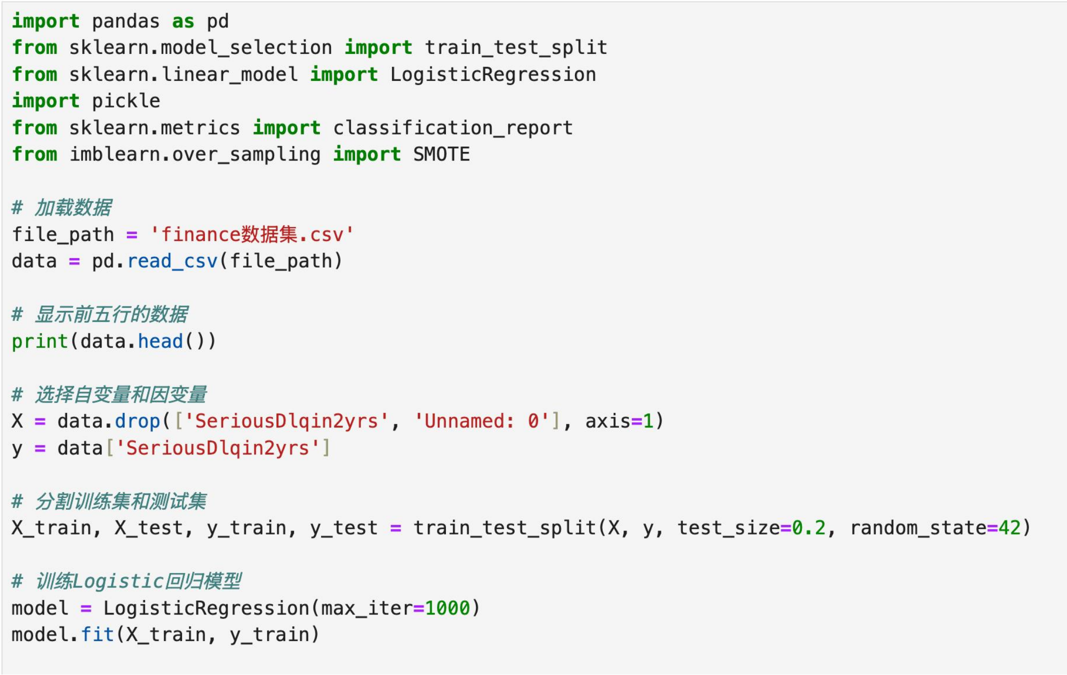
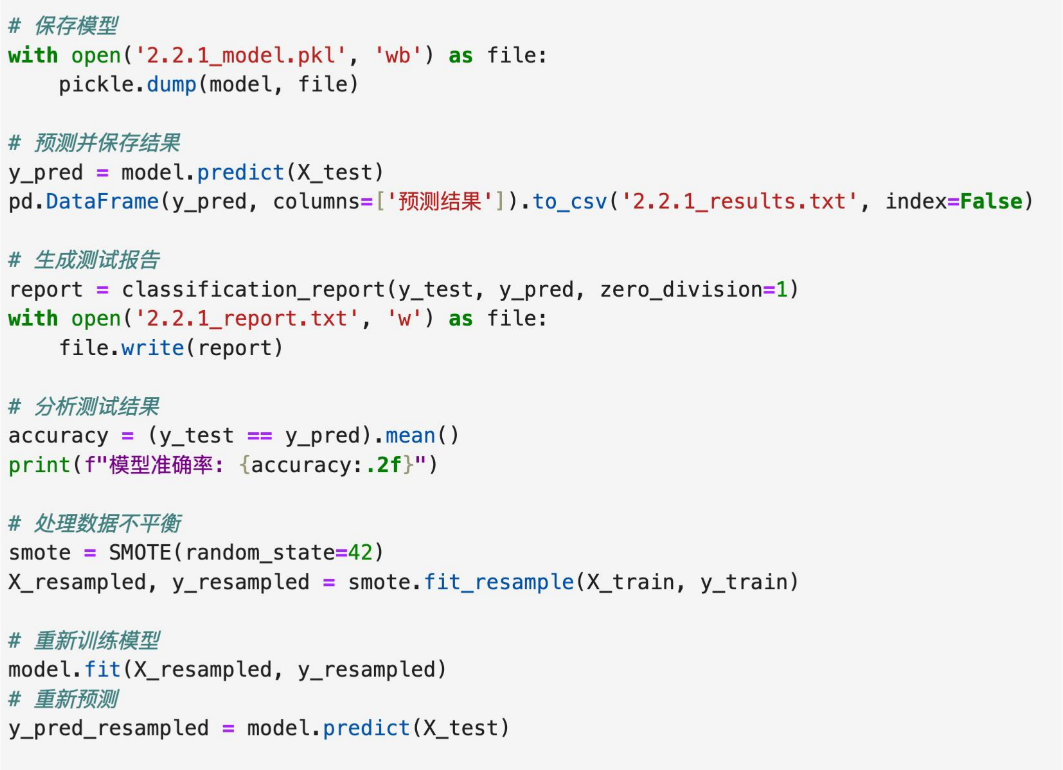
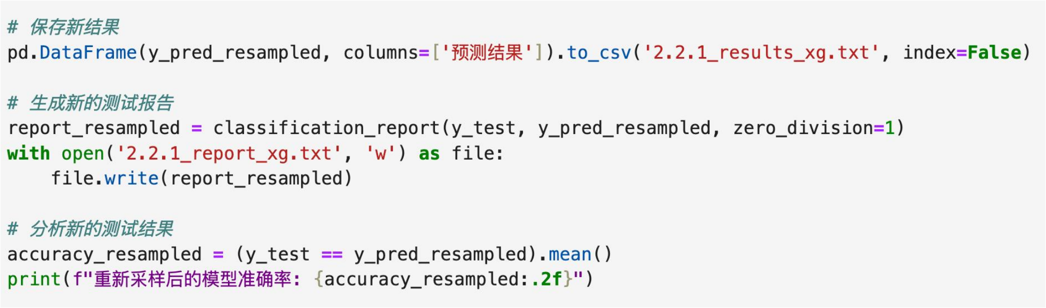
2.2.1



