Описание задачи

Прежде чем писать код, необходимо разобраться в решаемой задаче и доступных данных. В этом задании будем работать с выложенными в общий доступ данными об энергоэффективности зданий в Нью-Йорке.

Цель: использовать имеющиеся данные для построения модели, которая прогнозирует количество баллов Energy Star Score для конкретного здания, и интерпретировать результаты для поиска факторов, влияющих на итоговый балл.

Данные уже включают в себя присвоенные баллы Energy Star Score, поэтому наша задача представляет собой машинное обучение с управляемой регрессией:

- Управляемая (Supervised): нам известны признаки и цель, и наша задача — обучить модель, которая сможет сопоставить первое со вторым.
- Регрессия (Regression): балл Energy Star Score — это непрерывная переменная.

Наша модель должна быть точная— чтобы могла прогнозировать значение Energy Star Score близко к истинному,— и интерпретируемая— чтобы мы могли понять её прогнозы. Зная целевые данные, мы можем использовать их при принятии решений по мере углубления в данные и создания модели.

Очистка данных

Далеко не каждый набор данных представляет собой идеально подобранное множество наблюдений, без аномалий и пропущенных значений. В реальных данных мало порядка, так что прежде чем приступить к анализу, их нужно очистить и привести к приемлемому формату. Очистка данных — неприятная, но обязательная процедура при решении большинства задач по анализу данных.

Сначала можно загрузить данные в виде кадра данных (dataframe) Pandas и изучить их:

```
import pandas as pd
import numpy as np

# Read in data into a dataframe
data =
pd.read_csv('data/Energy_and_Water_Data_Disclosure_for_Local_Law_84_2017__Data_for_Calendar_Year_2016_.csv')

# Display top of dataframe
data.head()
```

3rd Largest Property Use Type - Gross Floor Area (ft²)	Year Built	Number of Buildings - Self-reported	Occupancy	Metered Areas (Energy)	Metered Areas (Water)	ENERGY STAR Score	Site EUI (kBtu/ft²)	Weather Normalized Site EUI (kBtu/ft²)	Weather Normalized Site Electricity Intensity (kWh/ft²)	Weather Normalized Site Natural Gas Intensity (therms/ft²)
Not Available	1963	2	100	Whole Building	Not Available	Not Available	305.6	303.1	37.8	Not Available
Not Available	1969	12	100	Whole Building	Whole Building	55	229.8	228.8	24.8	2.4
Not Available	1924	1	100	Not Available	Not Available	Not Available	Not Available	Not Available	Not Available	Not Available

Так выглядят реальные данные.

Это фрагмент таблицы из 60 колонок. Даже здесь видно несколько проблем: нам нужно прогнозировать Energy Star Score, но мы не знаем, что означают все эти колонки. Хотя это не обязательно является проблемой, потому что зачастую можно создать точную модель, вообще ничего не зная о переменных. Но нам важна интерпретируемость, поэтому нужно выяснить значение как минимум нескольких колонок.

Воспользовавшись поисковиком и названием файл, можно понять, что последний составлен в соответствии с действующим в Нью-Йорке законом, согласно которому владельцы всех зданий определённого размера должны отчитываться о потреблении энергии. Далее представлены все значения колонок. Так что не пренебрегайте именами файлов, они могут быть хорошей отправной точкой.

Мы не будем изучать все колонки, но точно разберёмся с Energy Star Score, которая описывается так:

Ранжирование по перцентили от 1 до 100, которая рассчитывается на основе самостоятельно заполняемых владельцами зданий отчётов об энергопотреблении за год. Energy Star Score — это относительный показатель, используемый для сравнения энергоэффективности зданий.

Первая проблема решилась, но осталась вторая — отсутствующие значения, помеченные как «Not Available». Это строковое значение в Python, которое означает, что даже строки с числами будут храниться как типы данных object, потому что, если в колонке есть какое-нибудь строковое значение, Pandas конвертирует её в колонку, полностью состоящую из строковых. Типы данных колонок можно узнать с помощью метода dataframe.info():

```
# See the column data types and non-missing values
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11746 entries, 0 to 11745
Data columns (total 60 columns):
Order                                                    11746 non-null int64
Property Id                                              11746 non-null int64
Property Name                                            11746 non-null object
Parent Property Id                                       11746 non-null object
Parent Property Name                                     11746 non-null object
BBL - 10 digits                                          11735 non-null object
NYC Borough, Block and Lot (BBL) self-reported           11746 non-null object
NYC Building Identification Number (BIN)                 11746 non-null object
Address 1 (self-reported)                                11746 non-null object
Address 2                                                11746 non-null object
Postal Code                                              11746 non-null object
Street Number                                            11622 non-null object
Street Name                                              11624 non-null object
Borough                                                  11628 non-null object
DOF Gross Floor Area                                     11628 non-null float64
Primary Property Type - Self Selected                    11746 non-null object
List of All Property Use Types at Property               11746 non-null object
Largest Property Use Type                                11746 non-null object
Largest Property Use Type - Gross Floor Area (ft²)       11746 non-null object
2nd Largest Property Use Type                            11746 non-null object
2nd Largest Property Use - Gross Floor Area (ft²)        11746 non-null object
3rd Largest Property Use Type                            11746 non-null object
3rd Largest Property Use Type - Gross Floor Area (ft²)   11746 non-null object
```

