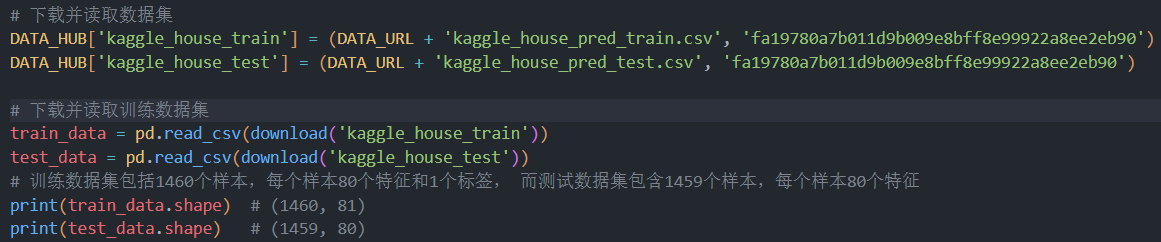
实战：Kaggle房价预测

1. 下载并读取数据集（从指定的URL下载并读取训练和测试数据集。）

**2．数据预处理**

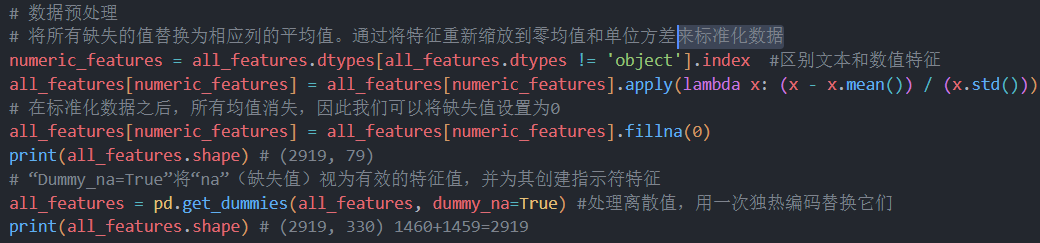
将所有缺失的值替换为相应列的平均值，并将特征重新缩放到零均值和单位方差来标准化数据（均值变为0，方差变为1）。在标准化数据之后，所有均值消失，因此我们可以将缺失值设置为0。最后处理离散值，用一次独热编码替换它们。

使用 pandas 库中的 get\_dummies 函数对 all\_features 数据框中的离散值进行独热编码处理。具体来说，它将数据框中的分类变量（通常是离散值）转换为多个二进制列，每个分类值对应一个二进制列。

**dummy\_na=True**：这个参数表示在独热编码过程中，缺失值（NaN）也会被处理。默认情况下，get\_dummies 会忽略缺失值，但通过设置 dummy\_na=True，缺失值会被转换为一个单独的虚拟变量

**独热编码**（One-Hot Encoding）是一种常用的技术，用于处理分类数据。在机

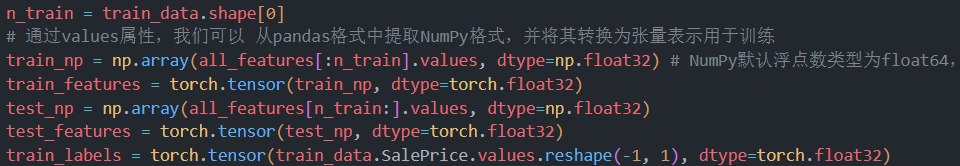
器学习中，分类变量通常需要转换为数值形式才能被模型处理。独热编码通过为每个分类值创建一个二进制列，使得每个分类值都有一个唯一的表示，从而避免了数值变量之间的线性关系

假设 all\_features 数据框包含一个名为 color 的列，其值为 ['red', 'blue', 'green', 'red', 'blue', None]。执行这行代码后，color 列将被转换为多个二进制列，例如 color\_red、color\_blue、color\_green 和 color\_nan，其中每个二进制列表示原始 color 列中的一个分类值。

训练数据集前80列为特征，最后一列为标签，即根据这些特征，比如房子面积，房屋类型等（有数值有文本），来预测房子买了多少钱

all\_features = pd.concat((train\_data.iloc[:, 1:-1], test\_data.iloc[:, 1:]))

