目录

1. 总体概况

1.1 实验背景和目的

随着数字化时代的到来，大数据关系到我们生活的方方面面。电影行业作为文化产业的重要组成部分，其数据量也在迅速增长。从电影的票房收入、放映时间、导演、演员，到观众的观影评价，都构成了庞大的数据集。这些数据不仅反映了电影市场的动态，也为预测算法提供了丰富的信息来源。

预测算法旨在利用这些数据集，通过机器学习和数据分析的方法，对电影的票房收入、口碑等未来表现进行预测。这样的预测可以为电影制作方、发行方以及观众提供有价值的参考信息。例如，制作方可以根据预测结果调整拍摄策略，发行方可以更有针对性地进行宣传和营销，而观众则可以根据预测评分来选择观影。

为了实现这一目标，探究不同预测算法在电影评分预测中的准确性和适用性。我们将收集电影相关的历史数据，包括电影的类型和观众的评分数据。然后，我们将采用多种预测算法，如线性回归、KNN、随机森林和逻辑回归等，对电影的评分进行预测。

通过对比不同算法的预测结果，我们将评估其准确率。

1.2 总体设计思路

根据提供的数据集，对数据进行预处理，拼接成一个总体的 dataframe，使用独热编码对 Genres 项进行编码，采用 sklearn 包预置的几种函数进行数据预测处理，多次重复，取平均值，得到最终准确率。

1.3 项目模块结构

图 1-1

2. 环境介绍

2.1 环境构成

CPU: AMD Ryzen R7-6800H
GPU:NVIDIA Geforce RTX 3050
Python 3.11.5
Conda 23.9.0
Jetbrains PyCharm 2023.2.1

2.2 依赖环境

可以通过 pip 安装到全局，也可以通过 conda 创建虚拟环境后再安装依赖,所需依赖如下：

pandas：2.1.3
scikit-learn：1.3.2
time：python 内置
matplotlib：3.8.2
numpy：1.26.0

3. 数据处理

3.1 数据初始化

在 dataset.py 中定义函数 data_init()，目的是从给定的数据目录（'/data'）中读取电影、评分和标签数据，进行数据预处理，然后返回一个合并后的 DataFrame。

1. **读取数据：**

从 data/movies.csv 读取电影信息，设置列名为 "MovieID", "Title", "Genres"。删除第一行（df_mv.drop(df_mv.index[0], inplace=True)）。

从 data/ratings.csv 读取用户评分信息，设置列名为 "UserID", "MovieID", "Rating", "Timestamp"。删除第一行（df_rt.drop(df_rt.index[0], inplace=True)）和 "Timestamp" 列（df_rt.drop("Timestamp", axis=1, inplace=True)）。

从 data/tags.csv 读取用户标签信息，设置列名为 "UserID", "MovieID", "Tag","Timestamp"。删除第一行（df_tg.drop(df_tg.index[0], inplace=True)）和"Timestamp" 列（df_tg.drop("Timestamp", axis=1, inplace=True)）。

2. **数据合并：**

使用 merge 方法将电影信息与评分信息合并，基于 MovieID 进行外连接（outer join）。使用同样的方法将电影信息与标签信息合并，同样基于 MovieID 进行外连接。最后，将前两步得到的合并数据再合并，形成一个完整的包含电影、评分和标签的数据集。

3. **数据规范化：**

使用 fillna 方法将缺失值（NaN）替换为 0。将 MovieID 和 UserID 列的数据类型转换为整数。将 Rating 列的数据类型转换为浮点数。

4. **添加平均评分列：**

使用 groupby 方法按 MovieID 对评分进行分组，并计算每部电影的平均评分。然后将这个平均评分列添加到数据框中。

5. **处理年份数据：**

从标题中提取年份，并创建一个新的列 Year。将缺失值替换为 0。将 Year 列的数据类型转换为整数。

6. **整数化：**

将 Rating 列的数据类型转换为整数。将 Year 列的数据类型转换为整数。返回

值：返回处理后的合并数据框 df_merged。

本部分代码实现了从指定目录中读取电影、评分和标签数据，进行数据的清洗、合并和规范化处理，然后返回一个包含电影、评分和标签的完整数据框。

3.2 独热编码

在 dataset.py 中定义函数 onehot_encoding，主要目的是对输入数据框 df_merged 中的 Genres 列进行独热编码（One-Hot Encoding）。