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import pickle
from sklearn.metrics import classification_report
from imblearn.over_sampling import SMOTE

# 加载数据
file_path = 'finance数据集.csv'
data = pd.read_csv(file_path)

# 显示前五行的数据
print(data.head())

# 选择自变量和因变量
X = data.drop(['SeriousDlqin2yrs', 'Unnamed: 0'], axis=1)
y = data['SeriousDlqin2yrs']

# 分割训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 训练Logistic回归模型
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

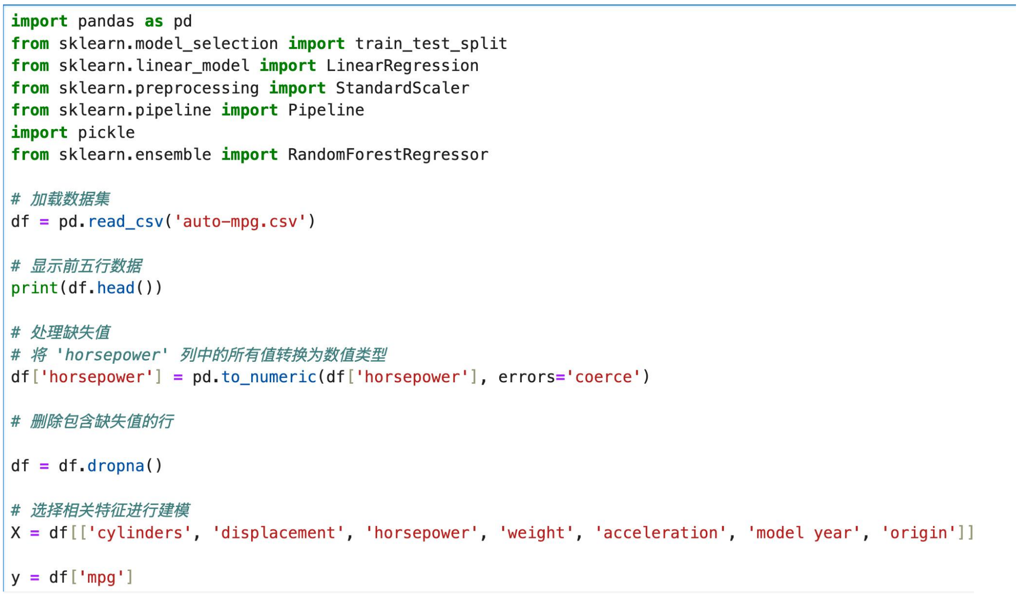


```python
# 保存模型
with open('2.2.1_model.pkl', 'wb') as file:
    pickle.dump(model, file)

# 预测并保存结果
y_pred = model.predict(X_test)
pd.DataFrame(y_pred, columns=['预测结果']).to_csv('2.2.1_results.txt', index=False)

# 生成测试报告
report = classification_report(y_test, y_pred, zero_division=1)
with open('2.2.1_report.txt', 'w') as file:
    file.write(report)

# 分析测试结果
accuracy = (y_test == y_pred).mean()
print(f"模型准确率: {accuracy:.2f}")

# 处理数据不平衡
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

# 重新训练模型
model.fit(X_resampled, y_resampled)
# 重新预测
y_pred_resampled = model.predict(X_test)
```

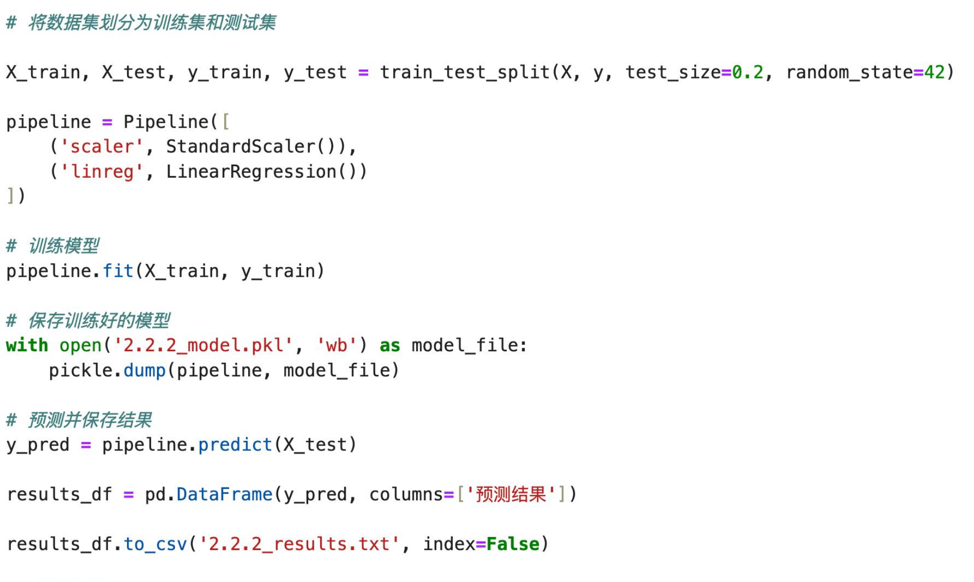


```python
# 保存新结果
pd.DataFrame(y_pred_resampled, columns=['预测结果']).to_csv('2.2.1_results_xg.txt', index=False)

# 生成新的测试报告
report_resampled = classification_report(y_test, y_pred_resampled, zero_division=1)
with open('2.2.1_report_xg.txt', 'w') as file:
    file.write(report_resampled)

# 分析新的测试结果
accuracy_resampled = (y_test == y_pred_resampled).mean()
print(f"重新采样后的模型准确率: {accuracy_resampled:.2f}")
```