这行代码是去掉id列 这是不能用的，提交结果的时候用



**n\_train = train\_data.shape[0]**：获取训练数据的样本数量，即train\_data DataFrame的行数

DataFrame 是 Pandas 中的另一个核心数据结构，用于表示二维表格型数据。

DataFrame 是一个表格型的数据结构，它含有一组有序的列，每列可以是不同的值类型（数值、字符串、布尔型值）。

DataFrame 既有行索引也有列索引，它可以被看做由 Series 组成的字典（共同用一个索引）。

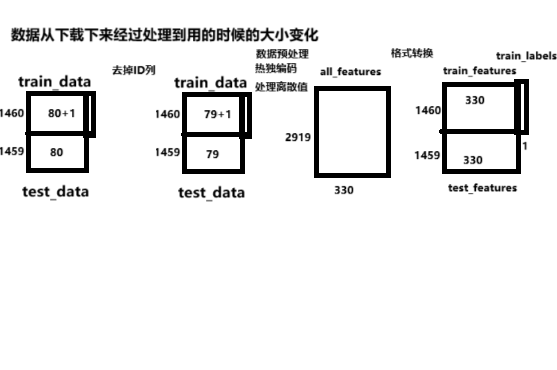
DataFrame 提供了各种功能来进行数据访问、筛选、分割、合并、重塑、聚合以及转换等操作。

**all\_features[:n\_train].values**提取了前n\_train行（即训练数据）的特征值，并将其转换为NumPy数组（在数据预处理的时候把训练数据集和验证数据集合并起来了，这里是分开了转换格式）

**dtype=np.float32**指定了数组的数据类型为32位浮点数。这是因为在PyTorch中，默认的浮点数类型是32位浮点数（float32），而NumPy的默认浮点数类型是64位浮点数（float64）。使用32位浮点数可以减少内存使用，并可能提高计算速度

将训练数据和测试数据从**Pandas DataFrame**格式转换为**NumPy数组**，然后再转换为**PyTorch张量**

* train\_data.SalePrice.values获取了SalePrice列的所有值。
* reshape(-1, 1)将一维数组转换为二维数组，其中-1表示自动计算该维度的大小，1表示将数组转换为一列（标签列）



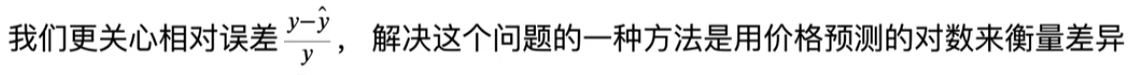


使用均方误差（Mean Squared Error, MSE）作为损失函数。MSE是回归问题中常用的损失函数，用于衡量预测值与真实值之间的差异

获取输入特征维度（这里是330）：train\_features是一个包含训练数据的张量，shape[1]表示输入特征的数量，即每个样本的特征维度

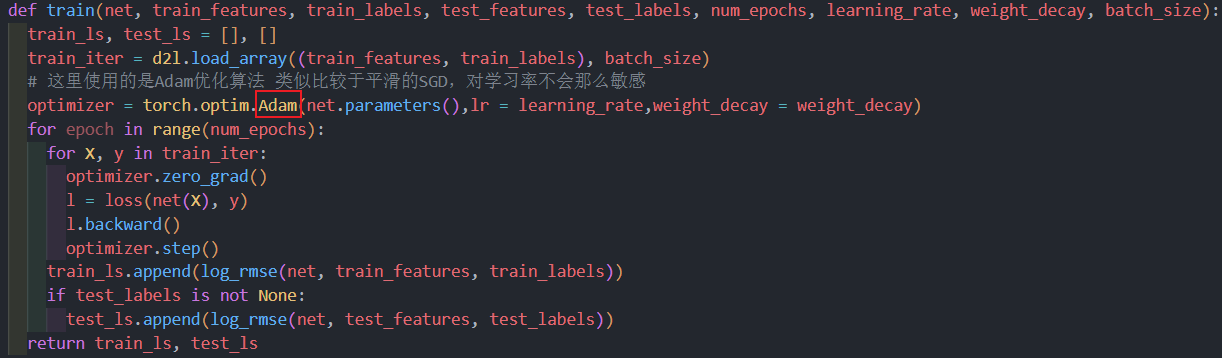
get\_net定义了一个简单的线性回归模型，即一个包含一个线性层的神经网络。nn.Sequential是一个容器，用于顺序地堆叠多个神经网络层。nn.Linear(in\_features, 1)表示一个线性层，输入特征维度为in\_features，输出特征维度为1

