机器学习模型算法

主讲：韩老师

目录

CONTNETS

概述

01

PART ONE

时间序列数据定义/ Teacher says lesson

- 线性回归模型：用于建立因变量和一个或多个自变量之间的线性关系。
  - 经济预测、市场趋势分析等。
- 逻辑回归模型：用于处理分类问题，预测二元或多元分类结果的概率。
  - 风险评估、医学诊断等。
- 决策树模型：通过树形结构进行决策，可用于分类和回归问题。
  - 客户分类、产品推荐等。
- 随机森林模型：由多个决策树组成，用于提高预测准确性和抑制过拟合。
  - 信用评分、股票预测等。

时间序列数据定义/ Teacher says lesson

- 支持向量机（SVM）模型：用于分类和回归分析，能够处理高维数据并找出数据间的复杂关系。
  - 文本分类、图像识别等。
- 神经网络模型：包括深度学习模型，如卷积神经网络（CNN）和循环神经网络（RNN），用于处理复杂的非线性关系。
  - 语音识别、自然语言处理等。
- 聚类模型：如K均值聚类、层次聚类等，用于将数据分成不同的类别或群组。
  - 市场细分、社交网络分析等。
- 关联规则模型：用于发现数据中的关联规则和模式。
  - 购物篮分析、交叉销售分析等。

常用模型算法

02

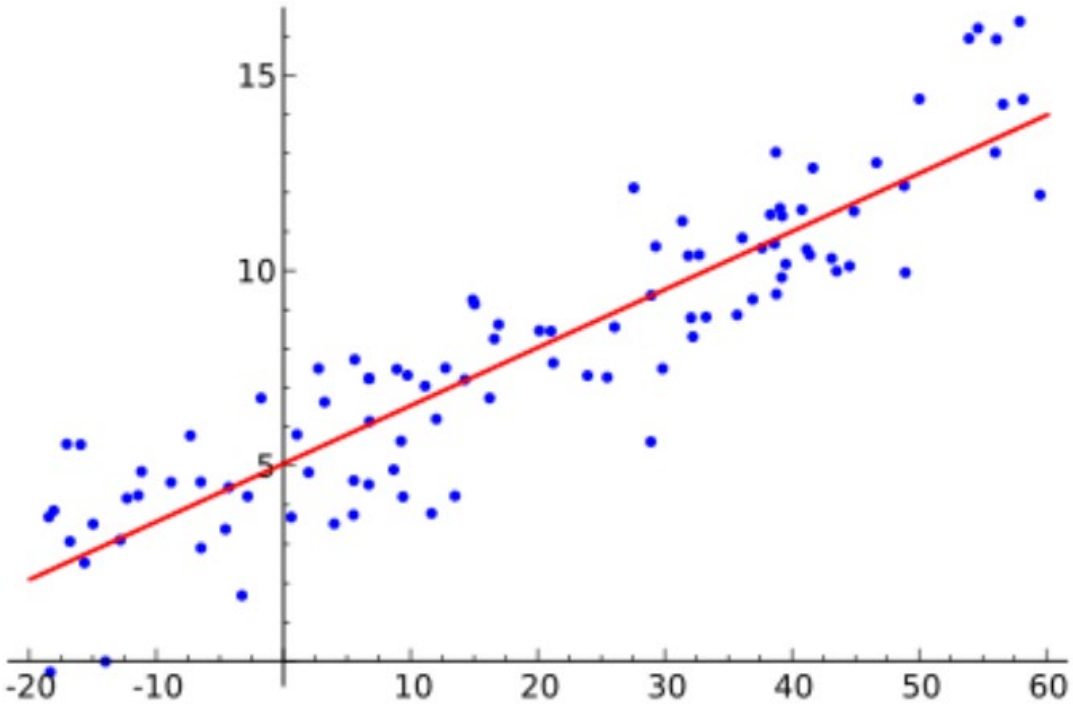
PART TWO

线性回归模型/Teacher says lesson

原理：使用数据点来寻找最佳拟合线。

公式，y = mx + c，其中 y 是因变量，x 是自变量，利用给定的数据集求 m 和 c 的值。

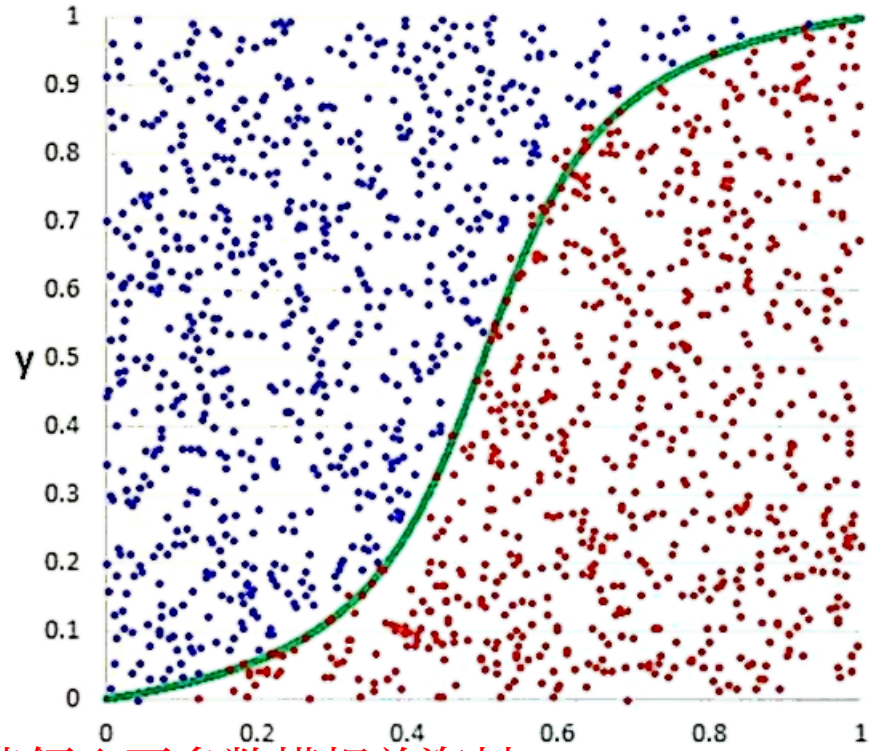
```
# 建立线性回归模型
model = LinearRegression()

# 拟合模型
model.fit(X, y)

# 输出模型参数
print('斜率:', model.coef_)
print('截距:', model.intercept_)

# 进行预测
X_new = np.array([[6], [7]])  # 新的自变量
y_pred = model.predict(X_new)  # 预测因变量
print('预测结果:', y_pred)
```



