机器学习

作业四

一. (12 分) 朴素贝叶斯分类器

一家电商公司希望通过用户评论的关键词来预测评论的情感（**正面**或 **负面**）。假设已经收集了一小部分用户评论，并从中提取出以下四个关键词作为特征：**“good”**、**“bad”**、**“quality”** 和 **“price”**。每条评论可以被标记为“正面”或“负面”。

评论情感	good 出现次数	bad 出现次数	quality 出现次数	price 出现次数
正面评论	50	5	45	20
负面评论	10	25	5	25

假设正面评论和负面评论的先验概率分别为 $P(\text{正面评论}) = 0.7$ 和 $P(\text{负面评论}) = 0.3$。

问题

1. **(5 分)** 基于上述数据，使用**拉普拉斯修正**（$\alpha = 1$）计算以下条件概率：

   - $P(\text{good}|\text{正面评论})$
   - $P(\text{bad}|\text{正面评论})$
   - $P(\text{quality}|\text{正面评论})$
   - $P(\text{price}|\text{正面评论})$

   同时，计算上述特征在负面评论下的条件概率。

2. **(7 分)** 假设我们有一条新评论 $X = \{\text{good}, \text{quality}, \text{price}\}$，请使用朴素贝叶斯分类器计算该评论属于**正面评论**和**负面评论**的后验概率 $P(\text{正面评论}|X)$ 和 $P(\text{负面评论}|X)$，并根据计算结果确定该评论的情感类别。

注：

- 本题的答案请以四舍五入后的小数点后两位的形式给出，比如 P=0.67。如果给出答案为分式，会扣分。
- 本题要平滑的属性为四个关键词，即 $\hat{P}_{good|\text{正面评论}}$。

二. (20 分) EM 算法

假设有一个包含 6 次硬币抛掷结果的数据集，记录了每次抛掷是否得到“正面”：

$$X = \{\text{正面, 反面, 反面, 正面, 正面, 反面}\}$$

假设这些结果是由两枚硬币 A 和 B 生成的，每次抛掷时选择使用硬币 A 或 B 的概率均为 0.5。然而，具体每次抛掷使用的是哪一枚硬币是未知的。硬币 A 和 B 的正面概率分别为 θ_A 和 θ_B。我们的目标是通过 EM 算法估计这两枚硬币的正面概率 θ_A 和 θ_B。

已知：1. 初始参数：硬币 A 的正面概率 $\theta_A^{(0)} = 0.6$ 和硬币 B 的正面概率 $\theta_B^{(0)} = 0.5$。2. 每次抛掷使用硬币 A 和硬币 B 的概率均为 0.5，即 $P(A) = 0.5$ 和 $P(B) = 0.5$。

请通过一轮 EM 算法的迭代步骤，估计硬币 A 和 B 的正面概率 θ_A 和 θ_B。本题的答案请以分式或者小数点后两位的形式给出，比如 P=0.67.

问题：

1. **E 步**（10 分）：对于每一次抛掷结果，使用当前的参数估计值（$\theta_A^{(0)}$ 和 $\theta_B^{(0)}$），计算该结果由硬币 A 和硬币 B 生成的后验概率，即每次抛掷属于硬币 A 和硬币 B 的“软分配”概率。

   请计算以下内容：

   - 在第 1 次到第 6 次抛掷中，每个结果（正面或反面）由硬币 A 生成的概率 $P(A|x_i)$。
   - 每个结果由硬币 B 生成的概率 $P(B|x_i)$。

2. **M 步**（10 分）：基于 E 步计算出的“软分配”概率，计算硬币 A 和 B 的正面和反面出现的期望次数，并更新硬币 A 和 B 的正面概率 θ_A 和 θ_B。

   请计算以下内容：

   - 硬币 A 的正面和反面期望出现次数，并据此更新硬币 A 的正面概率 $\theta_A^{(1)}$。
   - 硬币 B 的正面和反面期望出现次数，并据此更新硬币 B 的正面概率 $\theta_B^{(1)}$。

三. (28 分) 集成学习

集成学习是一种通用技术，通过将多个不同模型的预测结果结合起来，以平均值或多数投票的方式生成单一预测，从而有效应对过拟合问题。

1. （3 分）考虑一组互不相关的随机变量 $\{Y_i\}_{i=1}^n$，其均值为 μ，方差为 σ^2。请计算这些随机变量平均值的期望和方差，给出计算过程。提示：在集成方法的背景下，这些 Y_i 类似于分类器 i 所作的预测。

2. 在第 1 小问中，我们看到对于不相关的分类器，取平均可以有效减少方差。尽管现实中的预测不可能完全不相关，但降低决策树之间的相关性通常能减少最终方差。现在，重新考虑一组具有相关性的随机变量 $\{Z_i\}_{i=1}^n$，其均值为 μ，方差为 σ^2，每个 $Z_i \in \mathbb{R}$ 为标量。假设对任意 $i \neq j$，$\mathrm{Corr}(Z_i, Z_j) = \rho$。提示：如果你不记得相关性与协方差之间的关系，请回顾你的概率论等课程内容。

- （5 分）请计算随机变量 Z_i 平均值的方差，以 σ、ρ 和 n 为变量表示，给出计算过程。
- （6 分）当 n 非常大时，会发生什么？这对于取平均的潜在有效性说明了什么？（……如果 ρ 很大（$|\rho| \approx 1$）会怎样？……如果 ρ 非常小（$|\rho| \approx 0$）又会怎样？……如果 ρ 处于中等水平（$|\rho| \approx 0.5$）呢?）无需严格推导——基于你得出的方差公式，用定性分析进行讨论即可。

