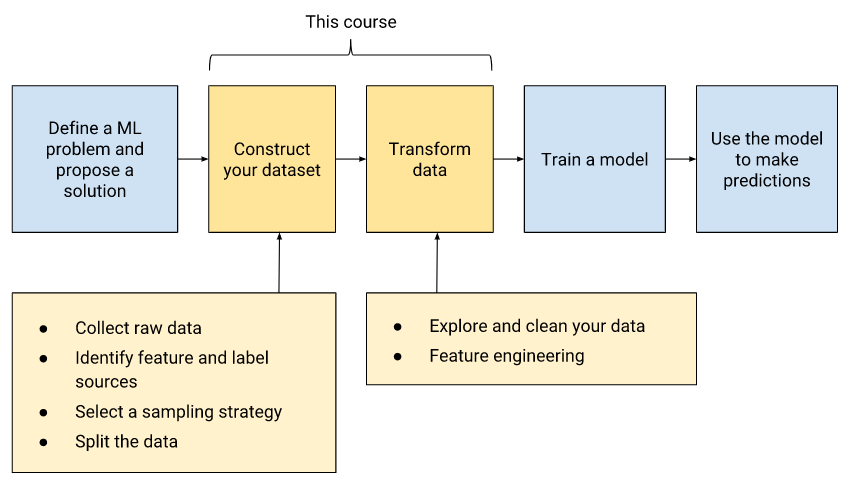
为什么学习数据准备和特征工程？

您可以将特征工程视为帮助模型以与您相同的方式理解数据集。学习者经常参加以模型构建为重点的机器学习课程，但最终花更多的时间关注数据。

数据准备和特征工程的流程



请记住：

图中显示了一个典型的流程，这可能不是每个项目的理想流程。本课程主要适用于线性回归和神经网络。

所示过程并不总是连续的。例如，您可以在转换数据后将其拆分。你可能需要收集更多的数据。您可能需要修改特性集，即使在训练开始之后，因为您可以根据经验了解哪些有效哪些无效。

构建数据集

构建数据集的步骤：

1. 收集原始数据
2. 明确特征和标签来源
3. 选择一个采样策略
4. 拆分数据

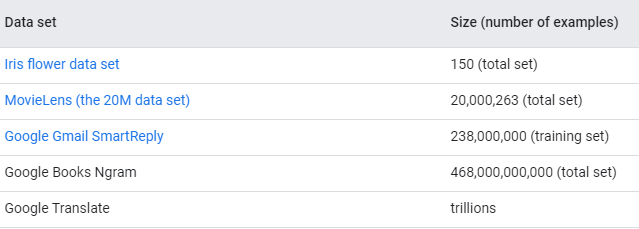
收集数据

数据集大小和质量

大小

作为一个粗略的经验法则，模型应该使用的样本数要比模型可训练参数多一个数量级。大数据集上简单模型通常比小数据集上的复杂模型效果要好。谷歌在大型数据集上训练简单线性回归模型取得了巨大成功。

许多数据是什么量级，不同项目有不同要求：



数据质量

如果都是脏数据，数据再多也没有用。但是质量怎么评估，这没有明确的答案。考虑采用经验方法，选择产生最佳结果的选项。有了这种心态，一个高质量的数据集可以让你成功地处理你关心的业务问题。换句话说，如果数据完成了预期的任务，那么它是好的。

然而，在收集数据的同时，对质量有一个更具体的定义是有帮助的。质量的某些方面往往对应于性能更好的模型：

可靠性

特征表达

最小化的倾斜

可靠性

可靠性是指您可以信任数据的程度。在可靠数据集上训练的模型比在不可靠数据上训练的模型更有可能产生有用的预测。在测量可靠性时，必须确定：

标签错误的可能性，如果标签由人工标记，那么人有时会犯错误

特征是否有噪音？例如，GPS测量会波动，一定的噪音没有关系，数据集中总会有噪音

的，你也可以收集更多的例子。

数据是否针对您的问题进行了适当的筛选？例如，您的数据集是否应该包括来自bot的

搜索查询？如果你正在构建一个垃圾邮件检测系统，那么答案很可能是肯定的，但是如

果你试图改善人类的搜索结果，那么就不是。

什么会导致数据不可靠？以下情况：

缺失值

重复样本

错误标签

错误的特征值

特征表达

特征表达是将数据映射为有用的特征。你需要考虑以下问题：

数据如何展示给模型

是否应该将数值类数据归一化

如何处理异常值

训练与预测

如果在线下有很好的结果，但是在线上，这些结果站不住脚，发生了什么？

始终考虑在预测时能获得哪些数据，在训练中只用可以在服务时能够获得的特征，并确保训练集能够代表服务流量。

数据联合

识别标签和来源

直接标签和派生标签

当标签定义良好时，机器学习更容易。最好的标签是你想要预测的东西的直接标签。例如，如果你想预测一个用户是否是泰勒·斯威夫特的粉丝，一个直接的标签将是“用户是泰勒·斯威夫特的粉丝”。

