Makine Öğrenmesi Projesi Bölüm-1

Bu çalışmamızda gerçek veriseti kulanarak bir makine öğrenmesi çalışmasının tüm adımlarını uygulamalı olarak göreceğiz.
Bizden istenen durum:
Bir binanın "Energy Star puanını" tahmin edebilen bir model geliştirmek için eldeki bina enerji verilerini kullanın ve sonuçları enerji puanını en iyi tahmin eden değişkenleri bulmak için yorumlayın.
Bu işlem anlaşıldığı üzere bir yapılandırılmış(supervised), regresyon makine öğrenmesi görevidir. Çünkü, bir veri seti verilmiş ve hedeflenen durum(yani maksimum puan) belirtilmiş, burada yapmak istediğimiz belirtilen özellikleri istenen hedef ile eşleştirebilen(karşılaştırabilen) bir model geliştirmek istiyoruz.
Yapılandırılmış Problem(Supervised Problem) : Açıkça anlaşıldığı üzere problemimiz yapılandırılmış çünkü özellik ve hedef her ikiside verilmiş.
Regresyon Problemi(Resgression Problem) : Hedef değişkenler sürekli ve 0-100 arasında değer alıyor.
Eğitim sürecinde modelimizin özellikler ve enerji puanı arasındaki ilişkiyi öğrenmesini istiyoruz. Hem özellikleri hem de cevabı vererek eğitim sırasında, modelin özellikler ve puan arasındaki ilişkiyi öğrenmesini bekliyoruz. Ondan sonra modelimizi daha önce hiç bilinmeyen veriler ile test edip, modelin ne kadar iyi eğitildiğini kontrol ediyoruz.
Makine Öğrenmesi Adımları.
Aslında ne kadarda vereceğimiz işlem adımları değişkenlik gösterebilse de ana iskelet olarak bir makine öğrenmesi çalışması aşağıda belirtilen adımlardan oluşmaktadır:

1. Veri Temizleme ve biçimlendirme(Data cleaning and formatting)
2. Açılayıcı veri analizi(Exploratory data analysis)
3. Özellik belirleme ve seçme(Feature engineering and selection)
4. Temel işlemleri uygulayıp birkaç farklı makine öğrenmesi modeli ile deneyerek performanslarını karşılaştırma. (Establish a baseline and compare several machine learning models on a performance metric)
5. Problem için optimize etmek üzere en iyi modele hiperparametre ayarlaması yapma!
   (Perform hyperparameter tuning on the best model to optimize it for the problem)
6. Test verileri üzerinde en iyi modeli değerlendirme. (Evaluate the best model on the testing set)
7. Model sonuçlarını en iyi şekilde yorumlama. (Interpret the model results to the extent possible)

8. Sonuçları görselleştirme ve raporumuzu en iyi şekilde hazırlama.
(Draw conclusions and write a well-documented report)

Yukarıda belirttiğimiz işlem adımlarının birbirini nasıl takip ettiğini anlamaya çalışalım. Daha öncede belirttiğimiz üzere makine öğrenmesi işlem adımları burada belirtilen kuralların katı bir şekilde uygulanacağı anlamına gelmiyor. Örnek olarak, herhangi bir model oluşturmadan önce bir özellik belirleme işlemini deneyimleyebiliriz yada geri dönmek ve farklı bir özellik grubu seçmek için modelleme sonuçlarını kullanabiliriz veya bizim oluşturduğumuz model çok aykırı sonuçlarda üretebilir bunun için verilerin anlaşılmasını başka bir açıdan da anlamaya çalışabiliriz.
Genellikle olan şudur, bir adımı tamamlayıp diğerine geçmek ve sonuçları gözlemlemektir. Şöyle düşünmeyin biz bir işlemi yaptık bitti ve diğer adıma geçtik artık bir daha geri dönemeyiz.
Bu çalışmamız yukarıda belirtmiş olduğumuz işlem adımlarının ilk üç adımını kapsayacak, diğer adımlar için serinin devamı olan yeni çalışma örneklerinde ilgili konulara değineceğiz. Çalışma örneğimizde daha çok işlemlerin uygulanma şekline değineceğiz konular ile ilişkili derin bilgiler almak isterseniz sayfa sonunda belirteceğim linklerden ilgili konular için detaylı bilgiler edinebilirsiniz.
Evet, çalışmamıza artık başlayabiliriz, çok fazla laf kalabalığı yapmak benimde hoşuma gitmiyor. E o zaman hadi bize lazım olacak ve çalışmamızda kullanacağımız python kütüphanelerini eklemekle projemize başlayalım. Bu çalışmamızda standart veri bilimi kütüphaneleri olan: numpy, pandas ve scikit-learn kullanacağız, yaptığımız çalışmaları görselleştirmek içinde matplotlib ve seaborn kütüphanelerinden faydalanacağız.
E hadi başlayalım :)

In [1]:
```
#Verileri işlemek için Pandas ve numpy kütüphanelerini ekliyoruz.
import pandas as pd
import numpy as np

#Bölüm kopyaları üzerinde değer ayarlamaya ilişkin uyarıları kapatmak için
pd.options.mode.chained_assignment = None

#60 sütüna kadar verilerimizi görüntülüyoruz
pd.set_option('display.max_columns', 60)

#Görselleştirme arası olan Matplotlib ekliyoruz
import matplotlib.pyplot as plt
%matplotlib inline
```

```
#Yazı fontlarını varsayılan olarak ayarlıyoruz
plt.rcParams['font.size'] = 24

#Şekil boyutlarını ayarlamak için dahili Ipython
kütüphanesinide ekliyoruz
from IPython.core.pylabtools import figsize

#Yine görselleştirme aracı olarak seaborn kütüphanesini
ekliyoruz.
import seaborn as sns
sns.set(font_scale = 2)

#Verilerimize eğitim ve test verileri olarak bölüyoruz,
genel kabul olarak (%80 eğitim, %20 test)
from sklearn.model_selection import train_test_split
```

1. Verilerin temizlenmesi ve düzenlenmesi

a. Verileri yükleme ve inceleme

Veri bilimi için en iyi veri yapılarından biri olarak kullanılan pandas kütüphansesini kullanarak verilerimizi çalışma ortamına yüklüyoruz. Pandas kütüphanesini Python'un bir sayfası olarak düşünebilirsiniz, bu sayfada temizleme, düzenleme ve görselleştirme işlemlerini yapacağız. Pandas'ta birçok method var bunlar bilimsel çalışmalar için ihtiyaç olan kütüphanelerdir ve amacına uygun olarak bu kütüphaneler kullanılabilir.

In [2]:

```
#Verilerimizi ilgili dataframe(veri çerçevesi)'nden
okuyoruz
data = pd.read_csv('../saitalay/Projeler/VeriBilimi/
Kaggle/
Energy_and_Water_Data_Disclosure_for_Local_Law_84_2015__D
ata_for_Calendar_Year_2014_.csv')

#Şimdi veri çerçevemizi görüntüleyelim- özet olarak ilk 5
satır görüntüleniyor
data.head()
```

Out[2]:

Evet eldeki verilere ilişkin yukarıdaki çıktıda birtakım değerler görünüyor ancak hepsinin ne anlama geldiğini bilmiyoruz. Veri seti içerisinde 60 sütün var ve bu sütünların ne demek olduğunu biz bilmiyoruz. Tek bildiğimiz şey bize problemde verilen ve bizden istenen skor(puan) sütünu, çünkü bizden enerji puanı için hangi özelliğin en önemli olduğunu bulmamız istenmişti. Bu veri seti içerisinde bulunan her sütunun ne anlama geldiğini bilmemiz gerekmiyor, bizim için önemli olan hangi özelliğin enerji puanı için en önemli olduğunu bulmak. Makine öğrenmesi içinde problem teşkil etmiyor çünkü modelin hangi özelliklerin önemli olduğuna karar vermesine izin veriyoruz. Hatta bazen sütun isimlerinide vermeyebiliriz yada ne tahmin etmeye çalıştığımız şeyleri söylemeyiz. Yine de, problemi mümkün olduğunca anlamak istiyoruz ve model sonuçlarını da yorumlamak istediğimizden, sütunlarla ilgili bazı bilgilere sahip olmak iyi bir fikir olabilir.(ne diyoruz; bilgi faydalıdır.)
Tabi veri seti içerisinde bize faydalı olacak bilgiler(sütunlar), üzerinde yoğunlaşmak daha iyi olacaktır. Çünkü biz birşeyler tahmin etmek istiyoruz bunun için hedeflenen veriler üzerinde çalışacağız.
Şimdi bizden istenen hedef neydi tekrar hatırlayalım,
Ilgili raporlama yılı(2015) için kendinden bildirilen enerji kullanımına dayalı olarak Portföy Yöneticisi'nde hesaplanan ve belirtilen bina türleri için 1-100 değerlerinde yüzdelik sıralamasının yapılması işlemi.
Bu oldukça basit : Enerji Yıldızı Puanı, enerji verimliliği açısından binaları en kötü olanın 1 ve en iyi olanı ise 100 ile derecelendirme yöntemidir. Göreceli bir yüzdelik sıralamasıdır. Bu, binaların birbirlerine göre puanlandırıldığı ve değerler aralığında düzgün bir dağılım göstermesi gerektiği anlamına gelir.

Veri Tipleri ve Kayıp Değerler

Dataframe.info yöntemi, her bir sütunun veri türlerini ve eksik olmayan değerlerin sayısını görüntüleyerek verileri değerlendirmenin hızlı bir yoludur. Veri içeriğine neden bakmalıyız, çünkü veriler içerisinde bir problem olabilir örneğin eksik değerler np.nan (bir sayı değil) yerine "Kullanılamaz" olarak kodlanır. Bu, sayıları olan sütunların sayısal olarak gösterilmeyeceği anlamına gelir çünkü Pandas, herhangi bir dizgisi olan sütunları tüm dizelerin sütunlarına dönüştürür.