逻辑回归模型/Teacher says lesson

原理：通过将线性回归模型的输出通过一个逻辑函数（通常是Sigmoid函数）进行转换，将连续的预测值映射到0和1之间，表示属于某一类的概率。逻辑回归的模型假设因变量服从伯努利分布，通过最大似然估计来拟合模型参数。

```
# 建立逻辑回归模型
model = LogisticRegression()

# 拟合模型
model.fit(X, y)

# 输出模型参数
print('斜率:', model.coef_)
print('截距:', model.intercept_)

# 进行预测
X_new = np.array([[6], [7]])  # 新的自变量
y_pred = model.predict(X_new)  # 预测分类结果
print('预测结果:', y_pred)
```

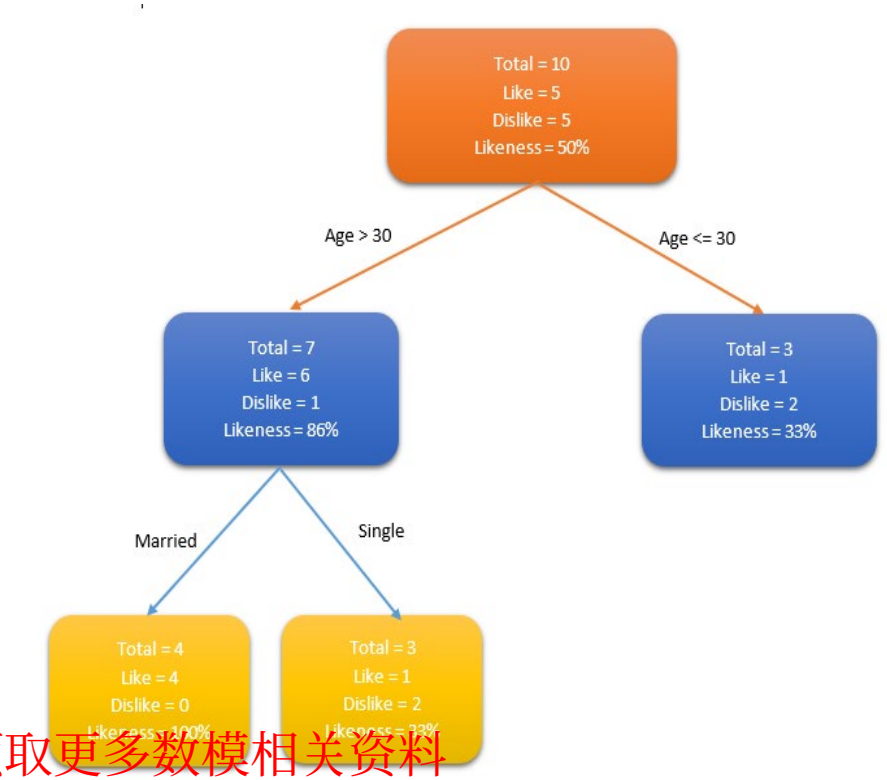


决策树模型/

Teacher says lesson

原理：决策树通过对数据集进行递归地划分，以建立一个树形结构，每个内部节点表示一个属性上的测试，每个分支代表一个测试输出，每个叶节点代表一个类别标签或者一个连续值。在分类问题中，决策树的目标是创建一个能够对实例进行准确分类的模型。在建立决策树的过程中，通常使用信息增益（ID3算法）、基尼不纯度（CART算法）或者增益率等指标来选择最佳划分属性。

```
# 准备数据
iris = datasets.load_iris()
X = iris.data
y = iris.target

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 建立决策树模型
model = DecisionTreeClassifier()

# 拟合模型
model.fit(X_train, y_train)

# 进行预测
y_pred = model.predict(X_test)

# 输出预测准确率
print('预测准确率:', accuracy_score(y_test, y_pred))
```