Наверняка некоторые колонки, которые явно содержат числа (например, ft²), сохранены как объекты. Мы не можем применять числовой анализ к строковым значениям, так что конвертируем их в числовые типы данных (особенно float)!

Этот код сначала заменяет все «Not Available» на *not a number* (np.nan), которые можно интерпретировать как числа, а затем конвертирует содержимое определённых колонок в тип float:

```
# Replace all occurrences of Not Available with numpy not a number
data = data.replace({'Not Available': np.nan})

# Iterate through the columns
for col in list(data.columns):
# Select columns that should be numeric
if ('ft²' in col or 'kBtu' in col or 'Metric Tons CO2e' in col or 'kWh' in
col or 'therms' in col or 'gal' in col or 'Score' in col):
# Convert the data type to float
data[col] = data[col].astype(float)
```

Когда значения в соответствующих колонках у нас станут числами, можно начинать исследовать данные.

Отсутствующие и аномальные данные

Наряду с некорректными типами данных одна из самых частых проблем — отсутствующие значения. Они могут отсутствовать по разным причинам, и перед обучением модели эти значения нужно либо заполнить, либо удалить. Сначала давайте выясним, сколько у нас не хватает значений в каждой колонке (код здесь).

```
Your selected dataframe has 60 columns.
There are 46 columns that have missing values.
```

	Missing Values	% of Total Values
Fuel Oil #1 Use (kBtu)	11737	99.9
Diesel #2 Use (kBtu)	11730	99.9
Address 2	11539	98.2
Fuel Oil #5 & 6 Use (kBtu)	11152	94.9
District Steam Use (kBtu)	10810	92.0
Fuel Oil #4 Use (kBtu)	10425	88.8
3rd Largest Property Use Type - Gross Floor Area (ft²)	10262	87.4
3rd Largest Property Use Type	10262	87.4
Fuel Oil #2 Use (kBtu)	9165	78.0
2nd Largest Property Use Type	8005	68.2
2nd Largest Property Use - Gross Floor Area (ft²)	8005	68.2
Metered Areas (Water)	4609	39.2

Убирать информацию всегда нужно с осторожностью, и если много значений в колонке отсутствует, то она, вероятно, не пойдёт на пользу нашей модели. Порог, после которого колонки лучше выкидывать, зависит от вашей задачи, а в нашем проекте мы будем удалять колонки, пустые более чем на половину.

Также на этом этапе лучше удалить аномальные значения. Они могут возникать из-за опечаток при вводе данных или из-за ошибок в единицах измерений, либо это могут быть корректные, но экстремальные значения. В данном случае мы удалим «лишние» значения, руководствуясь определением экстремальных аномалий:

- Ниже первого квартиля − 3 ∗ интерквартильный размах.
- Выше третьего квартиля + 3 ∗ интерквартильный размах.

Код, удаляющий колонки и аномалии, приведён в блокноте на Github. По завершении процесса очистки данных и удаления аномалий у нас осталось больше 11 000 зданий и 49 признаков.