In [3]:

```
#Sütünların veri tiplerini ve eksik olmayan değerleri
görüntülüyoruz.
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13974 entries, 0 to 13973
Data columns (total 33 columns):
```

Record Number
11804 non-null float64
BBL
13974 non-null int64
Co-reported BBL Status
668 non-null object
BBLs Co-reported
668 non-null object
Reported NYC Building Identification Numbers (BINs)
11504 non-null object
Street Number
13634 non-null object
Street Name
13678 non-null object
Borough
13974 non-null object
Postcode
11614 non-null float64
BBL on the Covered Buildings List
13974 non-null object
DOF Benchmarking Submission Status
13974 non-null object
Site EUI(kBtu/ft2)
10610 non-null float64
Weather Normalized Site EUI(kBtu/ft2)
11916 non-null object
Source EUI(kBtu/ft2)
11916 non-null object
Weather Normalized Source EUI(kBtu/ft2)
11916 non-null object
Municipally Supplied Potable Water - Indoor Intensity (gal/ft²) 11916 non-null object
Automatic Water Benchmarking Eligible
13974 non-null object
Reported Water Method
13974 non-null object
ENERGY STAR Score
13974 non-null object
Total GHG Emissions(MtCO2e)
13974 non-null object
Direct GHG Emissions(MtCO2e)
13974 non-null object

```
Indirect GHG Emissions(MtCO2e)
13974 non-null object
Reported Property Floor Area (Building(s)) (ft²)
13974 non-null object
DOF Property Floor Area (Buildngs and Parking)(ft2)
13678 non-null float64
Primary Property Type - Self Selected
13678 non-null object
DOF Number of Buildings
13678 non-null float64
Latitude
11138 non-null float64
Longitude
11138 non-null float64
Community Board
11138 non-null float64
Council District
11138 non-null float64
Census Tract
11138 non-null float64
BIN
10903 non-null float64
NTA
11141 non-null object
dtypes: float64(11), int64(1), object(21)
memory usage: 3.5+ MB
```

Yukarıdaki çıktıda görüldüğü üzere, nesne veri türü olarak kaydedilmiş değerler içeren birkaç sütun var. Herhangi bir sayısal analiz yapmadan önce bunları float veri türüne dönüştürmemiz gerekiyor.

Verileri doğru veri türlerine dönüştürme

Rakamlarla sütunları sayısal veri türlerine dönüştürerek "Kullanılamıyor(nan)" girişlerini np.nan ile değiştirerek float olarak yorumlatabiliriz. Daha sonra sayısal değerler içeren (kareler veya enerji kullanımı gibi) sütunları sayısal veri türlerine dönüştüreceğiz.
Rakamlarla sütunları sayısal veri türlerine dönüştürüp "Kullanılamıyor" girişlerini np.nan ile değiştirerek float olarak yorumlatabiliriz. Daha sonra sayısal değerler içeren (kareler veya enerji kullanımı gibi) sütunları sayısal veri türlerine dönüştüreceğiz.

In [4]:

```
#Veri seti içerisindeki tüm değerleri numpy ile nan
olarak değiştiriyoruz
data = data.replace({'Not Available': np.nan})
```

```
#Kolonlara uyguluyoruz
#for col in list(data.columns):
    #Seçilen sütunların numeric(sayısal)  olması
gerekiyor
   # if('BBL' in col or 'kBtu' in col or 'MtCO2e' in col
or 'ft²' in col or 'gal' in col
    #    or 'Score' in col or 'BBL' in col):
        #Artık veri türlerini float olarak
dönüştürebiliriz
     #   data[col] = data[col].astype(float)
```

In [5]:

```
# herbir kolonun istatistiksel oldak görüntülenmesi
data.describe()
```

Out[5]:

	Record Number	BBL	Postcode	Site EUI(kBtu/ft2)	DOF Property Floor Area (Buildngs and Parking) (ft2)	DOF Number of Buildings	Latitude	Longitude	Community Board	Council District	Census Tract	BIN
count	1.180400e+04	1.397400e+04	11614.000000	1.061000e+04	1.367800e+04	13678.000000	11138.000000	11138.000000	11138.000000	11138.000000	11138.000000	1.090300e+04
mean	3.215713e+06	2.211436e+09	10569.945411	2.412011e+03	1.803697e+05	1.629551	40.752561	-73.956826	7.022625	16.534566	4929.372957	1.874689e+06
std	6.817299e+05	1.225264e+09	569.729511	1.428145e+05	4.754714e+05	5.559912	0.080874	0.049258	4.006866	15.899826	13552.934878	1.016036e+06
min	7.365000e+03	1.000010e+09	10001.000000	0.000000e+00	5.000800e+04	1.000000	40.515850	-74.243582	1.000000	1.000000	1.000000	1.000000e+06
%2	2.721424e+06	1.013100e+09	10024.000000	6.800000e+01	6.540000e+04	1.000000	40.702197	-73.984811	4.000000	4.000000	99.000000	1.040022e+06

%5(	2.831214e+06	2.030840e+09	10455.000000	8.610000e+01	9.482800e+04	1.000000	40.757485	-73.962176	7.000000	10.000000	205.000000	1.087548e+06
%7	3.895795e+06	3.061323e+09	11219.000000	1.069000e+02	1.650912e+05	1.000000	40.817572	-73.928557	9.000000	33.000000	534.000000	3.022168e+06
max	4.495856e+06	5.080080e+09	11694.000000	1.080412e+07	2.760000e+07	152.000000	40.912869	-73.715543	56.000000	51.000000	155101.000000	5.166094e+06

Eksik Değerler

Artık doğru sütun veri türlerine sahip olduğumuza göre, her bir sütunda eksik değerlerin yüzdesine bakarak analiz başlatabiliriz. Keşfedici Veri Analizi(Exploratory Data Analysis) yaptığımızda eksik değerler iyidir, ancak makine öğrenimi yöntemleri için eksik değerlerin doldurulmaları gerekir. Aşağıda, eksik değerlerin sayısını ve her sütun için eksik olan toplam değerlerin yüzdesini hesaplayan bir fonksiyondur(işlev). Veri bilimindeki birçok görevde olduğu gibi, bu fonksiyonu kendim yazmamıştım, bu fonksiyonu da bir Stack Overflow forumunda buldum!

In [6]:

```
#Sütunlardaki eksik olan değerleri hesaplıyoruz
def missing_values_table(df):
    #Toplam  eksik değer sayısı
    mis_val = df.isnull().sum()

    #eksik değerlerin yüzde olarak hesaplanması
    mis_val_percent = 100 * df.isnull().sum() / len(df)

    #sonuçları bir tablo haline dönüştürüyoruz
    mis_val_table = pd.concat([mis_val, mis_val_percent],
axis = 1)

    #Sütunları Yeniden isimlendiriyoruz
    mis_val_table_ren_columns = mis_val_table.rename(
    columns = {0 : 'Missing Values', 1 : '%of Total
Values'})

    #Tabloyu eksik değerlerin azalan sırasına göresı
ralama
    mis_val_table_ren_columns =
mis_val_table_ren_columns[
```

```
        mis_val_table_ren_columns.iloc[:,1] !
=0].sort_values(
        '%of Total Values', ascending=False).round(1)
    #Bazı özet bilgileri yazdıralım
    print ("Bizim seçtiğimiz veri çerçevesinde olan sütun
sayısı : " + str(df.shape[1]) + "\n"
           "Ve burada eksik veriler ile bulunan sütun
sayısı : " + str(mis_val_table_ren_columns.shape[0]))

    # Eksik veriler içeren veri çerçevesine dallan
    return mis_val_table_ren_columns
```

In [7]:

```
missing_values_table(data)
```

```
Bizim seçtiğimiz veri çerçevesinde olan sütun sayısı : 33
Ve burada eksik veriler ile bulunan sütun sayısı : 28
```

Out[7]:

	Missing Values	%of Total Values
BBLs Co-reported	13306	95.2
Co-reported BBL Status	13306	95.2
Municipally Supplied Potable Water - Indoor Intensity (gal/ft²)	10019	71.7
ENERGY STAR Score	4985	35.7
Weather Normalized Site EUI(kBtu/ft2)	4779	34.2
Weather Normalized Source EUI(kBtu/ft2)	4779	34.2
Site EUI(kBtu/ft2)	3364	24.1
BIN	3071	22.0
Source EUI(kBtu/ft2)	2881	20.6
Latitude	2836	20.3
Census Tract	2836	20.3
Council District	2836	20.3
Community Board	2836	20.3
Longitude	2836	20.3
NTA	2833	20.3
Total GHG Emissions(MtCO2e)	2802	20.1

Direct GHG Emissions(MtCO2e)	2692	19.3
Reported NYC Building Identification Numbers (BINs)	2678	19.2
Primary Property Type - Self Selected	2648	18.9
Indirect GHG Emissions(MtCO2e)	2499	17.9
Postcode	2360	16.9
Reported Property Floor Area (Building(s)) (ft²)	2189	15.7
Reported Water Method	2174	15.6
Record Number	2170	15.5
Street Number	340	2.4
DOF Property Floor Area (Buildngs and Parking)(ft2)	296	2.1
DOF Number of Buildings	296	2.1
Street Name	296	2.1

Her ne kadar bilgi atmamak ve sütunları düşürürken dikkatli olmak istiyorsak da, eğer bir sütun eksik değerlerin yüksek bir yüzdesine sahipse, o zaman muhtemelen çok fazla kullanılmayacaktır.
Hangi sütunların saklanması biraz keyfi olabilir, ancak bu proje için% 50'den fazla eksik değer içeren sütunları kaldıracağız. Genel olarak, herhangi bir bilgiyi düşürme konusunda dikkatli olun, çünkü tüm gözlemler için olmasa bile, hedef değeri tahmin etmek için hala bazı bilgiler yararlı olabilir.