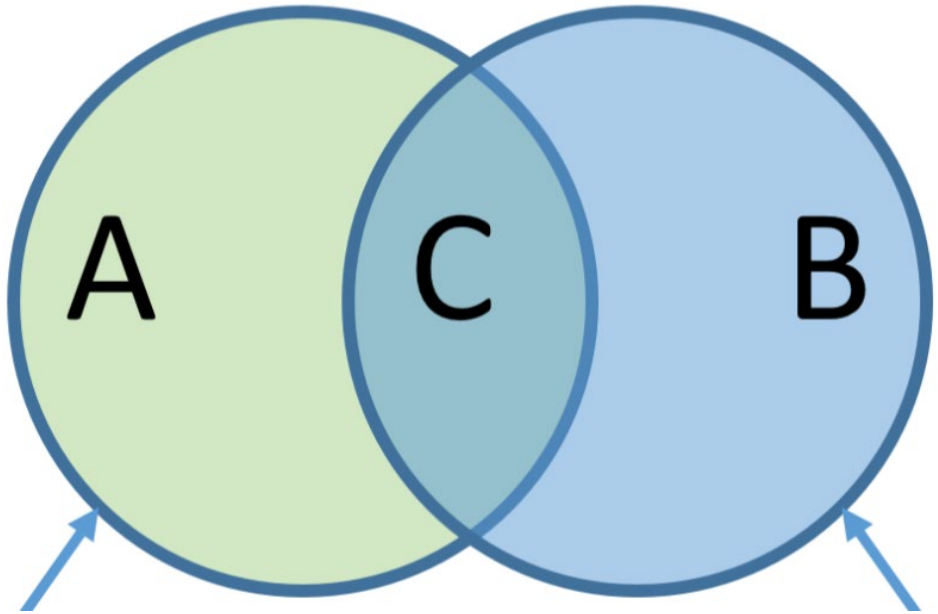


朴素贝叶斯模型/

Teacher says lesson

原理：朴素贝叶斯模型基于贝叶斯定理，它假设特征之间相互独立，即给定类别下特征之间是条件独立的。基于这个假设，可以利用训练数据计算出每个特征在各个类别下的条件概率，并结合贝叶斯定理计算出给定特征条件下各个类别的概率，最终选择概率最大的类别作为预测结果。

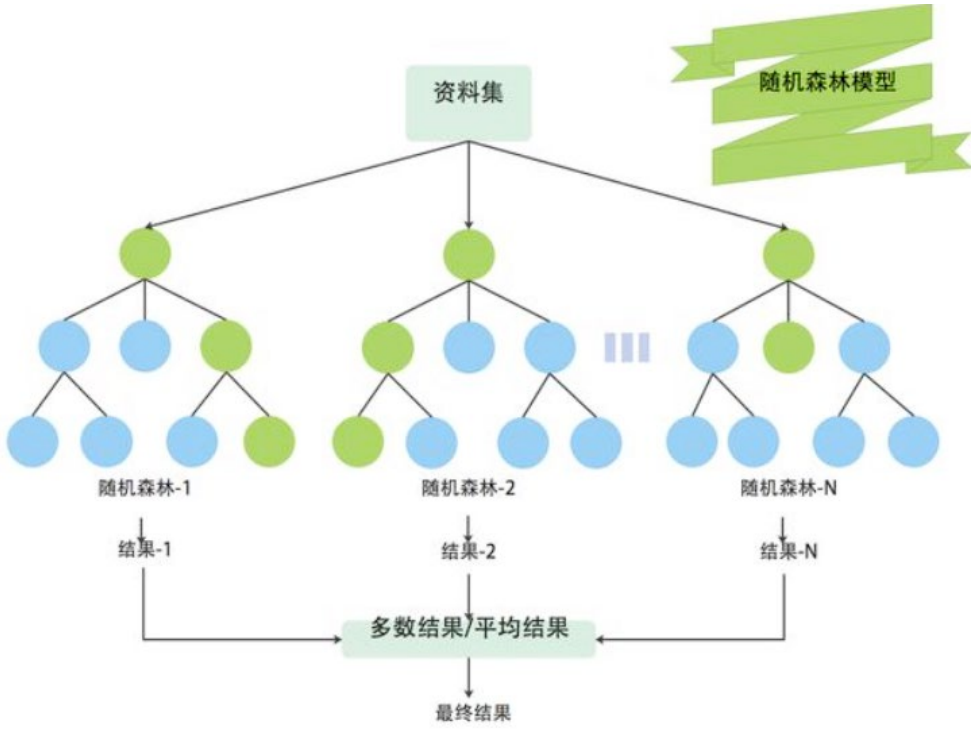
```
# 建立朴素贝叶斯模型
model = GaussianNB()

# 拟合模型
model.fit(X_train, y_train)

# 进行预测
y_pred = model.predict(X_test)
```



随机森林模型/Teacher says lesson

原理：随机森林由多棵决策树组成，每棵树都是使用随机选择的特征和随机选择的样本进行训练。在进行预测时，随机森林对所有树的预测结果进行平均或多数投票，以得到最终的预测结果。通过引入随机性，随机森林能够降低模型的方差，提高模型的鲁棒性。