对粉丝的一个更简单的测试可能是用户是否在YouTube上看过Taylor Swift视频。标签“用户在YouTube上观看了Taylor Swift视频”是一个派生标签，因为它不能直接衡量你想要预测的内容。这个派生的标签是用户喜欢Taylor Swift的可靠指标吗？您的模型性能将由派生标签和所需预测之间的连接决定。

标签来源

模型的输出可以是事件或属性。这将产生以下两种类型的标签：

事件的直接标签，例如“用户是否单击了顶部搜索结果？”

属性的直接标签，例如“广告客户在下一周内会花费超过$X吗？”

事件的直接标签

对于事件，直接标签通常很简单，因为您可以将事件期间的用户行为记录下来用作标签。在标记事件时，请问自己以下问题：

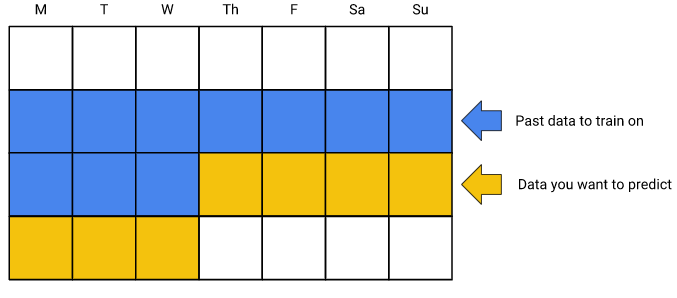
你的日志结构如何？

在你的日志中什么是“事件”？

例如，当用户进行搜索时，系统是否记录用户单击搜索结果或者没有点击？如果你有点击日志，那么你永远不会看到没有点击的情况。

属性的直接标签

假设你的标签是“广告客户下周将花费超过X美元”，通常，你会使用前几天的数据来预测接下来几天会发生什么。例如，下图显示了预测随后七天的十天培训数据：



记住要考虑季节性或周期性的影响；例如，广告商可能会在周末花更多的钱。因此，您可能更喜欢使用14天的窗口，或者使用日期作为特征，以便模型可以学习年度效果。

如果你还没有记录数据呢

也许你的产品还不存在，所以你没有任何数据要登录。在这种情况下，可以执行以下一个或多个操作：

使用启发式方法作为起始，然后根据记录的数据训练系统。

使用类似问题的日志来引导系统。

使用人工评分器通过完成任务生成数据。

**采样和分割数据**

采样

为一个机器学习项目收集足够的数据常常是一件困难的事。但是，有时数据太多，您必须选择一个子集示例进行培训。

你如何选择那个子集？以Google搜索为例。您将在多大的粒度上对它的大量数据进行采样？你会使用随机查询吗？随机会话？随机用户？

最终，答案取决于问题：我们想要预测什么，我们想要什么特性？

若要使用“上一个查询”功能，需要在会话级别进行采样，因为会话包含一系列查询。

要使用前几天的功能用户行为，需要在用户级别进行采样。

过滤个人识别信息如果你的数据包括个人识别信息，你可能需要从你的数据中过滤它。例如，策略可能要求您删除不常见的特征。

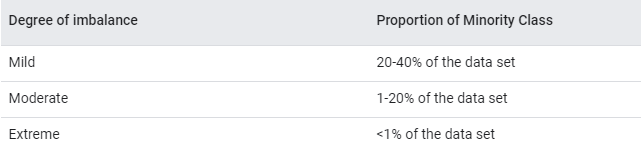
此筛选将扭曲您的分布。您将丢失尾部的信息（分布中值非常低的部分，与平均值相差很远）。

这种过滤是有帮助的，因为非常罕见的功能很难学习。但重要的是要认识到，您的数据集将偏向于head查询。在提供服务的时候，尾部数据预测会差一些，因为这些是从你的训练数据中筛选出来的。尽管这种偏差无法避免，但在分析过程中要注意这一点。