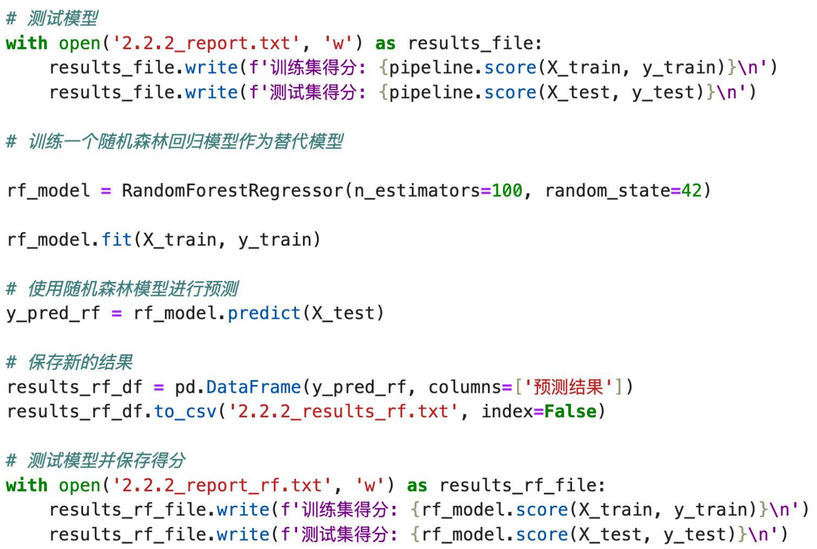
2.2.2



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
import pickle
from sklearn.ensemble import RandomForestRegressor

# 加载数据集
df = pd.read_csv('auto-mpg.csv')

# 显示前五行数据
print(df.head())

# 处理缺失值
# 将 'horsepower' 列中的所有值转换为数值类型
df['horsepower'] = pd.to_numeric(df['horsepower'], errors='coerce')

# 删除包含缺失值的行

df = df.dropna()