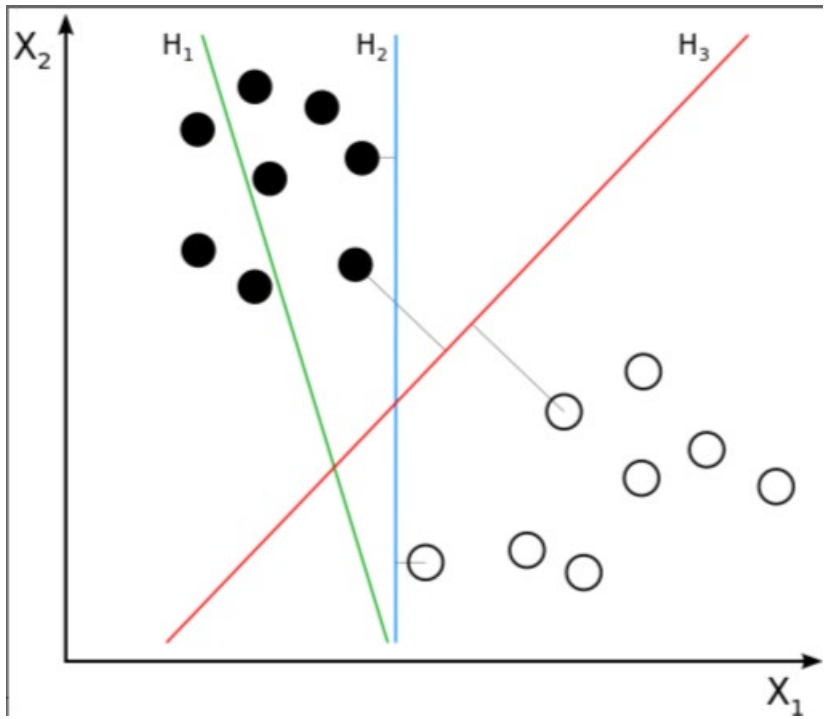
```
# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 建立随机森林模型
model = RandomForestClassifier(n_estimators=100, random_state=42)  # 这里指定了100棵树

# 拟合模型
model.fit(X_train, y_train)

# 进行预测
y_pred = model.predict(X_test)
```



支持向量机模型/ Teacher says lesson

原理：支持向量机的目标是找到一个能够将不同类别的数据分隔开的超平面，使得两个类别的间隔尽可能地大。在实际问题中，如果数据线性不可分，支持向量机会使用核函数将数据映射到高维空间中，以找到一个能够分隔数据的超平面。在分类问题中，支持向量机会选择支持向量（离超平面最近的数据点），并通过对支持向量进行处理来构建分类模型。

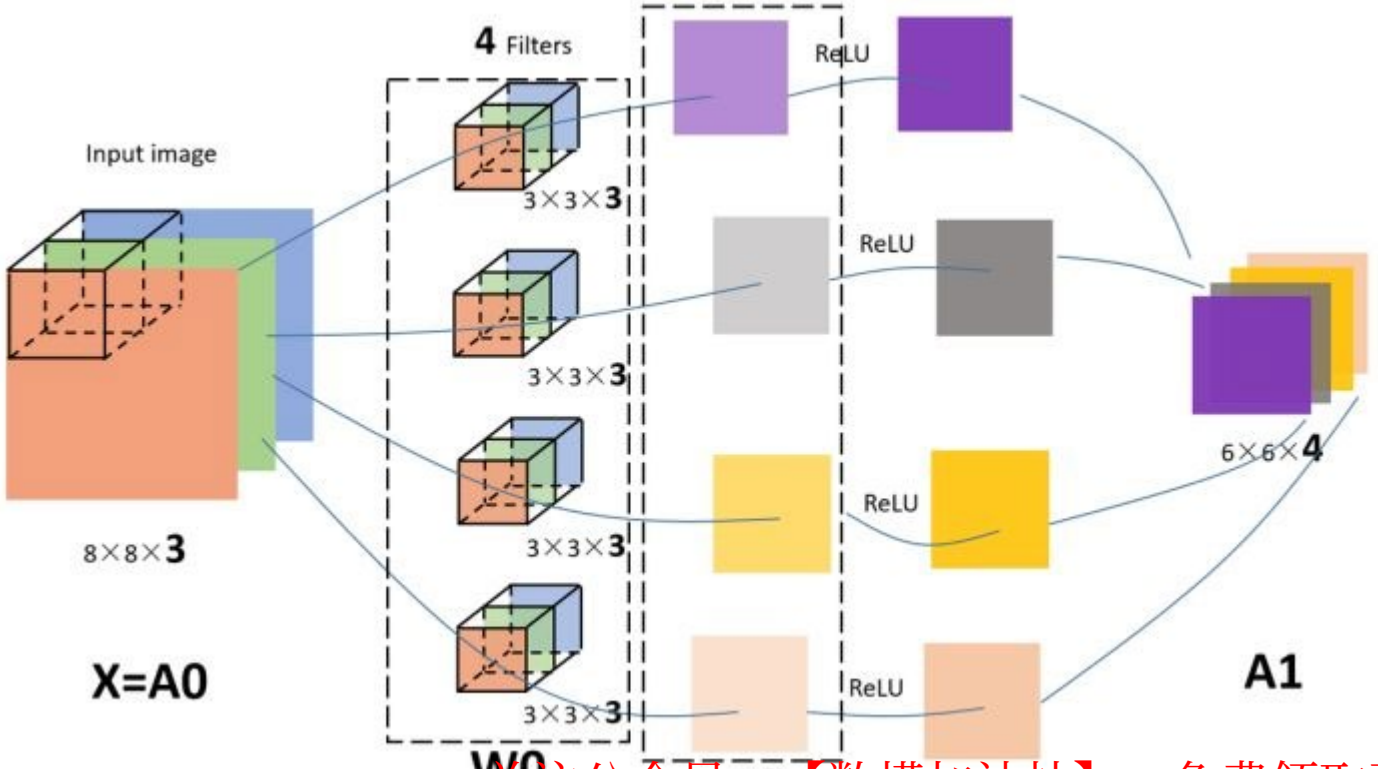
```
# 划分训练集和测试集
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42)

# 建立支持向量机模型
model = SVC(kernel='rbf', C=1.0, gamma='scale')  #
这里使用了径向基函数核

# 拟合模型
model.fit(X_train, y_train)

# 进行预测
y_pred = model.predict(X_test)
```