不平衡数据

类比例倾斜的分类数据集称为不平衡数据集。占数据集很大比例的类称为多数类。占较小比例的是少数类。

不平衡怎么衡量，下表所示：



当在不平衡数据集上进行分类任务时，需要使用特别的采样技术。

为什么会有问题？相对于负样本来说，正样本数量太少了，训练模式会把大部分时间花在负样本上，而不会从正样本中学到足够的东西。例如，如果批处理大小为128，则许多批样本中将没有正样本，因此梯度的信息量将更少。

如果你有一个不平衡的数据集，首先尝试在真实数据集上训练。如果这个模型很好地工作并且推广，工作就完成了！如果没有，请尝试以下降采样和升权重技术。

Downsampling and Upweighting

处理不平衡数据的一种有效方法是对多数类进行降采样和升权重。让我们从定义这两个新术语开始：

降采样（在这种情况下）意味着训练使用多数类样本的一个子集。

增加权重意味着对于降采样类的样本，增加一个和降采样比例相等的权重。

第一步：多数类降采样。考虑一个欺诈数据集示例，包括1个正样本和200个负样本，我们可以减少负样本数量为10，降采样比例为1/20，现在我们大约有10%的正面数据，这样有利于模型训练。

第二部：升权重。最后一步是向降采样类添加样本权重。因为我们减少了20倍的采样，所以示例权重应该是20。

权重应等于用于缩小采样的因子：



为什么要进行降采样和升权重？

在降采样后添加样本权重可能看起来很奇怪。我们试图使我们的模式在少数类中得到改进——为什么我们要提高多数类的地位？以下是由此产生的变化：

更快收敛

在训练过程中，我们更经常地看到少数类，这将有助于模型更快地收敛。

节省磁盘空间

通过将多数类合并为具有更大权重的更少示例，我们将花费更少的磁盘空间来存

储它们。这种节省为少数类提供了更多的磁盘空间，因此我们可以从该类中收集更

多的示例和更广泛的示例。

校准

提高权重可以确保我们的模型仍然被校准；输出仍然可以解释为概率。

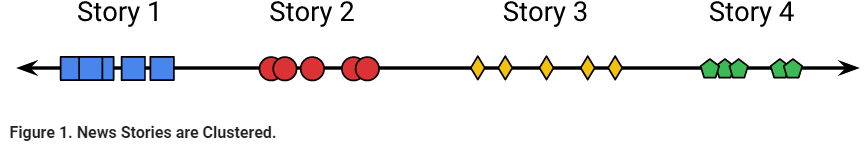
数据分割

在收集数据，采样之后，下一步是将数据分割成训练集、验证集和测试集。

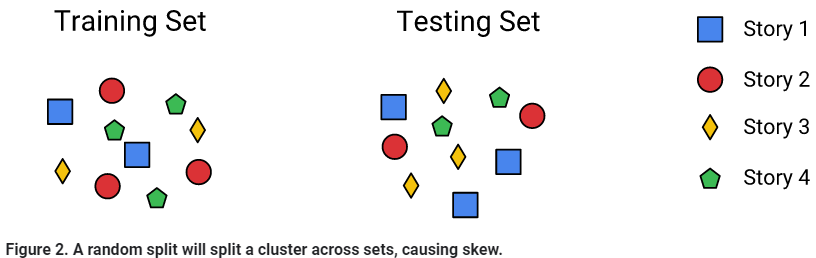
什么时候随机分割不是最好的方式

虽然随机分割是许多ML问题的最佳方法，但它并不总是正确的解决方案。例如，考虑这样的数据集，其中的样本自然地聚集成相似样本集合。

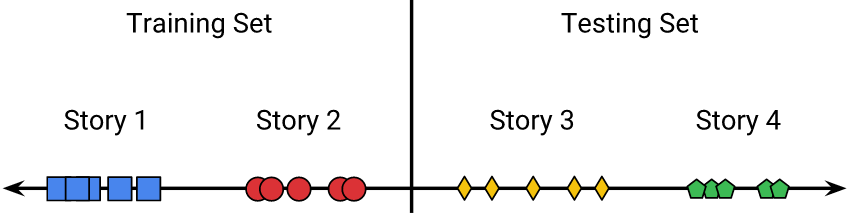
假设您希望您的模型从新闻文章的文本中对主题进行分类。为什么随机拆分会有问题？



新闻故事以集群的形式出现：同一主题的多个故事在同一时间发布。因此，如果我们随机分割数据，测试集和训练集可能包含相同的故事。事实上，这样做是行不通的，因为所有的故事都会同时出现，所以这样的分裂会导致数据倾斜。