Разведочный анализ данных

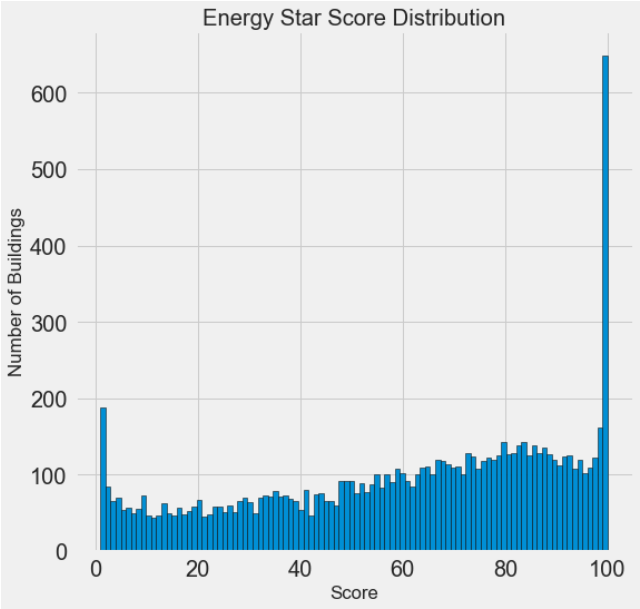
Скучный, но необходимый этап очистки данных закончен, можно перейти к исследованию! Разведочный анализ данных (РАД) — неограниченный по времени процесс, в ходе которого мы вычисляем статистику и ищем в данных тенденции, аномалии, шаблоны или взаимосвязи.

Коротко говоря, РАД — это попытка выяснить, что нам могут сказать данные. Обычно анализ начинается с поверхностного обзора, затем мы находим интересные фрагменты и анализируем их подробнее. Выводы могут быть интересными сами по себе, или они могут способствовать выбору модели, помогая решить, какие признаки мы будем использовать.

Одноперменные графики

Цель — прогнозировать значение Energy Star Score (в наших данных переименовано в score), так что имеет смысл начать с исследования распределения этой переменной. Гистограмма — простой, но эффективный способ визуализации распределения одиночной переменной, и её можно легко построить с помощью matplotlib.

```
import matplotlib.pyplot as plt
# Histogram of the Energy Star Score
plt.style.use('fivethirtyeight')
plt.hist(data['score'].dropna(), bins = 100, edgecolor = 'k');
plt.xlabel('Score'); plt.ylabel('Number of Buildings');
plt.title('Energy Star Score Distribution');
```



Выглядит подозрительно! Балл Energy Star Score является перцентилем, значит следует ожидать единообразного распределения, когда каждый балл присваивается одному и тому же количеству зданий. Однако высший и низший результаты получило непропорционально большое количество зданий (для Energy Star Score чем больше, тем лучше).

Если мы снова посмотрим на определение этого балла, то увидим, что он рассчитывается на основе «самостоятельно заполняемых владельцами зданий отчётов», что может объяснить избыток очень больших значений. Просить владельцев зданий сообщать

о своём энергопотреблении, это как просить студентов сообщать о своих оценках на экзаменах. Так что это, пожалуй, не самый объективный критерий оценки энергоэффективности недвижимости.

Если бы у нас был неограниченный запас времени, то можно было бы выяснить, почему так много зданий получили очень высокие и очень низкие баллы. Для этого нам пришлось бы выбрать соответствующие здания и внимательно их проанализировать. Но нам нужно только научиться прогнозировать баллы, а не разработать более точный метод оценки. Можно пометить себе, что у баллов подозрительное распределение, но мы сосредоточимся на прогнозировании.

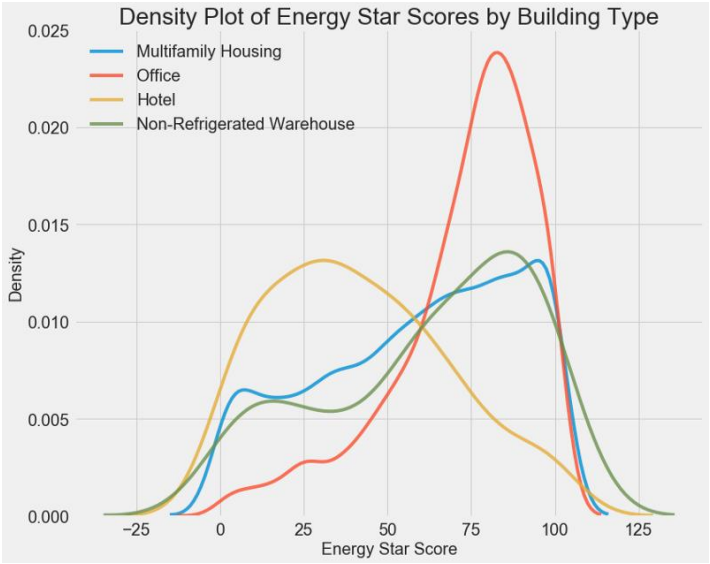
Поиск взаимосвязей

Главная часть РАД — поиск взаимосвязей между признаками и нашей целью. Коррелирующие с ней переменные полезны для использования в модели, потому что их можно применять для прогнозирования. Один из способов изучения влияния категориальной переменной (которая принимает только ограниченный набор значений) на цель — это построить график плотности с помощью библиотеки Seaborn.

