一、选择题可能考点 (Python编程、NumPy、Pandas、机器学习基础)

Python编程基础：

1. 函数定义与调用：
    - `def` 关键字定义函数，`return` 关键字返回值。
    - 函数参数：位置参数、默认参数、关键字参数、可变参数 (`*args`, `**kwargs`)。
    - 函数调用时参数的匹配规则。
    - 示例：`def my_func(a, b=10): return a + b`，调用 `my_func(5)` 和 `my_func(5, 20)` 的结果。
2. 变量作用域：
    - 局部变量 (Local) vs. 全局变量 (Global)。
    - `global` 关键字的用法。
    - 函数内部定义的变量默认为局部变量。
    - 示例：在一个函数内部尝试修改全局变量（不使用 `global` 和使用 `global` 的区别）。
3. **Lambda** 表达式：
    - 匿名函数的定义：`lambda arguments: expression`。
    - 常用于简单的、一次性的函数。
    - 示例：`square = lambda x: x * x`。
4. 模块的概念与导入：
    - 模块：一个包含 Python 定义和语句的文件 (.py 文件)。
    - 导入模块：`import module_name`。
    - 从模块导入特定项：`from module_name import item_name`。
    - 导入并使用别名：`import module_name as alias_name`。
    - 包：包含模块的目录，必须有 `__init__.py` 文件。
    - 相对导入与绝对导入的概念（课件中有提及）。
    - 示例：`import math`，然后使用 `math.sqrt()`。
5. 库函数的引用：
    - 理解如何调用标准库或第三方库中的函数。

- 示例：`import random; random.randint(1, 10)`。

6. 基本数据类型与操作：
    - `print()` 函数的用法 (sep, end 参数)。
    - `input()` 函数的用法。
    - 字符串的 `.split()` 方法。
    - `type()` 函数判断数据类型。
    - `isinstance()` 函数判断对象是否为某个类的实例。
    - `int()`, `float()`, `str()`, `bool()` 类型转换函数。
    - 循环语句：`for`, `while` 的基本结构和用法。
    - 条件语句：`if`, `elif`, `else` 的基本结构和用法。
    - 循环控制：`break`, `continue`, `pass`。
    - `range()` 函数的用法。

NumPy：

1. 数组创建：
    - `np.array()`：从列表或元组创建数组。
    - `np.arange()`：创建等差序列数组。
    - `np.ones()`, `np.zeros()`, `np.eye()`：创建特殊数组。
    - 示例：`arr = np.array([1, 2, 3])`。
2. 数组属性：
    - `.shape`：数组维度。
    - `.dtype`：数据类型。
    - `.size`：元素总数。
    - `.T`：数组转置。
3. 数组索引与切片：
    - 一维数组索引和切片 (类似列表)。
    - 多维数组索引 (`arr[row, col]`, `arr[row][col]`) 和切片 (`arr[:2, 1:3]`)。
    - 布尔索引：`arr[arr > 5]`。
    - `np.where()` 函数的用法。
4. 数组运算：

- 逐元素算术运算 (+, -, *, /)。
- 数组与标量运算。
- `np.dot()`：点积/矩阵乘法。
- `np.sum()`, `np.mean()`, `np.std()`, `np.min()`, `np.max()` 等聚合函数及 `axis` 参数。

5. 数组形状修改：
    - `.reshape()` vs. `.resize()` vs. `np.resize()` 的区别。
6. 数组合并与分割：
    - `np.concatenate()`, `np.vstack()`, `np.hstack()`, `np.append()`。
    - `np.delete()`。

Pandas：

1. 数据结构：
    - `pd.Series()`：创建 Series (带标签的一维数组)。
    - `pd.DataFrame()`：创建 DataFrame (带标签的二维表格)。
    - Series 的索引 (`.index`) 和值 (`.values`)。
    - DataFrame 的行索引 (`.index`) 和列索引 (`.columns`)。
2. 数据读写：
    - `pd.read_csv()`, `pd.read_excel()`, `pd.read_table()`。
    - `.to_csv()`, `.to_excel()`。
    - `index=False` 参数在写入时的作用。
3. 数据选择与索引：
    - 选择列：`df['column_name']` (返回 Series), `df[['col1', 'col2']]` (返回 DataFrame)。
    - `.loc[]`：基于标签的索引和切片。
    - `.iloc[]`：基于整数位置的索引和切片。
    - 布尔索引：`df[df['column'] > value]`。
    - `.query()` 方法。
4. 数据处理与转换：
    - 处理缺失值：`.isnull()`, `.notnull()`, `.dropna()`, `.fillna()`。
    - `.astype()`：数据类型转换。

- `.apply()`：应用函数到行或列。
- `.replace()`：替换值。
- `.drop_duplicates()`：删除重复行。
- `.sort_values()`, `.sort_index()`：排序。

5. 分组与聚合 (`.groupby()`):
    - `df.groupby('column_name').agg_function()` (如 `.mean()`, `.sum()`, `.count()`)。
    - `.agg()` 方法：应用多个聚合函数，对不同列应用不同函数，使用自定义函数，结果重命名。
6. 数据合并与连接：
    - `pd.merge()`：数据库风格的连接 (on, how, left_on, right_on)。
    - `.join()`：基于索引的连接。
    - `pd.concat()`：沿轴连接 (axis, join)。
7. 数据重塑：
    - `.pivot()`：长表转宽表 (唯一性要求)。
    - `.pivot_table()`：类似数据透视表 (可聚合)。
    - `.melt()`：宽表转长表。
    - `.stack()`, `.unstack()`：索引和列的层级转换。
8. 字符串处理 (`.str` 访问器):
    - `.str.split()`, `.str.contains()`, `.str.replace()`, `.str.extract()` 等。

