#### （一）比赛介绍

#### 比赛主要是一个商品价格的回归预测问题，比赛提供了用户在二手商品交易网站平台上卖方提供的商品相关信息以及一些物品的价格，通过提取特征训练回归模型从而预测没有价格的物品。比赛提供的字段有文本类的字段包括商品名称、商品所属的类别名称，商品所属的品牌名以及商品描述等，另外还有一些数值类型的字段包括是否包邮，以及商品自身的状况等。

（二）比赛整体的方案

#### 比赛总共采用了两种解决方案：一种是采用RNN and Ridge两种算法，并采用bagging策略，针对两种模型预测的结果最后求取加权和得到最后的结果。另外一种也是采用bagging融合的策略，不过采用的是FTRL+FM+LGB三种模型。

（三）方案一介绍：

首先来讲第一种方案：

1. **停用词过滤**：首先利用nltk文本处理包中的stopwords对文本类数据字段（商品名称和商品描述）做一个简单的停用词过滤【这步最后没有用】
2. **计算文本类数据字段描述的长度**：计算文本（商品名称和商品描述）所使用的单词的个数，因为文本长度与最后商品的价格会有一定的相关性。【其中商品描述是“No description yet”，最后文本长度设为0；商品名称可能影响不大，但最后加入计算也不会对模型性能降低】
3. **拆分类别字段形成更细致的类别信息**：将商品所属类别字段进一步划分成三个更细致的分类，以使得模型获取更多的信息维度。
4. **填充缺失值**：品牌名称字段缺失数量比较多，先用“missing”填充，然后针对品牌名是“missing”的查看其商品名中是否出现品牌名，若出现则将出现的品牌名替换掉“missing”。将商品描述中的“No description yet”信息替换成“missing”以提升模型的性能
5. **类别型字段编码（RNN）**： 利用scikit中的LabelEncoder()函数对类别型的文本字段（类别名，品牌名以及各个子类别名）进行数字化编码转换。每一类是一种数值。
6. **文本数据转换数值序列（RNN）：**利用keras.preprocessing.text中的Tokenizer将商品描述、商品名称，商品类别的文本数据大小写转换后进行单词的分割，最后用texts\_to\_sequences转换成数值的序列。
7. **序列长度截断补长保持相同长度作为RNN的输入**：主要对商品名称和商品描述文本数值序列进行pad\_sequences处理，转换为numpy类型的数组【其余其他字段转换为numpy的array】
8. 每个字段都作为RNN的一个输入（InputLayer），然后Embedding，然后经过GRU层和Flatten层，最后将经过GRU层和Flatten层后的输出Concatenate到一起形成一个dense层，最后经过几层的dense和dropout层后输出到最后一个神经元的层，得到预测的输出。

【这里需要注意几个网络层：

* model.add(Embedding(1000, 64, input\_length=10))

# the model will take as input an integer matrix of size (batch, input\_length).

# the largest integer (i.e. word index) in the input should be no larger than 999 (vocabulary size).

# now model.output\_shape == (None, 10, 64), where None is the batch dimension.

# **input\_length**: Length of input sequences, when it is constant. This argument is required if you are going to connect Flatten then Dense layers upstream (without it, the shape of the dense outputs cannot be computed).**这里的大小是10**

# **input\_dim**: int > 0. Size of the vocabulary, i.e. maximum integer index + 1**.这里的大小是1000**

# **output\_dim**: int >= 0. Dimension of the dense embedding.**这里的大小是64**

# input\_array = np.random.randint(1000, size=(32, 10))输入层的大小就是32\*10，每个元素在0~999之间

* keras.layers.core.Flatten() Flatten层用来将输入“压平”，即把多维的输入一维化，常用在从卷积层到全连接层的过渡。Flatten不影响batch的大小。
* 使用GRU而不用LSTM原因：GRUs are faster than LSTMs
* *# as the first layer in a Sequential model*

*model = Sequential()*

*model.add(LSTM(32, input\_shape=(10, 64)))*

model.input\_shape：(None, 10, 64) 形如（samples，timesteps，input\_dim）的3D张量

model.output\_shape：(None, 32) 形如（samples，output\_dim）的2D张量

# *to stack recurrent layers, you must use return\_sequences=True* *# on any recurrent layer that feeds into another recurrent layer.* *# note that you only need to specify the input size on the first layer.*

model = Sequential()

model.add(LSTM(64,*input\_shape=(10,64)*,return\_sequences=**True**))

model.add(LSTM(32, return\_sequences=**True**))

model.add(LSTM(10))】

1. RNN模型训练：设置epoch、初始学习率、最终学习率以及衰减学习率等
2. 向量化所有字段数据（Ridge）：利用CountVectorizer和TfidfVectorizer函数将文本数据字段进行向量化转换，并用pipeline包提供的FeatureUnion类来进行整体并行处理。最后得到sparse matrix。
3. 训练Ridge和RidgeCV模型（2-fold）
4. 综合几个模型（RNN/Ridge/RidgeCV）的预测结果，求取各个模型的权重最优的组合，使用一个简单的循环遍历所有可能的比率，以在验证集中找到最佳模型组合比例。
5. 方案二介绍：

利用FTRL+FM+LGB model来做predict，用到一个**wordbatch**的包(Parallel text feature extraction for machine learning.)

1. 数据预处理和特征提取：

* **拆分类别字段形成更细致的类别信息**
* **cutting函数：**求数据集中相关字段中不等于‘missing’的出现频率最高的前NUM\_BRANDS的值，然后将该字段不在前NUM\_BRANDS的值赋值为‘missing’
* **定义文本正则的函数，首先获取停用词表，然后定义正则函数，并制定规则过滤词句**
* **针对‘name’'item\_description'用wordbatch.WordBatch生成hashtf稀疏向量；针对'general\_cat'/'subcat\_1'/'subcat\_2'用CountVectorizer方法生成稀疏向量；针对'brand\_name'字段将品牌名二值稀疏化LabelBinarizer(sparse\_output=True)，sparse\_output产生csr稀疏矩阵形式；'item\_condition\_id', 'shipping'字段onehot编码后转换为csr稀疏矩阵形式**

1. 利用FTRL和FM\_FTRL模型训练及预测
2. 利用lgb模型训练及预测