3. Bagging 是一种通过随机化从同一数据集生成多个不同学习器的方法。给定一个大小为 n 的训练集，Bagging 会通过有放回抽样生成 T 个随机子样本集，每个子样本集大小为 n'。在每个子样本集中，一些数据点可能会被多次选中，而另一些可能完全未被选中。当 $n' = n$ 时，大约 63% 的数据点会被选中，而剩下的 37% 被称为袋外样本点（Out-of-Bag，OOB）。

- （7 分）为什么是 63%? 提示：当 n 很大时，某个样本点未被选中的概率是多少？请只考虑在任意一个子样本集中（而不是所有 T 个子样本集）未被选中的概率。
- （7 分）关于决策树的数量 T。集成中的决策树数量通常需要在运行时间和降低方差之间进行权衡（典型值范围从几十到几千棵树）。而子样本大小 n' 对运行时间的影响较小，因此选择 n' 时主要考虑如何获得最优预测结果。虽然常见实践是设置 $n' = n$，但这并非总是最佳选择。你会如何建议我们选择超参数 n'?

四. (20 分) 聚类理论

聚类是一种无监督学习任务，其核心是根据数据样本之间的相似性（通常由距离度量定义）将数据分成若干个簇。常用聚类算法如 k-均值和 DBSCAN 都需要特定的距离度量和超参数设置。以下问题围绕距离度量、目标函数、以及超参数设置展开：

1. （5 分）在聚类算法中，距离度量是衡量样本间相似性的基础，选择合适的距离度量对聚类效果有显著影响。给定欧几里得距离、曼哈顿距离和余弦相似度，在以下场景中，分析哪种距离更适合使用，并简要说明原因：

- 场景 1：高维稀疏特征向量（如文本数据的 TF-IDF 表示）
- 场景 2：二维几何分布数据（如图像中的空间点分布）

2. （15 分）k-均值聚类的目标函数与迭代过程, k-均值聚类的目标是最小化以下目标函数：

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

其中，C_i 表示第 i 个聚类簇，μ_i 为第 i 个簇的中心。

(a) 推导在分配样本点到最近的簇中心时，为什么目标函数 J 会减少。

(b) 推导为什么更新簇中心为簇内样本点的均值时，目标函数 J 会减少。

(c) k-均值的超参数 k 对结果有何影响?

- 如果 k 设置过大或过小，分别可能会导致什么问题?
- 提出一种确定 k 的方法，并解释其原理。

五. (20 分) 聚类实战

使用商场客户数据集 (Mall Customer Dataset) 完成客户分群分析。该数据集包含客户的年龄 (Age)、年收入 (Annual Income) 和消费积分 (Spending Score) 等特征。你需要通过实现和优化聚类算法来完成客户画像分析。数据随作业提供。

为方便起见，每小题我们提供了部分初始代码，其中包括预处理步骤和部分功能的实现。你可以自由选择使用或不使用这些代码来完成实现。

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler

# 加载数据
def load_mall_data():
    df = pd.read_csv("Mall_Customers.csv")
    X = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']].values
    # 数据标准化
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    return X_scaled, X, df
```

Listing 1: 数据加载

1. 在不借助外部实现的情况下，手动实现 KMeans 方法 (5 分)，在数据集上进行聚类，可视化聚类结果 (2 分)，并解决下列问题 (3 分)：

- 如何使用肘部法则确定合适的 k 值，绘图说明
- 简单分析每个客户群的特征
- 计算和分析簇内平方和 (inertia)

```python
class KMeans:
    def __init__(self, n_clusters=3, max_iters=100):
        self.n_clusters = n_clusters
        self.max_iters = max_iters
        self.centroids = None
        self.labels = None
        self.inertia_ = None  # 簇内平方和

    def fit(self, X):
        """
        实现K-means算法
        参数:
            X: shape (n_samples, n_features)
        返回:
```

```
            self
        """
        # TODO: 实现K-means算法
        # 1. 随机初始化聚类中心
        # 2. 迭代优化直到收敛
        pass

    def predict(self, X):
        """返回每个样本的聚类标签"""
        pass

def plot_clusters(X, labels, centroids=None):
    """绘制聚类结果的散点图"""
    pass
```

Listing 2: Kmeans 部分实现

2. 在实际业务中，不同特征的重要性可能不同，且某些客户群可能需要大小相近。请实现带权重和大小约束的改进版本：

- 实现带约束的聚类算法，需要支持特征加权和簇大小约束 (6 分)
- 在以下两个场景下重新进行实验：收入特征权重加倍，限制每个客户群至少包含 20% 的客户，绘制聚类结果 (4 分)

```
class ConstrainedKMeans(KMeansPlusPlus):
    def __init__(self, n_clusters=3, max_iters=100, weights=None, size_constraints=None):
        """
        参数:
            weights: 特征权重向量
            size_constraints: 每个簇的最小和最大样本数量限制
        """
        super().__init__(n_clusters, max_iters)
        self.weights = weights
        self.size_constraints = size_constraints

    def _weighted_distance(self, X, centroids):
        """计算加权欧氏距离"""
        pass

    def _reassign_clusters(self, X, labels, distances):
        """在满足大小约束的情况下重新分配样本到簇"""
        pass
```

Listing 3: 带约束的聚类算法部分实现