机器学习基础：

1. 回归问题 vs. 分类问题：
    - 回归：预测连续值 (如房价、销售额)。
    - 分类：预测离散类别 (如是否幸存、邮件是否垃圾)。
2. 线性回归：
    - 基本原理：拟合一条直线 ($y = \beta_0 + \beta_1 x + \epsilon$) 来描述变量间关系。
    - 最小二乘法：通过最小化残差平方和来估计参数。
    - `sklearn.linear_model.LinearRegression` 类的基本使用：`fit()`, `predict()`, `coef_`, `intercept_`。
3. 逻辑回归：

- 用于解决二分类问题。
- 将线性回归的输出通过 Sigmoid 函数映射到 (0,1) 区间，表示概率。
- `sklearn.linear_model.LogisticRegression` 类的基本使用 (课件中“数据分析”部分有)。

4. **聚类问题：**
   - 无监督学习，将数据点分组成相似的簇。
   - K-Means 算法原理：选择 K 个初始质心，迭代分配数据点到最近质心并更新质心。
   - `sklearn.cluster.KMeans` 类的基本使用：`n_clusters`, `init`, `fit()`, `labels_`, `cluster_centers_`, `inertia_`。
   - 确定 K 值的方法：肘部法 (SSE)，轮廓系数法。
5. **模型评估：**
   - 回归模型：MSE (均方误差), RMSE (均方根误差), MAE (平均绝对误差), R^2 (决定系数)。
   - 分类模型 (逻辑回归)：准确率 (Accuracy), 精确率 (Precision), 召回率 (Recall), F1-score, ROC曲线, AUC值。
   - 聚类模型：轮廓系数 (Silhouette Score), DB指数 (DBI), Dunn指数 (DI)。
6. **数据预处理：**
   - `sklearn.model_selection.train_test_split()`：划分训练集和测试集。
   - `sklearn.preprocessing.StandardScaler()`：特征标准化。
   - `sklearn.preprocessing.MinMaxScaler()`：特征归一化。
   - `sklearn.preprocessing.LabelEncoder()`：类别标签编码。

二、简答题可能考点

1. **数据可视化：**
   - **什么是数据可视化？** (将数据转换成图或表等视觉形式，以更直观的方式展现和呈现数据，帮助理解、分析和沟通信息。)
   - **数据可视化的重要性？** (揭示数据模式和趋势、简化复杂数据、辅助决策、增强沟通效果、提高数据分析效率。)
   - **常用的统计图类型及其适用场景？** (折线图-趋势，柱状图/条形图-比较，饼图-占比，散点图-关系/分布，直方图-频率分布，箱线图-数据分布/异常值，热力图-矩阵数据。)

- **Matplotlib 和 Seaborn 的关系和区别？** (Matplotlib 是基础绘图库，提供底层API，灵活但代码可能较多；Seaborn 基于 Matplotlib，提供更高级的统计图形接口，默认样式更美观，API 更简洁。)

2. **数据分析流程：**
    - **典型的数据分析/机器学习项目流程是什么？** (课件“数据分析” P4-P5)
        i. **问题定义** (明确目标、确定评估指标)
        ii. **数据收集与理解** (数据来源、数据探索 EDA - 分布、可视化、缺失/异常值检测)
        iii. **数据预处理** (清洗 - 缺失/异常值处理；转换 - 标准化/归一化/编码；特征工程 - 构造/选择/降维)
        iv. **模型选择与训练** (划分数据集、选择模型、训练与验证、超参数调优)
        v. **模型评估** (使用合适的评估指标评估模型性能)
        vi. **(可选) 模型部署与监控**

3. **维度灾难 (Curse of Dimensionality)：**
    - **什么是维度灾难？** (当数据的维度（特征数量）非常高时，会出现一系列问题，如数据稀疏性、计算复杂度增加、模型性能下降、过拟合风险增加等。)
    - **维度灾难的影响有哪些？** (数据点在空间中变得非常稀疏，使得距离度量失去意义；需要更多的数据来覆盖高维空间；计算成本显著增加；更容易过拟合。)

4. **降维方法：**
    - **为什么要进行降维？** (解决维度灾难、去除冗余特征、降低计算复杂度、提高模型性能、便于数据可视化。)
    - **常见的降维方法有哪些？**
        - **特征选择 (Feature Selection):** 过滤法 (如卡方检验、方差选择)、包裹法 (如递归特征消除 RFE)、嵌入法 (如L1正则化)。(课件“数据分析”P4 提及)
        - **特征提取 (Feature Extraction):** 主成分分析 (PCA)、线性判别分析 (LDA)。(课件“数据分析”P4 提及 PCA, LDA)

5. **大数据特征 (4V 或 5V)：**
    - **请简述大数据的 4V (或 5V) 特征。**
        - **Volume (大量):** 数据规模巨大。
        - **Velocity (高速):** 数据产生和处理速度快。
        - **Variety (多样):** 数据类型和来源多样化 (结构化、半结构化、非结构化)。

    - **Value (价值):** 数据具有潜在的巨大价值，但价值密度可能较低。
    - **(可选) Veracity (真实性):** 数据的准确性和可信度。