График плотности можно считать сглаженной гистограммой, потому что он показывает распределение одиночной переменной. Можно раскрасить отдельные классы на графике, чтобы посмотреть, как категориальная переменная меняет распределение. Этот код строит график плотности Energy Star Score, раскрашенный в зависимости от типа здания (для списка зданий с более чем 100 измерениями):

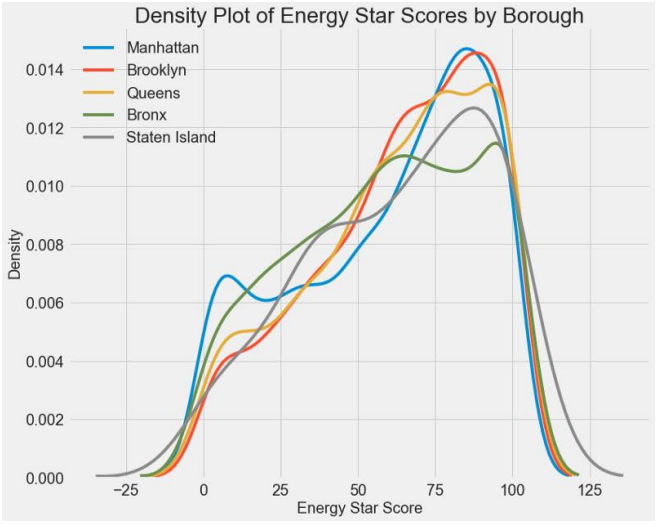
```
# Create a list of buildings with more than 100 measurements
types = data.dropna(subset=['score'])
types = types['Largest Property Use Type'].value_counts()
types = list(types[types.values > 100].index)

# Plot of distribution of scores for building categories
figsize(12, 10)

# Plot each building
for b_type in types:
# Select the building type
subset = data[data['Largest Property Use Type'] == b_type]

# Density plot of Energy Star Scores
```

```
sns.kdeplot(subset['score'].dropna(),
label = b_type, shade = False, alpha = 0.8);

# label the plot
plt.xlabel('Energy Star Score', size = 20); plt.ylabel('Density', size = 20);
plt.title('Density Plot of Energy Star Scores by Building Type', size = 28);
```



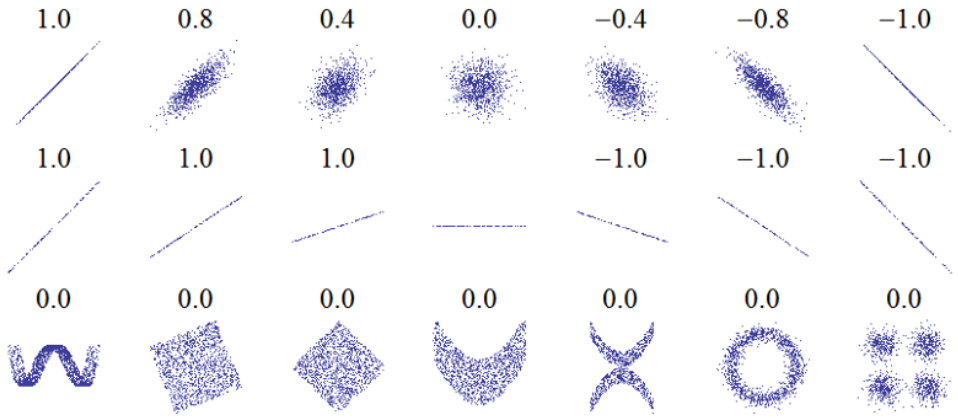
Как видите, тип здания сильно влияет на количество баллов. Офисные здания обычно имеют более высокий балл, а отели более низкий. Значит нужно включить тип здания в модель, потому что этот признак влияет на нашу цель. В качестве категориальной переменной мы должны выполнить one-hot кодирование типа здания.

Аналогичный график можно использовать для оценки Energy Star Score по районам города:



Район не так сильно влияет на балл, как тип здания. Тем не менее мы включим его в модель, потому что между районами существует небольшая разница.