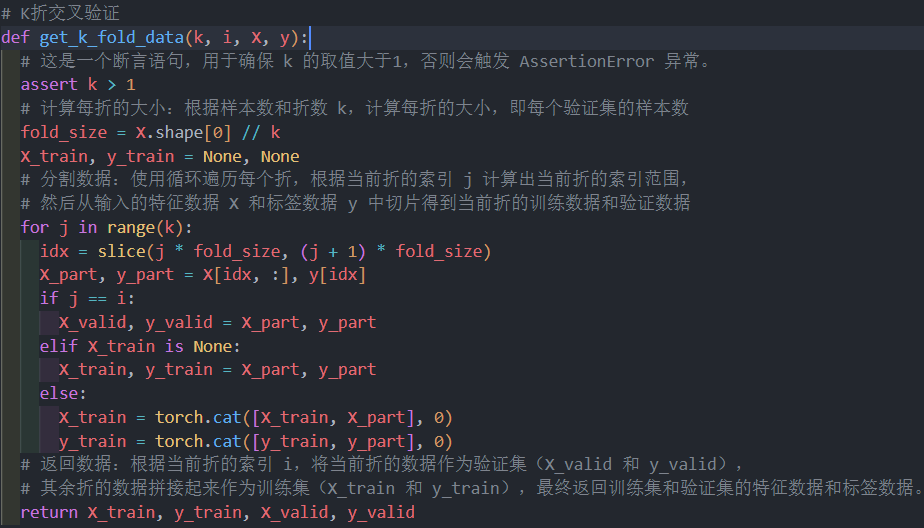
（真实值-预测值）/真实值



相当与先把预测值和真实值取对数，然后再正常的做线性回归

log\_rmse函数用于计算相对误差，并返回均方根误差（RMSE）。具体步骤如下：

1. torch.clamp(net(features), 1, float('inf'))：将预测值限制在1到正无穷之间，以避免在取对数时出现负值或零值，从而提高数值稳定性。
2. torch.log(clipped\_preds)和torch.log(labels)：对预测值和真实值取对数。
3. loss(torch.log(clipped\_preds),torch.log(labels))：计算对数预测值和真实值之间的均方误差。
4. torch.sqrt(...)：对均方误差取平方根，得到均方根误差。
5. rmse.item()：将张量转换为Python标量，并返回。

这里跟之前不一样的是使用的Adam优化算法 类似比较于平滑的SGD，对学习率不会那么敏感

* **k**：表示将数据集分成 k折，通常是一个大于1的整数。(5)
* **i**：表示当前是第几折，取值范围是 0 到 k-1。
* **X**：表示输入特征数据，是一个形状为 (样本数, 特征数) 的张量。(1460,330)
* **y**：表示对应的标签数据，是一个形状为 (样本数,) 的张量。(1460)

1.使用断言语句 assert k > 1 确保参数 k 的取值大于1，否则会触发 AssertionError 异常。

2.计算每折的大小：样本数除以折数 k，得出每折的大小，即每个验证集的样本数。

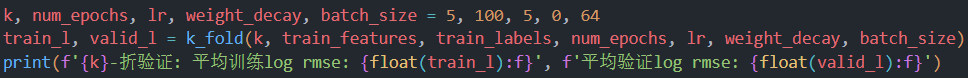
3.初始化训练集和验证集的特征数据和标签数据为 None。

4.使用循环遍历每个折，根据当前折的索引 j 计算出当前折的索引范围，然后从输入的特征数据 X 和标签数据 y 中切片得到当前折的训练数据和验证数据。

5.根据当前折的索引 i，将当前折的数据作为**验证集**（X\_valid 和 y\_valid），其余折的数据拼接起来作为**训练集**（X\_train 和 y\_train）。

6.返回训练集的特征数据和标签数据和验证集的特征数据和标签数据。

**torch.cat**是 PyTorch 库中的一个函数，用于在指定维度上连接两个或多个张量（tensor）。这个函数在处理多维数据时非常有用，尤其是在需要将多个数据块合并成一个连续的数据块时

计算并打印出k折交叉验证的平均训练log rmse和平均验证log rmse。（求和/k做平均）

float(train\_l):f 和 float(valid\_l):f 将列表转换为浮点数，并使用固定小数点格式化输出

* **k**: 交叉验证的折数设置为5，意味着数据将被分成5个子集，每次训练时使用其中4个子集作为训练集，剩下的1个子集作为验证集。
* **num\_epochs**: 训练模型时的迭代次数设置为100。
* **lr**: 学习率设置为5，通常学习率是一个小于1的正数，但在实际应用中，学习率通常较小，如0.001。
* **weight\_decay**: 权重衰减系数，用于防止过拟合，这里设置为0，表示不使用权重衰减。
* **batch\_size**: 每次训练时使用的样本数为64