6. **典型数据结构 (Python 中的列表 List, 字典 Dict)：**
    - **列表 (List) 的特点和常用操作？**
        - 特点：有序、可变序列，可以包含不同类型的元素。
        - 常用操作：索引、切片、添加元素 (`append`, `insert`, `extend`)、删除元素 (`remove`, `pop`, `del`)、查找 (`in`, `index`, `count`)、排序 (`sort`)、反转 (`reverse`)、长度 (`len`)。
    - **字典 (Dict) 的特点和常用操作？**
        - 特点：无序 (Python 3.7+ 为有序)、可变键值对集合，键必须唯一且不可变。
        - 常用操作：通过键访问值 (`dict[key]`, `.get(key)`)、添加/修改键值对、删除键值对 (`del dict[key]`, `.pop(key)`)、获取键/值/项 (`.keys()`, `.values()`, `.items()`)、检查键是否存在 (`in`)、长度 (`len`)。

7. **拟合 (Fitting)、过拟合 (Overfitting)、欠拟合 (Underfitting)：**
    - **什么是模型拟合？** (模型学习训练数据的过程，试图找到数据中的模式和关系。)
    - **什么是欠拟合？如何判断和解决？**
        - 欠拟合：模型在训练集和测试集上表现都很差，未能捕捉到数据的基本模式。
        - 判断：训练误差和验证/测试误差都很高。学习曲线中训练和验证误差都较高且收敛到一起。
        - 解决：增加模型复杂度 (如使用更复杂的模型、增加特征、减少正则化)、获取更多特征、减少正则化强度、训练更长时间。
    - **什么是过拟合？如何判断和解决？**
        - 过拟合：模型在训练集上表现很好，但在未见过的数据（测试集）上表现很差，学习了训练数据中的噪声和特有模式。
        - 判断：训练误差很低，但验证/测试误差很高。学习曲线中训练误差低而验证误差高，两者差距大。
        - 解决：获取更多数据、降低模型复杂度 (如使用更简单的模型、特征选择、增加正则化)、交叉验证、早停法 (Early Stopping)。
    - **如何通过学习曲线和验证曲线判断拟合情况？** (线性回归课件 P44, P46)

- 学习曲线：展示训练误差和验证误差随训练样本数量变化的趋势。
- 验证曲线：展示训练误差和验证误差随模型某个超参数变化的趋势。

三、程序设计题可能考点 (填空为主)

Python 基础填空：

1. 函数定义与参数：
    - 补全函数定义中的参数或 `return` 语句。
    - 示例：`def calculate_area(length, ____): return length * width` (填 `width`)
2. 循环与条件：
    - 补全 `for` 循环的 `range()` 部分或循环体。
    - 补全 `if/elif/else` 的条件判断。
    - 示例：`for i in range(1, ____, 2): print(i)` (填 `10` 表示1到9的奇数)
3. 列表/字典操作：
    - 补全列表的添加/删除/访问元素的代码。
    - 补全字典的键值对操作。
    - 示例：`my_list = [1, 2]; my_list.____(3)` (填 `append`)
    - 示例：`my_dict = {'a':1}; my_dict['b'] = ____` (填 `2`)
4. 模块导入：
    - 补全导入语句。
    - 示例：`____ math` (填 `import`) 或 `from math import ____` (填 `sqrt`)

NumPy 填空：

1. 数组创建：
    - `arr = np.____([1,2,3])` (填 `array`)
    - `zeros_arr = np.zeros((2,____))` (填 `3` 表示2行3列)
2. 数组属性与方法：
    - `print(arr.____)` (填 `shape` 或 `dtype` 或 `size`)
    - `reshaped_arr = arr.reshape((____, 2))`

3. 索引与切片：
    - `element = arr[1, ____]`
    - `sub_array = arr[____, 1:3]`
    - `filtered_arr = arr[arr ____ 0]` (填 `>` 或 `<`)
4. 常用函数：
    - `mean_val = np.____(arr, axis=0)` (填 `mean`)
    - `sorted_arr = np.sort(arr, axis=____)` (填 `1`)

Pandas 填空：

1. **Series/DataFrame** 创建：
    - `s = pd.Series([1,2,3], index=['a',____,'c'])` (填 `'b'`)
    - `df = pd.DataFrame({'col1': [1,2], 'col2': [3,____]})` (填 `4`)
2. 数据读取：
    - `df = pd.read_csv('data.csv', index_col=____)` (填 `0` 或列名)
3. 数据选择：
    - `col_data = df[____]` (填 `'column_name'`)
    - `row_data = df.loc[____, 'column_name']` (填行标签)
    - `subset = df.iloc[0:2, ____]` (填列的整数位置)
4. 常用方法：
    - `missing_values = df.isnull().____()` (填 `sum`)
    - `df_grouped = df.groupby('category')[____].mean()` (填 `'value_column'`)
    - `merged_df = pd.merge(df1, df2, on='key_column', how=____)` (填 `'left'` 或 `'inner'`)
    - `df['new_col'] = df['old_col'].apply(lambda x: x * ____)` (填 `2`)

机器学习 **(sklearn)** 填空：

1. 模型导入与实例化：
    - `from sklearn.linear_model import ____` (填 `LinearRegression` 或 `LogisticRegression`)