Чтобы посчитать взаимосвязи между переменными, можно использовать коэффициент корреляции Пирсона. Это мера интенсивности и направления линейной зависимости между двумя переменными. Значение +1 означает идеально линейную положительную зависимость, а -1 означает идеально линейную отрицательную зависимость. Вот несколько примеров значений коэффициента корреляции Пирсона:



Хотя этот коэффициент не может отражать нелинейные зависимости, с него можно начать оценку взаимосвязей переменных. В Pandas можно легко вычислить корреляции между любыми колонками в данных (dataframe):

```
# Find all correlations with the score and sort
correlations_data = data.corr()['score'].sort_values()
```

Самые отрицательные корреляции с целью:

```
Site EUI (kBtu/ft²)                                -0.723864
Weather Normalized Site EUI (kBtu/ft²)             -0.713993
Weather Normalized Source EUI (kBtu/ft²)           -0.645542
Source EUI (kBtu/ft²)                              -0.641037
Weather Normalized Site Electricity Intensity (kWh/ft²)   -0.358394
```

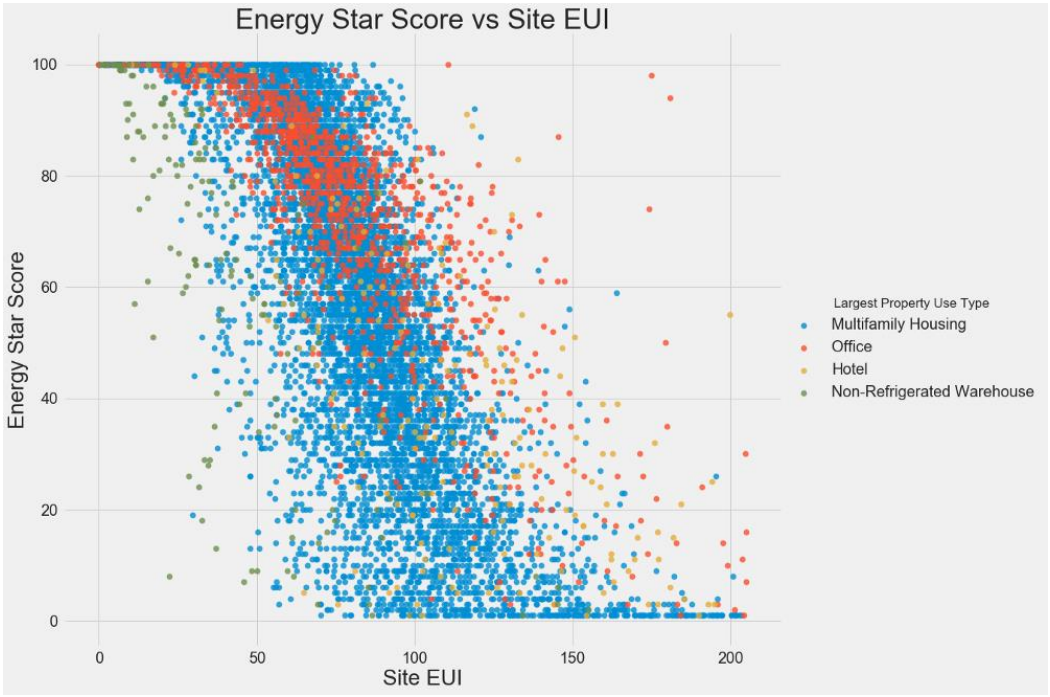
и самые положительные:

```
DOF Gross Floor Area                               0.013001
Property GFA - Self-Reported (ft²)                 0.017360
Largest Property Use Type - Gross Floor Area (ft²) 0.018330
Order                                              0.036827
Community Board                                    0.056612
Council District                                   0.061639
```

Есть несколько сильных отрицательных корреляций между признаками и целью, причём наибольшие из них относятся к разным категориям EUI (способы расчёта этих показателей слегка различаются). EUI (Energy Use Intensity, интенсивность использования энергии) — это количество энергии, потреблённой зданием, делённое на квадратный фут площади. Эта удельная величина используется для оценки энергоэффективности, и чем она меньше, тем лучше. Логика подсказывает, что эти корреляции оправданны: если EUI увеличивается, то Energy Star Score должен снижаться.

Двухпеременные графики

Воспользуемся диаграммами рассеивания для визуализации взаимосвязей между двумя непрерывными переменными. К цветам точек можно добавить дополнительную информацию, например, категориальную переменную. Ниже показана взаимосвязь Energy Star Score и EUI, цветом обозначены разные типы зданий:



