TFT

> 一个好的时序模型需要考虑的东西可能有哪些：
> 单变量、多变量时间序列通吃
> 既能利用序列自身的历史信息，也能利用一些其他变量的信息
> 除了动态信息，**静态信息也能利用**，比如一些统计特征
> 能够**适应复杂模式以及简单模式的时间序列**
> 能够多步预测显然比单步预测再来递推来得好（避免累积误差
> 不仅是点预测，还能知道预测值的不确定区间
> Explainability is all we need!

变量类别

时间序列预测中，所有的变量都划分为两个大类：

1. 静态变量；
2. 动态变量；

1. 静态变量：

再细分可以划分为**静态连续变量**和**静态离散变量**。
1）**静态离散变量：** 例如商品所在的城市，商品的大类，商品所在销售区域等等，这些变量都是不会随着时间发生变化的。
2）**静态连续变量：** 例如，商品 A 在 2020 年的年总销售金额，商品 B 去年双十一期间的销量等，这些不会随着时间变化的连续变量都是静态连续变量。

2. 动态变量：

再细分可以划分为**动态时变量**和**动态时不变变量**。
1）**动态时变量**：随着时间变化的特征（无法推断），例如我们要预测的销售金额，明天的问题等等；
2）**动态时不变变量**：随着时间变化的特征（可以推断），例如月份，星期几
那么二者的区别是什么？二者的核心区别在于**是否可以推断出来**，**动态时变变量是无法推断的**，比如在经典的温度预测的例子里，温度，湿度，气压这些都是无法推断的，随着时间变化的，我们无法事先知道的。而动态时不变变量，最典型的就是月份，星期几了，这些变量他们虽然也是随着时间变化的，但是**可以轻而易举的进行推断**从而应用到模型训练的过程中。
在将来时间段的数据集中：**动态时不变信息**可以知晓、**静态特征**因为固定也可以知晓。
常规的向量输出的深度学习模型在训练的过程中是不会用到未来的特征的，即使是静态特征，例如我们用历史的 30 天的销量数据预测未来 10 天的销量数据，我们在模型训练的过程中不会用到未来十天的任何特征。

单步预测与多步预测

1. 单步预测

所谓单步预测，就是每一次预测的时候 输入窗口 只 预测 未来一个值。
单步预测的两个策略：
输入窗口 全部使用 真实值作为 输入窗口，这种情况是只预测未来一个值的时候这个情况的。
预测未来第一个值的时候输入窗口使用全部真实值，预测后面 n-1 个的时候，预测窗口将包含有预测值，这种情况是单步预测预测未来的多个值的时候。

2. 多步预测

所谓多步预测，就是每一次预测的时候 输入窗口 预测 未来 n 个值（也叫 n 步）。
多步预测的策略：
只预测未来一次，即只预测一个 n 输入的 n 个输出，n 是滑动窗口个数，即输入 n 个滑动窗口，直接输出未来的 n 个，利用的输入全部是历史数据的真实值。
预测未来多次，即预测一个 m✖n 输入的 m✖n 个输出，n 是滑动窗口个数，m=1、2、3、4、5、6…，即输入 n 个滑动窗口作为一次输入，整体输入是 m 次这样的窗口，直接输出未来的 m✖n 个，利用的输入第一次是历史数据的真实值，第一次之后包含有预测值。