神经网络模型-CNN/

Teacher says lesson

原理：CNN通过使用卷积层、池化层和全连接层来提取图像中的特征，并将这些特征用于图像分类和识别。卷积层通过滤波器对图像进行卷积操作，提取出图像中的局部特征。池化层则用于减小特征图的尺寸，降低模型的复杂度。全连接层则将提取出的特征进行分类。

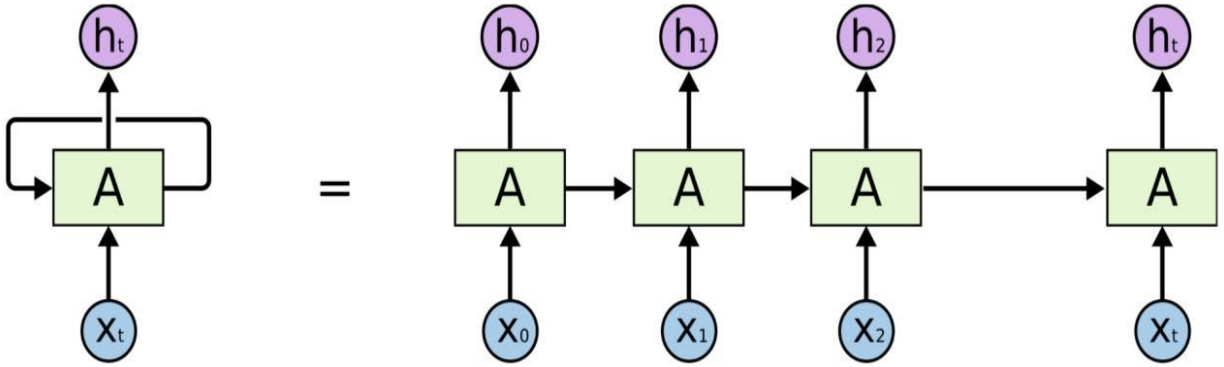



```
# 建立CNN模型
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3),
activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

# 编译模型
model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])

# 训练模型
model.fit(x_train, y_train, batch_size=128,
epochs=5, validation_data=(x_test, y_test))
```

神经网络模型-RNN/

Teacher says lesson

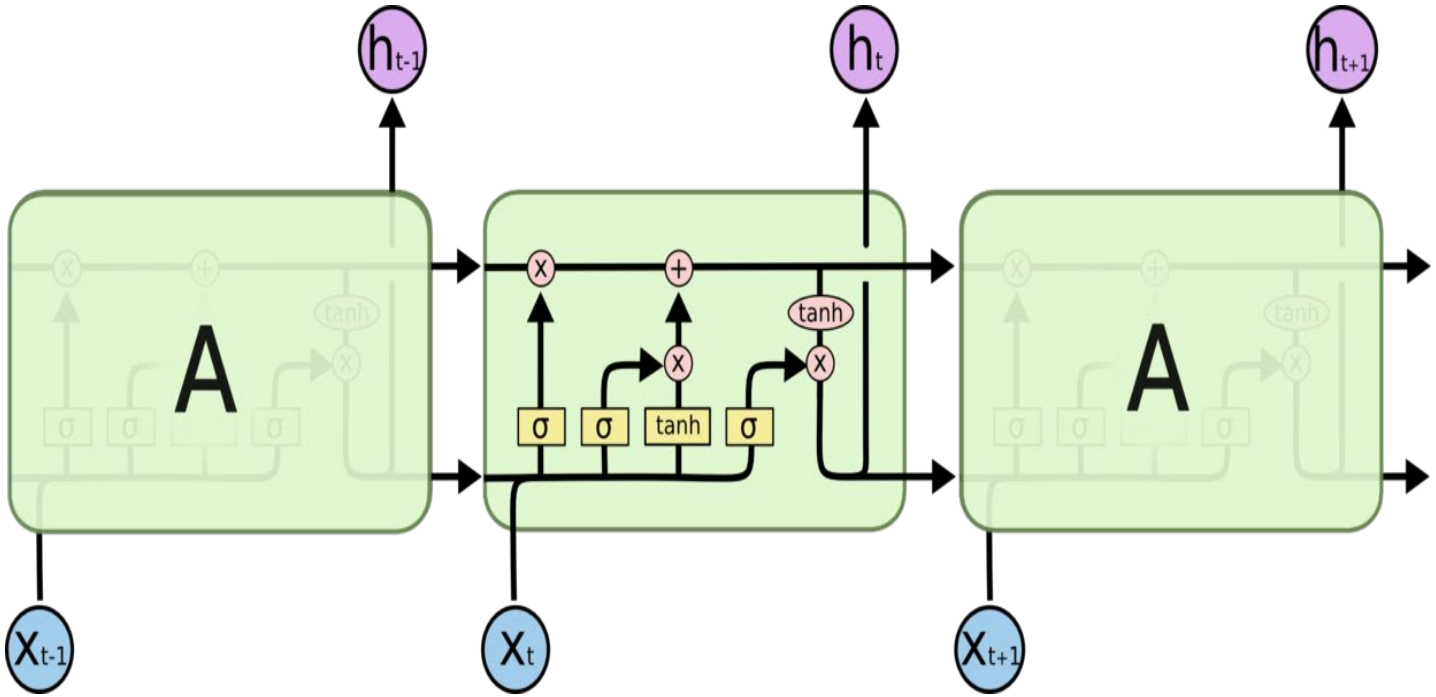
原理：RNN中的神经元通过循环的方式将上一个时间步的输出作为输入，从而使得网络能够保持对过去信息的记忆。




```
# 建立RNN模型
model = Sequential()
model.add(SimpleRNN(32, input_shape=(10, 5)))
model.add(Dense(1, activation='sigmoid'))

# 编译模型
model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])

# 训练模型
model.fit(x_train, y_train, batch_size=32,
epochs=10)
```