解决这个问题的一个简单方法是，根据故事发表的时间，也许是在故事发表的那天，来分割我们的数据。这将导致同一天的故事被放在同一个拆分中。



数万条或更多的新闻，一些样本可能会在不同的日子里被分配。不过，没关系；事实上，这些故事是在新闻周期的两天内被分割的。或者，您可以抛出离截止点一定距离内的数据，以确保没有任何重叠。例如，您可以为4月份的故事进行培训，然后使用5月份的第二周作为测试集，使用周间隔防止重叠。

在线系统常用的是通过时间分割数据，即：

收集30天的数据

在1-29天数据上训练

使用第30天的数据进行评估

对于在线系统，培训数据比服务数据旧，因此此技术确保验证集反映培训和服务之间的延迟。

然而，基于时间的拆分对于非常大的数据集最有效，比如那些有上千万个示例的数据集。在数据较少的项目中，培训、验证和测试之间的分布最终会有很大的不同。

还记得机器学习速成课程中描述的机器学习文献项目中的数据分割缺陷。

这些数据是由三位作者中的一位撰写的文献，因此这些数据主要分为三组。由于团队采用了随机分割，因此在训练、评估和测试集中存在来自每个组的数据，因此模型学到了在预测时不需要的信息。无论您的数据是作为时间序列数据分组，还是按其他条件分组，都可能发生此问题。领域知识可以告诉您如何分割数据。

要设计代表数据的拆分，请考虑数据所代表的内容。黄金法则同样适用于数据拆分：测试任务应该尽可能与生产任务匹配。

随机化

考虑实际情况

使数据生成管道可复制。假设要添加特征以查看它如何影响模型质量。为了公平的实验，除了这个新特性之外，您的数据集应该是相同的。如果数据生成运行不可复制，则无法生成这些数据集。

本着这种精神，确保数据生成中的任何随机化都可以确定：

生成随机数使用相同的seed，相同的seed保证每次运行都会生成相同的输出，这样可

以重新生成数据集

使用固定散列键，散列是分割或采样数据的常用方法。您可以散列每个示例，并使用得

到的整数来决定将示例放在哪个拆分位置。哈希函数的输入不应在每次运行数据生成程

序时更改。例如，如果要按需重新创建哈希，请不要在哈希中使用当前时间或随机数。

上述方法同时适用于数据的采样和拆分。

散列的注意事项

再次想象一下，您正在收集搜索查询并使用散列来包含或排除查询。如果哈希key只使用查询，那么在多天的数据中，您要么总是包含该查询，要么总是排除它。始终包含或始终排除查询是错误的，因为：

您的训练集将只能包括有限类别的查询。

您的评估集将不会和训练数据有重叠，实际上，在提供服务时，在训练数据中能够看到

实时的数据。

相反，您可以在query+date上散列，这将导致每天不同的散列。

**数据处理**

数据处理介绍

特征工程是确定哪些特征在训练模型时可能有用，然后通过转换日志文件和其他源中的原始数据来创建这些特征的过程。在本节中，我们将重点讨论何时以及如何转换数值和分类数据，以及不同方法之间的权衡。

数据处理的原因

1. 数据兼容性的强制转换。包括：

将非数字特征转换为数字特征。不能对字符串进行矩阵乘法，因此必须将字符串转换为某种数字表示形式。

将输入调整为固定大小。线性模型和前馈神经网络有固定数量的输入节点，因此输入数据必须始终具有相同的大小。例如，图像模型需要将其数据集中的图像重塑为固定大小。

1. 可选的质量转换，可以帮助模型更好地执行。包括：

文本特征的符号化或小写。

标准化的数值特征（大多数模型在之后表现更好）。

允许线性模型将非线性引入特征空间。

严格地说，质量转换是不必要的——您的模型仍然可以在没有它们的情况下运行。但是使用这些技术可以使模型产生更好的结果。

在什么地方做转换

可以在数据产生阶段或者在模型中进行数据转换。

训练前转化

优点

计算只执行一次。

计算可以查看整个数据集来确定转换。

缺点

转换需要在预测时再现。小心歪斜！

任何转换更改都需要重新运行数据生成，从而导致较慢的迭代。

对于涉及在线服务的案件，歪斜更为危险。在离线服务中，您可以重用生成培训数据的代码。在在线服务中，创建数据集的代码和用于处理实时流量的代码几乎必然不同，这使得很容易引入歪斜。