In [8]:
```
# Eksik değerin %50'den büyük olduğu sütunları seçiyoruz
missing_df = missing_values_table(data);
missing_columns = list(missing_df[missing_df['%of Total Values'] > 50].index)
print('Bizim silmemiz gereken %d sütun.' % len(missing_columns))
```

```
Bizim seçtiğimiz veri çerçevesinde olan sütun sayısı : 33
Ve burada eksik veriler ile bulunan sütun sayısı : 28
Bizim silmemiz gereken 3 sütun.
```

In [9]:
```
# Sütunları siliyoruz
data = data.drop(columns = list(missing_columns))
```

Keşifsel Veri Analizi(Exploratory Data Analysis)

Keşifsel veri analizi: Verilerin anlaşılması için çizimler hazırladadığımız ve istatistikler hesapladığımız açık uçlu bir süreçtir. Burada amaç anormallikleri, modelleri, şablonları veya veriler arasındaki ilişkileri bulmaktır. Bunlar kendileri tarafından ilginç olabilir (örneğin iki değişken arasında bir korelasyon bulmak) veya hangi özelliklerin kullanılacağı gibi modelleme kararlarını bildirmek için kullanılabilirler. Kısacası, EDA'nın amacı verinin bize neler söyleyeceğini belirlemektir! EDA(Exploratory Data Analysis) genel olarak yüksek düzeyli bir genel bakışla başlar ve incelemek için ilginç alanlar bulduğumuzda veri kümesinin belirli kısımlarına kadar daraltılır. Veri keşfine başlarken(EDA), biz tekbir değişkene odaklanacağız, bu çalışmada bizim hedef değişkenimiz "Enerji Star Puanı" çünkü makine öğrenmesi modeli için hedefimiz "Enerji Star Puanı"nını bulmaktır. Basitlik birşekilde puan almak için sütunu yeniden adlandırabilir ve sonra bu değeri keşfetmeye başlayabiliriz.

Tek Değişkini Çizdiriyoruz

Histogramda tek bir değişkenin dağılımını gösteriyoruz.

Geri kalan değerleri, makine öğrenimi yapmadan önce uygun bir strateji (doldurulmalıdır) uygulanmalıdır.

In [14]:

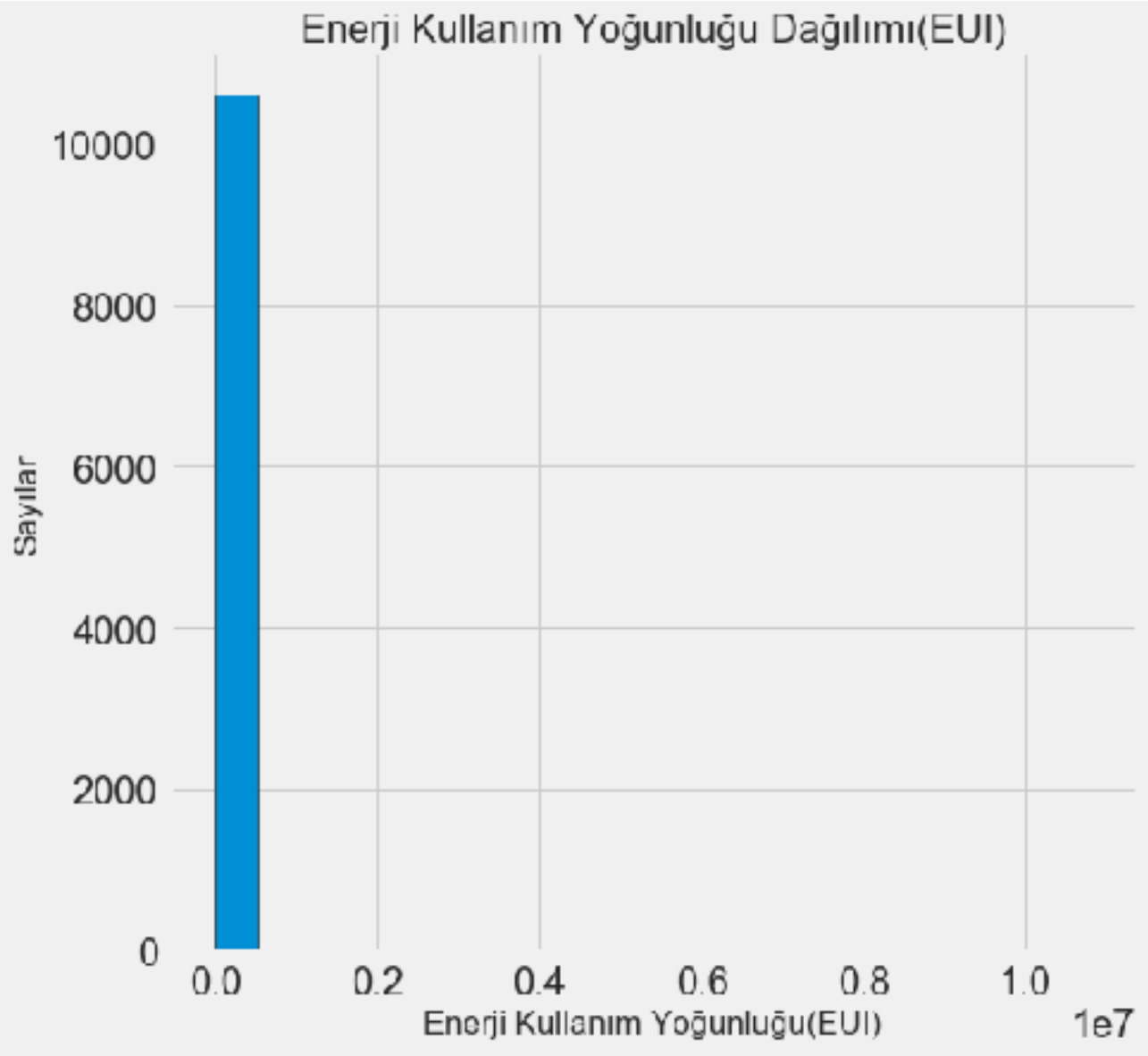
```
figsize(8,8)

#Puan sütununu yeniden isimlendiriyoruz
data = data.rename(columns = {'ENERJI STAR Score':
'score'})

#Enerji Star Puanının Histogramı
plt.style.use('fivethirtyeight')
plt.hist(data['score'].dropna(), bins = 100, edgecolor =
'k');
plt.xlabel('Puan');
plt.ylabel('Bina Numarası');
plt.title('Enerji Star Puanı Dağılımı')
```

İlk histogramımızda, bazı şaşırtıcı (ve şüpheli) bilgileri zaten ortaya çıkardı! Enerji Yıldızı Puanı bir yüzdelik derecesi olduğundan, her bir skorun dağılımın %1'ini (yaklaşık 90 bina) oluşturan tamamen düz bir dağılım görmesini beklerdik. Bununla birlikte, bu en yaygın iki puanın (1 ve 100), genel puanların büyük bir kısmını oluşturduğunu görebildiğimiz için kesinlikle su durum geçerli değildir.
Skorun tanımına geri dönersek, bu sonucun kendinden bildirilen enerji kullanımına dayalı olduğunu görürüz. Bu durum bir problem oluşturuyor çünkü bir bina sahibi, binalarının performansını yapay olarak artırmak için

daha düşük elektrik kullanımını rapor etmek cazip gelebilir. Örneğin okulda öğrencilere, test puanlarına kendi kendilerine bir not vermek gibidir. Bu davranış, binaların yüksek yüzdesini mükemmel bir 100 puanla açıklayacaktır, ancak neden bu kadar çok binanın en altta olduğunu bu sonuç açıklamıyor! Energy Star Puanı'nı karşılaştırmak için, binanın toplam enerji kullanımından meydana gelen Enerji Kullanım Yoğunluğuna (EUI-EKY) bakabiliriz. Burada enerji kullanımı kendiliğinden bildirilmemiştir, bu yüzden bir binanın enerji verimliliğinin daha objektif bir ölçüsü olabilir. Dahası, bu yüzdelik bir oran değildir, bu yüzden mutlak değerler önemlidir ve bunların normalde düşük ya da yüksek uçta bir kaç aykırı durum ile normal olarak dağılmasını beklerdik.

In [215]:

```
#Enerji Kullanım Yoğunluğunu(EUI) Histogramını
çizdiriyoruz
figsize(8, 8)
plt.hist(data['Site EUI(kBtu/ft2)'].dropna(), bins = 20,
edgecolor = 'black');
plt.xlabel('Enerji Kullanım Yoğunluğu(EUI)');
plt.ylabel('Sayılar');
plt.title('Enerji Kullanım Yoğunluğu Dağılımı(EUI)');
```



Peki bu bize başka bir problemimiz olduğunu gösteriyor: aykırı durumlar! Grafik, çok yüksek puanlara sahip birkaç binanın varlığı nedeniyle inanılmaz derecede çarpık. Aykırı olan durumlarla başa çıkmak için birşeyler yapmamız gerekecek gibi görünüyor. Bu durum için istatistiklere bakalım.

In [32]:

```
data['Site EUI(kBtu/ft2)'].describe()
```

Out[32]:

```
count    1.061000e+04
mean     2.412011e+03
std      1.428145e+05
min      0.000000e+00
25%      6.800000e+01
50%      8.610000e+01
75%      1.069000e+02
max      1.080412e+07
```

```
Name: Site EUI(kBtu/ft2), dtype: float64
```

In [33]:

```
data['Site EUI(kBtu/
ft2)'].dropna().sort_values().tail(10)
```

Out[33]:

```
10184       91990.6
2933        93989.9
6335       108572.9
7668       109454.1
7646       117755.5
10260      288491.5
9051       466788.9
2909       778850.6
8931      9932717.0
11523    10804120.0
Name: Site EUI(kBtu/ft2), dtype: float64
```

Görüldüğü üzere biri diğerlerinin açık ara önünde bir puana sahip.