1. **作用：**

对 df_merged 数据框中的 Genres 列进行独热编码。独热编码是一种将类别型变量转换为机器学习模型可读特征的技术。在编码中，每个唯一的类别值在输出中都是一个单独的二进制特征，且只有当该特征对应的类别值出现时，该二进制特征才为 1，否则为 0。

2. **使用的方法：**

pd.Series(df_merged.Genres.str.split('|').tolist())：

使用 split 方法将 Genres 列中的每个值按 '|' 分割，然后将结果转换为列表。

list2series.fillna('0', inplace=True)：将缺失值（NaN）替换为 0。

MultiLabelBinarizer()：

这是一个用于多标签分类的类，可以用来对数据集中的多标签进行独热编码。

mlb.fit_transform(list2series)：

这里，首先使用 fit 方法拟合转换器以确定标签和类别的对应关系，然后使用 transform 方法对数据进行独热编码。

pd.DataFrame(...)：

将独热编码后的数据转换为 dataframe 格式。

df_merged.join(df_genres)：

将原始数据框与独热编码后的 dataframe 合并。

df_merged.drop("Title",axis=1,inplace=True)

df_merged.drop("Genres",axis=1, inplace=True)：

删除原始 df 中的 Title 和 Genres 列。

df_merged.drop("0", axis=1, inplace=True)：

删除独热编码后添加的字符串'0'列。

3. **优点：**

可读性：独热编码是一种非常直观的表示方式，可以清晰地表示类别变量的每一个

可能值。

模型兼容性：许多机器学习模型，特别是分类模型，需要数值型输入。独热编码为这些模型提供了这样的输入。

避免类别偏斜问题：在某些情况下，某些类别可能比其他类别更常见，这可能导致模型的偏差。独热编码可以确保每个类别都有等可能的表示，从而避免这种偏斜问题。

方便扩展：随着新类别的出现，独热编码可以很容易地扩展到新的二进制特征，而无需修改现有代码。

3.3 数据分割

在 dataset.py 中定义函数 data_split，目的是将给定的数据集 df_merged 按照 9:1 的比例划分为训练集和测试集。

1. **作用：**

将数据集 df_merged 划分为比例为 90%的训练集和 10%的训练集和测试集。

2. **使用方法：**

x_feature = df_merged.drop("Rating", axis=1):

从 df_merged 中移除了"Rating" 列，并将结果存储在 x_feature 中。将所有特征（除 "Rating" 以外）用作输入特征 x。

y_target = df_merged['Rating']:

从 df_merged 中提取"Rating"列，并将其存储在 y_target 中。将 "Rating" 列用作目标变量或标签 y。

x_train, x_test, y_train, y_test = train_test_split(x_feature, y_target, test_size=0.1, random_state=None):

这行代码使用了 train_test_split 函数，这是一个常用的函数，用于将数据集随机划分为训练集和测试集。这里，测试集的大小设置为 10%（即 0.1）。random_state=None 表示随机数生成器的种子没有设置，因此结果是可变的。

3. **实现的特性：**

函数首先提取输入特征和目标标签。然后，使用 train_test_split 函数将数据集划分为训练集和测试集。最后，返回训练集的输入特征、测试集的输入特征、训练集的目标标签和测试集的目标标签。

4. **优点：**

数据拆分：该代码使用了一个简单但有效的方法来拆分数据集，确保了数据的随机分布。

灵活性：可以调整 `test_size` 参数来改变训练和测试集的划分比例。

可复用性：代码使用了广泛接受的库和方法，可以很容易地与其他机器学习或数据分析流程集成。

4. 预测算法

4.1 线性预测

线性回归是一种回归分析的方法，通过找出一个线性模型来预测一个响应变量和一个或多个自变量之间的关系。

创建线性回归模型 lr，然后使用训练数据 x_train 和 y_train 来拟合模型。后使用测试数据 x_test 来预测 y_test 的值，得到预测结果 y_pred。最后，使用 metrics.accuracy_score() 函数来计算预测结果的准确率。

