

【导读】这是一篇完全手把手进行机器学习项目构建的教程，包含：1. 数据清理和格式化 2. 探索性数据分析 3. 特征工程和特征选择 4. 在性能指标上比较几种机器学习模型 5. 对最佳模型执行超参数调整 6. 在测试集合中评估最佳模型 7. 解释模型结果 8. 得出结论。今天是第一部分（1-3）从数据清理，到数据分析，到特征工程，再到Baseline的构建，作者以浅显易懂的语言和清晰的示例和代码教你从头开始走过一个机器学习之旅，并且附详细的代码，大家可以收藏和学习。

作者 | William Koehrsen

编译 | 专知

参与 | Chaofan, Xiaowen



用python完成一个完整的机器学习项目：第一部分 ——Putting the machine learning pieces together

阅读一本数据科学书籍或学习一门相关的课程，你可能感觉你有了独立的碎片，但不知道如何将它们拼在一起。想要继续推进下去并解决完整的机器学习问题可能令人望而生畏，但完成第一个项目后将使你有信心应对任何数据科学问题。本系列文章将介绍——使用了真实世界数据集的机器学习项目的完整解决方案，让你了解所有碎片是如何拼接在一起的。

我们将按照一般机器学习的工作流程逐步进行：

1. 数据清理和格式化
2. 探索性数据分析
3. 特征工程和特征选择
4. 在性能指标上比较几种机器学习模型
5. 对最佳模型执行超参数调整
6. 在测试集合中评估最佳模型
7. 解释模型结果
8. 得出结论

按照上述流程，我们将介绍每个步骤如何进入到下一步，以及如何用Python实现每个部分。完整的项目在GitHub上可以找到，第一个notebook在这里。 第一篇文章将涵盖步骤1-3，其余的内容将在后面的文章中介绍。

GitHub完整项目链接：

<https://github.com/WillKoehrsen/machine-learning-project-walkthrough>

问题定义

编码之前的第一步是了解我们试图解决的问题和可用的数据。在这个项目中，我们将使用公共可用的纽约市的建筑能源数据【1】。

目标是使用能源数据建立一个模型，来预测建筑物的Energy Star Score（能源之星分数），并解释结果以找出影响评分的因素。

数据包括Energy Star Score，意味着这是一个监督回归机器学习任务：

监督：我们可以知道数据的特征和目标，我们的目标是训练可以学习两者之间映射关系的模型。

回归：Energy Star Score是一个连续变量。

我们想要开发一个模型，在准确性上——它可以实现预测Energy Star Score，并且结果接

近真实值。在解释上—— 我们可以理解模型的预测。

一旦我们知道了目标，在深入挖掘数据并构建模型时，就可以用它来指导我们的决策。

数据清洗

与大多数数据科学课程所相信的相反，并非每个数据集都是一组完美的观测数据，没有缺失值或异常值（你可以查看你的mtcars【2】和iris数据集【3】）。现实世界的的数据很乱，这意味着在我们开始分析之前，我们需要清理并将其转换为可接受的格式【4】。数据清理，是大多数实际的数据科学问题中不具吸引力，但必不可少的一部分。

首先，我们可以将数据用Pandas DataFrame加载并查看：


```
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

What Actual Data Looks Like!

这是包含60列的完整数据的子集。 我们已经可以看到几个问题：首先，我们知道我们想要预测的Energy Star Score的情况，但我们不知道任何一列的含义。 虽然这不一定是个问题——我们通常可以在没有任何变量知识的情况下创建一个准确的模型——我们想把重点放在模型的可解释性上，而至少了解一些列可能是重要的。

起初，我从初创阶段得到任务时，我不想问所有的列名是什么意思，所以我查看了csv文件的名称，

 Energy_and_Water_Data_Disclosure_for_Local_Law_84_2017_Data_for_Calendar_Year_2016_.csv

并决定搜索“Local Law 84”。所以在上文我解释——这是纽约法律要求的，所有具有一定规模的建筑物报告其能源使用情况。关于列的更多搜索内容在这里。也许看一个文件名是一个明显的开始，但对我来说，这是提醒你要放慢速度，这样你才不会错过任何重要的东西！

我们不需要研究所有的列的定义，但我们至少应该了解Energy Star Score，它被描述为：

根据报告年度中，自我报告的能源使用情况而进行的1至100百分位的排名。 Energy Star Score是用于比较建筑物能效的相对度量。【5】

这解决了第一个小问题，但第二个问题是缺少的值被编码为“Not Available”。这是Python中的一个字符串，这意味着甚至包含数字的列都将被存储为object数据类型，因为Pandas会将包含任何字符串的列转换为所有元素都为字符串的列。我们可以使用dataframe.info () 方法来查看列的数据类型：