- `model = LinearRegression(____)` (可能为空，或填参数如 `fit_intercept=False`)
- `from sklearn.cluster import ____` (填 `KMeans`)
- `kmeans = KMeans(n_clusters=____, random_state=0)` (填 `3`)

2. 模型训练与预测：
   - `model.____(X_train, y_train)` (填 `fit`)
   - `y_pred = model.____(X_test)` (填 `predict`)
3. 数据划分与预处理：
   - `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=____, random_state=42)` (填 `0.2` 或 `0.3`)
   - `scaler = StandardScaler(); X_scaled = scaler.fit_transform(____)` (填 `X_train`)
4. 模型评估：
   - `mse = mean_squared_error(y_test, ____)` (填 `y_pred`)
   - `accuracy = ____(y_test, y_pred_class)` (填 `accuracy_score` - 逻辑回归中)
   - `silhouette = silhouette_score(X_features, kmeans._____)` (填 `labels_`)

四、综合分析题：泰坦尼克数据分析 (结合课件)

这部分需要你详细回顾 "数据分析" 课件中 P6-P52 的泰坦尼克号案例分析流程。分析题通常会给出一个新的、类似的数据集或场景，要求你模仿泰坦尼克号的分析步骤进行。

核心分析步骤 (需要结合课件中的代码和图表进行阐述)：

1. 问题定义与理解：
   - 明确分析目标：泰坦尼克号案例是预测乘客是否幸存 (二分类问题)。
   - 理解数据：变量含义 (课件 P8 详细列出)。
2. 数据加载与初步探索 (EDA)：
   - 导入必要的库 (课件 P9)：`pandas`, `numpy`, `matplotlib`, `seaborn`, `sklearn` 中的相关模块。
   - 加载数据集 (课件 P10)：`pd.read_csv()`。
   - 查看数据基本信息 (课件 P11-P13)：
     - `.head()`, `.tail()`：查看首尾数据。

- `.info()`：查看列名、非空值数量、数据类型、内存占用。
- `.describe()`：查看数值型特征的描述性统计 (均值、标准差、分位数等)。
- `.isnull().sum()`：查看各列缺失值数量。
- `.columns.values`：获取所有特征名。

3. 数据可视化与假设分析 **(EDA 核心)**：
    - 分析单个特征与目标变量 **(Survived)** 的关系：
        - **Pclass (船舱等级)** (课件 P15)：使用 `.groupby()` 计算不同等级的平均存活率，并进行可视化 (如条形图)。得出 Pclass=1 存活率高。
        - **Sex (性别)** (课件 P16)：同上，计算不同性别的平均存活率。得出女性存活率远高于男性。
        - **SibSp (兄弟姐妹/配偶数) & Parch (父母/子女数)** (课件 P17)：计算不同数量下的平均存活率。发现单独看可能不是强相关，引出特征工程的思路 (如合并为 FamilySize)。
        - **Age (年龄)** (课件 P18)：
            - 使用 `sns.FacetGrid()` 结合 `plt.hist()` 绘制不同存活情况下年龄的直方图。
            - 分析结论：婴儿存活率高，老年人也可能幸存，15-25 岁死亡率高。需要处理缺失值，可能需要分箱。
        - **Embarked (登船港口)** (课件 P20)：
            - 使用 `sns.FacetGrid()` 结合 `sns.pointplot` (或 `sns.barplot`) 可视化不同港口、不同性别下的存活率与票价的关系。
            - 结论：不同港口存活率有差异，可能与 Pclass 和 Fare 有关。需要处理缺失值。
        - **Fare (票价)** (课件 P20-P21, 与 Embarked 和 Pclass 结合分析)：
            - 高票价乘客存活率更高。
    - 分析特征之间的关系：
        - **Age 与 Pclass** (课件 P19)：使用 `sns.FacetGrid()` 绘制不同 Pclass 下年龄的直方图 (按是否存活进行颜色区分)。结论：Pclass=1 中老年乘客多，Pclass=3 中年轻乘客多。
    - 基于分析提出假设 (课件 P14)：哪些特征可能重要，哪些可能需要处理或丢弃。

4. 数据清洗与特征工程：

- 处理缺失值：
    - **Age** (课件 P29-P30)：
        - 方法：使用相关特征 (Sex, Pclass) 的中位数/均值来填充。课件中展示了基于 Sex 和 Pclass 组合的中位数填充。
        - 具体实现：循环遍历 Sex 和 Pclass 的组合，计算对应分组的 Age 中位数，然后用该中位数填充对应分组的 Age 缺失值。最后用全局中位数填充剩余无法匹配的。
    - **Embarked** (课件 P36)：使用众数填充。`train_df.Embarked.dropna().mode()[0]`。
    - **Fare** (测试集中的缺失值) (课件 P38)：使用中位数填充。`test_df['Fare'].fillna(test_df['Fare'].dropna().median(), inplace=True)`。
- 特征转换/创建：
    - **Title (称谓)** (课件 P23-P26)：
        - 从 Name 中提取 Title (如 Mr, Miss, Mrs)。使用正则表达式 `str.extract('([A-Za-z]+)\.')`。
        - 合并稀有 Title 为 'Rare'。
        - 将 Title 映射为数值。`title_mapping = {"Mr": 1, ...}`。
    - **Sex (性别)** (课件 P28)：将 'female' 转为 1, 'male' 转为 0。`.map({'female': 1, 'male': 0}).astype(int)`。
    - **AgeBand (年龄分箱)** (课件 P31-P32)：
        - 使用 `pd.cut()` 将连续的 Age 分为几个区间。
        - 将这些区间映射为序数 (0, 1, 2, 3, 4)。
    - **FamilySize (家庭成员数量)** (课件 P33)：`SibSp + Parch + 1`。
    - **IsAlone (是否独自一人)** (课件 P34)：根据 FamilySize 创建，如果 FamilySize=1 则 IsAlone=1，否则为0。
    - **Embarked (登船港口)** (课件 P37)：将 S, C, Q 映射为数值 0, 1, 2。
    - **FareBand (票价分箱)** (课件 P39-P40)：
        - 使用 `pd.qcut()` (等频分箱) 将 Fare 分为几个区间。
        - 将这些区间映射为序数。
- 删除无用特征：