神经网络模型-LSTM/

原理：LSTM通过引入门控机制来解决RNN中的梯度消失和梯度爆炸问题，从而使得网络能够更好地捕捉长期依赖关系。LSTM包含遗忘门、输入门和输出门，以及一个内部细胞状态，这些门结构能够控制信息的流动，允许网络选择性地记住或遗忘过去的信息。



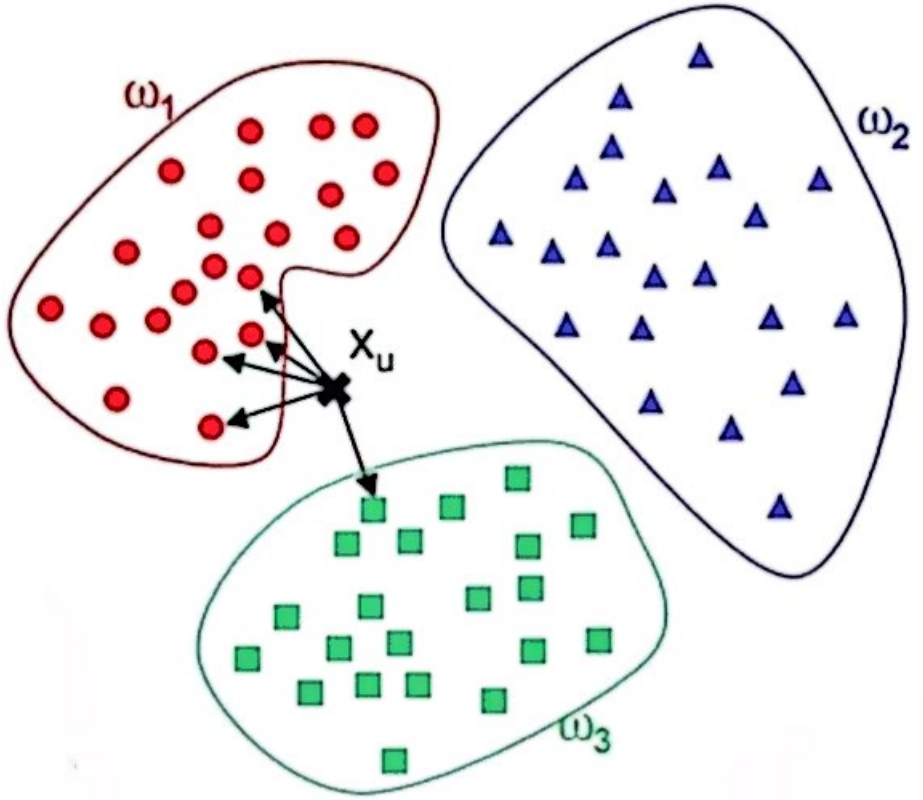

```
# 建立LSTM模型
model = Sequential()
model.add(LSTM(32, input_shape=(10, 5)))
model.add(Dense(1, activation='sigmoid'))

# 编译模型
model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])

# 训练模型
model.fit(x_train, y_train, batch_size=32,
epochs=10)
```

聚类模型-KNN/

Teacher says lesson

原理：KNN算法的核心思想是通过计算待预测样本与训练集中各个样本的距离，然后选取距离最近的K个样本，根据这K个样本的类别进行投票决定待预测样本的类别。

```
# 划分训练集和测试集
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42)

# 建立KNN模型
model = KNeighborsClassifier(n_neighbors=3)  # 这里
指定了K=3

# 拟合模型
model.fit(X_train, y_train)

# 进行预测
y_pred = model.predict(X_test)
```