在模型中转化

对于这种方法，转换是模型代码的一部分。该模型将未转换的数据作为输入，并将其转换为模型中的数据。

优点

简单的迭代。如果更改转换，则仍可以使用相同的数据文件。

在训练和预测时间上，你可以保证同样的转换。

缺点

昂贵的转换会增加模型延迟。

按批转换。

每批转换有许多注意事项。假设您希望通过一个特征的平均值来规范化它——也就是说，您希望将特征值更改为平均值0和标准偏差1。在模型内部进行转换时，此规范化将只能访问一批数据，而不能访问完整的数据集。您可以按批处理中的平均值进行规范化（如果批数据不均匀，则很危险），也可以提前计算平均值并将其作为模型中的常量进行修复。我们将在下一节探讨规范化。

探索、清理和可视化您的数据

在对数据执行任何转换之前，请先对其进行浏览和清理。在收集和构建数据集时，您可能已经完成了以下一些任务：

检查几行数据。

检查基本统计数据。

修复丢失的数字项。

经常可视化数据。考虑一下Anscombe的四重奏：你的数据可以从一个角度看基本的统计数据，而当图表化的时候则可以从另一个角度看。在你开始深入分析之前，用图形的方式来查看你的数据，可以是散点图，也可以是直方图。不仅在管道开始时查看图形，而且在转换过程中查看图形。可视化将帮助您不断检查您的假设，并看到任何重大变化的影响。

转换数值型数据

数值型数据有两种转化方法：

归一化-将数字数据转换为与其他数字数据相同的比例。

分箱-将数值（通常是连续的）数据转换成分类数据。

为什么需要归一化

如果在同一个特征中有非常不同的值（例如，城市人口），则需要进行规范化。如果没有标准化，如果梯度更新太大，你的训练可能会都是NaN。

你可能有两个不同的特征，范围大不相同（例如年龄和收入），导致梯度下降“反弹”并减慢收敛速度。像Adagrad和Adam这样的优化器通过为每个特性创建单独的有效学习率来防止这个问题。但是优化器不能从单个特性中的大量值中拯救您；在这些情况下，您必须规范化。

归一化

标准化的目标是将特征转换为相似的尺度。这提高了模型的性能和训练的稳定性。

归一化的技术

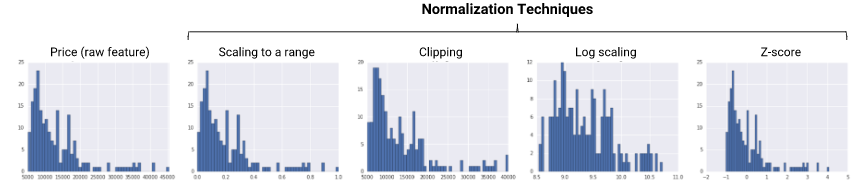
缩放

修剪

Log缩放

z-score

下图显示了每种规格化技术对左侧原始特征（价格）分布的影响。这些图表基于1985年沃德汽车年鉴中的数据集，该年鉴是汽车数据集下UCI机器学习库的一部分。



缩放

缩放意味着将浮点特征值从其自然范围（例如，100到900）转换为通常为0和1（或有时为-1到+1）的标准范围。使用以下简单公式缩放到一个范围：



当满足以下两个条件时，缩放到范围是一个不错的选择：

你知道你的数据的近似上界和下界，只有很少或没有异常值。

您的数据大致均匀分布在该范围内。

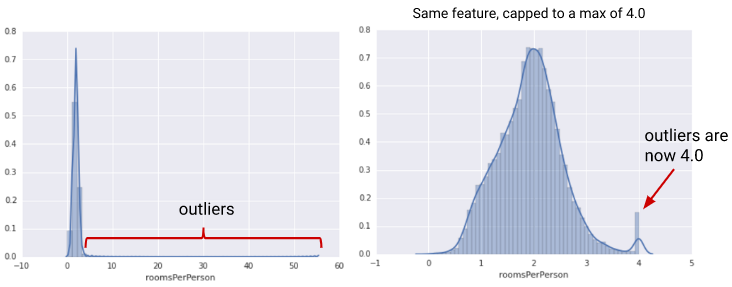
一个很好的例子就是年龄。大多数年龄值在0到90岁之间，范围的每一部分都有大量的人。

