作业答案

1.使用思维导图总结集成学习部分的内容

略

2.说明集成学习的方式

- 集成学习思想：多个弱学习器组合成一个更强大的学习器，解决单一预测，进步一得到更好性能。
- 主要有以下两种方式
  - Bagging
    - 主要思想：又称**装袋算法**或者自举汇聚法，**有放回的抽样**(booststrap抽样)产生不同的训练集，从而训练不同的学习器，通过**平权投票**、多数表决的方式决定预测结果，在分类问题中，会使用多数投票统计结果，在回归问题中，会使用求均值统计结果，弱学习器**并行训练**
    - 代表算法：随机森林
  - Boosting：
    - 主要思想：每一个训练器重点**关注前一个训练器不足的地方**进行训练，通过**加权投票**的方式，得出预测结果，**串行训练**
    - 代表算法：Adaboost、GBDT、XGBoost、LightGBM

3.实现本部分所有的案例

3.1 RandomForest随机森林算法

```
# 1.导入依赖包
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

# 2.读取数据
data = pd.read_csv('./data/titanic/train.csv')
print(data.head())
print(data.info)

# 3.数据处理
x = data[['Pclass', 'Sex', 'Age']].copy()
y = data['Survived'].copy()
print(x.head(10))
x['Age'].fillna(x['Age'].mean(), inplace=True)
print(x.head(10))
x = pd.get_dummies(x)
```

```
print(x.head(10))

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

# 4.模型训练
# 4.1 决策树
tree = DecisionTreeClassifier()
tree.fit(x_train, y_train)

# 4.2 随机森林
rf = RandomForestClassifier()
rf.fit(x_train, y_train)

# 4.3 网格搜索交叉验证
params = {'n_estimators': [10, 20], 'max_depth': [2, 3, 4, 5]}
model = GridSearchCV(estimator=rf, param_grid=params, cv=3)
model.fit(x_train, y_train)
print(model.best_estimator_)

rfs = RandomForestClassifier(max_depth=4, n_estimators=10)
rfs.fit(x_train, y_train)

# 5.模型评估
# 5.1 决策树
print(tree.score(x_test, y_test))

# 5.2 随机森林
print(rf.score(x_test, y_test))

# 5.3 网格搜索交叉验证
print(rfs.score(x_test, y_test))
```

3.2 Adaboost算法

```
# 1.导入依赖包
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split

# 2.读取数据及数据预处理
# 2.1 读取数据
data = pd.read_csv('./data/wine0501.csv')
print(data.info)
# 2.2 数据预处理
data = data[data['Class label'] != 1]
x = data[['Alcohol', 'Hue']].copy()
y = data['Class label'].copy()
print(y)

pre = LabelEncoder()
y = pre.fit_transform(y)
print(y)
```

```
# 2.3 数据集划分
x_train, x_test, y_train, y_test = train_test_split(x, y)

# 3.模型训练
ada = AdaBoostClassifier()
ada.fit(x_train, y_train)

# 4.模型评估
print(ada.score(x_test, y_test))
```

3.3 GBDT算法

```
# 1.导入依赖包
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

# 2.读取数据及数据预处理
# 2.1读取数据
data = pd.read_csv('./data/titanic/train.csv')
print(data.head())
print(data.info)

# 2.2 数据预处理
x = data[['Pclass', 'Sex', 'Age']].copy()
y = data['Survived'].copy()
print(x.head(10))
x['Age'].fillna(x['Age'].mean(), inplace=True)
print(x.head(10))
x = pd.get_dummies(x)
print(x.head(10))

# 2.3 划分数据集
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

# 3.模型训练
model = GradientBoostingClassifier()
model.fit(x_train, y_train)

# 4.模型评估
print(model.score(x_test, y_test))
```

3.4 XGBoost算法

```
# 1.导入依赖包
import pandas as pd
from sklearn.model_selection import train_test_split
```

```python
from xgboost import XGBClassifier
from sklearn.metrics import classification_report
from sklearn.utils import class_weight
from sklearn.ensemble import GradientBoostingClassifier

# # 2.数据读取及数据预处理
# # 2.1 数据获取
# data =pd.read_csv('./data/红酒品质分类.csv')
# print(data.head())
# # 2.2 数据预处理
# x = data.iloc[:,:-1]
# y = data.iloc[:,-1]-3
# # 2.3 数据集划分
#
x_train,x_test,y_train,y_test=train_test_split(x,y,stratify=y,test_size=0.2
)

# pd.concat([x_train,y_train],axis=1).to_csv('红酒品质分类_train.csv')
# pd.concat([x_test,y_test],axis=1).to_csv('红酒品质分类_test.csv')

# 2.数据读取及数据预处理
# 2.1 数据获取
train_data = pd.read_csv('红酒品质分类_train.csv')
test_data = pd.read_csv('红酒品质分类_test.csv')
# 2.2 数据预处理
x_train = train_data.iloc[:, :-1]
y_train = train_data.iloc[:, -1]
x_test = test_data.iloc[:, :-1]
y_test = test_data.iloc[:, -1]
class_weight = class_weight.compute_sample_weight(class_weight='balanced',
y=y_train)

# 3.模型训练
model = XGBClassifier(n_estimators=5, objective='multi:softmax')
# model =GradientBoostingClassifier(n_estimators=5)
model.fit(x_train, y_train, sample_weight=class_weight)

# 4.模型预测
y_pre = model.predict(x_test)

# 5.模型评估
print(classification_report(y_test, y_pre))
```