线性回归模型通过最小二乘法来拟合数据，并使用训练数据来找到最佳的线性模型参数。然后使用这个模型对测试数据进行预测，并计算预测结果的准确率。

4.2 K 近邻预测

使用了 K 近邻（K-Nearest Neighbors，KNN）回归算法来进行预测。KNN 是一种简单且常用的监督学习算法，可以用于分类和回归任务。

在 KNN 回归中，算法通过测量不同数据点之间的距离来确定它们之间的相似性。对于给定的测试样本，KNN 回归会找到训练数据集中与测试样本最接近的 K 个邻居，并使用这些邻居的目标变量值来进行预测。通常情况下，预测结果是这 K 个邻居目标变量值的平均值。

首先创建了一个 KNN 回归模型 knn，并设置 n_neighbors=32，即使用 32 个最近的邻居来进行预测。然后使用训练数据 x_train 和 y_train 来拟合模型。接着使用测试数据 x_test 来进行预测，得到预测结果 y_pred。最后使用 metrics.accuracy_score() 函数来计算预测结果的准确率。

KNN 回归算法是一种基于距离度量的方法，通过找到与测试样本最近的 K 个邻居并使用它们的目标变量值来进行预测。

4.3 随机森林回归预测

随机森林是一种集成学习算法，通过构建多个决策树并结合它们的预测结果来提高模型的准确性和稳定性。

在随机森林中，算法首先生成多棵决策树，每棵树都基于训练数据的随机子集进行训练。每棵决策树都对一个样本进行预测，并将预测结果存储在每个树的输出中。然后，随机森林通过投票或平均值的方式将多棵树的预测结果结合起来，得到最终的预

测结果。

创建随机森林回归模型 rf，并设置 n_estimators，使用一定数量的决策树来构建随机森林。random_state 用于确保每次运行代码时都能得到相同的结果。然后使用训练数据 x_train 和 y_train 来拟合模型。拟合模型的过程就是训练决策树，并存储它们的预测结果。接着，使用测试数据 x_test 来进行预测，得到预测结果 y_pred。这里的预测结果是这些决策树的预测结果的平均值。最后，使用 metrics.accuracy_score() 函数来计算预测结果的准确率。

随机森林回归算法是一种基于集成学习的方法，通过构建多棵决策树并综合它们的预测结果来提高模型的准确性和稳定性。

4.4 逻辑回归预测

逻辑回归（Logistic Regression）是一种广泛用于分类问题的统计方法，尤其适用于二元分类问题。

在逻辑回归中，算法通过拟合一个逻辑函数来预测一个样本属于某个类别的概率。逻辑函数通常使用 sigmoid 函数，可以将任何实数映射到 0 和 1 之间，表示概率。逻辑回归模型的训练过程是通过最大化训练数据的似然函数来估计模型参数。

创建逻辑回归模型 logreg，并设置 max_iter=1000，即最大迭代次数为 1000。然后使用训练数据 x_train 和 y_train 来拟合模型。拟合模型的过程就是通过迭代优化算法来找到最佳的模型参数，使得模型能够最好地拟合训练数据。接着使用测试数据 x_test 来进行预测，得到预测结果 y_pred。最后使用 metrics.accuracy_score() 函数来计算预测结果的准确率。