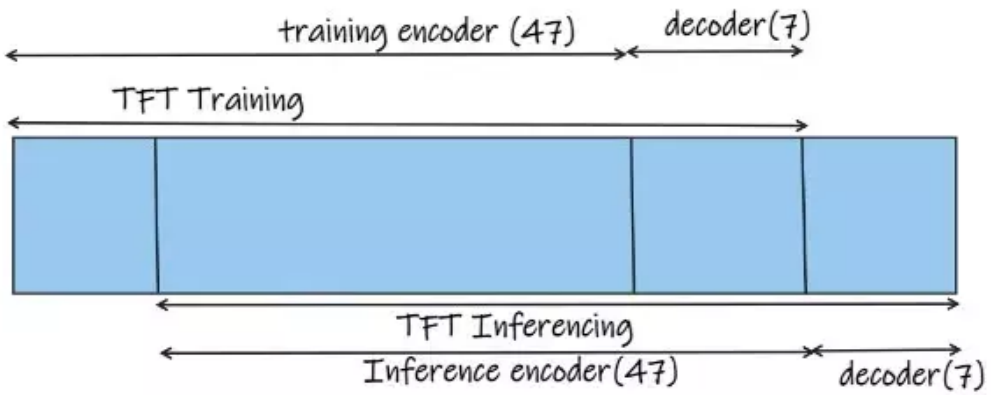
3. 单步预测和多步预测的特点

（1）多步预测的预测误差会随着步数的增加而累积，多步预测的结果也会越来越不准。但是单步预测很多时候不会有这种误差累积。

4. 总结

（1）单步预测预测的时候输入是最近的数据，而长期依赖和短期依赖已经在模型中训练了，准确率比多步预测要好。
（2）多步预测的有点就是能直接预测多步，而单步预测不能，因为单步预测直接预测多步的话会造成时间悖论（但是单步预测可以进行叠加训练预测，从而达到多步预测的目的且误差也比多步预测要小很多很多、因为输入都是利用的最近的数据）。
（3）短期预测用单步预测，长期预测用多步预测或者单步迭代训练预测。

https://blogger.googleusercontent.com/img/a/AVvXsEjn-GEpuwiBa4Od21FBnTST8-z2j...
https://blogger.googleusercontent.com/img/a/AVvXsEjn-GEpuwiBa4Od21FBnTST8-z2jAgyw3rq68A...



1. Static covariates：这些信息在整个时间序列中都是静态的，即它们不随时间变化而变化，例如人口普查数据、性别、年龄等。
2. Past-observed time-dependent inputs：这些信息是过去时间点上观察到的变量值，它们随着时间而变化，例如气温、交通流量等。
3. Apriori-known future time-dependent inputs：这些信息是我们在建模或预测时已知的将来时间点上的变量值，例如未来天气预报、未来经济指标等。

Exogenous time series

是指在时间序列分析中，**与我们所关注的主要时间序列具有相关性但不受其影响的其他时间序列。**与之相对的是内生时间序列（Endogenous time series），它是指在时间序列分析中，我们所关注的主要时间序列，其变化是由自身内在因素决定的。
举个例子，假设我们想预测某个**城市的房价(主)**，我们可能会考虑到一些可能会影响房价的因素，例如**人口增长率**、**GDP增长率**、**利率水平**等。这些因素虽然与房价具有相关性，但它们不是由房价本身决定的，而是受到其他因素的影响，比如经济发展水平、政策等。因此，这些因素就可以被看做是**exogenous time series（人口增长率、GDP增长率、利率水平）。**
在时间序列建模中，我们通常会将这些exogenous time series视为模型的额外输入，以提高模型的预测准确性。这样，我们就可以同时考虑多个与主要时间序列相关的因素，从而更好地解释和预测主要时间序列的变化。

static metadata

Static metadata是指在数据收集和处理过程中，不随时间变化的元数据。例如，在一份数据集中，数据的列名称、列数据类型、列的含义和数据的来源等都可以被视为静态元数据。

假设有一个电商平台的销售数据集，其中包含每个订单的时间、买家ID、商品ID、订单金额等信息。这个数据集中的静态元数据可以包括：

- 列名称：时间、买家ID、商品ID、订单金额
- 列数据类型：时间为日期时间类型、买家ID和商品ID为字符串类型、订单金额为浮点数类型
- 列的含义：时间列表示订单的下单时间、买家ID列表示下单买家的ID、商品ID列表示订单中的商品ID、订单金额列表示订单的总金额
- 数据来源：数据来自于该电商平台的销售记录

这些静态元数据可以帮助数据分析师或机器学习算法更好地理解数据的结构和特征，例如：

- 了解每个列的含义，可以更好地理解数据
- 确定每个列的数据类型，可以确保数据的正确性
- 确定数据来源，可以验证数据的可靠性
- 对于机器学习算法来说，理解数据的静态元数据可以帮助算法选择合适的特征和模型。例如，在这个数据集中，订单金额可能是一个重要的预测因素，因此可以在机器学习模型中将其作为一个特征来进行预测。

line

LIME（Local Interpretable Model-Agnostic Explanations）是一种用于**解释机器学习模型的算法**，它可以帮助人们理解机器学习模型对于输入数据的预测结果的解释。LIME主要通过对于单个数据点进行解释来提高对于整个模型的理解。

