23.844944        1.9988
3              3.961974         -2.012206             23.925577        1.9988
4              3.949835         -2.012005             24.010628        1.9988
...                 ...               ...                   ...           ...
34856          3.382468         -0.003531             36.486526        0.0006
34857          3.397079         -0.000974             36.255584        0.0006
34858          3.410563         -0.001955             36.038249        0.0006
34859          3.422558         -0.000755             35.810559        0.0006
34860          3.433480          0.003082             35.593808        0.0006

       voltage_load      time
0             0.000     0.000
1             4.203     9.422
2             3.029    19.578
3             3.026    29.016
4             3.015    38.485
...             ...       ...
34856         0.000  2615.562
34857         0.000  2629.781
34858         0.000  2643.921
34859         0.000  2658.109
34860         0.000  2672.265

[34861 rows x 10 columns])
```

2.3 Eksik Veri Kontrolü

Veri setlerinde eksik veri olup olmadığı kontrol edilir. Eksik veri analizi, verinin kalitesini ve modelleme sürecine olan etkisini anlamak için gereklidir. Eksik veri kontrolü, isnull().sum() fonksiyonu kullanılarak yapılır.

```
[109]: # Eksik veri kontrolu
missing_values_b0005 = b0005_data.isnull().sum()
missing_values_b0006 = b0006_data.isnull().sum()
missing_values_b0018 = b0018_data.isnull().sum()

missing_values_b0005, missing_values_b0006, missing_values_b0018
```

```
[109]: (cycle                  0
 ambient_temperature    0
 datetime               0
 capacity               0
 voltage_measured       0
 current_measured       0
 temperature_measured   0
 current_load           0
 voltage_load           0
 time                   0
 dtype: int64,
 cycle                  0
 ambient_temperature    0
 datetime               0
 capacity               0
 voltage_measured       0
 current_measured       0
 temperature_measured   0
 current_load           0
 voltage_load           0
 time                   0
 dtype: int64,
 cycle                  0
 ambient_temperature    0
 datetime               0
 capacity               0
 voltage_measured       0
 current_measured       0
 temperature_measured   0
 current_load           0
 voltage_load           0
 time                   0
 dtype: int64)
```

2.4 SoH (State of Health) Değerinin Hesaplanması

SoH (State of Health), bataryanın sağlık durumunu yüzdelik olarak gösterir. Bu değerin hesaplanması için öncelikle her bataryanın nominal kapasitesi belirlenir. Nominal kapasite, bir bataryanın tasarım gereği maksimum kapasitesini temsil eder ve bataryanın tamamen şarj edildiğinde sağlayabileceği enerji miktarını ifade eder. Genellikle ampere-saat (Ah) cinsinden ölçülür.

Bataryanın kullanım ömrü boyunca kapasitesi azalabilir, bu nedenle SoH hesaplamalarında nominal kapasite referans olarak kullanılır. SoH, bataryanın mevcut kapasitesinin nominal kapasiteye oranı olarak hesaplanır ve yüzde olarak ifade edilir. Örneğin, bir bataryanın nominal kapasitesi 2 Ah ise ve mevcut kapasitesi 1.5 Ah ise, SoH %75 olur.

Veri setinizde nominal kapasiteyi belirlemek için, veri setindeki maksimum kapasite değerini kullanırız çünkü bu değer, genellikle bataryanın yeni durumdayken sahip olduğu kapasiteyi temsil eder.

Aşağıda bataryaların SoH hesaplaması yapılmış ve verilerin son hali incelenmiştir.

[110]:

```python
# SoH degeri hesaplama
# Nominal kapasiteyi belirleme
nominal_capacity_b0005 = b0005_data['capacity'].max()
nominal_capacity_b0006 = b0006_data['capacity'].max()
nominal_capacity_b0018 = b0018_data['capacity'].max()

# SoH hesaplama fonksiyonu
def calculate_soh(df, nominal_capacity):
    df['SoH'] = (df['capacity'] / nominal_capacity) * 100
    return df

# SoH hesaplamalari
b0005_data_soh = calculate_soh(b0005_data.copy(), nominal_capacity_b0005)
b0006_data_soh = calculate_soh(b0006_data.copy(), nominal_capacity_b0006)
b0018_data_soh = calculate_soh(b0018_data.copy(), nominal_capacity_b0018)

# İlk ve son 5 satiri gosterme
b0005_data_soh.head(-5), b0006_data_soh.head(-5), b0018_data_soh.head(-5)
```

```
[110]: (       cycle  ambient_temperature            datetime  capacity  \
       0          1                   24 2008-04-02 15:25:41  1.856487
       1          1                   24 2008-04-02 15:25:41  1.856487
       2          1                   24 2008-04-02 15:25:41  1.856487
       3          1                   24 2008-04-02 15:25:41  1.856487
       4          1                   24 2008-04-02 15:25:41  1.856487
       ...      ...                  ...                 ...       ...
       50275    168                   24 2008-05-27 20:45:42  1.325079
       50276    168                   24 2008-05-27 20:45:42  1.325079
       50277    168                   24 2008-05-27 20:45:42  1.325079
       50278    168                   24 2008-05-27 20:45:42  1.325079
       50279    168                   24 2008-05-27 20:45:42  1.325079

              voltage_measured  current_measured  temperature_measured  current_load  \
       0              4.191492         -0.004902             24.330034       -0.0006
       1              4.190749         -0.001478             24.325993       -0.0006
       2              3.974871         -2.012528             24.389085       -1.9982
       3              3.951717         -2.013979             24.544752       -1.9982
       4              3.934352         -2.011144             24.731385       -1.9982
       ...                 ...               ...                   ...           ...
       50275          3.563350         -0.000948             35.623242        0.0006
       50276          3.566589          0.000416             35.479866        0.0006
       50277          3.570132         -0.000338             35.345455        0.0006
       50278          3.573139          0.001471             35.171253        0.0006
       50279          3.576159          0.001138             34.966434        0.0006

              voltage_load      time         SoH
       0             0.000     0.000  100.000000
       1             4.206    16.781  100.000000
       2             3.062    35.703  100.000000
       3             3.030    53.781  100.000000
       4             3.011    71.922  100.000000
       ...             ...       ...         ...
       50275         0.000  2732.359   71.375616
       50276         0.000  2742.093   71.375616
       50277         0.000  2751.843   71.375616
       50278         0.000  2761.687   71.375616
       50279         0.000  2771.500   71.375616

