哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：2024 春季

课程名称：统计机器学习

实验名称：构建决策树模型实现银行借贷预测

实验性质：设计型

实验学时：2 地点：T2102

学生班级：计算机 10 班

学生学号：220111028

学生姓名：许辰涛

实验与创新实践教育中心制

2024 年 1 月

一、 实验环境

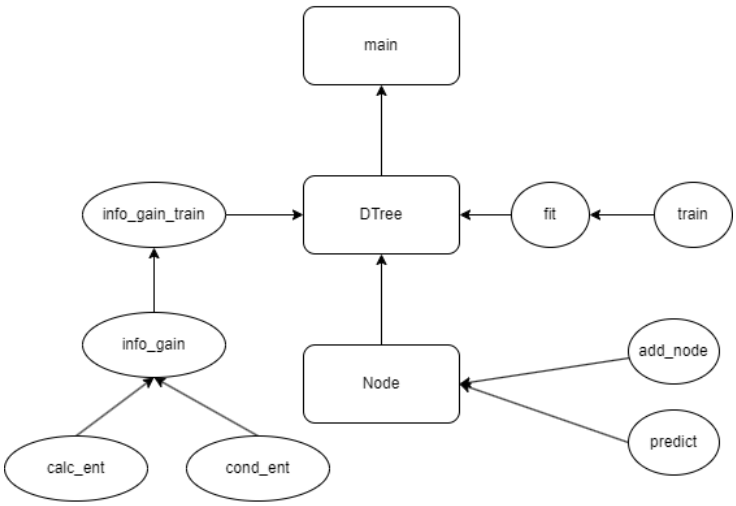
请填写用到的操作系统和主要软件版本。

操作系统：Windows11
Python：3.10.12

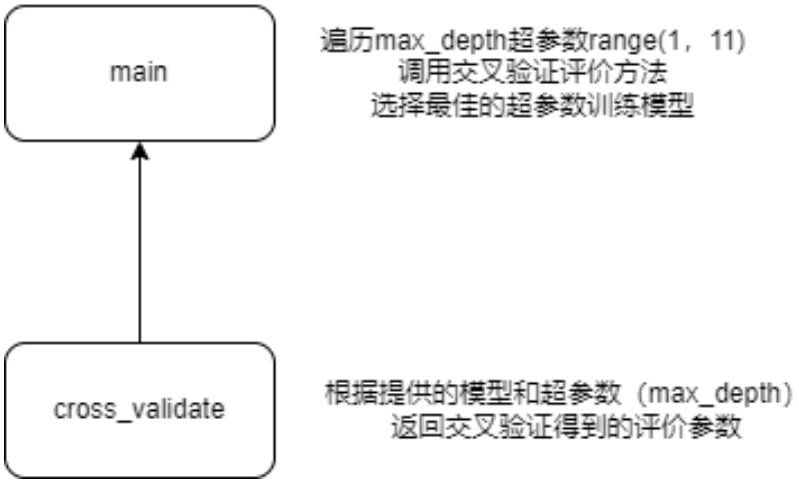
二、 实验过程

1. *程序总体结构设计：main 函数与各子函数之间的调用和返回关系，程序运行流程等，可以用文字、流程图、或图表的方式描述。*

任务一



任务二



2. 数据集分割

```
# 加载数据
train_data = pd.read_excel('银行借贷数据集train.xls')
test_data = pd.read_excel('银行借贷数据集test.xls')

# drop id
train_data = train_data.drop("nameid", axis=1)
test_data = test_data.drop("nameid", axis=1)

# revenue离散化
re = [0,10000,20000,30000,40000,50000]
train_data['revenue'] = pd.cut(train_data["revenue"], re, labels=False)
test_data["revenue"] = pd.cut(test_data["revenue"], re, labels=False)

# 假设最后一列是目标变量
X_train = train_data.iloc[:, :-1]
y_train = train_data.iloc[:, -1]
X_test = test_data.iloc[:, :-1]
y_test = test_data.iloc[:, -1]
```

要点：

- revenue 离散化
- 去除 id 列

K 折交叉验证

```
# 定义一个函数进行交叉验证
def cross_validate(X, y, model, cv=10):
    kf = KFold(n_splits=cv, shuffle=True, random_state=1)
    precision_scores = []
    recall_scores = []
    f1_scores = []

    for train_index, val_index in kf.split(X):
        X_kf_train, X_kf_val = X.iloc[train_index], X.iloc[val_index]
        y_kf_train, y_kf_val = y.iloc[train_index], y.iloc[val_index]

        clf = model
        clf.fit(X_kf_train, y_kf_train)
        y_kf_pred = clf.predict(X_kf_val)

        precision_scores.append(precision_score(y_kf_val, y_kf_pred))
        recall_scores.append(recall_score(y_kf_val, y_kf_pred))
        f1_scores.append(f1_score(y_kf_val, y_kf_pred))

    return np.mean(precision_scores), np.mean(recall_scores), np.mean(f1_scores)
```