# 选择相关特征进行建模
X = df[['cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'model year', 'origin']]

y = df['mpg']
```

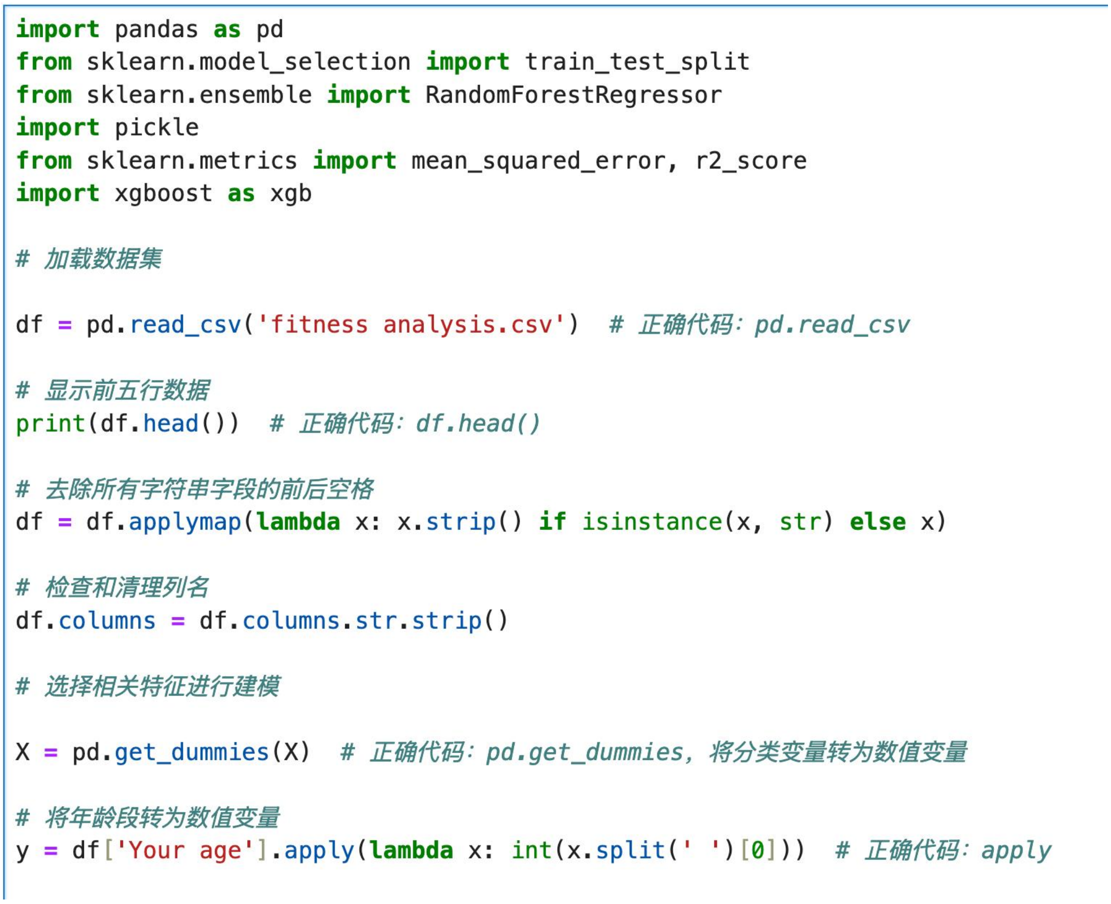


```
# 将数据集划分为训练集和测试集

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('linreg', LinearRegression())
])

# 训练模型
pipeline.fit(X_train, y_train)

# 保存训练好的模型
with open('2.2.2_model.pkl', 'wb') as model_file:
    pickle.dump(pipeline, model_file)

# 预测并保存结果
y_pred = pipeline.predict(X_test)

results_df = pd.DataFrame(y_pred, columns=['预测结果'])

results_df.to_csv('2.2.2_results.txt', index=False)
```

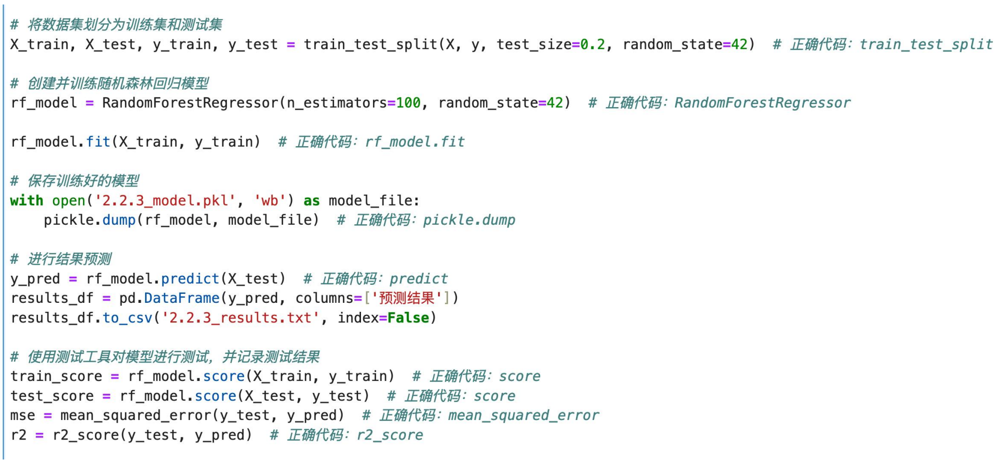


```
# 测试模型
with open('2.2.2_report.txt', 'w') as results_file:
    results_file.write(f'训练集得分: {pipeline.score(X_train, y_train)}\n')
    results_file.write(f'测试集得分: {pipeline.score(X_test, y_test)}\n')

# 训练一个随机森林回归模型作为替代模型

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

rf_model.fit(X_train, y_train)