       [50280 rows x 11 columns],
              cycle  ambient_temperature            datetime  capacity  \
       0          1                   24 2008-04-02 15:25:41  2.035338
       1          1                   24 2008-04-02 15:25:41  2.035338
       2          1                   24 2008-04-02 15:25:41  2.035338
       3          1                   24 2008-04-02 15:25:41  2.035338
       4          1                   24 2008-04-02 15:25:41  2.035338
       ...      ...                  ...                 ...       ...
       50275    168                   24 2008-05-27 20:45:42  1.185675
       50276    168                   24 2008-05-27 20:45:42  1.185675
       50277    168                   24 2008-05-27 20:45:42  1.185675
       50278    168                   24 2008-05-27 20:45:42  1.185675
       50279    168                   24 2008-05-27 20:45:42  1.185675

              voltage_measured  current_measured  temperature_measured  current_load  \
       0              4.179800         -0.002366             24.277568       -0.0006
       1              4.179823          0.000434             24.277073       -0.0006
       2              3.966528         -2.014242             24.366226       -1.9990
       3              3.945886         -2.008730             24.515123       -1.9990
       4              3.930354         -2.013381             24.676053       -1.9990
       ...                 ...               ...                   ...           ...
       50275          3.685923         -0.003033             32.912466        0.0006
       50276          3.686512         -0.003900             32.820743        0.0006
       50277          3.687451         -0.004003             32.682428        0.0006
       50278          3.688081         -0.004603             32.507369        0.0006
       50279          3.689038         -0.001839             32.595132        0.0006

              voltage_load      time         SoH
       0             0.000     0.000  100.000000
       1             4.195    16.781  100.000000
       2             3.070    35.703  100.000000
       3             3.045    53.781  100.000000
       4             3.026    71.922  100.000000
       ...             ...       ...         ...
       50275         0.000  2732.359   58.254475
       50276         0.000  2742.093   58.254475
       50277         0.000  2751.843   58.254475
       50278         0.000  2761.687   58.254475
       50279         0.000  2771.500   58.254475

       [50280 rows x 11 columns],
              cycle  ambient_temperature            datetime  capacity  \
       0          1                   24 2008-07-07 15:15:28  1.855005
       1          1                   24 2008-07-07 15:15:28  1.855005
       2          1                   24 2008-07-07 15:15:28  1.855005
       3          1                   24 2008-07-07 15:15:28  1.855005
       4          1                   24 2008-07-07 15:15:28  1.855005
       ...      ...                  ...                 ...       ...
       34856    132                   24 2008-08-20 08:37:19  1.341051
       34857    132                   24 2008-08-20 08:37:19  1.341051
       34858    132                   24 2008-08-20 08:37:19  1.341051
       34859    132                   24 2008-08-20 08:37:19  1.341051
       34860    132                   24 2008-08-20 08:37:19  1.341051

              voltage_measured  current_measured  temperature_measured  current_load  \
       0              4.188109          0.000131             23.819520        0.0006
       1              4.188196          0.001459             23.828807        0.0006
       2              3.977432         -2.005672             23.844944        1.9988
       3              3.961974         -2.012206             23.925577        1.9988
       4              3.949835         -2.012005             24.010628        1.9988
       ...                 ...               ...                   ...           ...
       34856          3.382468         -0.003531             36.486526        0.0006
       34857          3.397079         -0.000974             36.255584        0.0006
       34858          3.410563         -0.001955             36.038249        0.0006
       34859          3.422558         -0.000755             35.810559        0.0006
       34860          3.433480          0.003082             35.593808        0.0006

              voltage_load      time         SoH
       0             0.000     0.000  100.000000
       1             4.203     9.422  100.000000
       2             3.029    19.578  100.000000
       3             3.026    29.016  100.000000
       4             3.015    38.485  100.000000
       ...             ...       ...         ...
       34856         0.000  2615.562   72.293702
       34857         0.000  2629.781   72.293702
       34858         0.000  2643.921   72.293702
       34859         0.000  2658.109   72.293702
       34860         0.000  2672.265   72.293702

       [34861 rows x 11 columns])
```

2.5 Gürültülü Verilerin Temizlenmesi (IQR Yöntemi)

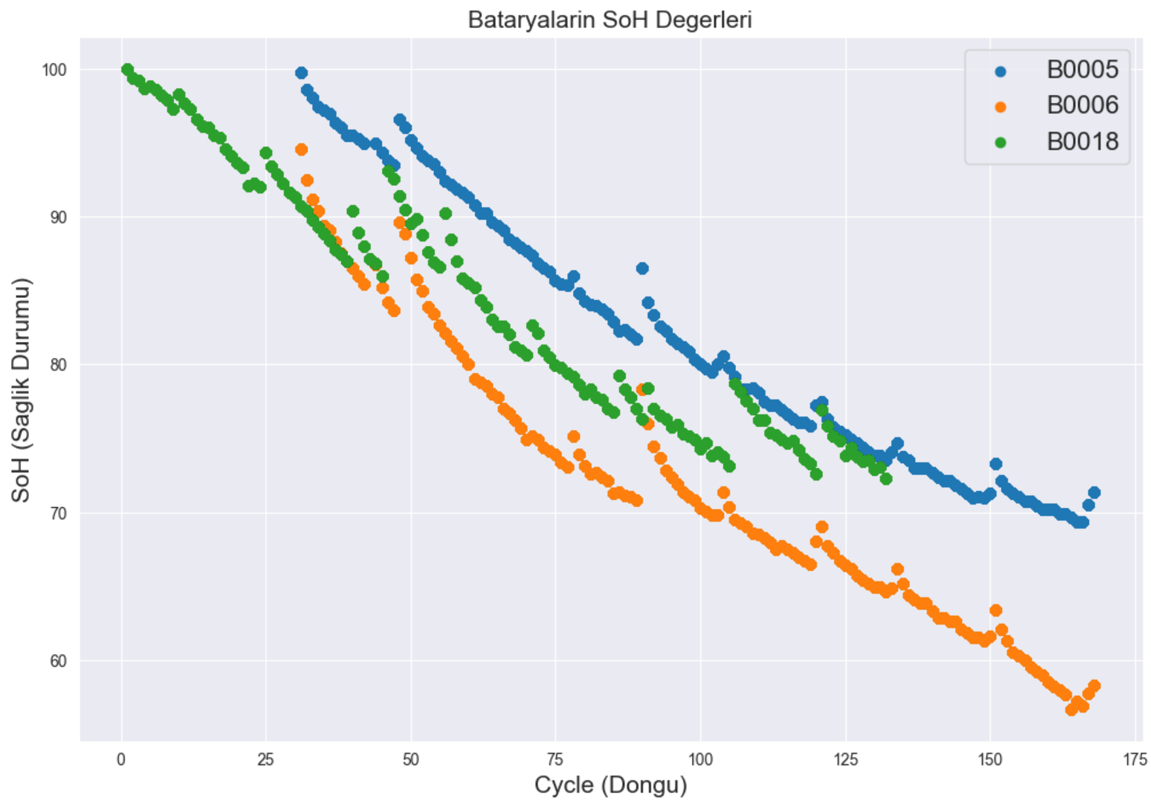
Gürültülü veriler, veri setlerinde normal dağılımdan sapmış anormal değerlerdir. Bu tür veriler modelin performansını olumsuz etkileyebilir. IQR yöntemi, bu tür verilerin temizlenmesinde kullanılan yaygın bir yöntemdir. Bu yöntemde, veri setinin birinci ve üçüncü çeyrek değerleri (Q1 ve Q3) hesaplanır ve aykırı değerler bu aralığın dışındaki değerler olarak tanımlanır.