```
# 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

```

当然，一些明确包含数字（例如ft²）的列被存储为object类型。我们不能对字符串进行数值分析，因此必须将其转换为数字（特别是浮点数）数据类型！

这里有一个简短的Python代码，用不是数字（np.nan）代替所有“Not Available”条目，np.nan可以被解释为数字，这样就可以将相关列转换为float数据类型：

```

# 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)

```

一旦列是数字，我们就可以开始进行调查数据。

缺少数据和异常值

除了不正确的数据类型外，处理真实世界数据时的另一个常见问题是缺失值。这可能是由于许多原因引起的，在我们训练机器学习模型之前必须填写或删除。首先，让我们了解每列中有多少缺失值（请参阅notebook中的代码）。

```
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 (ft ²)	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 (ft ²)	8005	68.2
Metered Areas (Water)	4609	39.2

（为了创建这个表，我使用了这个Stack Overflow论坛的一个函数【6】）。

尽管我们总是希望小心删除信息，但如果列中缺失值的比例很高，那么它对我们的模型可能不会有用了。删除列的阈值应该取决于实际问题，并且对于此项目，我们将删除缺失值超过50%的列。

此时，我们可能还想要移除异常值。这些异常可能是由于数据输入中的拼写错误，单位中的错误，或者它们可能是合法但是极端的值。对于这个项目，我们将根据极端异常值的定义来消除异常：

1. the first quartile - 3 * interquartile range
2. the third quartile + 3 * interquartile range

（有关删除列和异常的代码，请参阅notebook）。在数据清理和异常清除过程结束时，我们剩下11,000多个建筑物和49个特征。

探索性数据分析

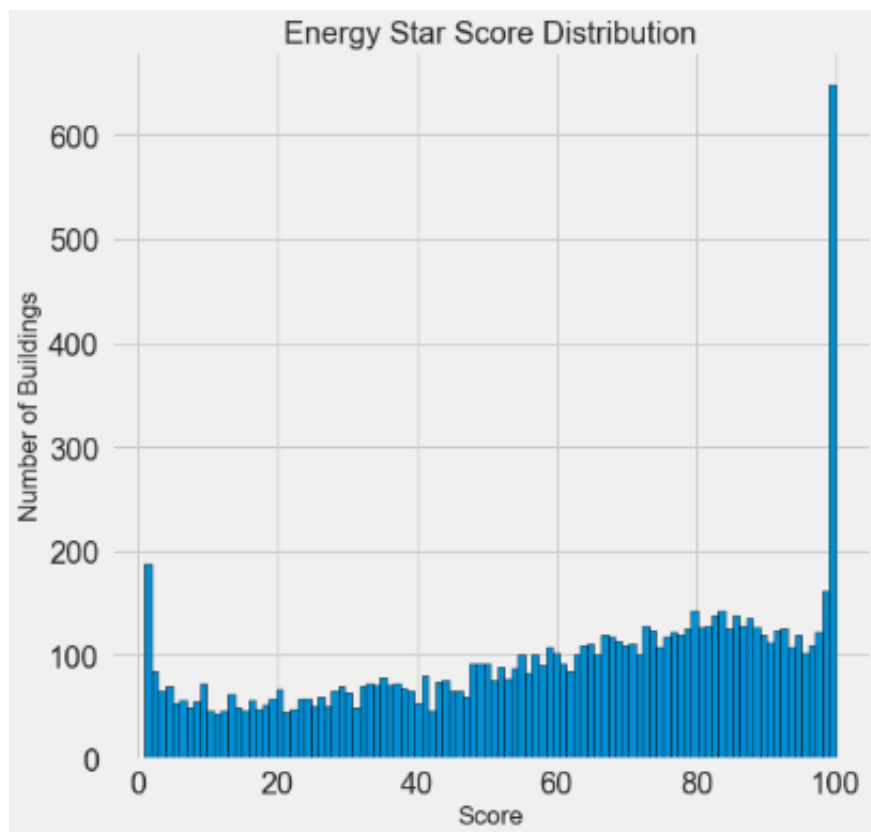
现在，数据清理这个乏味但必要的步骤已经完成，我们可以继续探索我们的数据！探索性数据分析（EDA）是一个开放式的过程，我们可以计算统计数据，并画图去发现数据中的趋势，异常，模式或关系。（ trends, anomalies, patterns, or relationships ）