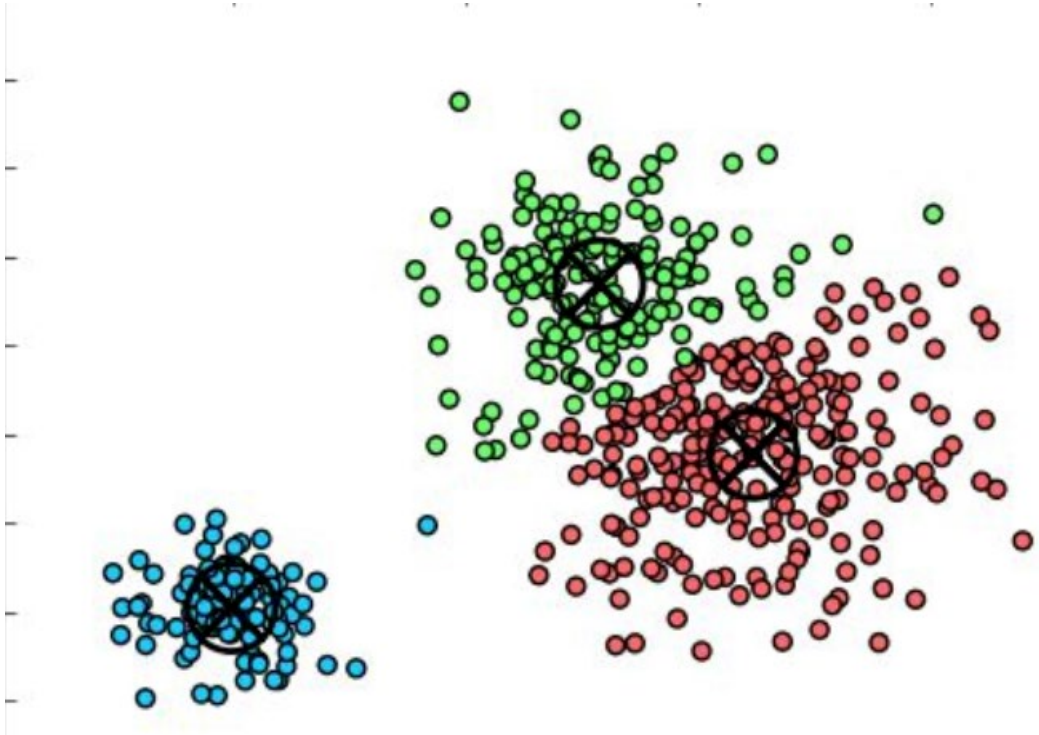
聚类模型-K-Means/

Teacher says lesson

原理：通过迭代的方式将数据点分配到K个簇中，使得每个数据点到其所属簇的中心点（质心）的距离最小化。

把 n 个点（可以是样本的一次观察或一个实例）划分到 k 个集群（cluster），使得每个点都属于离他最近的均值（即聚类中心，centroid）对应的集群。重复上述过程一直持续到重心不改变。

```
# 建立K均值模型
kmeans = KMeans(n_clusters=4)

# 拟合模型
kmeans.fit(X)

# 可视化结果
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_,
cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black',
s=200, alpha=0.5)
plt.show()
```

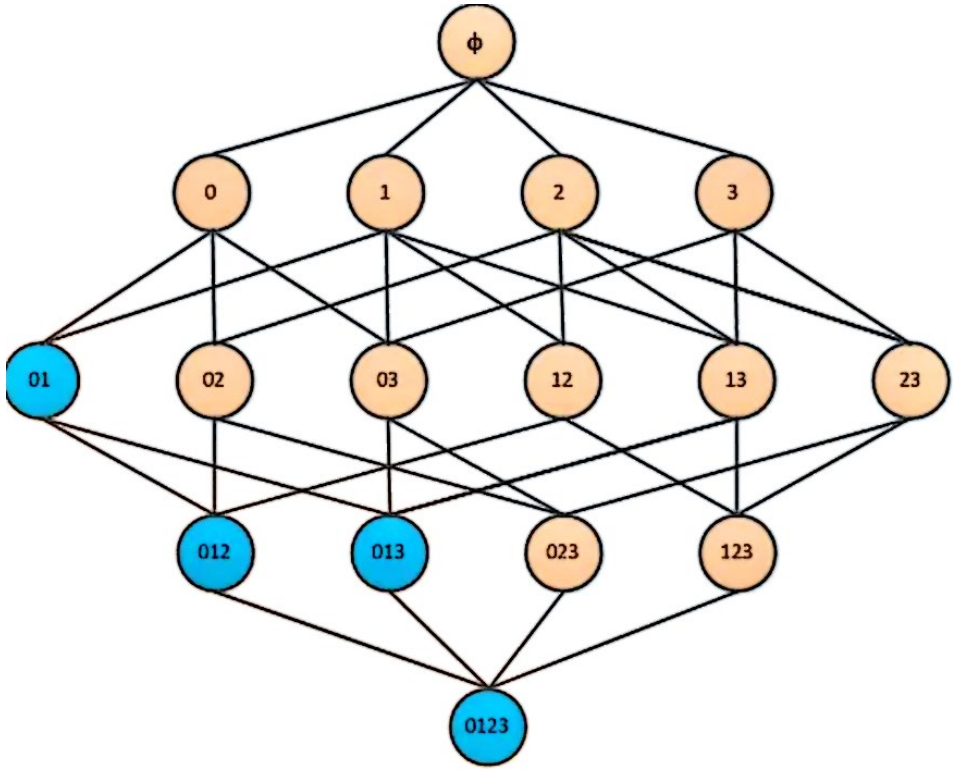


关联规则模型-Apriori/

Teacher says lesson

原理：Apriori算法的核心思想是基于频繁项集的性质来发现数据集中的频繁项集。该算法使用了一种叫做Apriori原理的性质，即如果一个项集是频繁的，那么它的所有子集也必定是频繁的。Apriori算法通过迭代的方式生成候选项集，并使用逐层扫描数据集来逐步发现频繁项集。