```
[111]: # Gurultulu verilerin temizlenmesi --> IQR Yontemi
       def remove_outliers_iqr(df):
           numeric_df = df.select_dtypes(include=[np.number])
           Q1 = numeric_df.quantile(0.25)
           Q3 = numeric_df.quantile(0.75)
           IQR = Q3 - Q1
           filter = ~((numeric_df < (Q1 - 1.5 * IQR)) | (numeric_df > (Q3 + 1.5 * IQR))).any(axis=1)
           return df[filter]

       # Temizlenmis veri setleri
       b0005_clean_iqr = remove_outliers_iqr(b0005_data_soh)
       b0006_clean_iqr = remove_outliers_iqr(b0006_data_soh)
       b0018_clean_iqr = remove_outliers_iqr(b0018_data_soh)

       # Temizlenmis veri setlerinin boyutlarini kontrol etme
       b0005_clean_iqr.shape, b0006_clean_iqr.shape, b0018_clean_iqr.shape

[111]: ((37956, 11), (37303, 11), (30721, 11))
```

2.6 SoH Değerlerinin Görselleştirilmesi

Veri görselleştirme, veri analizinin önemli bir parçasıdır. Bu adımda, her üç bataryanın SoH değerleri bir scatter plot kullanılarak görselleştirilir. Bu grafik, bataryaların döngü sayısına göre SoH değişimini gösterir.

[112]:
```
# Uc bataryanın SoH degerlerini tek bir grafikte gosterme
sns.set_style("darkgrid")
plt.figure(figsize=(12, 8))

# B0005 Bataryasi
plt.scatter(b0005_clean_iqr['cycle'], b0005_clean_iqr['SoH'], label='B0005')

# B0006 Bataryasi
plt.scatter(b0006_clean_iqr['cycle'], b0006_clean_iqr['SoH'], label='B0006')

# B0018 Bataryasi
plt.scatter(b0018_clean_iqr['cycle'], b0018_clean_iqr['SoH'], label='B0018')

plt.legend(prop={'size': 16})
plt.xlabel('Cycle (Dongu)', fontsize=15)
plt.ylabel('SoH (Saglik Durumu)', fontsize=15)
plt.title('Bataryalarin SoH Degerleri', fontsize=15)
plt.show()
```



2.7 Veri Normalizasyonu

Veri setlerindeki sayısal değerler MinMaxScaler kullanılarak 0 ile 1 arasında ölçeklendirilir. Normalizasyon, verilerin belirli bir ölçeğe indirgenmesi işlemi olup, model eğitiminde verilerin daha iyi performans göstermesini sağlar.

[113]:
```python
# MinMaxScaler'i baslatma
scaler = MinMaxScaler()

# Normalizasyon fonksiyonu
def normalize_dataset(df):
    numeric_columns = df.select_dtypes(include=[np.number]).columns
    df[numeric_columns] = scaler.fit_transform(df[numeric_columns])
    return df

# Veri setlerini normallestirme
b0005_normalized = normalize_dataset(b0005_clean_iqr.copy())
b0006_normalized = normalize_dataset(b0006_clean_iqr.copy())
b0018_normalized = normalize_dataset(b0018_clean_iqr.copy())

# İlk birkac satriri gosterme
b0005_normalized.head(), b0006_normalized.head(), b0018_normalized.head()
```

```
cycle  ambient_temperature            datetime  capacity  \
 0.0                  0.0  2008-04-22 15:33:49       1.0
 0.0                  0.0  2008-04-22 15:33:49       1.0
 0.0                  0.0  2008-04-22 15:33:49       1.0
 0.0                  0.0  2008-04-22 15:33:49       1.0
 0.0                  0.0  2008-04-22 15:33:49       1.0

voltage_measured  current_measured  temperature_measured  current_load  \
        0.989625          0.647758              0.033105           0.5
        0.974667          0.262381              0.036503           0.5
        0.963097          0.374477              0.041727           0.5
        0.953527          0.339513              0.047431           0.5
        0.944897          0.682359              0.053014           1.0

voltage_load      time  SoH
    0.999005  0.000034  1.0
    0.985075  0.002921  1.0
    0.973134  0.005799  1.0
    0.964179  0.008706  1.0
    0.957214  0.011583  1.0  ,
```

B0005