3. *决策树算法：选用合适的决策树算法。*

任务 1：

```
# 信息增益比
def info_gain(self, ent, cond_ent):
    if cond_ent == 0:
        return 100000
    else:
        return (ent - cond_ent) / cond_ent
```

C4.5 算法，挑选信息增益比最大的特征

任务 2：
Scikit-learn 中的 DecisionTreeClassifier，使用优化后的 CART 算法

4. *评估模型：选用合适的评估指标。*

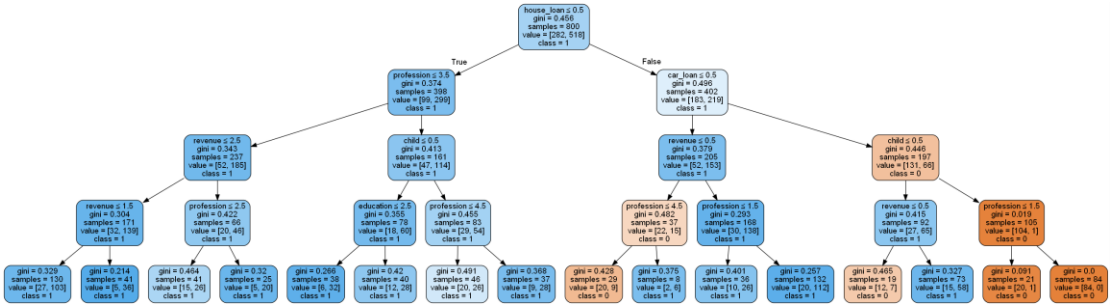
```
# 计算评估指标
def calculate_precision_recall_f1(y_true, y_pred):
    tp = sum((y_true == 1) & (y_pred == 1))
    fp = sum((y_true == 0) & (y_pred == 1))
    fn = sum((y_true == 1) & (y_pred == 0))

    precision = tp / (tp + fp) if (tp + fp) > 0 else 0
    recall = tp / (tp + fn) if (tp + fn) > 0 else 0
    f1 = 2 * precision * recall / (precision + recall) if (precision + recall) > 0 else 0

    return precision, recall, f1
```

使用精确率、召回率得出的 F1-score 来评估模型

5. *总结分析：得到结论。*



借贷策略见决策树，在测试集上的指标为

```
Precision: 0.9760479041916168, Recall: 0.9819277108433735, F1 Score: 0.978978978978979
```

三、 收获与反思

请填写本次实验的收获，记录实验过程中出现的值得反思的问题及你的思考

收获：熟悉了决策树算法，将理论与实践相结合，进一步熟悉了 Python 语法

值得反思的问题：任务二在训练集上的准确度并不高，在 0.85 左右，但是在测试机上却可以达到 0.97，这使我产生了疑问，也不是欠拟合也不是过拟合

我的思考：考虑可能是训练集和测试集分布不均，可以尝试打乱后重新测试

```
# 合并训练集和测试集
combined_data = pd.concat([train_data, test_data], ignore_index=True)

# 混洗数据集
combined_data = combined_data.sample(frac=1, random_state=1).reset_index(drop=True)

# 重新分割成新的训练集和测试集，假设80%为训练集，20%为测试集
train_size = int(0.8 * len(combined_data))
train_data = combined_data.iloc[:train_size]
test_data = combined_data.iloc[train_size:]
```

在混合训练集测试集并打乱后得到了解决

```
Max Depth: 1, Precision: 0.6812499999999999, Recall: 1.0, F1 Score: 0.8094367346295412
Max Depth: 2, Precision: 0.7852082176399928, Recall: 0.8675113862771369, F1 Score: 0.8237369293171912
Max Depth: 3, Precision: 0.8064848988314939, Recall: 0.9706965332298794, F1 Score: 0.8803032275628212
Max Depth: 4, Precision: 0.8112864560749374, Recall: 0.9482824695741539, F1 Score: 0.8738703903357153
Max Depth: 5, Precision: 0.8061498175331503, Recall: 0.9453875118652857, F1 Score: 0.8691406580194434
Max Depth: 6, Precision: 0.8042165306872396, Recall: 0.910387627926099, F1 Score: 0.852939216488607
Max Depth: 7, Precision: 0.8054974451824686, Recall: 0.8771810683031258, F1 Score: 0.8383886753331711
Max Depth: 8, Precision: 0.8027207995250499, Recall: 0.8173059288043334, F1 Score: 0.8077474965920931
Max Depth: 9, Precision: 0.8087308300898748, Recall: 0.7939367599211022, F1 Score: 0.799859775604672
Max Depth: 10, Precision: 0.7984236869437688, Recall: 0.7721089006316898, F1 Score: 0.7827383682461803
{'max_depth': 3}
Test Set Evaluation - Precision: 0.8154761904761905, Recall: 0.9856115107913669, F1 Score: 0.8925081433224756
```