In [216]:

```
data.loc[data['Site EUI(kBtu/ft2)'] == 10804120, :]
```

Bu bina sahibini kontrol etmek gerekebilir :)! Ancak, bu bizim sorunumuz değil ve sadece bu bilginin nasıl ele alınacağını bulmamız gerekiyor. Aykırı değerler birçok nedenden ötürü olabilir: yazım hataları, ölçüm cihazlarındaki arızalar, yanlış birimler ya da yasal olabilir ama değerler çok aşırı.

Aykırı Değerleri Siliyoruz

Aykırı değerleri ortadan kaldırdığımızda, sadece garip görünmeleri nedeniyle ölçümleri atmadığımıza dikkat etmek istiyoruz. Onlar daha fazla araştırmamız gereken gerçek bir değerin sonucu olabilirler. Aykırı değerleri ortadan kaldırırken, aşırı bir dışsallık tanımını kullanarak olabildiğince dikkatli olmaya çalışıyorum.

Düşük uçta, aşırı bir aykırı değer

$$\text{First Quartile} - 3 * \text{Interquartile Range}$$

First Quartile

−

3

∗

Interquartile Range

'ın altında. Yüksek uçta, aşırı bir aykırı değer

Üçüncü Çeyrek+3∗Interquartile Range

Üçüncü Çeyrek

+

3

∗

Interquartile Range

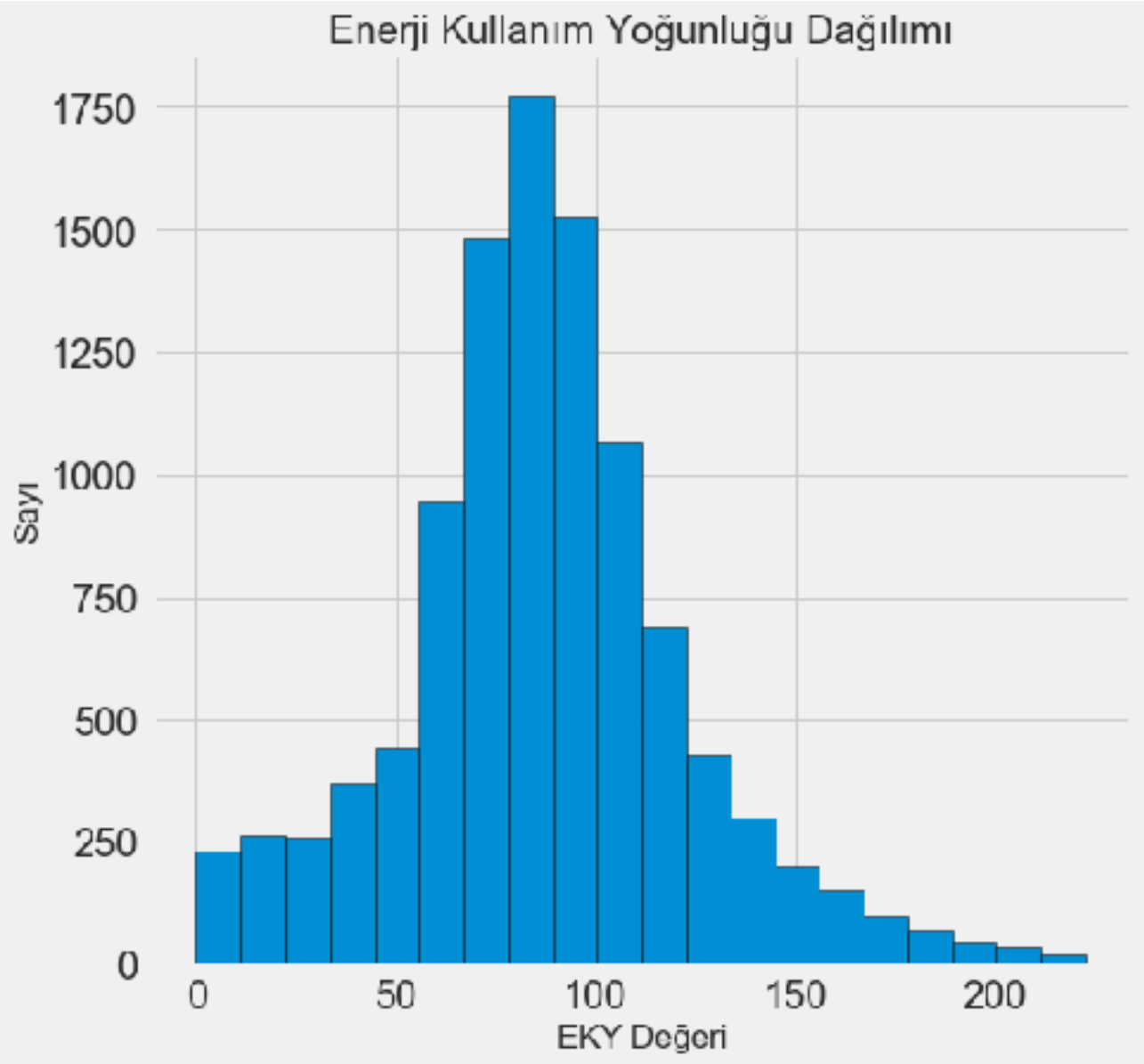
'in üzerindedir. Bu durumda, sadece tek bir çıkış noktasını kaldıracağım ve dağıtımın nasıl göründüğünü göreceğim.

In [217]:

```
#1. ve 3. Çeyrek değerlerini hesağluyoruz
first_quartile = data['Site EUI(kBtu/ft2)'].describe()
['25%']
third_quartile = data['Site EUI(kBtu/ft2)'].describe()
['75%']

#İki Çeyrek arasındaki fark
iqr = third_quartile - first_quartile

#Aykırı değerleri siliyoruz
data = data[(data['Site EUI(kBtu/ft2)'] > (first_quartile
- 3 * iqr)) &
            (data['Site EUI(kBtu/ft2)'] < (third_quartile
+ 3 * iqr))]
```

In [218]:

```
#Enerji Kullanım Yoğunluğu(EKY) histogramını çizdiriyoruz
figsize(8, 8)
plt.hist(data['Site EUI(kBtu/ft2)'].dropna(), bins = 20,
edgecolor = 'black');
plt.xlabel('EKY Değeri');
plt.ylabel('Sayı');
plt.title('Enerji Kullanım Yoğunluğu Dağılımı');
```



Aykırı değerleri çıkardıktan sonra, artık analize geri dönebiliriz.
Bu histogram biraz daha az şüpheli görünüyor ve normalde sağ taraftaki uzun bir kuyrukla dağılmaya yakın(pozitif bir eğriliğe sahip).
Bu daha objektif bir ölçüm olsa da, hedefimiz hala Energy Star Puanını tahmin etmektir, bu yüzden bu değişkeni incelemeye geri döneceğiz. Skor iyibir ölçü olmasa bile, bunu tahmin etmek bizim görevimizdir, bu yüzden görevimizi yapmaya çalışacağız! Dahası, bu projede daha fazla zamanımız olsaydı, ortak bir şeyleri olup olmadığını görmek için 1 ve 100 puanlık binalara bir göz atmak ilginç olabilirdi(Siz deneyebilirsiniz).

Değişkenler Arasındaki ilişkileri kontrol ediyoruz.

Kategorik değişkenlerin puan üzerindeki etkisine bakmak için kategorik değişkenin değeri ile renklendirilmiş bir yoğunluk grafiği çizebiliriz. Yoğunluk grafikleri ayrıca tek bir değişkenin dağılımını gösterir ve düzeltilmiş histogram

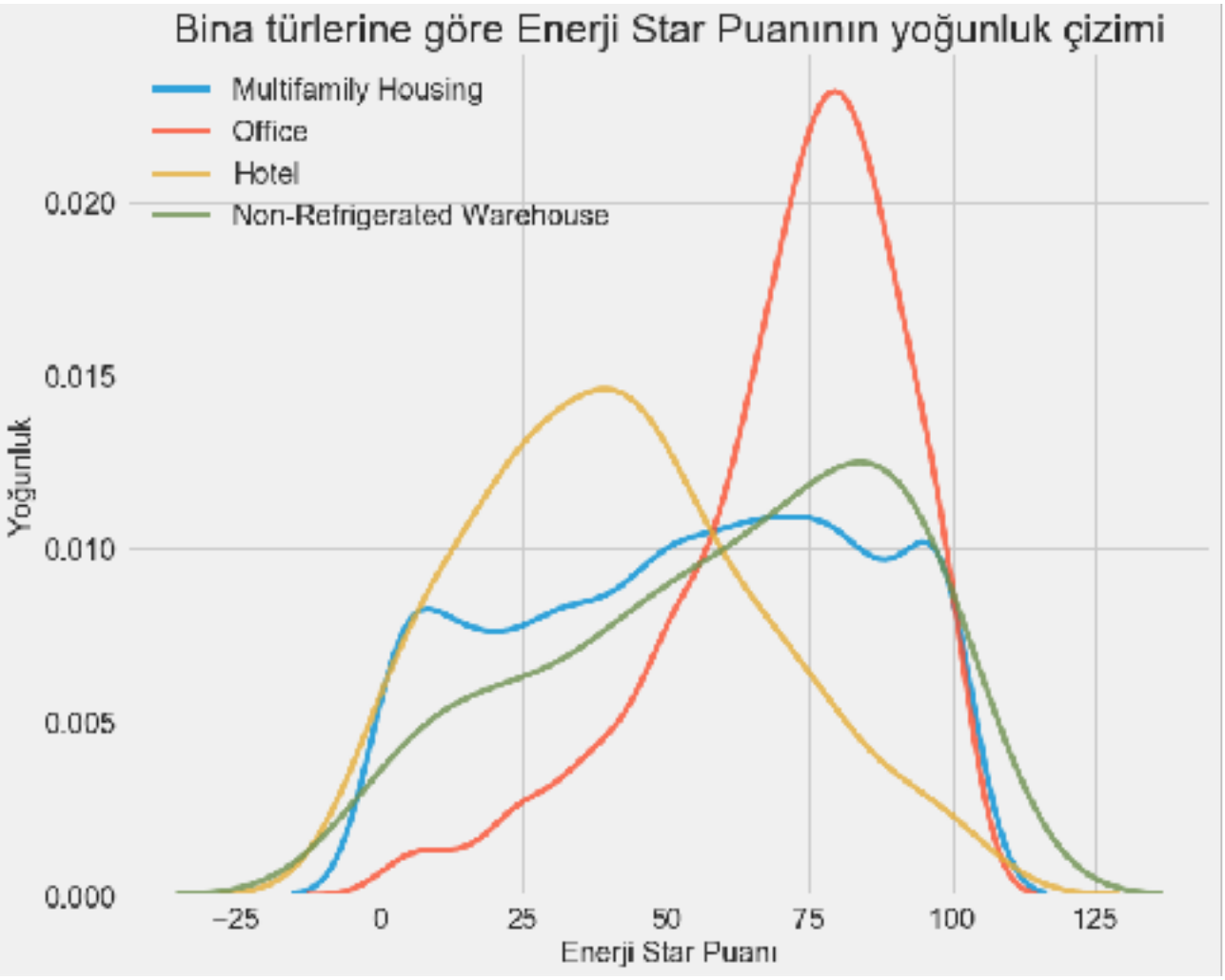
olarak düşünülebilir. Yoğunluk eğrilerini kategorik bir değişkenle renklendirirsek, bu bize dağılımın sınıfa göre nasıl değiştiğini gösterir. Yapacağımız ilk çizim, puanların mülk türüne göre dağılımını gösterecektir. Çizgiyi dağıtmamak için, grafiği veri kümesinde 100'den fazla gözlem içeren yapı türleriyle sınırlayacağız.