LIME的核心思想是**对于每一个数据点，通过构建一个局部的线性模型来近似代替原有的复杂模型，以达到更好的解释性能。**具体而言，LIME在一个数据点的附近随机生成一组数据，并对这些数据进行加权重的样本，然后通过训练一个简单的线性模型来近似复杂的模型，从而推导出该数据点的解释。

LIME的优势在于它是一种模型无关的解释方法，可以用于任何类型的机器学习模型，包括深度学习模型和黑盒模型。此外，LIME可以通过可视化工具将解释结果展现出来，让用户更容易理解和分析。

在实际应用中，LIME可以帮助人们了解一个模型对于不同特征的权重和影响，发现模型预测错误的原因，以及为模型的改进提供思路。它在解释复杂模型和提高模型可解释性方面有着广泛的应用。

假设我们有一个二元分类模型 $f(x)$ ，其中 x 是输入特征。LIME 的目标是解释在输入 x_0 处的预测结果 $f(x_0)$ 。LIME 的核心思想是在 x_0 的邻域内随机生成一些“伪数据”，并用这些数据拟合一个局部线性模型，从而近似解释 $f(x_0)$ 。伪数据的生成过程可以使用分布假设，例如高斯分布或均匀分布等。

这里给出一个简单的例子。假设我们有一个二元分类模型 $f(x) = \text{sign}(x_1 + x_2)$ ，其中 x_1 和 x_2 是输入特征。我们要解释在输入 $(2, 3)$ 处的预测结果 $f(2, 3)$ 。我们可以按照以下步骤进行解释：

1. 在 $(2, 3)$ 的邻域内随机生成一些“伪数据”，比如 $(2.1, 2.8)$ 、 $(2.3, 2.9)$ 、 $(2.2, 3.1)$ 等。
2. 对于每个“伪数据”，计算它们对应的模型输出。比如，对于 $(2.1, 2.8)$ ， $f(2.1, 2.8) = \text{sign}(2.1 + 2.8) = 1$ 。
3. 用伪数据和对应的模型输出来训练一个局部线性模型，比如一个线性回归模型。线性模型的输入是伪数据的特征，输出是对应的模型输出。
4. 根据线性模型的系数，计算每个特征的重要性。这些重要性可以用来解释模型对于输入特征的预测结果的贡献。例如，如果发现 x_1 的系数为正，说明 x_1 对于模型的预测结果为正有正向贡献。

LIME 的公式如下：

$$\phi(x) = \arg\min_{g \subset G} L\left(f, g, \pi_x\right) + \Omega(g)$$

其中， $\phi(x)$ 是解释结果， G 是可解释模型的集合， $L(f, g, \pi_x)$ 是拟合度量函数， π_x 是权重函数， $\Omega(g)$ 是复杂度函数。具体的实现方式和选择函数是根据具体情况而定的

SHAP

SHAP（SHapley Additive exPlanations）是一种用于解释模型预测结果的方法。它基于Shapley 值的概念，用于计算每个输入特征对于模型预测结果的贡献度，并且可以将这些贡献度分配给各个特征。

具体来说，假设我们有一个模型 f ，输入为特征向量 $x=(x_1,x_2,\cdots,x_p)$ ，输出为 $f(x)$ ，我们想要解释在输入 x_0 处的预测结果 $f(x_0)$ 。SHAP 的目标是计算每个特征 i 对于 $f(x_0)$ 的贡献度 $\phi_i(x_0)$ 。这些贡献度可以通过如下方式计算：