简而言之，EDA的目标是了解我们的数据可以告诉我们什么。它通常以高层概述开始，在我们发现有趣的数据部分后，再缩小到特定的区域。这些发现本身可能很有意思，或者可以用于通知我们的建模选择，例如帮助我们决定使用哪些特征。

单变量图

目标是预测Energy Star Score（将其重新命名为score），因此合理的开始是检查此变量的分布。直方图是可视化单个变量分布的简单而有效的方法，使用matplotlib很容易。

```
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');
```



这看起来很可疑！Energy Star Score是百分位数，这意味着我们期望看到一个统一的分布，即每个得分分配给相同数量的建筑物。然而，不成比例的建筑物具有最高的100分或最低的1分（对于Energy Star Score来说更高表示越好）。

如果我们回到score的定义，我们会看到它基于“自我报告能量使用”，这可能解释了分数偏高——要求建筑物所有者报告自己的能源使用情况就像要求学生在测试中报告自己的分数！因此，这可能不是衡量建筑能效的最客观标准。

如果我们有无限的时间，我们可能想要调查为什么这么多建筑物有非常高和非常低的分数——我们可以通过选择这些建筑物并查看它们的共同点。但是，我们的目标只是预测分数，而不是设计更好的建筑物评分方法！我们可以在我们的报告中记下分数具有可疑分布，但我们主要关注预测分数。

寻找关系

EDA的主要部分是搜索特征和目标之间的关系。与目标相关的变量对模型很有用，因为它们可用于预测目标。通过使用seaborn库的密度图可以检查目标上的分类变量（仅采用有限的一组值）的效果。

密度图可以被认为是平滑的直方图，因为它显示了单个变量的分布。我们可以按类别对密度图进行着色，以查看分类变量如何改变分布。下面的代码创建了一个用建筑物类型（仅限于具有超过100个数据点的建筑物类型）着色的Energy Star Score密度图：