```
cycle  ambient_temperature            datetime  capacity  \
 0.0                  0.0  2008-04-22 15:33:49       1.0
 0.0                  0.0  2008-04-22 15:33:49       1.0
 0.0                  0.0  2008-04-22 15:33:49       1.0
 0.0                  0.0  2008-04-22 15:33:49       1.0
 0.0                  0.0  2008-04-22 15:33:49       1.0

voltage_measured  current_measured  temperature_measured  current_load  \
        0.999984          0.198419              0.029182           0.5
        0.986588          0.488170              0.033551           0.5
        0.976350          0.227512              0.037598           0.5
        0.967818          0.567347              0.042899           0.5
        0.960236          0.436611              0.047219           0.5

voltage_load      time  SoH
    1.000000  0.000032  1.0
    0.984848  0.002783  1.0
    0.973262  0.005524  1.0
    0.964349  0.008293  1.0
    0.952763  0.011035  1.0  ,
```

B0006

```
cycle  ambient_temperature            datetime  capacity  \
 0.0                  0.0  2008-07-07 15:15:28       1.0
 0.0                  0.0  2008-07-07 15:15:28       1.0
 0.0                  0.0  2008-07-07 15:15:28       1.0
 0.0                  0.0  2008-07-07 15:15:28       1.0
 0.0                  0.0  2008-07-07 15:15:28       1.0

voltage_measured  current_measured  temperature_measured  current_load  \
        0.981370          0.770876              0.097408      0.666667
        0.966418          0.092755              0.102743      0.666667
        0.954677          0.113632              0.108370      0.666667
        0.944729          0.603454              0.114494      0.666667
        0.935930          0.390137              0.120344      0.666667

voltage_load      time  SoH
    0.972864  0.000000  1.0
    0.969849  0.002894  1.0
    0.958794  0.005797  1.0
    0.948744  0.008685  1.0
    0.940704  0.011574  1.0  )
```

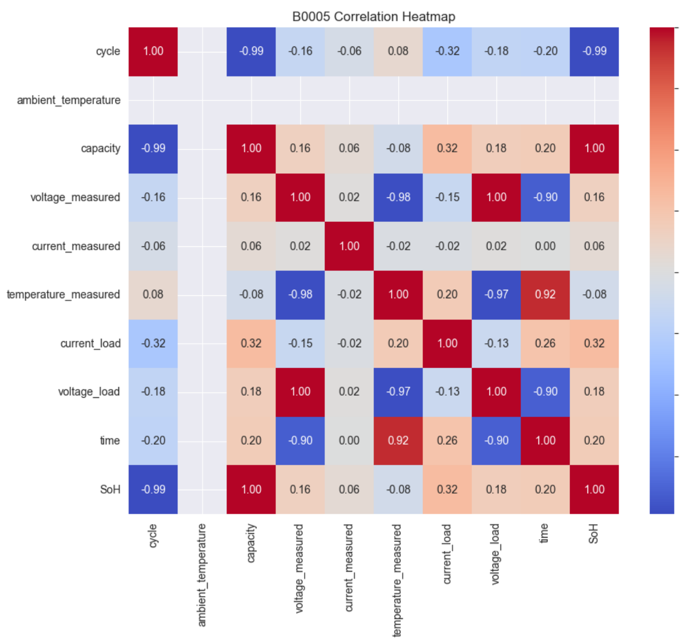
B0018

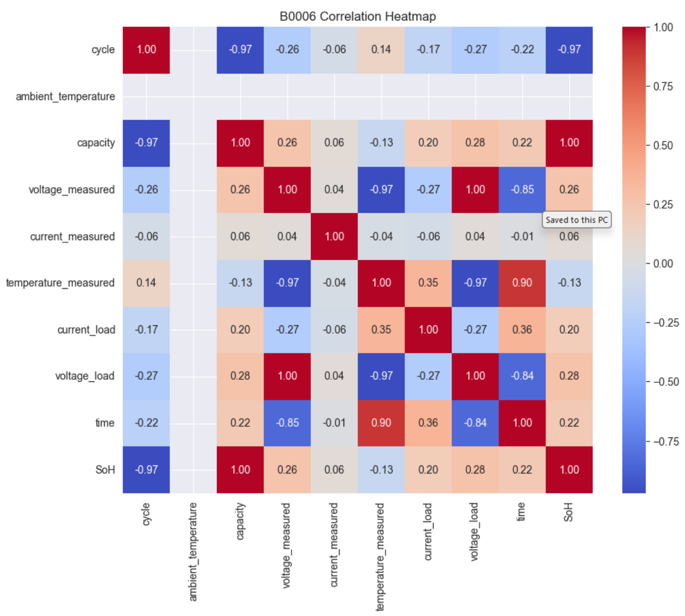
3. Keşifsel Veri Analizi (EDA)

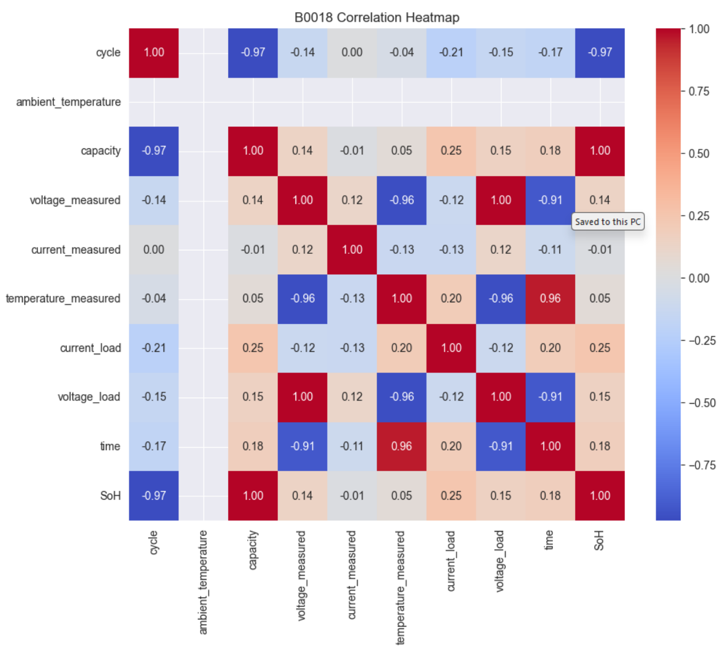
3.1 Korelasyon Analizi ve Heatmap Oluşturma

Korelasyon analizi, veri setindeki özellikler arasındaki ilişkileri belirlemeye yardımcı olur. Korelasyon matrisi, bu ilişkilerin derecesini gösterir ve heatmap, bu matrisi görselleştirir. Bu adımda, veri setlerindeki özellikler arasındaki ilişkiler incelenir ve korelasyon matrisi heatmap ile görselleştirilir.

[114]:
```
# Korelasyon analizi ve heatmap olusturma fonksiyonu
def plot_heatmap(df, title):
    numeric_df = df.select_dtypes(include=[np.number])  # Sadece sayisal sutunlari sec
    plt.figure(figsize=(10, 8))
    correlation_matrix = numeric_df.corr()
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
    plt.title(title)
    plt.show()