In [230]:

```
# Değeri 100 ve üzeri olan bir bina listesi oluşturuyoruz
types = data.dropna(subset = ['score'])
types = types['Primary Property Type - Self
Selected'].value_counts()
types = list(types[types.values > 100].index)
```

In [220]:

```
#Bina kategorilerine göre dağılım puanlarını çizdiriyoruz
figsize(12, 10)

#Herbir bina türü için çizdiriyoruz
for b_type in types:
    #Bina türünü seçiyoruz
    subset = data[data['Primary Property Type - Self
Selected'] == b_type]

    #Enerji Star puanı için yoğunluk çizimi
    sns.kdeplot(subset['score'].dropna(),
                label = b_type, shade = False, alpha =
0.8);
#Çizimimizi etiketlendiriyoruz
plt.xlabel('Enerji Star Puanı', size = 20);
plt.ylabel('Yoğunluk', size = 20);
plt.title('Bina türlerine göre Enerji Star Puanının
yoğunluk çizimi', size  = 28);
```



Bu grafikten, bina tipinin puan üzerinde bir etkisi olduğunu görebiliriz(grafik üzerindeki negatif puanlar, çekirdek yoğunluğu tahmin prosedürünün bir eseridir). İlk başta skoru tahmin etmek için sayısal sütunlara odaklanacağız, o halde, bu grafik bize özellik türünü eklememizi söyler çünkü bu bilgi skoru belirlemek için yararlı olabilir. Yapı tipi kategorik bir değişken olduğundan, bir makine öğrenim modeline dahil edilmeden önce tek kodlu kodlanmış olmalıdır.
Başka bir kategorik değişkeni incelemek için, aynı grafiği yapabiliriz, ancak bu kez grafik başka bir şekilde renklendirilmesi gerekir.

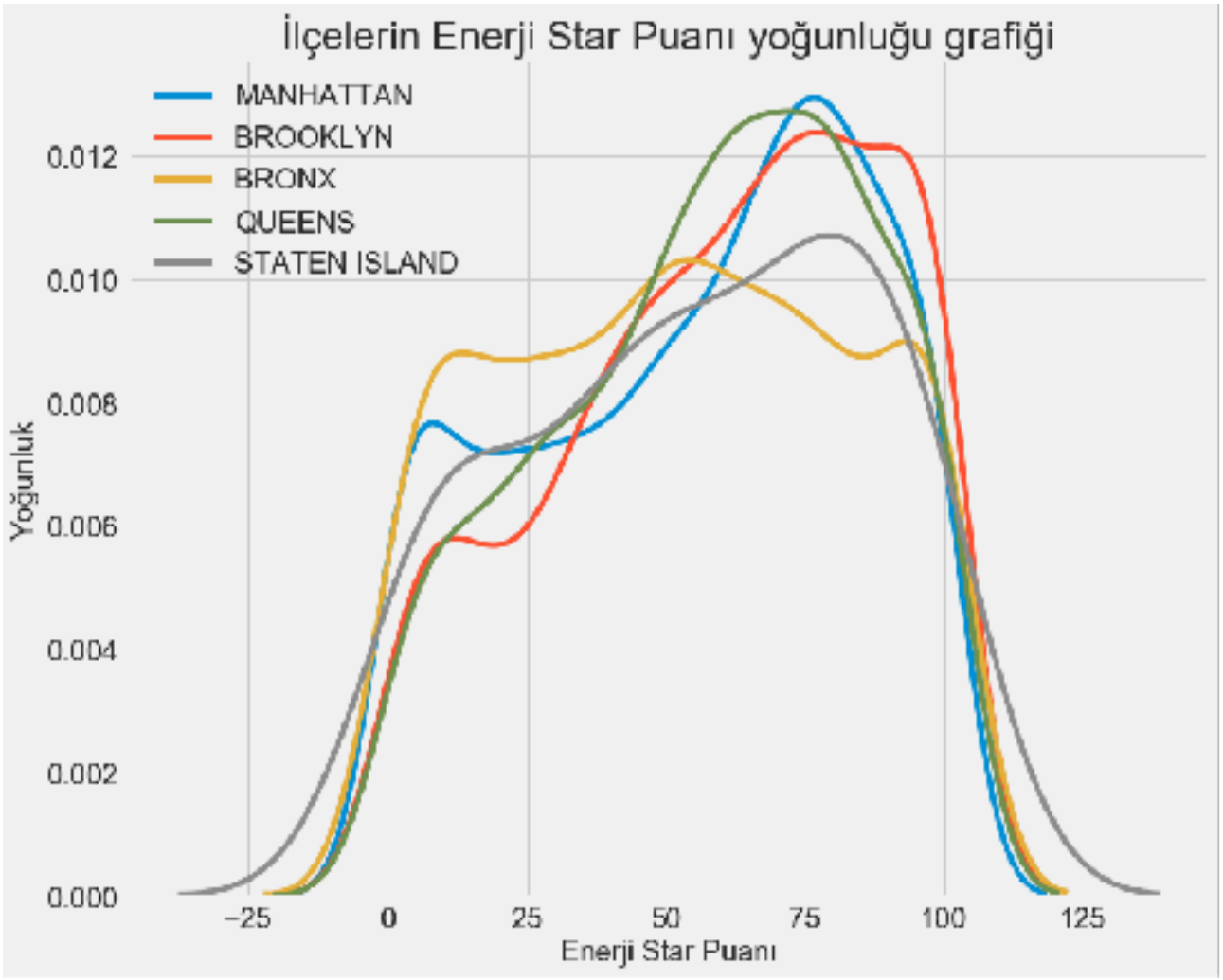
In [231]:

```
#Sayısı 100 gözlemden fazla olan ilçelerin bir listesini
oluşturuyoruz
boroughs = data.dropna(subset=['score'])
boroughs = boroughs['Borough'].value_counts()
boroughs = list(boroughs[boroughs.values > 100].index)
```

In [226]:

```
#İlçelere göre puan dağılımını çizdiriyoruz
figsize(12, 10)
```

```
#Herbir ilçe için dağılım puanını çizdiriyoruz
for borough in boroughs:
    #Bina türünü seçiyoruz
    subset = data[data['Borough'] == borough]

    #Enerji Star Puanı yoğunluğu çizimi
    sns.kdeplot(subset['score'].dropna(),
                label = borough);
#Çizimlerimizi etiketlendiriyoruz
plt.xlabel('Enerji Star Puanı', size = 20);
plt.ylabel('Yoğunluk', size = 20);
plt.title('İlçelerin Enerji Star Puanı yoğunluğu
grafiği', size = 28);
```



İlçe binası, bina tipi kadar, skor dağılımında da önemli bir fark yaratmıyor gibi görünmektedir. Bununla birlikte, ilçeyi kategorik bir değişken olarak dahil etmek mantıklı olabilir.

Özellikler ve Hedef Arasındaki Korelasyonlar

Özellikler (değişkenler) ile hedef arasındaki korelasyonları ölçmek için Pearson korelasyon katsayısını hesaplayabiliriz. Bu, iki değişken arasındaki doğrusal bir ilişkinin gücü ve yönünün bir ölçüsüdür: -1 değeri, iki değişkenin, mükemmel negatif olarak doğrusal korelasyonu olduğu ve bir +1 değerinin, iki değişkenin, mükemmel pozitif olarak doğrusal ilişkili olduğu anlamına gelir. Aşağıdaki şekil, korelasyon katsayısının farklı değerlerini ve grafiksel olarak nasıl göründüğünü göstermektedir.

Özellikler ve hedefler arasında doğrusal olmayan ilişkiler olsa da ve korelasyon katsayıları özellikler arasındaki etkileşimleri hesaba katmasa da, doğrusal ilişkiler verilerdeki eğilimleri keşfetmeye başlamak için iyi bir yoldur. Daha sonra modelimizde istihdam edilecek özellikleri seçmek için bu değerleri kullanabiliriz.
Aşağıdaki kod, tüm değişkenler ve skor arasındaki korelasyon katsayılarını hesaplar.