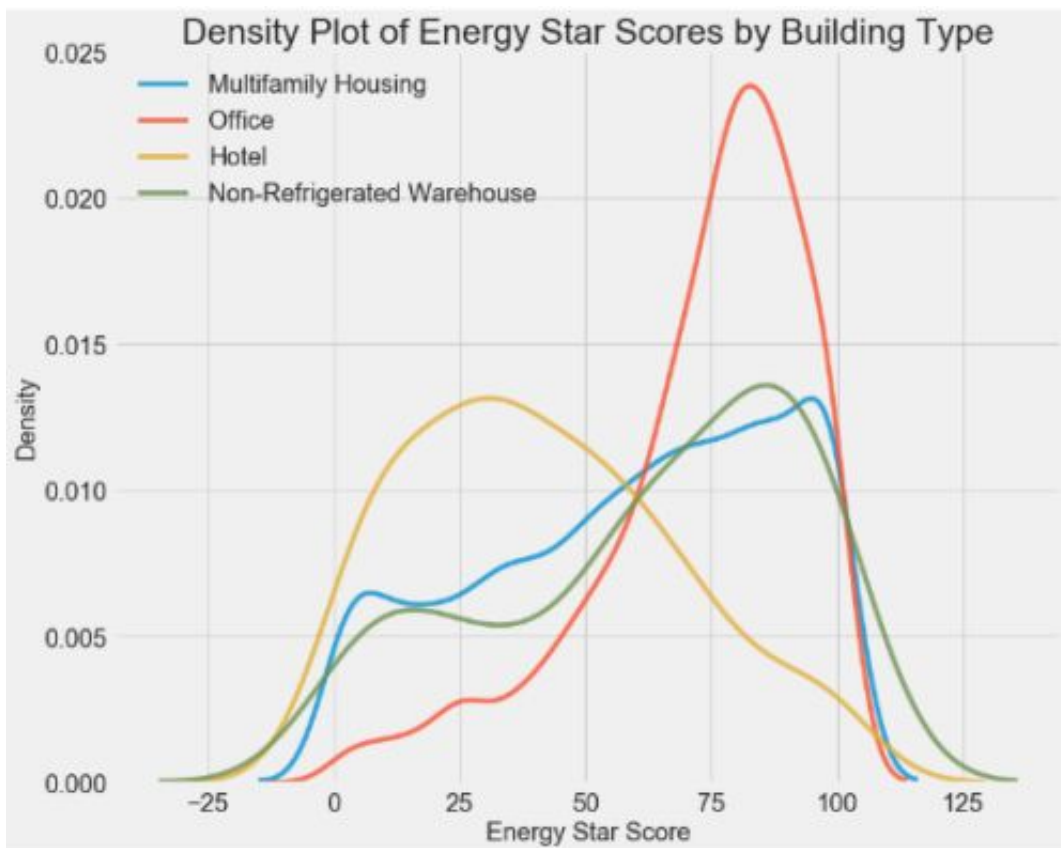
```
# Create a list of buildings with more than 100 measurements
types = data.dropna(subset=['score'])
types = types['Largest Property Use Type'].value_counts()
types = list(types[types.values > 100].index)

# 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]

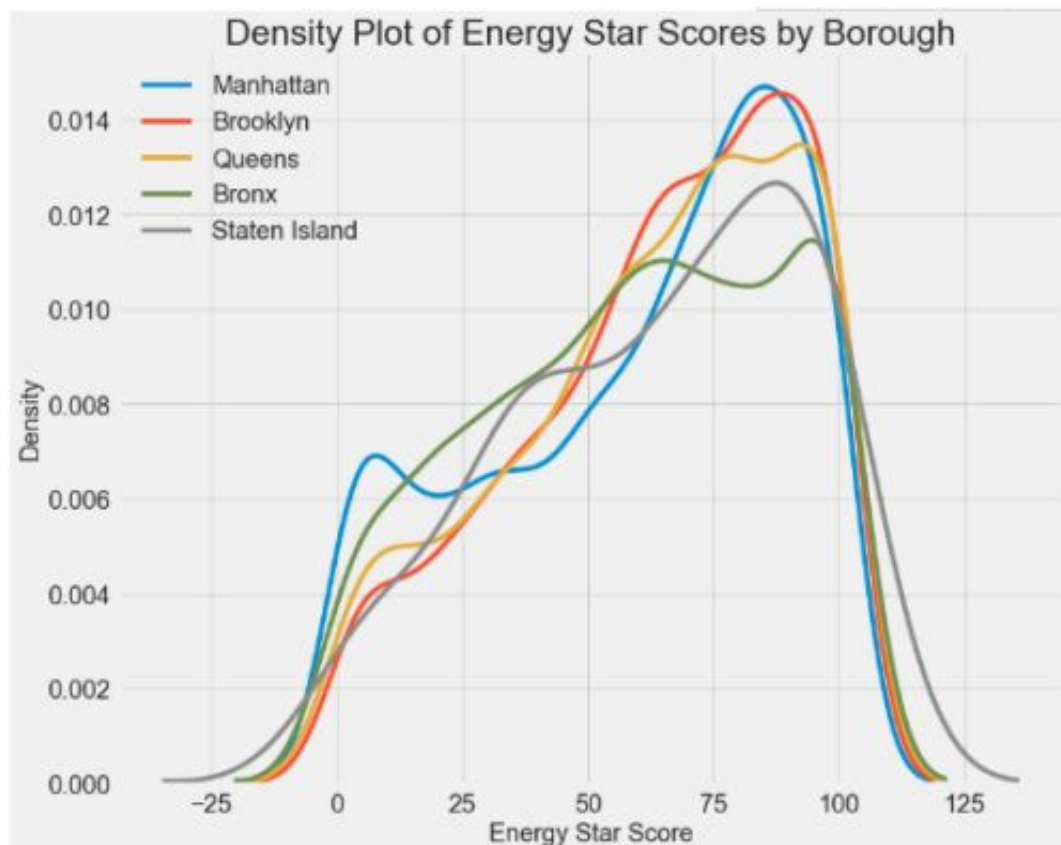
    # Density plot of Energy Star scores
    sns.kdeplot(subset['score'].dropna(),
                label = b_type, shade = False, alpha = 0.8);

# 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);
```



我们可以看到建筑类型对Energy Star Score有重大影响。 办公楼往往有较高的分数，而酒店的分数较低。这告诉我们，我们应该在建模中包含建筑类型，因为它确实对目标有影响。 作为分类变量，我们将不得不对建筑物类型进行one-hot编码。