- **Ticket, Cabin** (课件 P22)：缺失值过多或格式不统一。
- **Name, PassengerId** (课件 P27)：在提取 Title 后，Name 不再需要；PassengerId 通常对预测无用。
- **Parch, SibSp, FamilySize** (课件 P35)：在创建 IsAlone 后，这些可以被替代。

5. **模型选择与训练 (课件 P41-P51)：**
    - **准备数据：**
        - `X_train = train_df.drop("Survived", axis=1)`
        - `Y_train = train_df["Survived"]`
        - `X_test = test_df.drop("PassengerId", axis=1).copy()` (注意测试集没有 Survived 列)
    - **选择并训练多种模型：** 课件中尝试了逻辑回归、支持向量机 (SVC)、K 近邻 (KNN)、朴素贝叶斯、决策树、随机森林、感知机、线性SVC、SGD 分类器。
    - 对每个模型：
        - 实例化模型：`model = ModelClass()`
        - 训练模型：`model.fit(X_train, Y_train)`
        - 进行预测 (主要是在训练集上评估准确率)：`Y_pred_train = model.predict(X_train)` (课件中是直接用 `.score()` 评估训练集)
        - 计算训练集准确率：`acc = round(model.score(X_train, Y_train) * 100, 2)`

6. **模型评估与选择 (课件 P52)：**
    - 创建一个 DataFrame 汇总所有模型的训练集准确率。
    - `models.sort_values(by='Score', ascending=False)`
    - 根据得分选择最终模型 (课件中选择了随机森林，因为它能较好处理过拟合问题，尽管与决策树得分相同)。
    - **注意：** 实际项目中，更重要的是在**测试集**或通过**交叉验证**来评估模型，以判断其泛化能力。课件中主要展示了在训练集上的得分作为比较。

7. **(可选但重要) 生成提交文件：**
    - 使用选定的模型对 `X_test` 进行预测：`Y_pred_test = random_forest.predict(X_test)`
    - 创建包含 `PassengerId` 和预测的 `Survived` 的 DataFrame。
    - 保存为 CSV 文件提交。 (这部分课件未详细展示，但在 Kaggle 比赛中是必需的)

给一个案例做分析 (需要你自己根据新的数据集模仿上述步骤)

如果题目给出一个新的数据集 (例如，一个客户流失预测的数据集，包含客户年龄、性别、入网时长、消费金额、是否流失等字段)，你需要：

1. 理解问题： 目标是预测客户是否流失 (二分类)。
2. 加载数据并探索： 使用 Pandas 加载，查看 `.info()`, `.describe()`, 缺失值，数据类型。
3. 数据可视化：
    - 分析各特征 (年龄、消费、时长等) 与“是否流失”的关系 (使用直方图、条形图、箱线图)。
    - 分析特征间的关系 (如消费与时长的散点图)。
4. 数据清洗与特征工程：
    - 处理缺失值 (根据情况选择填充或删除)。
    - 转换类别特征为数值 (如 LabelEncoding 或 One-Hot Encoding)。
    - 创建新特征 (如果适用，如平均月消费 = 总消费 / 时长)。
    - 特征缩放 (如 StandardScaler)。
5. 模型选择与训练：
    - 划分训练集和测试集。
    - 尝试多种分类模型 (如逻辑回归、SVM、随机森林、决策树)。
    - 训练模型。
6. 模型评估：
    - 使用测试集评估模型 (准确率、精确率、召回率、F1-score, AUC)。
    - 进行比较，选择最优模型。
7. 结果解释与结论： 解释模型结果，哪些特征对流失影响大，模型的预测能力如何。

通用分析框架

对于任何数据分析和机器学习任务，我们通常会遵循一个相似的流程，这在您的 "数据分析" 课件 (P4-P5) 中也有体现：

1. 问题定义 **(Problem Definition):**
    - 明确任务目标：是回归、分类（二分类/多分类）、聚类还是推荐等。
    - 确定评估指标：根据任务类型选择合适的衡量标准。

2. 数据收集与理解 **(Data Acquisition and Understanding):**

- 加载数据：使用 `pandas` 的 `read_csv()` 等函数。
- 初步探索 (EDA - Exploratory Data Analysis)：
    - 查看数据基本信息：`.info()`, `.head()`, `.tail()`, `.shape`, `.describe()`。
    - 检查数据类型：`.dtypes`。
    - 识别缺失值：`.isnull().sum()`。
    - 可视化探索：使用 `matplotlib` 和 `seaborn` 绘制直方图、散点图、箱线图等，理解数据分布、特征间关系、特征与目标变量的关系。