# 使用随机森林模型进行预测
y_pred_rf = rf_model.predict(X_test)

# 保存新的结果
results_rf_df = pd.DataFrame(y_pred_rf, columns=['预测结果'])
results_rf_df.to_csv('2.2.2_results_rf.txt', index=False)

# 测试模型并保存得分
with open('2.2.2_report_rf.txt', 'w') as results_rf_file:
    results_rf_file.write(f'训练集得分: {rf_model.score(X_train, y_train)}\n')
    results_rf_file.write(f'测试集得分: {rf_model.score(X_test, y_test)}\n')
```

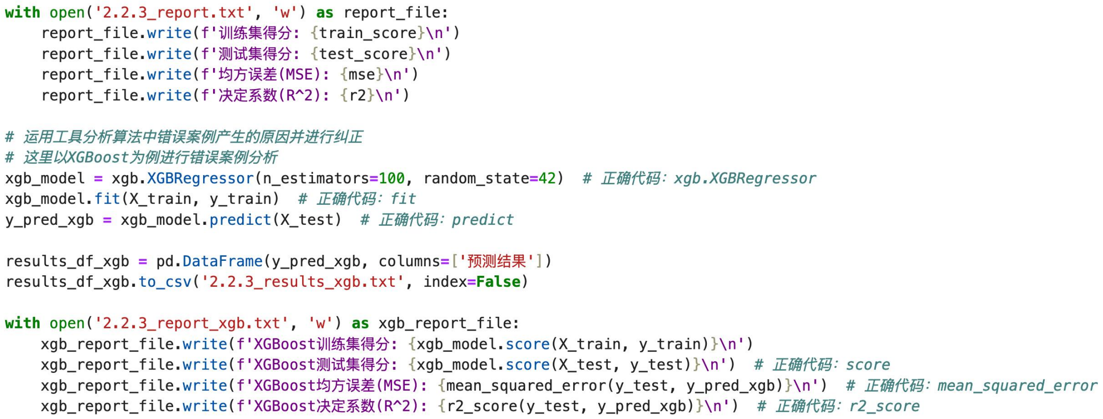
2.2.3



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
import pickle
from sklearn.metrics import mean_squared_error, r2_score
import xgboost as xgb

# 加载数据集

df = pd.read_csv('fitness analysis.csv')  # 正确代码：pd.read_csv

# 显示前五行数据
print(df.head())  # 正确代码：df.head()

# 去除所有字符串字段的前后空格
df = df.applymap(lambda x: x.strip() if isinstance(x, str) else x)

# 检查和清理列名
df.columns = df.columns.str.strip()

# 选择相关特征进行建模

X = pd.get_dummies(X)  # 正确代码：pd.get_dummies，将分类变量转为数值变量

# 将年龄段转为数值变量
y = df['Your age'].apply(lambda x: int(x.split(' ')[0]))  # 正确代码：apply
```