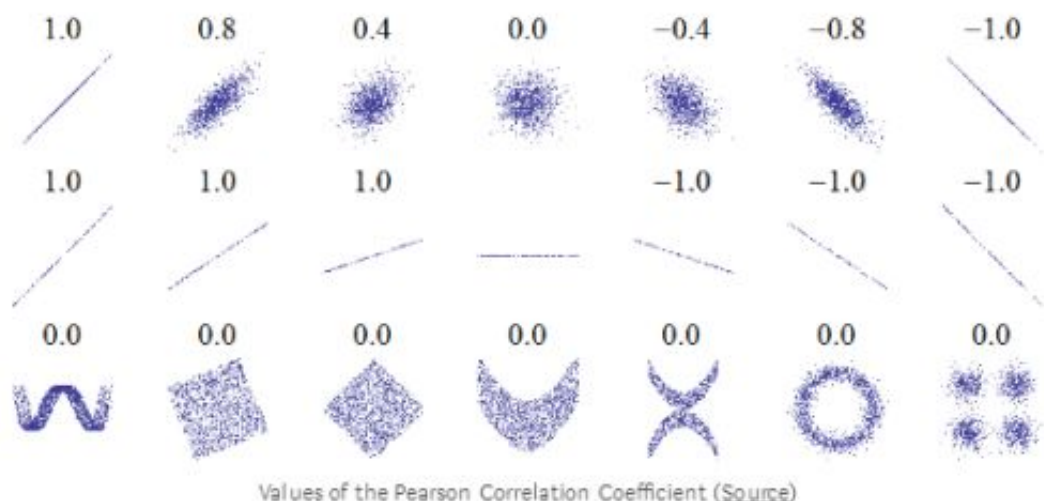
同上，可以显示自治市镇的Energy Star Score：



自治市镇对建筑类型的评分似乎没有太大的影响。 尽管如此，我们可能希望将其纳入我们

的模型中，因为各区之间存在细微的差异。

为了量化变量之间的关系，我们可以使用Pearson相关系数。它可以用来衡量两个变量之间的线性关系的强度和方向。+1分是完美的线性正相关关系，-1分是完美的负线性关系。相关系数的几个值如下所示：



虽然相关系数无法捕捉非线性关系，但它是开始计算变量如何相关的好方法。在Pandas中，我们可以轻松计算数据框中任何列之间的相关性：

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

与目标的最负面（左）和最正面（右）相关性：

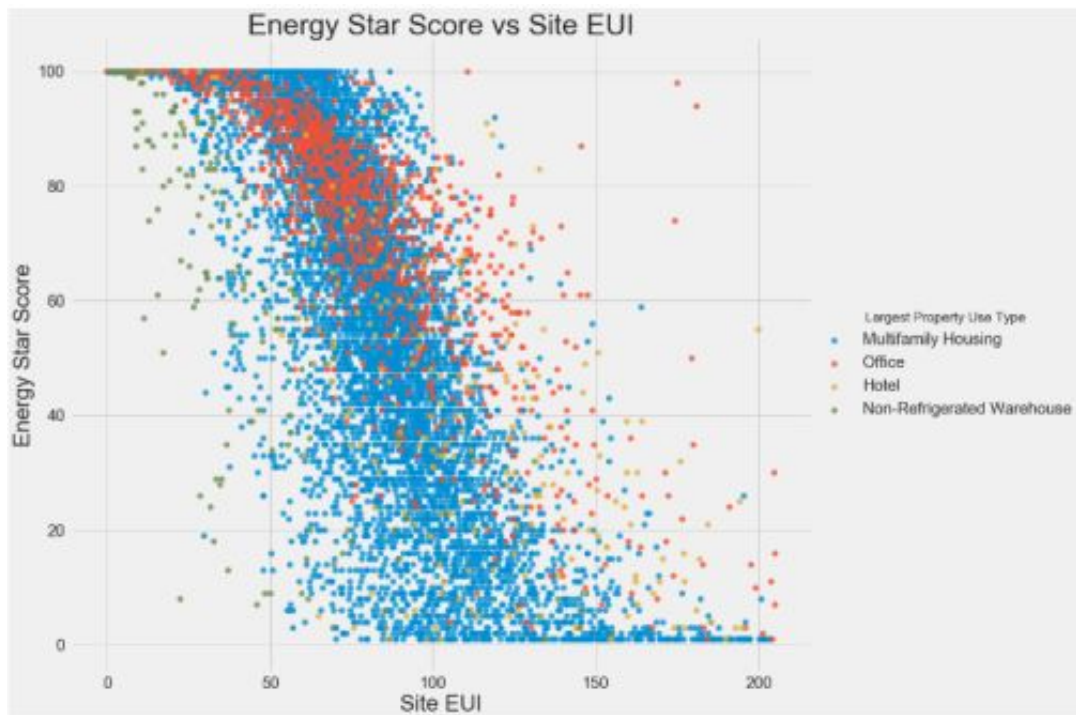
Site EUI (kBtu/ft ²)	-0.723864	D0F Gross Floor Area	0.013601
Weather Normalized Site EUI (kBtu/ft ²)	-0.713993	Property GFA - Self-Reported (ft ²)	0.017360
Weather Normalized Source EUI (kBtu/ft ²)	-0.645542	Largest Property Use Type - Gross Floor Area (ft ²)	0.018330
Source EUI (kBtu/ft ²)	-0.641037	Order	0.036827
Weather Normalized Site Electricity Intensity (kWh/ft ²)	-0.358394	Community Board	0.056612
		Council District	0.061639