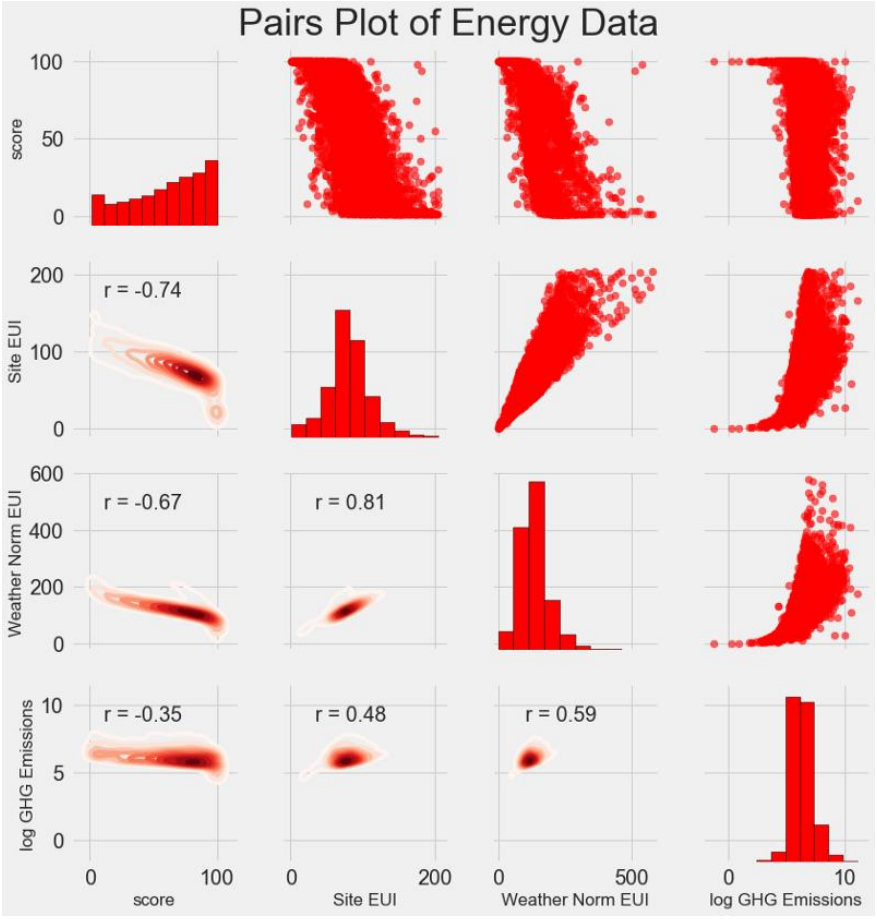
*Необходимо подписать: x='Site EUI (kBtu/ft²)', y='score', a size заменить на height в sns.lmplot.

Этот график позволяет визуализировать коэффициент корреляции -0,7. По мере уменьшения EUI увеличивается Energy Star Score, эта взаимосвязь наблюдается у зданий разных типов.

Наш последний исследовательский график называется Pairs Plot (парный график). Это прекрасный инструмент, позволяющий увидеть взаимосвязи между различными парами переменных и распределение одиночных переменных. Мы воспользуемся библиотекой Seaborn и функцией PairGrid для создания парного графика с диаграммой рассеивания в верхнем треугольнике, с гистограммой по диагонали, двухмерной диаграммой плотности ядра и коэффициентов корреляции в нижнем треугольнике.

```
# Extract the columns to plot
plot_data = features[['score', 'Site EUI (kBtu/ft²)',
'Weather Normalized Source EUI (kBtu/ft²)',
'log_Total GHG Emissions (Metric Tons CO2e)']]

# Replace the inf with nan
plot_data = plot_data.replace({np.inf: np.nan, -np.inf: np.nan})
```

```
# Rename columns
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'})

# Drop na values
plot_data = plot_data.dropna()

# Function to calculate correlation coefficient between two columns
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)

# Create the pairgrid object
grid = sns.PairGrid(data = plot_data)

# Upper is a scatter plot
grid.map_upper(plt.scatter, color = 'red', alpha = 0.6)

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

# Bottom is correlation and density plot
grid.map_lower(corr_func);
grid.map_lower(sns.kdeplot, cmap = plt.cm.Reds)

# Title for entire plot
plt.suptitle('Pairs Plot of Energy Data', size = 36, y = 1.02);
```



Чтобы увидеть взаимосвязи переменных, поищем пересечения рядов и колонок. Допустим, нужно посмотреть корреляцию Weather Norm EUI и score, тогда мы ищем ряд Weather Norm EUI и колонку score, на пересечении которых стоит коэффициент корреляции -0,67. Эти графики не только классно выглядят, но и помогают выбрать переменные для модели.