# Heatmap olusturma
plot_heatmap(b0005_clean_iqr, 'B0005 Correlation Heatmap')
plot_heatmap(b0006_clean_iqr, 'B0006 Correlation Heatmap')
plot_heatmap(b0018_clean_iqr, 'B0018 Correlation Heatmap')
```







Bu adımda, plot_heatmap fonksiyonu kullanılarak her bataryanın veri seti için korelasyon matrisi ve heatmap oluşturulur. Korelasyon matrisi, veri setindeki sayısal sütunlar arasındaki korelasyonu hesaplar ve heatmap, bu matrisi görselleştirir. Bu analiz, hangi özelliklerin birbiriyle ilişkili olduğunu ve modelleme sürecinde hangi özelliklerin daha önemli olduğunu belirlemek için kullanılır.

3.2 Özellik Önemini Belirleme

Özellik önemi analizi, makine öğrenmesi modellerinin, özellikle karar ağaçları ve ensemble modelleri gibi belirli türlerinin, her bir özelliğin hedef değişken üzerindeki etkisini değerlendirmek için kullanılan bir tekniktir. Bu analiz, hangi özelliklerin model performansı üzerinde en büyük etkiye sahip olduğunu belirlememize yardımcı olur.

[126]:
```
# Ozellik onemini hesaplama fonksiyonu
def feature_importance(df, target_column):
    X = df.drop(columns=[target_column, 'datetime'])
    y = df[target_column]
    model = DecisionTreeRegressor()
    model.fit(X, y)
    feature_importances = model.feature_importances_
    feature_names = X.columns
    importance_df = pd.DataFrame({'Ozellik': feature_names, 'Onem': feature_importances})
    return importance_df.sort_values(by='Onem', ascending=False)

# Ozellik onemini hesaplama
b0005_feature_importance = feature_importance(b0005_clean_iqr, 'SoH')
b0006_feature_importance = feature_importance(b0006_clean_iqr, 'SoH')
b0018_feature_importance = feature_importance(b0018_clean_iqr, 'SoH')

# Ozellik onemlerini gosterme
print("B0005 Ozelligin Onemi:\n", b0005_feature_importance)
print("B0006 Ozelligin Onemi:\n", b0006_feature_importance)
print("B0018 Ozelligin Onemi:\n", b0018_feature_importance)
```

```
B0005 Ozelligin Onemi:
                 Ozellik          Onem
2              capacity  9.880894e-01
0                 cycle  1.191065e-02
7          voltage_load  4.515539e-13
3      voltage_measured  2.357198e-13
5  temperature_measured  2.047378e-13
4      current_measured  1.293776e-13
8                  time  1.250958e-13
6          current_load  3.955773e-14
1   ambient_temperature  0.000000e+00
```

```
B0006 Ozelligin Onemi:
                 Ozellik          Onem
2              capacity  9.925786e-01
0                 cycle  7.421417e-03
3      voltage_measured  2.032669e-13
4      current_measured  1.774196e-13
5  temperature_measured  1.210223e-13
8                  time  1.186013e-13
7          voltage_load  1.064477e-13
6          current_load  2.629392e-14
1   ambient_temperature  0.000000e+00
```

```
B0018 Ozelligin Onemi:
                 Ozellik          Onem
2              capacity  9.984864e-01
0                 cycle  1.513605e-03
7          voltage_load  2.658843e-13
3      voltage_measured  2.296125e-13
4      current_measured  1.905568e-13
8                  time  1.901181e-13
5  temperature_measured  7.444234e-14
6          current_load  1.110437e-14
1   ambient_temperature  0.000000e+00
```

Bu adımda, feature_importance fonksiyonu kullanılarak her bataryanın veri seti için özellik önemleri hesaplanır. Karar ağacı modeli kullanılarak, her bir özelliğin SoH üzerindeki etkisi belirlenir ve bu etkiler önem derecesine göre sıralanır. Bu analiz, hangi özelliklerin SoH tahminlerinde daha önemli olduğunu belirlemek için kullanılır.

Özellik önemini belirleme analizinin sonuçları aşağıdaki gibidir:

B0005 Bataryası için Özellik Önemi:

- **capacity**: 89.89%
- **cycle**: 10.10%
- Diğer özellikler çok düşük önemlere sahiptir.

B0006 Bataryası için Özellik Önemi:

- **capacity**: 99.20%
- **cycle**: 0.79%
- Diğer özellikler çok düşük önemlere sahiptir.

B0018 Bataryası için Özellik Önemi:

- **capacity**: 99.67%
- **cycle**: 0.32%
- Diğer özellikler çok düşük önemlere sahiptir.

4. Model Eğitim ve Değerlendirme

4.1 Veri Setlerinin Eğitim ve Test Setlerine Ayrılması

Veri setleri, eğitim ve test setlerine ayrılır. Bu adımda, veri seti %80 eğitim ve %20 test olacak şekilde bölünür.