3. 数据预处理 **(Data Preprocessing):**

- **数据清洗：**
    - 处理缺失值：删除 (`dropna()`) 或填充 (`fillna()`，可使用均值、中位数、众数或模型预测)。
    - 处理异常值：检测（如箱线图）并处理（如截断、替换）。
- **数据转换：**
    - 类别特征编码：
        - 标签编码 (Label Encoding)：`sklearn.preprocessing.LabelEncoder` (用于序数特征或树模型)。
        - 独热编码 (One-Hot Encoding)：`pd.get_dummies()` (用于名义特征)。
    - 数值特征缩放：
        - 标准化 (Standardization)：`sklearn.preprocessing.StandardScaler` (均值为0，方差为1)。
        - 归一化 (Normalization)：`sklearn.preprocessing.MinMaxScaler` (缩放到或指定范围)。

4. 特征工程 **(Feature Engineering):**

- 特征构建：根据业务理解或数据洞察创建新的、更有意义的特征（如泰坦尼克案例中的 `FamilySize`, `IsAlone`, `Title`）。
- 特征选择：选择与目标变量最相关的特征，剔除冗余或不重要的特征（方法如过滤法、包裹法、嵌入法）。
- 降维：处理维度灾难，减少特征数量，保留主要信息 (如 PCA)。

5. **模型选择与训练 (Model Selection and Training):**
    - 划分数据集：使用 `sklearn.model_selection.train_test_split()` 将数据分为训练集和测试集。
    - 选择合适的模型：根据问题类型（回归、分类、聚类）选择一个或多个候选模型。
    - 训练模型：使用训练集的特征 (`X_train`) 和目标 (`y_train`) 调用模型的 `.fit()` 方法。
    - （可选）超参数调优：如网格搜索 (Grid Search)、随机搜索 (Random Search)。
6. **模型评估 (Model Evaluation):**
    - 使用测试集的特征 (`X_test`) 进行预测：`model.predict()`。
    - 使用选定的评估指标比较预测结果 (`y_pred`) 和真实目标 (`y_test`)。
    - 交叉验证 (K-fold Cross-Validation)：更稳健地评估模型性能 (`sklearn.model_selection.cross_val_score()`)。
    - 学习曲线 (Learning Curve) 和验证曲线 (Validation Curve)：判断过拟合或欠拟合。
7. **(可选) 模型部署与迭代 (Model Deployment and Iteration):** 将模型应用于实际场景，并根据反馈持续优化。

针对加利福尼亚房价预测 (程序填空)

这是一个典型的**回归问题**，目标是预测房价（连续值）。

程序填空可能涉及的知识点（基于您的课件）：

1. **加载数据:** (线性回归课件 P22)
    - `from sklearn.datasets import fetch_california_housing`
    - `housing = fetch_california_housing()`
    - `X = pd.DataFrame(housing.data, columns=housing.feature_names)`
    - `y = pd.Series(housing.target)`
2. **数据分割:** (线性回归课件 P17, P24)
    - `from sklearn.model_selection import train_test_split`
    - `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=____, random_state=42)` (填 `0.2` 或其他比例)

3. **特征缩放:** (线性回归课件 P19, P23)
    - `from sklearn.preprocessing import StandardScaler`
    - `scaler = StandardScaler()`
    - `X_train_scaled = scaler.fit_transform(____)` (填 `X_train`)
    - `X_test_scaled = scaler.transform(____)` (填 `X_test`)
4. **模型训练:** (线性回归课件 P15, P24)
    - `from sklearn.linear_model import LinearRegression`
    - `model = LinearRegression()`
    - `model.____(X_train_scaled, y_train)` (填 `fit`)
5. **模型预测:** (线性回归课件 P16, P25)
    - `y_pred = model.____(X_test_scaled)` (填 `predict`)
6. **模型评估:** (线性回归课件 P25, P28-P32)
    - `from sklearn.metrics import mean_squared_error, r2_score`
    - `mse = mean_squared_error(____, y_pred)` (填 `y_test`)
    - `r2 = r2_score(y_test, ____)` (填 `y_pred`)
    - `print(f"Intercept: {model.____}")` (填 `intercept_`)
    - `print(f"Coefficients: {model.____}")` (填 `coef_`)

【任务1】电影推荐系统 (MovieLens Dataset)

1. 问题定义:
* 任务：预测用户对未看过电影的评分，并基于预测评分推荐电影。
* 问题类型：这可以看作是一个**回归问题**（预测具体评分值），也是一个**推荐问题**。

2. 数据集理解与探索 (MovieLens 20M):
* `ratings.csv`: `userId`, `movieId`, `rating` (0.5-5), `timestamp`.
* `movies.csv`: `movieId`, `title`, `genres`.
* `tags.csv`: `userId`, `movieId`, `tag`, `timestamp`.
* **探索方向 (结合Pandas和可视化):**
    * 评分分布 (电影平均分、用户平均打分)。
    * 电影流行度 (被评分次数)。
    * 用户活跃度 (评分次数)。
    * 不同类型电影的评分情况。
    * 数据稀疏性：用户-电影评分矩阵会非常稀疏。

3. 数据预处理:

* 加载数据：`pd.read_csv()`。
* 合并数据：将 `ratings.csv` 与 `movies.csv` 基于 `movieId` 合并，方便获取电影信息。
* 处理缺失值 (如果存在于 `genres` 等字段)。
* 时间戳转换 (如果需要基于时间的分析)。
* 类型转换：`genres` 列通常是 `|` 分隔的字符串，可能需要拆分成列表或进行独热编码。