In [ ]:

```
# Tüm korelasyonları bulup sıralıyoruz
correlations_data = data.corr()[' score'].sort_values()

# Negatif korelasyonu en yüksek olan
print(correlations_data.head(15),'\n')

# Pozitif Korelasyonu en yüksek olan
print(correlations_data.tail(15))
```

Özellikler ve hedef arasında birkaç güçlü negatif korelasyon vardır. Skorla en olumsuz korelasyonlar, Enerji Kullanım Yoğunluğu'nun (EUI), Site EUI'sinin (kBtu/ft²) ve Hava Normalleştirilmiş Yerleşkenin EUI'sının (kBtu/ft²) farklı kategorileridir (bunlar, nasıl hesaplandıklarına göre biraz değişir). EUI, binaların kare görüntüleri ile bölünen bir binanın kullandığı enerji miktarıdır ve daha düşük bir puanın daha iyi olduğu bir binanın verimliliğinin bir ölçüsüdür. Sezgisel olarak, bu korelasyonlar daha mantıklı: EUI arttıkça, Enerji Yıldızı Puanı düşme eğilimindedir.
Olası doğrusal olmayan ilişkileri hesaba katmak için, özelliklerin karekökünü ve doğal log dönüşümlerini alabilir ve daha sonra puan ile korelasyon katsayılarını hesaplayabiliriz. İlçe veya bina tipi arasındaki olası ilişkileri denemek ve yakalamak için (bunların kategorik değişkenler olduğunu unutmayın) ve bu sütunları one-hot kodlamak zorunda kalacağız.
Aşağıdaki kodda, sayısal değişkenlerin log ve karekök dönüşümlerini, seçilen iki kategorik değişkeni (bina tipi ve ilçe) one-hot kodlama, tüm özellikler ve puan arasındaki korelasyonları hesaplayıp ve en yükse 15 olumlu korelasyon ile en iyi 15 negatif korelasyon sonuçlarını görüntülüyoruz. Bu işlemleri pandas ile yapmak göreceğiniz üzere kolay!

In [ ]:

```
#Sayısal değer içeren sütunları seçiyoruz
numeric_subset = data.select_dtypes('number')

#Sayısal sütunlardan karekök ve logaritmayı birlikle
oluşturacak bir sütun oluşturuyoruz
for col in numeric_subset.columns:
    #Enerji Star Puanı sütununu yoksayıyoruz
    if col == 'score':
        next
    else:
        numeric_subset['sqrt_' + col] =
np.sqrt(numeric_subset[col])
        numeric_subset['log_' + col] =
np.log(numeric_subset[col])
#Şimdi de kategorik veri içeren sütunları seçiyoruz
categorical_subset = data[['Borough', 'Primary Property
Type - Self Selected']]

#One hot kodlama
categorical_subset = pd.get_dummies(categorical_subset)

#concat kullanarak iki dataframe'i birleştiriyoruz,
sütunları birlşetirmek için axis = 1 yapınız
features = pd.concat([numeric_subset,
categorical_subset], axis = 1)

#Enerji Yıldız puanı olmayan binaları veri setinden
çıkarıyoruz
features = feature.dropna(subset =['score'])

#Şimdi Özellikler arasındaki korelasyona bakalım
correlation = features.corr()
['score'].dropna().sort_values()
```

In [ ]:

```
#Korelasyonu en olumsuz olan 15 tane özelliği
görüntülüyoruz
correlations.head(15)
```

In [ ]:

```
# Korelasyonu en iyi olan 15 özelliği görüntülüyoruz
correlations.tail(15)
```

Özellikleri dönüştürdükten sonra, en güçlü ilişkiler hala Enerji Kullanımı Yoğunluğu (EUI) ile ilgili olanlardır. Log ve karekök dönüşümleri, herhangi bir

güçlü ilişki ile sonuçlanmış gibi görünmüyor. Bina tipi bir ofisin (En Büyük Mülkiyet Kullanım-, Ofis Türü) puanla biraz pozitif ilişkili olduğunu görmemize rağmen güçlü olumlu doğrusal ilişkiler yoktur. Bu değişken, bina tipi için kategorik değişkenlerin one-hot kodlanmış bir temsilidir.
Bu seçimi, özellik seçimi yapmak için kullanabiliriz (birazdan geliyor). Şu anda, Site EUI (kBtu / ft ^ 2) olan veri kümesindeki en önemli korelasyonu (mutlak değer açısından) çizelim. Grafiği ilişkiyi nasıl etkilediğini göstermek için yapı tipine göre renklendirebiliriz.

İki Değişkenli Çizimler

İki değişken arasındaki ilişkiyi görselleştirmek için bir dağılım grafiği kullanırız. İşaretleyicilerin rengi veya işaretleyicilerin boyutu gibi özellikleri kullanarak ek değişkenler de ekleyebiliriz. Burada iki sayısal değişkeni birbiriyle karşılaştırır ve üçüncü kategorik bir değişkeni temsil etmek için renk kullanırız.

In [ ]:

```
figsize(12, 10)

#Bina Türlerini Ayıklıyoruz
features['Primary Property Type - Self Selected'] =
data.dropna(subset = ['score'])['Primary Property Type -
Self Selected']

#100'den fazla gözlemi bina türleri ile sınırlıyoruz
features = features[features['Primary Property Type -
Self Selected'].isin(types)]

#Seaborn kütüphanesini kullanarak Puan ve Lo değerlerinin
saçılım grafiğini çizdiriyoruz
sns.lmplot('Site EUI(kBtu/ft2)', 'score',
           hue = 'Primary Property Type - Self Selected',
data = features,
           scatter_kws = {'alpha' : 0.8, 's': 60}, fir_reg
= False,
           size = 12, aspect = 1.2);
#Son Olarak Şekillerimizi etiketlendirelim
plt.xlabel('Site EUI', size = 28)
plt.ylabel('Enerji Yıldız Puanı', size = 28)
plt.title('Enerji Yıldız Puanı& Site EUI', size = 36);
```

Site EUI'si ile skor arasında açık bir negatif ilişki vardır. İlişki mükemmel doğrusal değildir (korelasyon katsayısı -0.7 ile gözükmektedir, ancak bu özellik bir binanın skorunu tahmin etmek için önemli gibi görünmektedir).

Çiftli Çizim(Pairs Plot)

Keşifsel veri analizi için son bir alıştırma olarak, birkaç farklı değişken arasında çiftler çizebiliriz. Çiftler Çizgisi(Pairs Plot), değişkenlerin çiftleri ve diyagonaldeki tek değişkenli histogramlar arasındaki dağılımları gösterdiği anda birçok değişkeni aynı anda incelemenin harika bir yoludur.
Seaborn PairGrid fonksiyonunu kullanarak, farklı çizimleri gridin üç yönüne göre haritalayabiliriz. Üst üçgenin dağılma noktaları olacak, diyagonal histogramları gösterecek ve alt üçgen iki değişken arasındaki korelasyon katsayısını ve iki değişkenin 2-D çekirdek yoğunluk tahminini gösterecektir.

In [ ]:

```
# Sürunları çizdirmek için ayıklıyoruz
plot_data = features[['score', 'Site EUI(kBtu/ft2)',
                      'Weather Normalized Source EUI
kBtu/ft²)',
                      'log_Total GHG Emissions(MtCO2e)']]

#  inf değerlerini nan ile değiştiriyoruz
plot_data = plot_data.replace({np.inf: np.nan, -np.inf:
np.nan})

# Sütunları yeniden isimlendiriyoruz
plot_data = plot_data.rename(columns = {'Site EUI (kBtu/
ft²)': 'Site EUI',
                                        'Weather
Normalized Source EUI (kBtu/ft²)': 'Weather Norm EUI',
                                        'log_Total GHG
Emissions (Metric Tons CO2e)': 'log GHG Emissions'})

# na Değerlerini verisetinden ayırıyoruz
plot_data = plot_data.dropna()

# İki sütun arasındaki korelasyon katsayısını
hesaplayacak fonksiyonu yazıyoruz
def corr_func(x, y, **kwargs):
    r = np.corrcoef(x, y)[0][1]
    ax = plt.gca()
    ax.annotate("r = {:.2f}".format(r),
                xy=(.2, .8), xycoords=ax.transAxes,
                size = 20)

# pairgrid nesnesi oluşturuyoruz
grid = sns.PairGrid(data = plot_data, size = 3)

# Üste saçılım grafiği
```

```
grid.map_upper(plt.scatter, color = 'red', alpha = 0.6)

# Çapraza histogram
grid.map_diag(plt.hist, color = 'red', edgecolor =
'black')

#  Altta, korelasyon  ve yoğunluk çizimi
grid.map_lower(corr_func);
grid.map_lower(sns.kdeplot, cmap = plt.cm.Reds)

# Tüm çizim için başlığımız
plt.suptitle('Enerji Verisi için Çiftsel Çizim', size =
36, y = 1.02);
```

Çizelgedeki ilişkileri yorumlamak için, bir satırdaki değişkenlerin bir sütundaki değişkenlerle kesiştiği yerleri arayabiliriz. Örneğin, puan ile GHG Emisyonları günlüğü arasındaki ilişkiyi bulmak için skor sütununa bakar ve log GHG Emissions satırını buluruz. Kesişim noktasında (alt sol arsa), puanın bu değişkenle -0.35 korelasyon katsayısına sahip olduğunu görürüz. Sağ üst grafiğe bakarsak, bu ilişkinin bir dağılım grafiğini görebiliriz.

Özellik Mühendisliği ve Seçimi

Artık verilerdeki eğilimleri ve ilişkileri araştırdık, modellerimiz için bir dizi özellik mühendisliği üzerinde çalışabiliriz. Bu özellik mühendisliği için EDA'nın sonuçlarını kullanabiliriz. Özellikle, mühendislik / seçim özelliklerinde bize yardımcı olabilecek EDA'dan aşağıdakileri öğrendik:

- Skor dağılımı bina tipine göre ve ilçe tarafından daha az ölçüde değişmektedir. Sayısal özelliklere odaklanmamıza rağmen, bu iki kategorik özelliği de modele dahil etmeliyiz.
- Özelliklerin log dönüşümünün alınması, özellikler ve skor arasındaki doğrusal korelasyonlarda önemli artışlara yol açmaz.