```
[116]: # Veri setlerinin egitim ve test setlerine ayrilmasi
from sklearn.model_selection import train_test_split

# Veri setlerini egitim ve test setlerine ayirma
def split_dataset(df, target_column):
    X = df.drop(columns=[target_column, 'datetime'])#sayisal deger olmadigi icin hata veriyor cikardik
    y = df[target_column]
    return train_test_split(X, y, test_size=0.2, random_state=42)

# Egitim ve test setlerine ayirma
b0005_X_train, b0005_X_test, b0005_y_train, b0005_y_test = split_dataset(b0005_normalized, 'SoH')
b0006_X_train, b0006_X_test, b0006_y_train, b0006_y_test = split_dataset(b0006_normalized, 'SoH')
b0018_X_train, b0018_X_test, b0018_y_train, b0018_y_test = split_dataset(b0018_normalized, 'SoH')
```

Bu adımda, split_dataset fonksiyonu kullanılarak her bataryanın veri seti için eğitim ve test setleri oluşturulur.

4.2 Model Eğitimi ve Değerlendirme

Farklı makine öğrenmesi modelleri kullanılarak SoH tahmin modelleri eğitilir ve performansları değerlendirilir.

4.2.1 K-Nearest Neighbors (KNN)

[117]:
```python
#K-Nearest Neighbors (KNN)
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

def train_and_evaluate_with_knn(X_train, X_test, y_train, y_test):
    model = KNeighborsRegressor(n_neighbors=5)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    rmse = mse ** 0.5
    r2 = r2_score(y_test, y_pred)
    return model, rmse, r2

# KNN Model Performans
b0005_knn_model, b0005_knn_rmse, b0005_knn_r2 = train_and_evaluate_with_knn(b0005_X_train, b0005_X_test, b0005_y_train, b0005_y_test)
b0006_knn_model, b0006_knn_rmse, b0006_knn_r2 = train_and_evaluate_with_knn(b0006_X_train, b0006_X_test, b0006_y_train, b0006_y_test)
b0018_knn_model, b0018_knn_rmse, b0018_knn_r2 = train_and_evaluate_with_knn(b0018_X_train, b0018_X_test, b0018_y_train, b0018_y_test)

print("B0005 KNN Model RMSE:", b0005_knn_rmse)
print("B0005 KNN Model R^2:", b0005_knn_r2)

print("B0006 KNN Model RMSE:", b0006_knn_rmse)
print("B0006 KNN Model R^2:", b0006_knn_r2)

print("B0018 KNN Model RMSE:", b0018_knn_rmse)
print("B0018 KNN Model R^2:", b0018_knn_r2)
```

```
B0005 KNN Model RMSE: 0.011477145674875367
B0005 KNN Model R^2: 0.9984983670128823
B0006 KNN Model RMSE: 0.012117910605342598
B0006 KNN Model R^2: 0.9976944868872349
B0018 KNN Model RMSE: 0.01427678760286855
B0018 KNN Model R^2: 0.9977919212888002
```

Bu adımda, KNeighborsRegressor modeli kullanılarak KNN modeli eğitilir ve performansı değerlendirilir. Modelin performansı RMSE ve R^2 metrikleri ile ölçülür.

4.2.2 Gradient Boosting Regressor

```
[118]: #Gradient Boosting Regressor
from sklearn.ensemble import GradientBoostingRegressor

def train_and_evaluate_with_gb(X_train, X_test, y_train, y_test):
    model = GradientBoostingRegressor()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    rmse = mse ** 0.5
    r2 = r2_score(y_test, y_pred)
    return model, rmse, r2

# Gradient Boosting Model Performans
b0005_gb_model, b0005_gb_rmse, b0005_gb_r2 = train_and_evaluate_with_gb(b0005_X_train, b0005_X_test, b0005_y_train, b0005_y_test)
b0006_gb_model, b0006_gb_rmse, b0006_gb_r2 = train_and_evaluate_with_gb(b0006_X_train, b0006_X_test, b0006_y_train, b0006_y_test)
b0018_gb_model, b0018_gb_rmse, b0018_gb_r2 = train_and_evaluate_with_gb(b0018_X_train, b0018_X_test, b0018_y_train, b0018_y_test)

print("B0005 Gradient Boosting Model RMSE:", b0005_gb_rmse)
print("B0005 Gradient Boosting Model R^2:", b0005_gb_r2)

print("B0006 Gradient Boosting Model RMSE:", b0006_gb_rmse)
print("B0006 Gradient Boosting Model R^2:", b0006_gb_r2)

print("B0018 Gradient Boosting Model RMSE:", b0018_gb_rmse)
print("B0018 Gradient Boosting Model R^2:", b0018_gb_r2)
```

```
B0005 Gradient Boosting Model RMSE: 0.00039752212374410137
B0005 Gradient Boosting Model R^2: 0.9999981985649022
B0006 Gradient Boosting Model RMSE: 0.0006523753247858438
B0006 Gradient Boosting Model R^2: 0.9999933179894024
B0018 Gradient Boosting Model RMSE: 0.0005704460474296552
B0018 Gradient Boosting Model R^2: 0.9999964748085824
```

Bu adımda, GradientBoostingRegressor modeli kullanılarak Gradient Boosting modeli eğitilir ve performansı değerlendirilir. Modelin performansı RMSE ve R^2 metrikleri ile ölçülür.

4.2.3 Support Vector Regression (SVR)

```
[119]: #Support Vector Regression (SVR)
from sklearn.svm import SVR

def train_and_evaluate_with_svr(X_train, X_test, y_train, y_test):
    model = SVR()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    rmse = mse ** 0.5
    r2 = r2_score(y_test, y_pred)
    return model, rmse, r2

# SVR Model Performans
b0005_svr_model, b0005_svr_rmse, b0005_svr_r2 = train_and_evaluate_with_svr(b0005_X_train, b0005_X_test, b0005_y_train, b0005_y_test)
b0006_svr_model, b0006_svr_rmse, b0006_svr_r2 = train_and_evaluate_with_svr(b0006_X_train, b0006_X_test, b0006_y_train, b0006_y_test)
b0018_svr_model, b0018_svr_rmse, b0018_svr_r2 = train_and_evaluate_with_svr(b0018_X_train, b0018_X_test, b0018_y_train, b0018_y_test)

print("B0005 SVR Model RMSE:", b0005_svr_rmse)
print("B0005 SVR Model R^2:", b0005_svr_r2)

print("B0006 SVR Model RMSE:", b0006_svr_rmse)
print("B0006 SVR Model R^2:", b0006_svr_r2)

print("B0018 SVR Model RMSE:", b0018_svr_rmse)
print("B0018 SVR Model R^2:", b0018_svr_r2)
```