Конструирование и выбор признаков

Конструирование и выбор признаков зачастую приносит наибольшую отдачу с точки зрения времени, потраченного на машинное обучение. Сначала дадим определения:

- Конструирование признаков: процесс извлечения или создания новых признаков из сырых данных. Чтобы использовать переменные в модели, возможно, их придётся преобразовывать, скажем, брать натуральный логарифм, или извлекать квадратный корень, или применять one-hot кодирование категориальных переменных. Конструирование признаков можно рассматривать как создание дополнительных признаков из сырых данных.
- Выбор признаков: процесс выбора из данных самых релевантных признаков, в ходе которого мы удаляем часть признаков, чтобы помочь модели лучше обобщать новые данные ради получения более интерпретируемой модели. Выбор признаков можно рассматривать как удаление «лишнего», чтобы осталось только самое важное.

Модель машинного обучения может учиться только на предоставленных нами данных, поэтому крайне важно удостовериться, что мы включили всю релевантную для нашей задачи информацию. Если не предоставить модели корректные данные, она не сможет научиться и не будет выдавать точные прогнозы!

Мы сделаем следующее:

- Применим к категориальным переменным (квартал и тип собственности) one-hot кодирование.
- Добавим взятие натурального логарифма от всех числовых переменных.

One-hot кодирование необходимо для того, чтобы включить в модель категориальные переменные. Алгоритм машинного обучения не сможет понять тип «офис», так что если здание офисное, мы присвоим ему признак 1, а если не офисное, то 0.

Добавление преобразованных признаков поможет модели узнать о нелинейных взаимосвязях внутри данных. В анализе данных является нормальной практикой извлекать квадратные корни, брать натуральные логарифмы или ещё как-то преобразовывать признаки, это зависит от конкретной задачи или вашего знания лучших методик. В данном случае мы добавим натуральный логарифм всех числовых признаков.

Этот код выбирает числовые признаки, вычисляет их логарифмы, выбирает два категориальных признака, применяет к ним one-hot кодирование и объединяет оба множества в одно. Судя по описанию, предстоит куча работы, но в Pandas всё получается довольно просто!

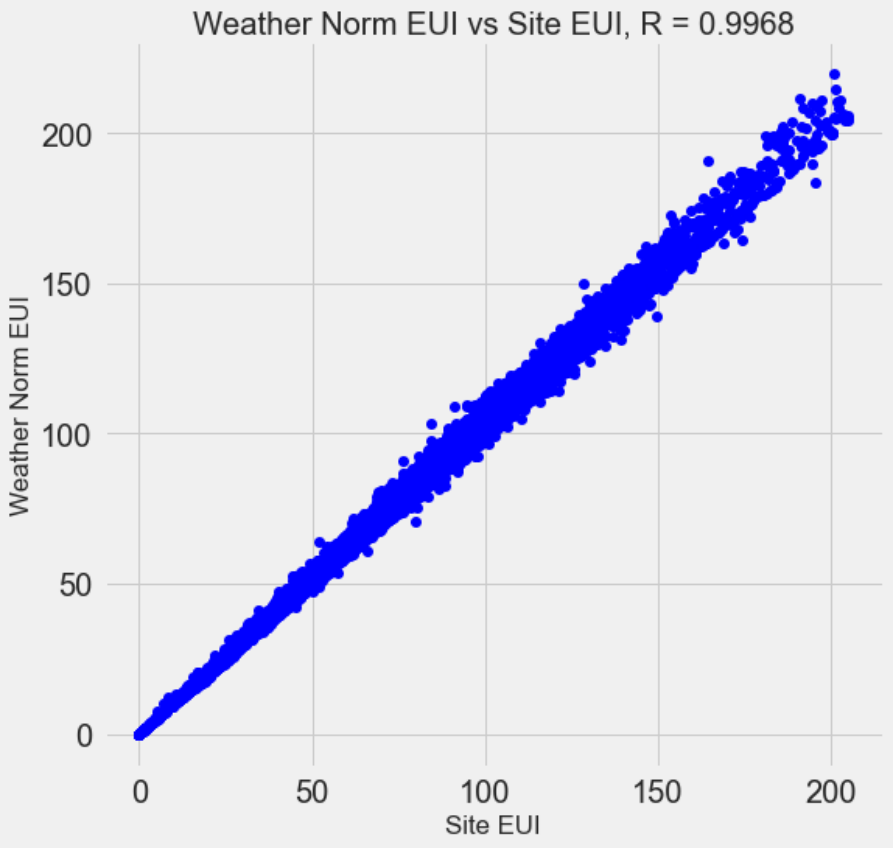
```
# Copy the original data
features = data.copy()

# Select the numeric columns
numeric_subset = data.select_dtypes('number')

# Create columns with log of numeric columns
for col in numeric_subset.columns:
# Skip the Energy Star Score column
if col == 'score':
next
else:
numeric_subset['log_' + col] = np.log(numeric_subset[col])

# Select the categorical columns
```

```
categorical_subset = data[['Borough', 'Largest Property Use Type']]

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

# Join the two dataframes using concat
# Make sure to use axis = 1 to perform a column bind
features = pd.concat([numeric_subset, categorical_subset], axis = 1)
```