Daha fazla ilerlemeden önce, özellik mühendisliği ve seçimin ne olduğunu tanımlamalıyız! Bu tanımlamalar gayri resmi ve büyük ölçüde çakışıyor, ancak bunları iki ayrı süreç olarak düşünmeyi seviyorum:

- Özellik Mühendisliği: Ham veri alma ve bir makine öğrenme modelinin bu özellikleri ve hedefi belirleyen bir haritalamayı öğrenmesini sağlayan yeni özellikler çıkarma veya oluşturma süreci. Bu, log ve karekök ile yaptığımız gibi değişkenlerin dönüşümlerini almak veya bir modelde kullanılabilmeleri için tek-kodlayıcı kategorik değişkenleri ifade etmek anlamına gelebilir. Genel olarak, özellik mühendisliğini ham verilerden türetilen ek özelliklerdir.
- Özellik Seçimi: Verilerinizdeki en uygun özellikleri seçme işlemidir. "En alakalı" birçok faktöre bağlı olabilir, ancak hedefle en yüksek korelasyon kadar basit veya en fazla varyansa sahip özellikler olabilir.

Özellik seçiminde, modelimizin özellikler ve hedef arasındaki ilişkiyi öğrenmesine yardımcı olmayan özellikleri çıkarırız. Bu, modelin yeni verilere daha iyi genelleştirilmesine ve daha yorumlanabilir bir modelle sonuçlanmasına yardımcı olabilir. Genel olarak, özellik seçimini çıkarma özellikleri olarak düşünürüm, çünkü özellik seçimi sonunda sadece en önemli olanlar kalıy elimizde.

Özellik mühendisliği ve seçimi, genellikle doğru bir şekilde yapılması gereken birkaç girişime ihtiyaç duyan iteratif süreçlerdir. Çoğu zaman, rastgele bir ormandaki(random forest) özellik itirazları gibi modelleme sonuçlarını geri dönüp, özellik seçimini tekrar yapmak için kullanırız veya daha sonra yeni değişkenler oluşturmayı gerektiren ilişkileri keşfedebiliriz. Ayrıca, bu süreçler genellikle verilerin bir alan bilgisi ve istatistik nitelikleri karışımını içerir. Özellik mühendisliği ve seçimi genellikle bir makine öğrenim problemine ayrılan zamanın en yüksek getirilerine sahiptir. Doğru olması biraz zaman alabilir, ancak genellikle model için kullanılan tam algoritma ve hipermetrelerden daha önemlidir. Eğer modeli doğru verilerle beslemezsek, o zaman başarısızlığa ayarlıyoruz demektir ve modelin iyi şeyler öğrenmesini beklememeliyiz!

Bu projede, özellik mühendisliği için aşağıdaki adımları yapacağız:

a- Sadece sayısal değişkenleri ve iki kategorik değişkeni (ilçe ve özellik kullanım tipi) seçiyoruz b- Sayısal değişkenlerin log dönüşümünü ekliyoruz c- Kategorik değişkenleri one-hot olarak kodluyoruz

Özellik seçimi için aşağıdakileri yapacağız,

a- Eşleme(collinearity) özelliklerini kaldırıyoruz.

Biz bu sürece vardığımızda collinearity (multiicollinearity olarak da adlandırılır) tartışacağız!

Aşağıdaki kod, sayısal özellikleri seçer, tüm sayısal özelliklerin log dönüşümlerine ekler, one-hot kategorik özellikleri kodlar ve birlikte özellikler kümesine katılır.

In [ ]:

```
#Orjinnal verileri kopyalıyoruz
features = data.copy()

#Sayısal sütunları seçiyoruz
numeric_subset = data.select_dtypes('number')

#Sayısal sütunları logaritmik olarak dönüştürüyoruz
for col in numeric_subset.columns:
    #Enerji Yıldıs Puanı sütununu daha öncede yaptığımız
gibi işleme almıyoruz
    if col == 'score':
        next
    else:
```

```
        numeric_subset['log_' + col] =
np.log(numeric_subset[col])

#Kategorik sütunları seçiyoruz
categorical_subset = data[['Borough', 'Primary Property
Type - Self Selected' ]]

#one-hot kodlama
categorical_subset = pd.get_dummies(categorical_subset)

#concatanete(birleştirme) komutunu kullanarak iki veri
çerçevesini birleştireceğiz
#axis = 1 seçiyoruz çünkü sütunlar üzerinde işlem
yapıyoruz
features = pd.concat([numeric_subset,
categorical_subset], axis = 1)
```

Bu noktada 109 farklı özellikte 11319 gözlem (bina) var (bir sütun puan). Bu özelliklerin hepsinin skoru tahmin etmek için önemli olma ihtimali yoktur ve bu özelliklerin birçoğu da yüksek oranda korundukları için gereksizdir. Aşağıda bu ikinci konuyla ilgileneceğiz.

Eşdoğrusal(Collinear) Özellikleri Kaldırma

Yüksek collinear özelliklerin aralarında anlamlı bir korelasyon katsayısı vardır. Örneğin, veri kümemizde, Site EUI ve Hava Normları EUI, enerji kullanımı yoğunluğunun hesaplanmasından sadece biraz farklı olduğu için yüksek derecede ilişkilidirler.

In [ ]:

```
# DİKKKKAAATTT Burada çizdirme işlemlerindeki ilgili
sütunları bulup değiştireceğiz
plot_data = dat[['Weather Normalized Site EUI (kBtu/
ft²)', 'Site EUI (kBtu/ft²)']].dropna()

plt.plot(plot_data['Site EUI (kBtu/ft²)'],
plot_data['Weather Normalized Site EUI (kBtu/ft²)'],
'bo')
plt.xlabel('Site EUI'); plt.ylabel('Weather Norm EUI')
plt.title('Weather Norm EUI vs Site EUI, R = %0.4f' %
np.corrcoef(data[['Weather Normalized Site EUI (kBtu/
ft²)', 'Site EUI (kBtu/ft²)']].dropna(), rowvar=False)[0]
[1]);
```

Bir veri kümesindeki değişkenler genellikle küçük bir dereceyle ilişkili olsa da, modele gerekli bilgileri vermek için sadece özelliklerden birini korumamız gerektiği anlamında yüksek oranda çapraz değişkenler gereksiz olabilir. Kolinear özelliklerin kaldırılması, özelliklerin sayısını azaltarak model karmaşıklığını azaltmanın ve model genellemesini artırmaya yardımcı olan bir yöntemdir. Aynı zamanda modeli yorumlamamıza da yardımcı olabilir, çünkü hem EUI hem de hava durumu normalleştirilmiş EUI'nın skoru nasıl etkilediğinden ziyade, EUI gibi tek bir değişken hakkında endişelenmemiz gerekir.
Varyans Enflasyon Faktörünün kullanılması gibi collinear(eşdoğgrusal) özelliklerin kaldırılması için çeşitli yöntemler vardır. Daha basit bir metrik kullanacağız ve birbiriyle belirli bir eşiğin üzerinde bir korelasyon katsayısına sahip olan özellikleri kaldıracağız (skorla değil, skorla yüksek oranda korelasyon gösteren değişkenler istiyoruz!) Kolinear değişkenleri kaldırma hakkında daha kapsamlı bir tartışma için, https://www.kaggle.com/robertoruiz/dealing-with-multicollinearity/code[Kaggle'da bu not defterini kontrol edebilirsiniz.]
Aşağıdaki kod, karşılaştırılan iki özellikten birini kaldırarak korelasyon katsayıları için seçtiğimiz bir eşiğe dayalı olarak collinear özellikleri kaldırır. Ayrıca, eşiği ayarlamanın etkisini görebilmemiz için çıkardığı korelasyonları da yazdırır. Özelliklerin arasındaki korelasyon katsayısı bu değeri aşarsa, bir çift özellikten birini kaldıran 0,6 eşiğini kullanırız.
Yine, bu kodu aslında sıfırdan yazmamıştım, bunun yerine bir [https://stackoverflow.com/questions/29294983/how-to-calculate-correlation-between-all-columns-and-remove-highly-correlated-on/43104383#43104383] den uyarladım.

In [ ]:

```
def remove_collinear_features(x, threshold):
    '''
    Amaç:
        Korelasyon katsayısına sahip bir dataframe'deki eşikten daha büyük olan collinear özellikleri kaldırmak
        Modelin yorumlanabilirliğini genelleştirmek ve geliştirmek eşleme özelliklerinin kaldırılması bir modele
        yardımcı olabilir.

    Girişler:
        Eşik değeri: Korelasyon ilişkisi bu değerden büyük olanları sil

    Çıkış   :
        Veriçerçevesi sadece yüksek olmayan eşdoğrusal değerler içerir
    '''
```

```
    #Enerji Yıldızz Puanları arasındaki korelasyonları
silmek istemiyoruz
    y = x['score']
    x = x.drop(columns = ['score'])

    #Korelasyon matrixi hesaplıyoruz
    corr_matrix = x.corr()
    iters = range(len(corr_matrix.columns) - 1)
    drop_cols = []

    #Korelasyon matrixksini uygulayıp, korelasyon
ilişkilerini karşılaştırıyoruz
    for i in iters:
        for j in range(i):
            item = corr_matrix.iloc[j:(j+1), (i+1):(i+2)]
            col = item.columns
            row =item.index
            val =abs.(item.values)

            #Korelasyon eğer eşik değerini aşarsa
            if val >= threshold:
                #Korelasyon değerlerini yazdır
                #print(col.values[0], "/", row.values[0],
"/", round(val[0][0], 2))
                drop_cols.append(col.values[0])

    #Kolelasyonlu çiftleri çıkarıyoruz
    drops = set(drop_cols)
    x = x.drop(columns = drops)
    x = x.drop(columns = ['Weather Normalized Site EUI
(kBtu/ft²)',
                          'Water Use (All Water Sources)
(kgal)',
                          'log_Water Use (All Water
Sources) (kgal)',
                          'Largest Property Use Type -
Gross Floor Area (ft²)'])

    #Puanları  artık verilere geri ekleyebiliriz
    x['score'] = y

    return x
```

In [ ]:
```
#Belirli bir korelasyon katsayısının üzerindeki collinear özelliklerini kaldırın
featureas = remove_collinear_features(feature, 0.6);
```

In [ ]:
```
#na içeren tüm sütunları siliyoruz
features = features.dropna(axis = 1, how = 'all')
features.shape
```

Son veri kümemiz şimdi ?? özelliğe sahiptir (sütunlardan biri hedeftir). Bu hala oldukça azdır, ancak çoğunlukla, bir kategorik değişkenleri kodlayan bir tane sıcaklığımız vardır. Ayrıca, doğrusal regresyon gibi modeller için çok sayıda özellik sorunlu olsa da, rasgele orman gibi modeller örtük özellik seçimi gerçekleştirmekte ve otomatik olarak hangi özelliklerin traning sırasında önemli olduğunu belirlemektedir. Almak için diğer özellik seçimi adımları vardır, ancak şimdilik sahip olduğumuz tüm özellikleri koruyacak ve modelin nasıl performans gösterdiğini göreceğiz.