特征与目标之间存在几个强烈的负相关性，而EUI对目标最为负面。（这些测量方法在计算方式上略有不同）EUI——能源使用强度（Energy Use Intensity）——是建筑物使用的能量除以建筑物的平方英尺。它意味着衡量一个建筑，效率越低越好。直观地说，这些相关性是有意义的：随着EUI的增加，Energy Star Score趋于下降。

双变量图

为了可视化两个连续变量之间的关系，我们使用散点图。我们可以在点的颜色中包含附加信息，例如分类变量。例如，下面的图表显示了不同建筑物类型的Energy Star Score与

site EUI的关系：



这个图让我们可以看到-0.7的相关系数。随着Site EUI减少，Energy Star Score增加，这种关系在建筑类型中保持稳定。

我们将做的最后的探索性plot被称为Pairs Plot。这是一个很好的探索工具，因为它可以让我们看到多个变量对之间的关系以及单个变量的分布。在这里，我们使用seaborn可视化库和PairGrid函数来创建上三角上具有散点图的配对图，对角线上的直方图以及下三角形上的二维核密度图和相关系数。

```
# 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)

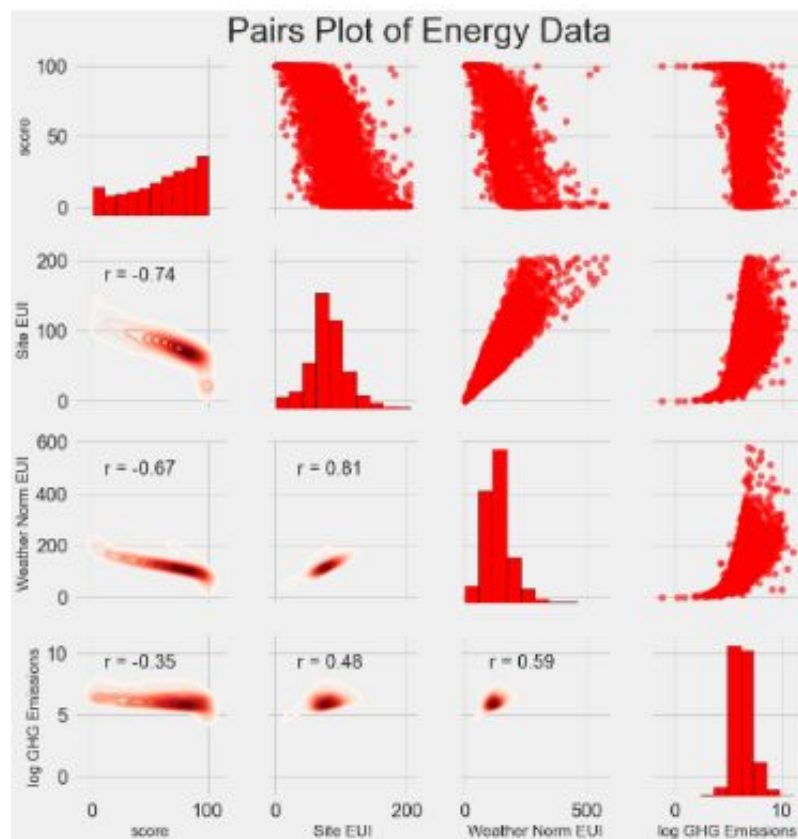
# Create the pairgrid object
grid = sns.PairGrid(data = plot_data, size = 3)

# 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);
```