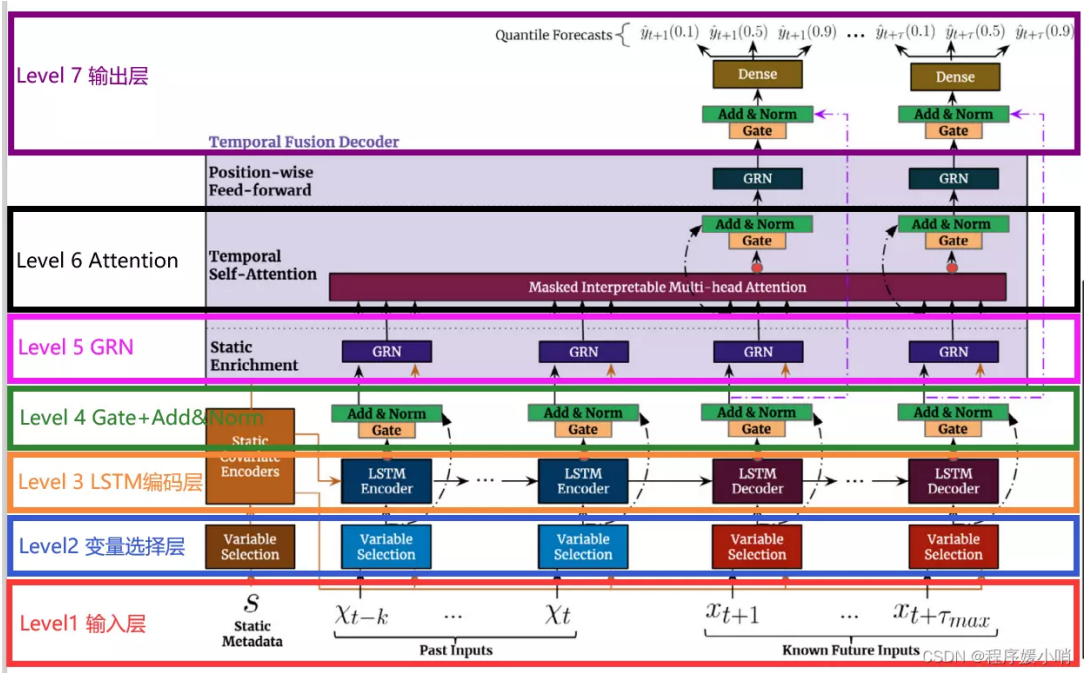
1. 假设我们有一个特征子集 S ，其中 $i \in S$$ ，x_{-i} 表示 x 去除特征 i 后的子集。我们随机选取一个特征子集 S 并计算 $f(x_S,x_{-S})$ 。
2. 计算特征 i 对于 $f(x_S,x_{-S})$ 的贡献度，即 $\phi_i(x_S,x_{-S})=f(x_S,x_{-S},i)-f(x_{-i,S},x_{-S})$ 。
3. 计算特征 i 的 Shapley 值，即 $\phi_i(x_0)=\sum_S \frac{|S|!(p-|S|-1)!}{p!}[\phi_i(x_S,x_{-S})-\mathbb{E}(\phi_i(x_S,x_{-S}))]$ ，其中 $\mathbb{E}$ 表示期望，$|S|$ 表示 S 中特征数量。

通过计算每个特征的 Shapley 值，我们可以得到每个特征对于模型预测结果的贡献度。这些贡献度可以用于解释模型的预测结果，并且可以通过可视化等方式呈现。SHAP 还提供了一些高级的算法和技术，例如 Tree SHAP、Kernel SHAP 等，用于加速计算和解决一些特殊问题。

LIME 的主要优点是它可以应用于任何机器学习模型，不需要了解模型的内部结构。然而，它需要进行采样和训练局部模型，计算量较大。

SHAP（SHapley Additive exPlanations）是一种基于 Shapley 值的方法，用于计算每个输入特征对于模型预测结果的贡献度，并且可以将这些贡献度分配给各个特征。SHAP 的具体实现方式有多种，例如 Tree SHAP、Kernel SHAP 等，它们的主要思想是基于 Shapley 值的概念，通过遍历特征子集并计算贡献度来解释模型的预测结果。

与 LIME 不同，SHAP 可以应用于多种模型，并且计算量相对较小。此外，SHAP 还可以提供全局的解释结果，而不仅仅是局部的解释结果。但是，SHAP 的计算复杂度较高，需要遍历特征子集，并且对于一些复杂的模型，可能需要进行一些特殊的处理。



Level 1 输入层

三部分：静态信息、历史（待预测变量）信息、未来（其他变量）信息

Level 2 变量选择层

说白了就是要做特征筛选

Level 3 LSTM 编码层

既然是时间序列，LSTM 来捕捉点长短期信息合情合理

Level 4 Gate + Add&Norm

门控可以理解是在进一步考虑不同特征的重要性，残差和 normalization 常规操作了

Level 5 GRN

跟 Level4 基本一样，可以理解就是在加深网络

Level 6 Attention

对不同时刻的信息进行加权

Level 7 输出层

做的是分位数回归，可以预测区间了