Ek Özellik Seçimi

Özellik seçimi için daha fazla yöntem var. Bazı popüler yöntemler, özellikleri en büyük varyansı koruyan daha az sayıda boyuta ya da bir takım özelliklerdeki bağımsız kaynakları bulmayı amaçlayan bağımsız bileşenler analizine (ICA) dönüştüren temel bileşenler analizini (PCA) içerir. Bununla birlikte, bu yöntemler özelliklerin sayısını azaltmada etkili olsa da, fiziksel bir anlamı olmayan ve dolayısıyla bir modeli yorumlamayı neredeyse imkansız kılan yeni özellikler yaratırlar.
Bu yöntemler, yüksek boyutlu verilerle uğraşmak için çok faydalıdır ve makine öğrenimi problemleriyle uğraşmayı planlıyorsanız, konu hakkında daha fazla bilgi edinmenizi tavsiye ederim!
link 1 -> [http://scikit-learn.org/stable/modules/feature_selection.html] link 2 -> [http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf] link 3 -> [http://cs229.stanford.edu/notes/cs229-notes11.pdf]

Eğitim ve Test Setlerine Bölünmüş

Makine öğrenmesinde, özelliklerimizi her zaman iki gruba ayırmamız gerekir- Eğitim ve Test olarak:
Eğitim sırasında modelimize verdiğimiz eğitim setinin yanı sıra, cevaplar ve hedefler arasında bir eşleme yapabilmeyi öğrenebilmekteyiz. Model tarafından öğrenilen haritalamayı değerlendirmek için kullandığımız test seti. Model test setindeki cevapları hiç görmemiş, bunun yerine yalnızca özellikleri kullanarak tahminlerde bulunmalıdır. Test setinin gerçek cevaplarını bildiğimiz gibi, gerçek dünyaya uyarlandığında modelimizin ne kadar iyi performans göstereceğine dair bir tahminde bulunmak için test tahminlerini gerçek test hedefleriyle karşılaştırabiliriz. Bizim sorunumuz için öncelikle Energy Star

Puanı olmaksızın tüm binaları çıkaracağız (bu binalar için doğru cevabı bilmeyiz, böylece eğitim veya test için yardımcı olmayacaktır). Ardından, binaları bir Energy Star Skoru ile binaların% 30'unu kapsayan bir test setine ve binaların% 70'ine yönelik bir eğitim setine bölebiliriz.
Verileri rastgele bir eğitime ayırmak ve test setini scikit-learn kullanarak yapmak oldukça basittir. Tutarlı sonuçlar sağlamak için bölünmenin rastgele durumunu ayarlayabiliriz.

In [ ]:
```
#Binaları Puansız ve Puanlı olarak ayıklıyoruz
no_score = features[features['score'].isna()]
score = features[features['score'].notnull()]

print(no_score.shape)
print(score.shape)
```

In [ ]:
```
#Özellik ve Hedef değerlerini ayırıyoruz
features = score.drop(columns = 'score')
targets = pd.DataFrame(score['score'])

#Bazı değerlerin isimlerini değiştiriyoruz
features = features.replace({np.inf: np.nan, -np.inf:
np.nan})

#Evet Veri setimizi %30 Test verisi ve %70 Eğitim verisi
olacak şekilde ayırıyoruz
X, X_test, y, y_test = train_test_split(features,
targets, test_size = 0.3, random_state = 42)

print(X.shape)
print(X_test.shape)
print(y.shape)
print(y_test_shape)
```

Puansız !!!!!(çıktı sonuçlarını yazacağız) bina, antrenman setinde 6622 bina ve test setinde 2839 bina var. Bu not defterine girmek için son bir adımımız var: Modellerimizin yenilmesi için saf bir temel belirleme!

Temel Bir Model Kurma

Makine öğrenimi modellerine başlamadan önce saf bir temel oluşturmak önemlidir. Oluşturduğumuz modeller saf bir tahminden daha iyi performans gösteremezse, o zaman o makine öğreniminin bu sorun için uygun olmadığını itiraf etmeliyiz. Bunun nedeni doğru modelleri kullanmamanız olabilir, çünkü daha fazla veriye ihtiyacımız var, ya da makine öğrenimi gerektirmeyen daha

basit bir çözüm var. Bir taban çizgisi oluşturmak çok önemlidir, bu yüzden biz sadece problemi çözemediğimizin farkına varmak için bir makine öğrenme modeli oluşturmaya son vermeyiz.
Bir regresyon görevi için, iyi bir naif taban çizgisi, test setindeki tüm örnekler için eğitim setindeki hedefin medyan değerini tahmin etmektir. Bu, bizim modellerimiz için nispeten düşük bir bar uygulamak ve ayarlamak için basittir: eğer medin değeri tahmin etmekten daha iyi bir şey yapamazlarsa, o zaman yaklaşımımızı yeniden gözden geçirmemiz gerekecektir.

Metrik: Mutlak Hata Ortalama

Makine öğrenimi görevlerinde kullanılan birtakım metrikler vardır ve hangisinin seçileceğini bilmek zor olabilir. Çoğu zaman, belirli bir soruna bağlı olacak ve belirli bir hedefe sahip olmanız gerekiyorsa dahada zor olacaktır. Değerlendirme sürecini basitleştirdiği için, modelleri karşılaştırmak için Andrew Ng'nin tek bir gerçek değer performans ölçüsü kullanmanızı tavsiye ediyorum. Birden çok metriği hesaplamaktan ve her birinin ne kadar önemli olduğunu belirlemekten ziyade, tek bir sayı kullanmamız gerekir. Bu durumda, regresyon yaptığımız için, ortalama mutlak hata(Mean Absolute Error) uygun bir metriktir. Bu aynı zamanda yorumlanabilir çünkü hedef değerindeki aynı birimlerde tahmin edilen ortalama tutarı temsil eder.
Aşağıdaki işlev, gerçek değerler ve tahminler arasındaki ortalama mutlak hatasını hesaplar.

In [ ]:

```
#Ortalama Mutlak Hata oranı hesaplama fonksiyonumuz
def mae(y_true, y_pred):
    return np.mean(abs(y_true - y_pred))
```

Şimdi ortanca tahmini yapabilir ve test setinde değerlendirebiliriz.

In [ ]:

```
baseline_guess = np.median(y)

print('Tahmin edilen temel puan : %0.2'f %
baseline_guess)
print('Test verileri üzerinde modelimizin temel
performansı : MAE = %0.4f' % (mae(y_test,
baseline_guess))
```

Bir üsteki kodun çıktısına göre değerlendirme yorumu yapacağız¶

Bu, test setindeki ortalama tahmimizin yaklaşık 25 puan olduğunu gösterir. Puanlar 1 ile 100 arasındadır, yani bu yaklaşık% 25 ise naif bir yöntemden ortalama hata anlamına gelir. Medyan eğitim değerini tahmin etmenin saf yöntemi, modellerimizin yenmesi için bize düşük bir temel sağlar!

Sonuçlar

Bu defterde, bir makine öğrenim probleminin ilk üç adımını gerçekleştirdik:

1- Ham verileri temizleyip biçimlendirdik
2- Bir keşifsel veri analizi gerçekleştirdik
3- Özellik mühendisliği ve özellik seçimi ile modelimizi eğitmek için bir dizi özellik geliştirildik

Ayrıca, bir temel metrik oluşturmanın çok önemli bir görevini de tamamladık, böylece modelimizin tahmin etmekten daha iyi olup olmadığını belirleyebiliriz! İnşallah, işlem basamaklarının her bir parçasının bir sonraki aşamaya nasıl aktığına dair bir fikir edinmeye başlıyorsunuz: verileri temizlemek ve doğru formatta almak, bir keşifsel veri analizini gerçekleştirmemize olanak sağlıyor. EDA daha sonra özellik mühendisliği ve seçim aşamasında kararlarımızı bilgilendirir. Bu üç aşama genellikle bu sırayla gerçekleştirilir, ancak daha sonra tekrar gelebilir ve modelleme sonuçlarımıza dayanarak daha fazla EDA veya özellik mühendisliği yapabiliriz. Veri bilimi, her zaman önceki çalışmamızı iyileştirmenin yollarını aradığımız yinelemeli bir süreçtir. Bu, ilk kez mükemmel şeyler elde etmek zorunda olmadığımız anlamına gelir (elimizden gelenin en iyisini yapmaya çalışsak da) çünkü sorun hakkında daha fazla bilgiye sahip olduktan hemen sonra kararlarımızı tekrar gözden geçirme fırsatları vardır.
İkinci bölümde, birkaç makine öğrenim yönteminin uygulanmasına, en iyi modelin seçilmesine ve çapraz doğrulama ile hiperparametre ayarlaması kullanılarak sorunumuz için optimize edilmeye odaklanacağız. Son bir adım olarak, geliştirdiğimiz veri kümelerini bir sonraki bölümde tekrar kullanacağız.