逻辑回归算法是一种分类算法，通过拟合一个逻辑函数来预测样本属于某个类别的概率，并使用准确率来评估模型的性能。

5. 实验结果

程序运行结果如下：

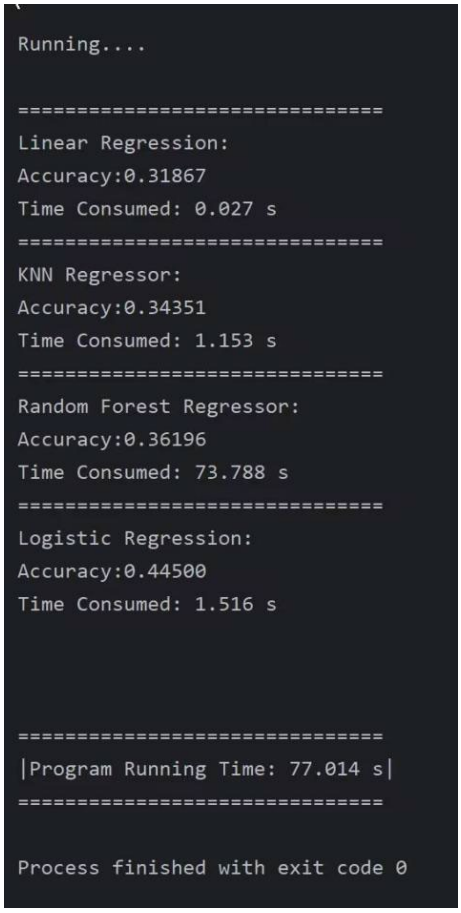

图 5-1 实验运行结果

根据提供的实验数据，线性回归、KNN、随机森林和逻辑回归的准确率分别为 31.9%，34.4%，36.2%和 44.5%。以下是针对这些结果的详细分析：

线性回归：线性回归主要用于预测连续值，而在本次实验中，我们尝试使用它来预测分类问题。这可能是导致线性回归准确率较低的原因。线性回归可能无法很好地捕获分类问题中的非线性关系。

KNN（K-最近邻）：KNN 是一种基于实例的学习算法，它根据输入数据的 k 个最近邻居的类别来预测输入数据的类别。KNN 的准确率略高于线性回归，但仍然相对较低。这可能是由于数据中的噪声、特征选择不当或 k 值的选择不合适等原因导致的。

随机森林：随机森林是一种集成学习方法，通过构建多个决策树并结合它们的预测来提高模型的准确性。在本实验中，随机森林的准确率高于线性回归和 KNN，这表明集成方法在处理分类问题时的有效性。然而，随机森林的性能仍然有提升的空间。

逻辑回归：逻辑回归是一种用于分类问题的统计方法，它通过拟合一个逻辑函数来预测样本属于某个类别的概率。在本实验中，逻辑回归的准确率最高，达到 44.5%。这表明逻辑回归在处理分类问题时能够较好地拟合数据并捕获类别之间的关系。

综上所述，虽然四种算法在处理分类问题时都取得了一定的效果，但逻辑回归在本次实验中的表现最佳。

6. 实验总结

在完成本次 Python 人工智能与大数据实验课程后，我们获得了许多宝贵的经验和感悟。

首先，我们深刻体会到了数据在人工智能领域的重要性。在本次实验中，我们需要收集并处理大量的电影评分数据，这些数据的质量和特征直接影响了我们模型的预测性能。通过数据预处理和特征工程，我们提取了有用的信息，为后续的模型训练提供了坚实的基础。

其次，我们感受到了不同算法在处理同一问题时的差异性和优势。线性回归、KNN、随机森林和逻辑回归等算法各有特点，它们在电影评分预测问题上的表现也各不相同。通过比较不同算法的性能指标，我们可以根据实际需求选择最合适的算法进行应用。

通过本次实验课程，我们掌握了数据预处理、特征工程、模型训练和评估等重要环节。这为我们在今后的学习与实践中应用机器学习算法打下了坚实的基础。未来，我们还可以尝试将更多先进的算法应用到实际问题中，例如集成学习、深度学习等，以进一步提高预测精度和泛化能力。同时，我们也要关注数据质量和特征选择对模型性能的影响，以实现更有效的预测结果。

最后，通过本次实验课程，我们不仅掌握了相关知识和技能，还培养了解决实际问题的能力。在实验过程中，我们遇到了许多挑战和困难，但通过不断尝试和调整，最终取得了令人满意的成果。这让我们更加自信，并激发了我们对人工智能和大数据领域的浓厚兴趣。

总之，本次 Python 人工智能与大数据实验课程让我们收获颇丰。我们相信在未来的学习和实践中，将继续运用所学知识和技能，探索更多有趣的问题和挑战。

7. 参考文献

[^1]: Python for Data Analysis, Third Edition, Wes McKinney, ISBN-9787111726722
[^2]: Movie Lens Dataset Visulisation and Prediction
(https://www.kaggle.com/code/aigamer/movie-lens-dataset-visulisation-and-prediction)
[^3]: 用 Python 如何进行预测型数据分析
(https://zhuanlan.zhihu.com/p/50456164)
[^4]: 机器学习实战：Python 基于 K 近邻 KNN 进行分类预测
(https://blog.csdn.net/weixin_48093827/article/details/130060221)
[^5]: Movie Lens Dataset
(https://www.kaggle.com/datasets/aigamer/movie-lens-dataset/code)
[^6]: 如何使用 Python scikit-learn 机器学习库做分类和回归预测
(https://zhuanlan.zhihu.com/p/53278304)