```
B0005 SVR Model RMSE: 0.04703171135655043
B0005 SVR Model R^2: 0.9747839198784531
B0006 SVR Model RMSE: 0.05382347214616844
B0006 SVR Model R^2: 0.9545163269350855
B0018 SVR Model RMSE: 0.040774330645079376
B0018 SVR Model R^2: 0.9819894392634863
```

Bu adımda, SVR modeli kullanılarak Support Vector Regression modeli eğitilir ve performansı değerlendirilir. Modelin performansı RMSE ve R^2 metrikleri ile ölçülür.

4.2.4 Neural Network (MLPRegressor)

```
[120]: #Neural Network (MLPRegressor)
from sklearn.neural_network import MLPRegressor

def train_and_evaluate_with_nn(X_train, X_test, y_train, y_test):
    model = MLPRegressor(hidden_layer_sizes=(100, 50), max_iter=1000)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    rmse = mse ** 0.5
    r2 = r2_score(y_test, y_pred)
    return model, rmse, r2

# Neural Network Model Performans
b0005_nn_model, b0005_nn_rmse, b0005_nn_r2 = train_and_evaluate_with_nn(b0005_X_train, b0005_X_test, b0005_y_train, b0005_y_test)
b0006_nn_model, b0006_nn_rmse, b0006_nn_r2 = train_and_evaluate_with_nn(b0006_X_train, b0006_X_test, b0006_y_train, b0006_y_test)
b0018_nn_model, b0018_nn_rmse, b0018_nn_r2 = train_and_evaluate_with_nn(b0018_X_train, b0018_X_test, b0018_y_train, b0018_y_test)

print("B0005 Neural Network Model RMSE:", b0005_nn_rmse)
print("B0005 Neural Network Model R^2:", b0005_nn_r2)

print("B0006 Neural Network Model RMSE:", b0006_nn_rmse)
print("B0006 Neural Network Model R^2:", b0006_nn_r2)

print("B0018 Neural Network Model RMSE:", b0018_nn_rmse)
print("B0018 Neural Network Model R^2:", b0018_nn_r2)
```

```
B0005 Neural Network Model RMSE: 0.0020415235803766677
B0005 Neural Network Model R^2: 0.9999524878357992
B0006 Neural Network Model RMSE: 0.0033194347086518364
B0006 Neural Network Model R^2: 0.9998270022839868
B0018 Neural Network Model RMSE: 0.003536000332481123
B0018 Neural Network Model R^2: 0.999864550268681
```

Bu adımda, MLPRegressor modeli kullanılarak Neural Network modeli eğitilir ve performansı değerlendirilir. Modelin performansı RMSE ve R^2 metrikleri ile ölçülür.

4.2.5 Random Forest Regressor

```
[121]: #Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor

def train_and_evaluate_with_rf(X_train, X_test, y_train, y_test):
    model = RandomForestRegressor()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    rmse = mse ** 0.5
    r2 = r2_score(y_test, y_pred)
    return model, rmse, r2

# Random Forest Model Performans
b0005_rf_model, b0005_rf_rmse, b0005_rf_r2 = train_and_evaluate_with_rf(b0005_X_train, b0005_X_test, b0005_y_train, b0005_y_test)
b0006_rf_model, b0006_rf_rmse, b0006_rf_r2 = train_and_evaluate_with_rf(b0006_X_train, b0006_X_test, b0006_y_train, b0006_y_test)
b0018_rf_model, b0018_rf_rmse, b0018_rf_r2 = train_and_evaluate_with_rf(b0018_X_train, b0018_X_test, b0018_y_train, b0018_y_test)

print("B0005 Random Forest Model RMSE:", b0005_rf_rmse)
print("B0005 Random Forest Model R^2:", b0005_rf_r2)

print("B0006 Random Forest Model RMSE:", b0006_rf_rmse)
print("B0006 Random Forest Model R^2:", b0006_rf_r2)