要查看变量之间的交互，我们查找行与列相交的位置。例如，要查看Weather EUorm EUI与score的相关性，我们查看Weather EUorm EUI行和score列，并查看相关系数为-0.67。除了看起来很酷之外，诸如这些图可以帮助我们决定在建模中应该包含哪些变量。

特征工程和选择

特征工程和选择通常会为机器学习问题投入最大的时间。首先，让我们来定义这两个任务是什么：

- **特征工程**：获取原始数据并提取或创建新特征的过程。这可能意味着需要对变量进行变换，例如自然对数和平方根，或者对分类变量进行one-hot编码，以便它们可以在模型中使用。一般来说，我认为特征工程是从原始数据创建附加特征。
- **特征选择**：选择数据中最相关的特征的过程。在特征选择中，我们删除特征以帮助模型更好地总结新数据并创建更具可解释性的模型。一般来说，我认为特征选择是减去特征，所以我们只留下那些最重要的特征。

机器学习模型只能从我们提供的数据中学习，因此确保数据包含我们任务的所有相关信息

至关重要。如果我们没有给模型提供正确的数据，那么我们将它设置为失败，我们不应该期望它学习！

对于这个项目，我们将采取以下功能设计步骤：

- One-hot编码分类变量 (borough and property use type)。
- 添加数值变量的自然对数转换。

在模型中，分类变量的One-hot编码是必要的。机器学习算法无法理解像“office”这样的建筑类型，因此如果建筑物是办公室，则必须将其记录为1，否则将其记录为0。

添加转换特征可以帮助我们的模型学习数据中的非线性关系。采用平方根，自然对数或特征的次幂是数据科学中的常见做法，也是基于领域知识或在实践中最有效的方法。这里我们将使用数字特征的自然对数。

以下代码选择数字特征，对这些特征进行对数转换，选择两个分类特征，对这些特征进行one-hot编码，然后将两个特征结合在一起。这似乎需要做很多工作，但在pandas中相对简单！

```
# 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
```

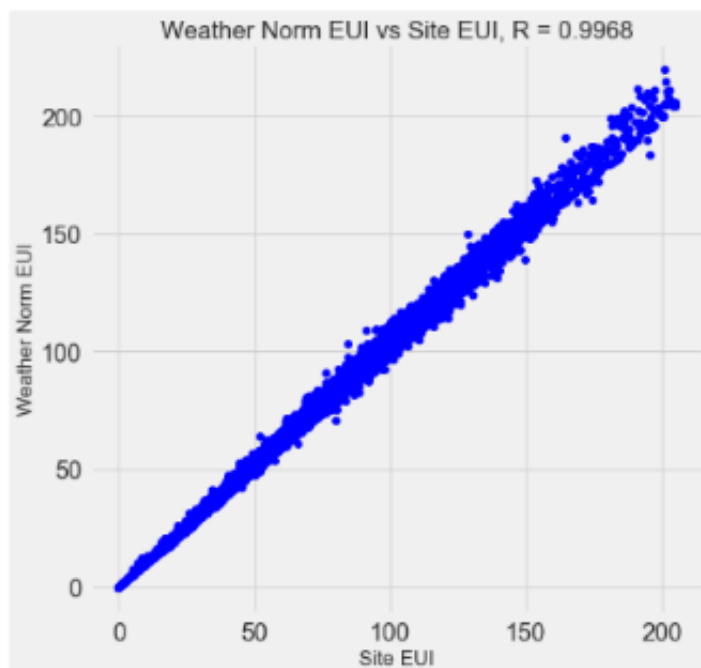
```
categorical_subset = pd.get_dummies(categorical_subset)

# 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)
```

在这个过程之后，我们有超过11,000个具有110列（特征）的观测值（建筑物）。并非所有这些特征都可能对预测Energy Star Score有用，所以现在我们将转向特征选择从而去除一些变量。

特征选择

我们数据中的110个特征中的许多特征是多余的，因为它们彼此高度相关。例如，以下是Site EUI与Weather Normalized SiteEUI的相关系数为0.997的图。