相反，你不会使用收入比例，因为只有少数人有很高的收入。收入线性标度的上限将非常高，大多数人将被挤在标度的一小部分。

修剪

如果数据集包含极端异常值，可以尝试要素剪裁，将高于（或低于）某个值的所有要素值限制为固定值。例如，可以将高于40的所有温度值剪裁为正好40。

可以在其他规格化之前或之后应用特征剪裁。



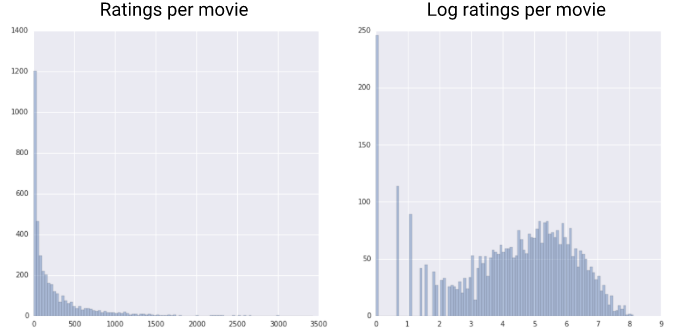
另一个简单的裁剪策略是通过z-score裁剪到+Nσ（例如，限制到+3σ）。注意，σ是标准差。

Log缩放

log缩放计算值的log以将宽范围压缩到窄范围。



当少数值有多个点，而大多数其他值只有几个点时，log缩放很有帮助。这种数据分布称为幂律分布。电影收视率就是一个很好的例子。在下面的图表中，大多数电影的收视率很低（数据在尾部），而少数电影的收视率很高（数据在头部）。对数标度改变了分布，有助于提高线性模型的性能。

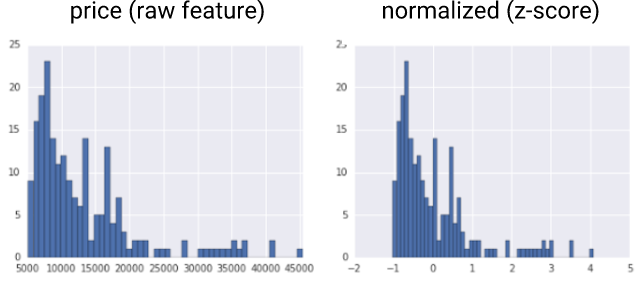


Z-score

Z分数是标度的一种变化，表示偏离平均值的标准差的数量。您可以使用z-score来确保您的特性分布具有mean=0和std=1。当有一些异常值时，它是有用的，但不是非常极端，以至于需要剪切。

计算点x的z-score的公式如下：

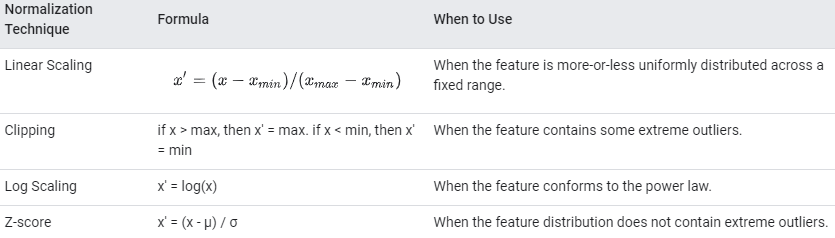




请注意，z-score将范围约为40000的原始值压缩到大约-1到+4的范围内。

假设您不确定异常值是否真的是极端的。在这种情况下，从z-score开始，除非您有不希望模型学习的特征值；例如，这些值是测量错误或怪癖的结果。

总结



分箱

如果选择对数值特征进行bucketize，请清楚如何设置边界以及应用哪种类型的bucketing：

具有等距边界的桶：边界是固定的，并且包含相同的范围（例如，0-4度、5-9度和10-14度，或$5000-$9999、10000-$14999和$15000-$19999）。有些水桶可能包含许多点，而其他水桶可能很少或没有。