Теперь у нас есть больше 11 000 наблюдений (зданий) со 110 колонками (признаками). Не все признаки будут полезны для прогнозирования Energy Star Score, поэтому займёмся выбором признаков и удалим часть переменных.

Выбор признаков

Многие из имеющихся 110 признаков избыточны, потому что сильно коррелируют друг с другом. К примеру, вот график EUI и Weather Normalized Site EUI, у которых коэффициент корреляции равен 0,997.



Признаки, которые сильно коррелируют друг с другом, называются коллинеарными. Удаление одной переменной в таких парах признаков часто помогает модели обобщать и

быть более интерпретируемой. Обратите внимание, что речь идёт о корреляции одних признаков с другими, а не о корреляции с целью, что только помогло бы нашей модели!

Существует ряд методов вычисления коллинеарности признаков, и один из самых популярных — фактор увеличения дисперсии (variance inflation factor). Мы для поиска и удаления коллинеарных признаков воспользуемся коэффициентом B-корреляции (thebcorrelation coefficient). Отбросим одну пару признаков, если коэффициент корреляции между ними больше 0,6. Код приведён в блокноте.

Машинное обучение эмпирично, и часто приходится экспериментировать, чтобы найти лучшее решение. После выбора у нас осталось 64 признака и одна цель.

```
# Remove any columns with all na values
features = features.dropna(axis=1, how = 'all')
print(features.shape)
(11319, 65)
```

Выбираем базовый уровень

Мы очистили данные, провели разведочный анализ и сконструировали признаки. И прежде чем перейти к созданию модели, нужно выбрать исходный базовый уровень (naive baseline) — некое предположение, с которым мы будем сравнивать результаты работы моделей. Если они окажутся ниже базового уровня, мы будем считать, что машинное обучение неприменимо для решения этой задачи, или что нужно попробовать иной подход.

Для регрессионных задач в качестве базового уровня разумно угадывать медианное значение цели на обучающем наборе для всех примеров в тестовом наборе. Эти наборы задают барьер, относительно низкий для любой модели.

В качестве метрики возьмём среднюю абсолютную ошибку (mae) в прогнозах. Для регрессий есть много других метрик. А среднюю абсолютную ошибку легко вычислить и интерпретировать.

Прежде чем вычислять базовый уровень, нужно разбить данные на обучающий и тестовый наборы:

- Обучающий набор признаков — то, что мы предоставляем нашей модели вместе с ответами в ходе обучения. Модель должна выучить соответствие признаков цели.
- Тестовый набор признаков используется для оценки обученной модели. Когда она обрабатывает тестовый набор, то не видит правильных ответов и должна прогнозировать, опираясь только на доступные признаки. Мы знаем ответы для тестовых данных и можем сравнить с ними результаты прогнозирования.

Для обучения используем 70 % данных, а для тестирования — 30 %:

```
# Split into 70% training and 30% testing set
X, X_test, y, y_test = train_test_split(features, targets,
test_size = 0.3,
random_state = 42)
```

Теперь вычислим показатель для исходного базового уровня:

```
# Function to calculate mean absolute error
def mae(y_true, y_pred):
return np.mean(abs(y_true - y_pred))

baseline_guess = np.median(y)

print('The baseline guess is a score of %0.2f' % baseline_guess)
print("Baseline Performance on the test set: MAE = %0.4f" % mae(y_test, baseline_guess))
```

The baseline guess is a score of 66.00
Baseline Performance on the test set: MAE = 24.5164

Средняя абсолютная ошибка на тестовом наборе составила около 25 пунктов. Поскольку мы оцениваем в диапазоне от 1 до 100, то ошибка составляет 25 % — довольно низкий барьер для модели!

Заключение

В этой статье мы прошли через три первых этапа решения задачи с помощью машинного обучения. После постановки задачи мы:

- Очистили и отформатировали сырые данные.
- Провели разведочный анализ для изучения имеющихся данных.
- Выработали набор признаков, которые будем использовать для наших моделей.

Наконец, мы вычислили базовый уровень, с помощью которого будем оценивать наши алгоритмы.