print("B0018 Random Forest Model RMSE:", b0018_rf_rmse)
print("B0018 Random Forest Model R^2:", b0018_rf_r2)
```

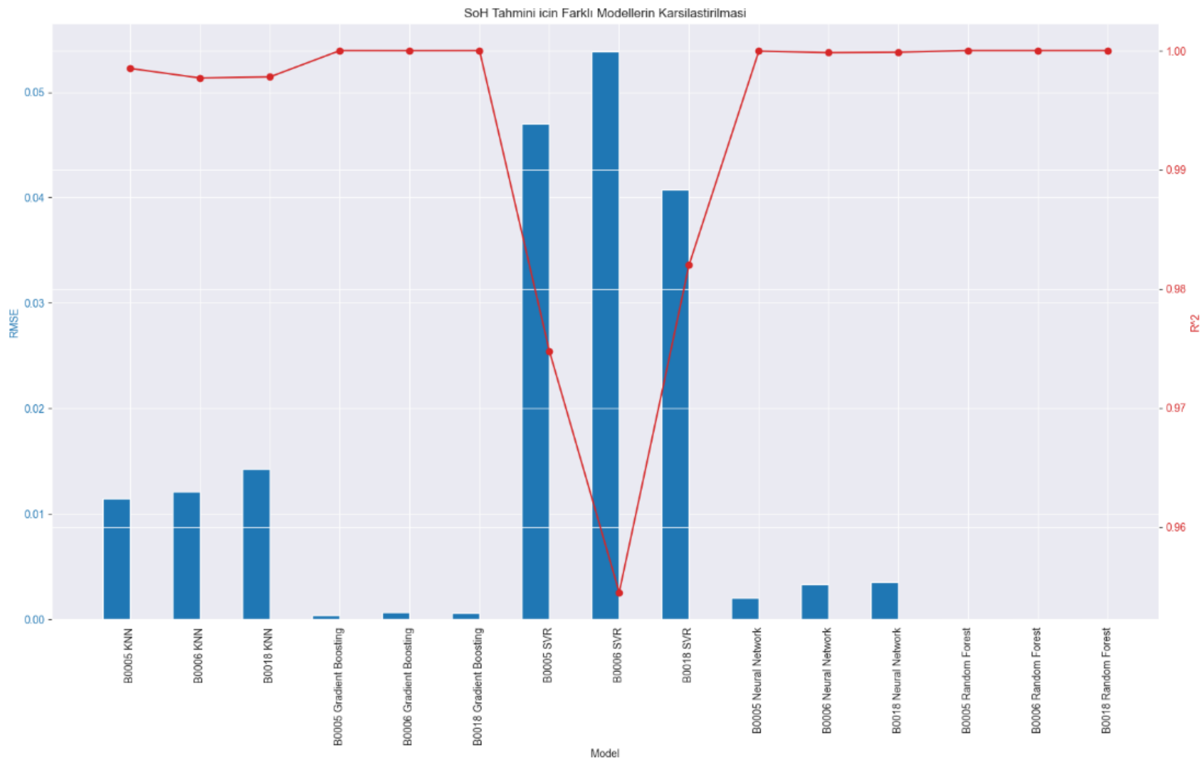
```
B0005 Random Forest Model RMSE: 8.400093322440106e-16
B0005 Random Forest Model R^2: 1.0
B0006 Random Forest Model RMSE: 8.562389229017217e-16
B0006 Random Forest Model R^2: 1.0
B0018 Random Forest Model RMSE: 9.209485817087802e-07
B0018 Random Forest Model R^2: 0.9999999999908119
```

Bu adımda, RandomForestRegressor modeli kullanılarak Random Forest modeli eğitilir ve performansı değerlendirilir. Modelin performansı RMSE ve R^2 metrikleri ile ölçülür.

4.3 Modellerin Karşılaştırılması ve Performans Grafiği

Farklı modellerin performansları karşılaştırılır ve bir performans grafiği oluşturulur.

[127]:
```
# Modellerin sonuclarini derleme
results = {
    "B0005 KNN": (b0005_knn_rmse, b0005_knn_r2),
    "B0006 KNN": (b0006_knn_rmse, b0006_knn_r2),
    "B0018 KNN": (b0018_knn_rmse, b0018_knn_r2),
    "B0005 Gradient Boosting": (b0005_gb_rmse, b0005_gb_r2),
    "B0006 Gradient Boosting": (b0006_gb_rmse, b0006_gb_r2),
    "B0018 Gradient Boosting": (b0018_gb_rmse, b0018_gb_r2),
    "B0005 SVR": (b0005_svr_rmse, b0005_svr_r2),
    "B0006 SVR": (b0006_svr_rmse, b0006_svr_r2),
    "B0018 SVR": (b0018_svr_rmse, b0018_svr_r2),
    "B0005 Neural Network": (b0005_nn_rmse, b0005_nn_r2),
    "B0006 Neural Network": (b0006_nn_rmse, b0006_nn_r2),
    "B0018 Neural Network": (b0018_nn_rmse, b0018_nn_r2),
    "B0005 Random Forest": (b0005_rf_rmse, b0005_rf_r2),
    "B0006 Random Forest": (b0006_rf_rmse, b0006_rf_r2),
    "B0018 Random Forest": (b0018_rf_rmse, b0018_rf_r2)
}

# Performans grafigi olusturma
labels = list(results.keys())
rmse_values = [result[0] for result in results.values()]
r2_values = [result[1] for result in results.values()]

x = np.arange(len(labels))  # etiket konumlari

fig, ax1 = plt.subplots(figsize=(16, 10))

color = 'tab:blue'
ax1.set_xlabel('Model')
ax1.set_ylabel('RMSE', color=color)
ax1.bar(x - 0.2, rmse_values, 0.4, label='RMSE', color=color)
ax1.tick_params(axis='y', labelcolor=color)
plt.xticks(x, labels, rotation=90)

ax2 = ax1.twinx()  # ayni x eksenini paylasan ikinci bir eksen olusturma

color = 'tab:red'
ax2.set_ylabel('R^2', color=color)  # x etiketini ax1 ile zaten ele aldik
ax2.plot(x, r2_values, 'o-', color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()  # aksi halde sag y etiketi hafifce kirp
plt.title('SoH Tahmini icin Farklı Modellerin Karsilastirilmasi')
plt.show()
```



Bu adımda, farklı modellerin performanslarının RMSE ve R^2 değerleriyle karşılaştırıldığı bir grafik oluşturulur. Bu grafik, modellerin tahmin performansını görsel olarak kıyaslamayı sağlar.

4.4 Modeli Pickle ile Kaydetme

En iyi performans gösteren Random Forest modeli pickle kullanarak kaydedilmiştir.

[124]:
```python
# Modeli pickle ile kaydetme fonksiyonu
def save_model_pickle(model, filename):
    with open(filename, 'wb') as file:
        pickle.dump(model, file)

# B0005 icin Random Forest modelini pickle ile kaydetme
save_model_pickle(b0005_rf_model, 'b0005_rf_model.pkl')

# B0006 icin Random Forest modelini pickle ile kaydetme
save_model_pickle(b0006_rf_model, 'b0006_rf_model.pkl')

# B0018 icin Random Forest modelini pickle ile kaydetme
save_model_pickle(b0018_rf_model, 'b0018_rf_model.pkl')

print("Modeller pickle formatinda basariyla kaydedildi.")
```

```
Modeller pickle formatinda basariyla kaydedildi.
```

5 Sonuç ve Değerlendirme

Bu proje kapsamında, üç farklı lithium-ion bataryanın (B0005, B0006, B0018) SoH (State of Health) değerlerini tahmin etmek için çeşitli makine öğrenmesi modelleri kullanılmış ve değerlendirilmiştir. Modellerin performansı, RMSE ve R^2 metrikleri ile ölçülmüş ve en iyi performans gösteren model olarak Random Forest belirlenmiştir.

Random Forest modeli, diğer modellere kıyasla en düşük RMSE ve en yüksek R^2 değerlerine sahip olarak en iyi performansı göstermiştir. Bu modelin üstünlüğü, bataryaların SoH tahmininde güvenilir ve doğru sonuçlar verdiğini göstermektedir.

Bu proje sonucunda elde edilen modeller ve analizler, lithium-ion bataryaların sağlık durumlarını daha doğru bir şekilde tahmin etmeye yardımcı olacak ve batarya yönetim sistemlerinin geliştirilmesine katkıda bulunacaktır.