带分位数边界的桶：每个桶有相同的点数。边界不是固定的，可以包含范围很窄或很宽的值。

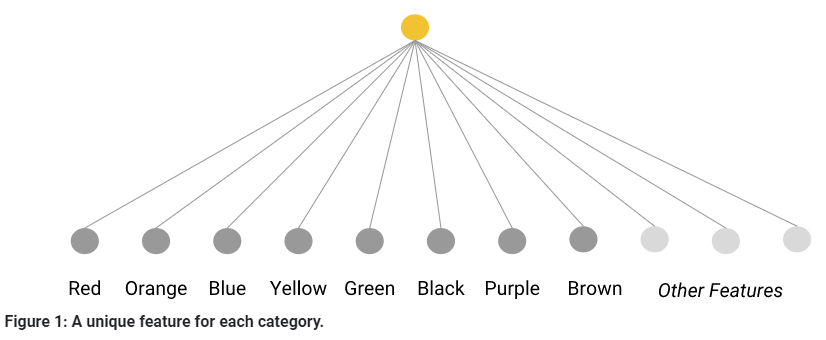
等间距边界屈曲是一种适用于大量数据分布的简单方法。但是，对于倾斜的数据，尝试使用分位数分桶。

转换分类数据

您的某些特征可能是不在有序关系中的离散值。例如狗的品种、文字或邮政编码。这些特性称为类别，每个值称为一个类别。您可以将分类值表示为字符串或偶数，但您将无法比较这些数字或从彼此中减去它们。

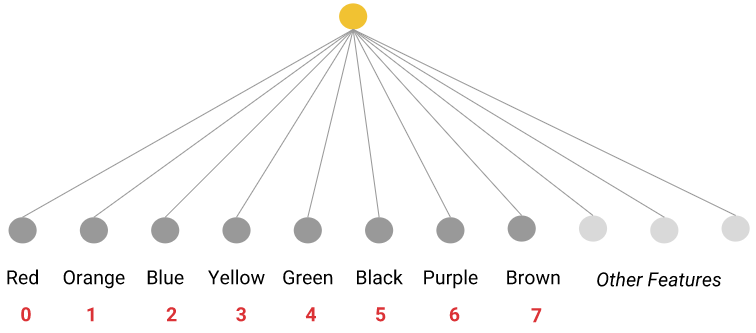
通常，应该将包含整数值的特征表示为分类数据，而不是数字数据。例如，考虑一个邮政编码特性，其中的值是整数。如果错误地用数字表示此功能，则要求模型在不同邮政编码之间找到数字关系；例如，您希望模型确定邮政编码20004是邮政编码10002的两倍（或一半）信号。通过将邮政编码表示为分类数据，可以使模型为每个单独的邮政编码找到单独的信号。

如果数据字段的类别数很小，例如星期几或有限的颜色调色板，则可以为每个类别创建唯一的功能。例如：



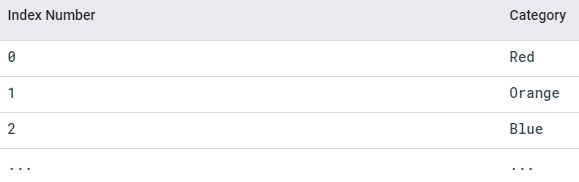
然后，模型可以学习每个颜色的单独权重。例如，也许模型可以了解到红色汽车比绿色汽车更贵。

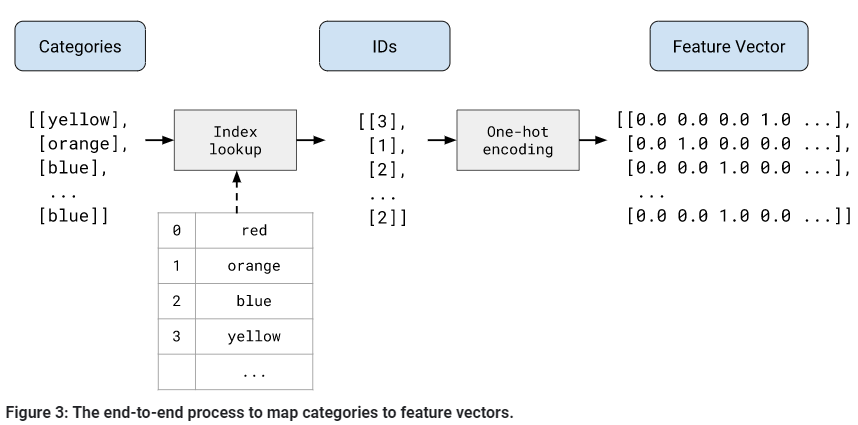
特征可以添加索引。



这个索引映射称为字典

在一个字典中，每一个值代表一个特征





使用稀疏的表达

如果您的类别是一周中的几天，那么您可能会以特征向量[0，0，0，0，1，0，0]来表示星期五。然而，大多数ML系统的实现将在内存中用稀疏表示来表示这个向量。常见的表示是非空值及其对应索引的列表，例如，值为1.0，索引为[4]。这允许您花费更少的内存存储大量的0，并允许更有效的矩阵乘法。就基础数学而言，[4]相当于[0，0，0，0，1，0，0]。