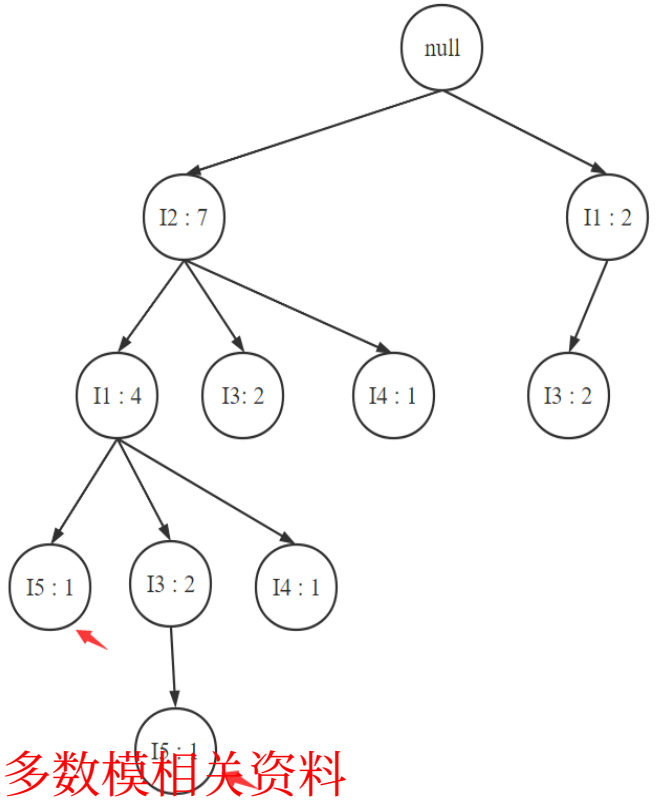
```
# 将数据集进行独热编码
df_encoded = pd.get_dummies(df.drop('ID', axis=1))

# 使用Apriori算法找出频繁项集
frequent_itemsets = apriori(df_encoded,
min_support=0.6, use_colnames=True)

# 找出关联规则
rules = association_rules(frequent_itemsets,
metric="confidence", min_threshold=0.7)
```



关联规则模型-FP-Growth/ Teacher says lesson

原理：FP-Growth算法通过构建FP树来高效地发现频繁项集。它包括两个主要步骤：首先构建FP树，然后通过FP树挖掘频繁项集。构建FP树的过程涉及压缩数据集，以减少内存消耗和加速挖掘过程。然后通过递归方式挖掘FP树，找出频繁项集。

```
# 将数据集进行独热编码
df_encoded = pd.get_dummies(df.drop('ID', axis=1))

# 使用FP-Growth算法找出频繁项集
frequent_itemsets = fpgrowth(df_encoded,
min_support=0.6, use_colnames=True)
```



模型使用流程

03

PART THREE

时序建模流程/

Teacher says lesson

1. 确定问题：首先要明确竞赛题目中需要解决的问题，明确问题的类型（分类、回归、聚类等），以及需要预测的变量或目标。
2. 数据理解：初步的探索性分析，包括了解数据的基本统计特征、缺失值处理、异常值处理等。
3. 特征工程：对原始数据进行特征提取、选择和转换，包括特征缩放、特征组合、特征选择、处理类别型特征等。
4. 模型选择：根据问题的类型和数据的特点，选择适当的机器学习模型，比如决策树、随机森林、支持向量机、神经网络等。
5. 模型训练：使用训练集对选定的模型进行训练，通过优化模型参数来使模型拟合训练数据。
6. 模型评估：使用测试集对训练好的模型进行评估，评估模型的性能指标，比如准确率、召回率、F1值、均方误差等。
7. 模型调优：根据评估结果对模型进行调优，可能包括调整超参数、交叉验证等。
8. 结果解释与报告：对模型的结果进行解释，分析模型对问题的贡献，撰写报告，呈现模型的应用效果和结论。

时序建模流程/

Teacher says lesson

历届题目分析

04

PART FOUR

2022 MCM Problem C: Trading Strategie/

Background

Market traders buy and sell volatile assets frequently, with a goal to maximize their total return. There is usually a commission for each purchase and sale. Two such assets are gold and bitcoin.

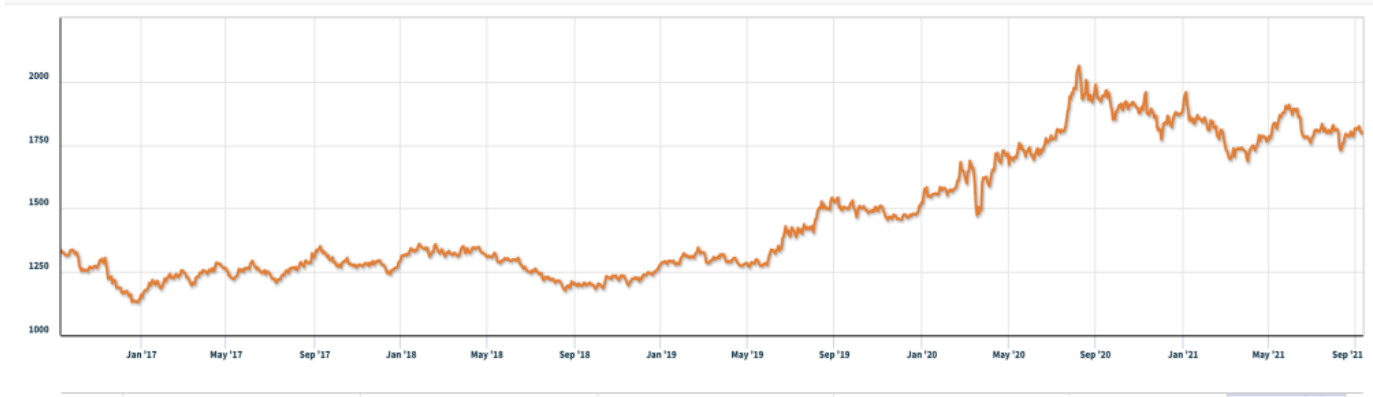


Figure 1: Gold daily prices, U.S. dollars per troy ounce. Source: London Bullion Market Association, 9/11/2021

Figure 2: Bitcoin daily prices, U.S. dollars per bitcoin. Source: NASDAQ, 9/11/2021

THANKS