4. 特征工程:

* **用户特征：** 用户平均评分、评分数量、偏好的电影类型 (可从已评分电影的类型统计)。
* **电影特征：** 电影平均评分、被评分数量、电影类型。
* **（高级）用户-物品交互特征：** 如果使用矩阵分解等模型，会学习用户和电影的隐向量。

5. 模型选择与训练:

* **基于协同过滤 (Collaborative Filtering):** (搜索：协同过滤算法)
    * **基于用户的 (User-Based CF):** 找到与目标用户评分模式相似的用户群，推荐这些相似用户喜欢且目标用户未看过的电影。
    * **基于物品的 (Item-Based CF):** 计算电影之间的相似度，推荐与用户喜欢的电影相似的电影。
    * **矩阵分解 (Matrix Factorization):** 如 SVD (奇异值分解)。将用户-电影评分矩阵分解为用户隐向量矩阵和电影隐向量矩阵，通过这两个矩阵的点积来预测评分。(Sklearn 中的 `TruncatedSVD` 可以用于降维，但专门的推荐库如 `Surprise` 更常用于 SVD 实现)。
* **基于内容的推荐 (Content-Based Filtering):** (搜索：内容推荐算法)
    * 根据电影的内容特征 (如类型、标签、描述) 和用户的历史偏好进行推荐。
* **混合推荐 (Hybrid Approaches):** 结合多种推荐策略。
* **将评分预测视为回归问题 (结合课件线性回归部分):**
    * **特征：** 可以是用户ID的独热编码、电影ID的独热编码 (对于大规模数据不现实)、用户平均分、电影平均分、电影类型 (独热编码后) 等。
    * **模型：** 线性回归 (`LinearRegression`) 可能过于简单，因为评分与特征间的关系可能非线性。更复杂的回归模型如梯度提升回归、随机森林回归可能效果更好 (但课件主要讲线性回归)。
    * **实现思路：** 构建包含用户特征、电影特征和对应评分的训练集。

6. 模型评估:

* **评分预测准确度 (回归指标):**
    * RMSE (均方根误差)：`np.sqrt(mean_squared_error(y_true, y_pred))` (课件中 MSE 有提及)。
    * MAE (平均绝对误差)：`mean_absolute_error(y_true, y_pred)`。
* **推荐列表评估 (排序指标):** (搜索：推荐系统评估指标 Precision@k, Recall@k)

    * Precision@k, Recall@k, F1-score@k: 推荐列表中前k个项目中有多少是用户真正喜欢的。
    * MAP (Mean Average Precision), nDCG (Normalized Discounted Cumulative Gain)。

7. 推荐生成:
* 对目标用户未看过的所有电影进行评分预测。
* 将预测评分从高到低排序，推荐 Top-N 部电影。

8. 潜在挑战:
* **数据稀疏性：** 用户通常只评价了很少一部分电影。
* **冷启动问题：** 新用户（没有评分历史）或新电影（没有被评分过）难以推荐。
* **可扩展性：** 数据量大 (20M 条评分)，需要高效的算法和实现。
* **评估：** 离线评估指标与在线用户满意度可能不完全一致。

【任务2】公司贷款违约预测

1. 问题定义:
* 任务：预测公司是否会违约。
* 问题类型：**二分类问题** (违约/不违约)。

2. 数据集理解与探索:
* 数据集包含公司财务数据、贷款信息等。
* **特征可能包括：** (需要查看数据集具体字段)
    * 公司基本信息：成立年限、行业、规模。
    * 财务指标：流动比率、速动比率、资产负债率、利润率、现金流等。
    * 贷款信息：贷款金额、期限、利率、贷款类型。
    * 历史信用记录：是否有逾期、破产历史等。
* **目标变量：** `Status` (0表示未违约, 1表示违约)。
* **探索方向：**
    * 各特征的分布情况，与违约状态的关系 (如高负债率的公司是否更容易违约)。
    * 特征之间的相关性。
    * 类别不平衡问题：通常违约的样本远少于未违约的样本。

3. 数据预处理:
* 加载数据。
* 处理缺失值：对于财务数据，缺失值处理尤为重要，可能需要结合业务理解或使用更复杂的插补方法。
* **类别特征编码：** 行业、贷款类型等需要编码 (`LabelEncoder` 或 `pd.get_dummies()`)。
* **数值特征缩放：** 财务指标和金额大小差异可能很大，需要缩放 (`StandardScaler` 或 `MinMaxScaler`)。

4. 特征工程:

* 创建新的财务比率 (如利息保障倍数、营运资本周转率等)。
* 根据时间序列数据创建趋势特征 (如果数据包含多年财务信息)。
* 离散化连续特征 (如将贷款金额分段)。
* 特征交叉。

5. 模型选择与训练:

* **逻辑回归 (`LogisticRegression`)**: 课件中提到的分类问题常用方法，可以输出概率，便于设定阈值。
* **决策树 (`DecisionTreeClassifier`) 和 随机森林 (`RandomForestClassifier`)**: 课件中泰坦尼克案例有应用，对特征缩放不敏感，易于解释。
* **支持向量机 (`SVC`)**: 课件中泰坦尼克案例有应用。
* **梯度提升机 (XGBoost, LightGBM, CatBoost)**: (搜索：梯度提升算法) 在这类表格数据分类任务中通常表现优异 (可能超出课件范围，但可以提及)。
* **处理类别不平衡：** (搜索：处理不平衡数据方法)
  * 过采样少数类 (SMOTE)。
  * 欠采样多数类。
  * 调整类别权重 (很多分类器支持 `class_weight` 参数)。