相互强相关的特征被称为共线，消除这些特征对中的一个变量通常可以帮助机器学习模型推广并更易于解释。（我应该指出，我们正在讨论特征与其他特征的相关性，而不是与目标的相关性，这有助于我们的模型！）

有许多方法可以计算特征之间的共线性，其中最常见的是方差扩大因子。在这个项目中，我们将使用相关系数来识别和删除共线特征。如果它们之间的相关系数大于0.6，我们将放弃一对特征中的一个。对于实现，看看notebook（和这个stack overflow答案）

虽然这个阈值可能看起来是任意的，但我尝试了几个不同的阈值，这个选择产生了最好的

模型机器学习是一个经验性领域，通过试验来发现性能最好的！特征选择后，我们剩下64个特征和1个目标。

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

建立Baseline

我们现在已经完成了数据清理，探索性数据分析和特征工程。开始建模之前要做的最后一步是建立一个Baseline。这实际上是我们可以比较我们的结果的一种猜测。如果机器学习模型没有超越这个猜测，那么我们可能必须得出结论，机器学习对于任务来说是不可接受的，或者我们可能需要尝试不同的方法。

对于回归问题，合理的Baseline是猜测测试集中所有示例的训练集上目标的中值。这设置了一个任何模型都要超越的相对较低的标准。

我们将使用的度量标准是平均绝对误差（Mean Absolute Error）（MAE），它测量预测的平均绝对误差。有很多回归的指标，但我喜欢Andrew Ng的建议【7】，选择一个指标，然后在评估模型时坚持使用它。平均绝对误差很容易计算，并且可以解释。

在计算Baseline之前，我们需要将我们的数据分成一个训练集和一个测试集：

- 1. 训练集是我们在训练期间给我们的模型提供特征以及答案的。目的是让模型学习特征与目标之间的映射。**
- 2. 测试集合的特征用于评估训练的模型。模型不允许查看测试集的答案，并且只能使用特征进行预测。我们知道测试集的答案，因此我们可以将测试预测与答案进行比较。**

我们将使用70%的数据进行训练，30%用于测试：

```
# 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)
```

现在我们可以计算出Baseline的性能：

```
# 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
```

Baseline的估计在测试集中约为25分。得分范围从1到100，所以这代表25%的误差，相当低的一个超越！

结论

在本文中，我们走过了机器学习问题的前三个步骤。在定义问题之后，我们：

1. 清理并格式化原始数据
2. 进行探索性数据分析以了解数据集
3. 开发了一系列我们将用于模型的特征

最后，我们还完成了建立我们可以判断我们的机器学习算法的Baseline的关键步骤。

第二篇文章将展示如何使用Scikit-Learn评估机器学习模型，选择最佳模型并执行超参数调整来优化模型。

- 1.http://www.nyc.gov/html/gbee/html/plan/l184_scores.shtml
- 2.<http://stat.ethz.ch/R-manual/R-devel/library/datasets/html/mtcars.html>
- 3.<https://archive.ics.uci.edu/ml/datasets/iris>
- 4.<https://www.springboard.com/blog/data-wrangling/>
- 5.<https://www.energystar.gov/buildings/facility-owners-and-managers/existing-buildings/use-portfolio-manager/interpret-your-results/what>
- 6.<https://stackoverflow.com/questions/26266362/how-to-count-the-nan-values-in-a-column-in-pandas-dataframe/39734251#39734251>
- 7.<https://www.coursera.org/learn/machine-learning-projects/lecture/wIKkC/single-number-evaluation-metric>

原文链接：

<https://towardsdatascience.com/a-complete-machine-learning-walk-through-in-python-part-one-c62152f39420>

代码链接：

<https://github.com/WillKoehrs/en/machine-learning-project-walkthrough/blob/master/Machine%20Learning%20Project%20Part%201.ipynb>

-END-

专·知

人工智能领域主题知识资料查看与加入专知人工智能服务群：

【专知AI服务计划】专知AI知识技术服务会员群加入与人工智能领域26个主题知识资料全集获取



[点击上面图片加入会员]

请PC登录www.zhuanzhi.ai或者点击[阅读原文](#)，注册登录专知，获取更多AI知识资料！



请加专知小助手微信（扫一扫如下二维码添加），加入专知主题群（请备注主题类型：AI、NLP、CV、KG等）交流~



请关注专知公众号，获取人工智能的专业知识！

点击“阅读原文”，使用专知

[阅读原文](#)