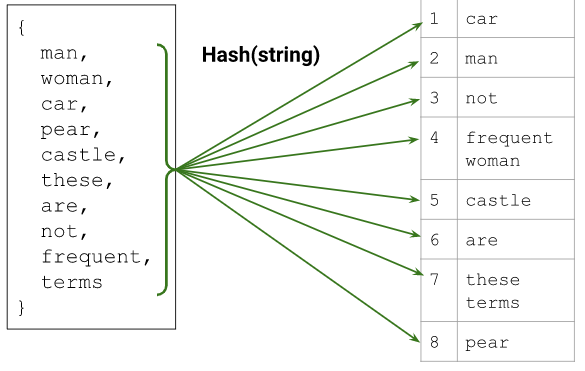
Out of Vocab(OOV)

正如数值数据包含离群值一样，分类数据也包含离群值。例如，考虑一个包含汽车描述的数据集。这个数据集的一个特点可能是汽车的颜色。假设常见的汽车颜色（黑色、白色、灰色等）在这个数据集中得到了很好的表示，并且您将它们中的每一种都划分为一个类别，这样您就可以了解这些不同的颜色是如何影响价值的。不过，假设这个数据集包含少量颜色古怪的汽车（淡紫色、深紫色、鳄梨色）。而不是给每种颜色一个单独的类别，你可以把它们归为一个全面的类别称为出声（OOV）。通过使用OOV，系统不会浪费时间对每种稀有颜色进行训练。

Hashing

另一个选择是将每个字符串（类别）散列到可用的索引空间中。散列通常会导致冲突，但是您依赖于模型来学习同一索引中类别的一些共享表示，这些表示对于给定的问题很有效。

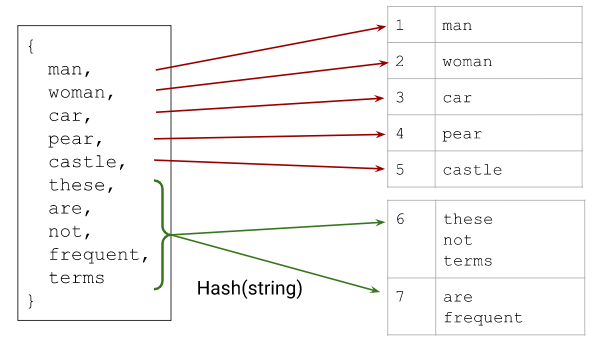
对于重要的术语，由于冲突，散列可能比选择词汇表更糟糕。另一方面，散列不需要组装词汇表，如果特性分布随着时间的推移发生了重大变化，这将非常有利。



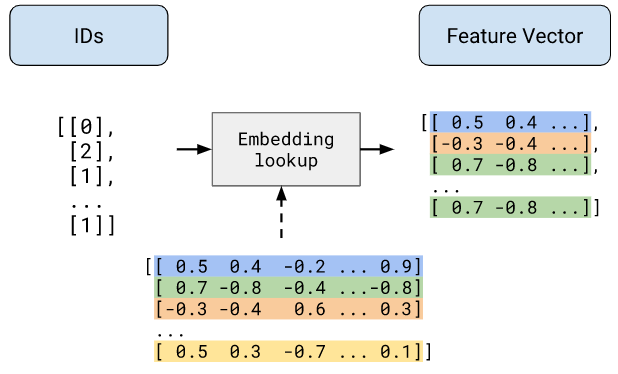
混合使用hashing和vocabulary

您可以采用混合方法并将散列与词汇表结合起来。为数据中最重要的类别使用词汇表，但是使用多个OOV，并使用散列将类别分配给桶。

哈希桶中的类别必须共享一个索引，并且模型可能不会做出很好的预测，但是我们已经分配了一些内存来尝试学习词汇表之外的类别。



注意Embeddings

回想一下机器学习速成课程，嵌入是表示为连续值特征的分类特征。深度模型经常将索引从索引转换为嵌入。

我们讨论过的其他转换可以存储在磁盘上，但是嵌入是不同的。由于嵌入是经过训练的，所以它们不是典型的数据转换它们是模型的一部分。它们使用其他模型权重进行训练，在功能上相当于一层权重。

预处理的嵌入件呢？预训练的嵌入在训练过程中仍然是典型的可修改的，因此它们在概念上仍然是模型的一部分。