6. 模型评估:

* **准确率 (`accuracy_score`)**: 在类别不平衡时具有误导性。
* **精确率 (`precision_score`)**: 关注预测为违约的准确性 (减少误判正常公司为违约)。
* **召回率 (`recall_score`)**: 关注识别出所有实际违约公司的能力 (减少漏判违约公司)。
* **F1 分数 (`f1_score`)**: 精确率和召回率的调和平均值。
* **ROC 曲线和 AUC 值 (`roc_auc_score`, `roc_curve`)**: 综合评估模型在不同阈值下的性能。
* **混淆矩阵 (`confusion_matrix`)**: 清晰展示各类别的预测情况。

7. 潜在挑战:

* **类别不平衡：** 违约样本少，模型可能倾向于预测多数类。
* **特征选择：** 金融指标众多，选择有预测能力的特征很重要。
* **模型解释性：** 银行需要理解模型为何做出违约判断。
* **经济周期影响：** 宏观经济状况可能影响违约率，模型需要考虑时效性。

【任务3】网络入侵检测 (KDD Cup 1999 Data)

1. 问题定义:

* 任务：区分正常网络连接和不良连接（入侵或攻击）。
* 问题类型：**多分类问题** (区分多种攻击类型和正常连接) 或 **二分类问题** (正常 vs. 攻击)。KDD Cup 1999 数据集本身是多分类的，但可以简化为二分类。

2. 数据集理解与探索 (KDD Cup 1999):

* 包含大量网络连接记录，每条记录有多个特征。
* **特征示例：** (搜索：KDD Cup 1999 features)
    * 基本特征：连接持续时间、协议类型 (`tcp`, `udp`, `icmp`)、服务 (`http`, `ftp`, `telnet`)、标志 (`SF`, `REJ`)。
    * 内容特征：源字节数、目标字节数、错误连接比例等。
    * 基于时间的流量特征 (窗口内)：相同主机连接数、相同服务连接数等。
* **目标变量：** 连接类型 (`normal.`，以及多种攻击类型如 `neptune.` (DoS), `ipsweep.` (Probe) 等)。
* **探索方向：**
    * 各类连接的占比 (通常攻击类型非常不平衡)。
    * 不同特征在正常连接和各类攻击中的分布差异。
    * 类别型特征的取值情况。

3. 数据预处理:

* 加载数据 (数据量可能很大，考虑分块读取或使用 Dask/Spark 等工具，但对于实验可能提供的是抽样数据)。
* **类别特征编码：** `protocol_type`, `service`, `flag` 等需要编码。由于类别较多且无序，**独热编码 (`pd.get_dummies()`)** 通常是更好的选择。
* **数值特征缩放：** 字节数、持续时间等数值差异大，需要缩放 (`StandardScaler` 或 `MinMaxScaler`)。

4. 特征工程:

* KDD Cup 1999 数据集本身已包含一些工程特征 (基于时间的流量特征)。
* 根据对攻击类型的理解，可以尝试构建新的组合特征。
* 特征选择：由于特征数量较多 (41个基础特征)，可以考虑特征选择方法。

5. 模型选择与训练:

* **决策树 (`DecisionTreeClassifier`) 和 随机森林 (`RandomForestClassifier`):** 对处理类别特征和数值特征混合的数据效果较好，且对特征缩放不敏感。
* **朴素贝叶斯 (`GaussianNB`, `MultinomialNB`):** 课件中泰坦尼克案例有提及，简单高效，但假设特征独立。
* **支持向量机 (`SVC`):** 可能需要较长时间训练，对参数和特征缩放敏感。
* **K近邻 (`KNeighborsClassifier`):** 计算开销较大，对特征缩放敏感。
* **神经网络/深度学习:** (搜索：深度学习入侵检测) 对于复杂模式识别可能有效，但实现和调参复杂。
* **处理类别不平衡：** 攻击类型通常高度不平衡。

6. 模型评估:

* 由于是多分类问题，且类别不平衡：
    * **混淆矩阵 (`confusion_matrix`):** 详细了解各类别的预测情况。
    * **准确率 (`accuracy_score`):** 整体准确率。

    * **精确率、召回率、F1 分数 (per-class, macro-average, weighted-average)**: `precision_recall_fscore_support` 或 `classification_report`。
        * 宏平均 (Macro-average): 对每一类的指标求平均，平等对待每一类。
        * 微平均 (Micro-average): 对所有样本的 TP, FP, FN 汇总后计算指标，受大类的影响大。
        * 加权平均 (Weighted-average): 按每一类的样本数量加权平均。
    * 对于二分类 (正常 vs. 攻击) 问题，可以使用 ROC AUC。

7. 潜在挑战:

* **类别高度不平衡：** 某些攻击类型可能非常罕见。
* **高维度数据：** 特征数量多。
* **实时性要求：** 实际入侵检测系统需要快速响应。
* **新型攻击 (零日攻击)：** 模型可能难以检测训练集中未出现过的新型攻击。
* **数据量大：** 完整 KDD Cup 数据集很大，对计算资源有要求。

collated by zjn , powered by Gemini in 2025