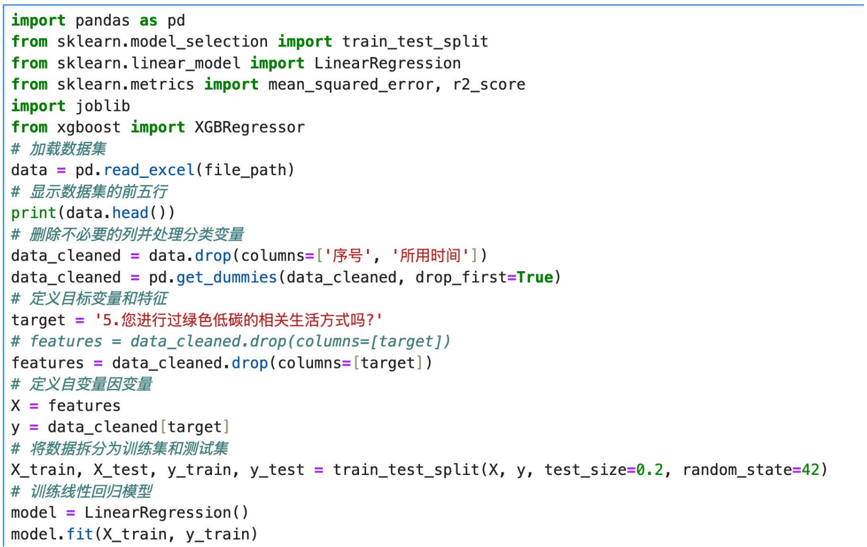


```python
# 将数据集划分为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  # 正确代码：train_test_split

# 创建并训练随机森林回归模型
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)  # 正确代码：RandomForestRegressor

rf_model.fit(X_train, y_train)  # 正确代码：rf_model.fit

# 保存训练好的模型
with open('2.2.3_model.pkl', 'wb') as model_file:
    pickle.dump(rf_model, model_file)  # 正确代码：pickle.dump

# 进行结果预测
y_pred = rf_model.predict(X_test)  # 正确代码：predict
results_df = pd.DataFrame(y_pred, columns=['预测结果'])
results_df.to_csv('2.2.3_results.txt', index=False)

# 使用测试工具对模型进行测试，并记录测试结果
train_score = rf_model.score(X_train, y_train)  # 正确代码：score
test_score = rf_model.score(X_test, y_test)  # 正确代码：score
mse = mean_squared_error(y_test, y_pred)  # 正确代码：mean_squared_error
r2 = r2_score(y_test, y_pred)  # 正确代码：r2_score
```

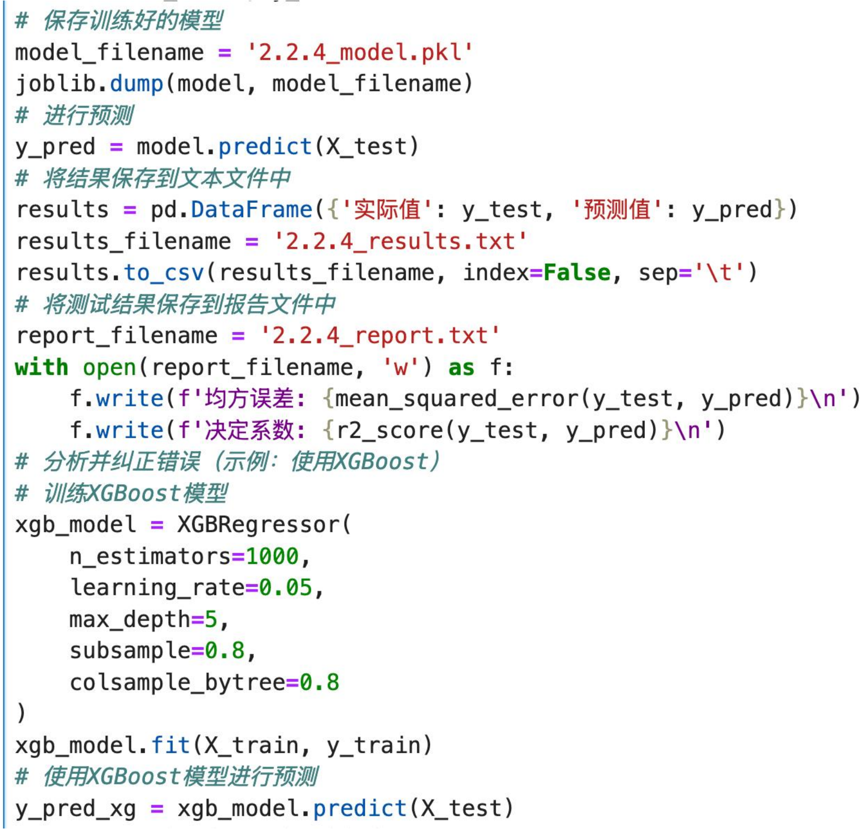


```python
with open('2.2.3_report.txt', 'w') as report_file:
    report_file.write(f'训练集得分：{train_score}\n')
    report_file.write(f'测试集得分：{test_score}\n')
    report_file.write(f'均方误差(MSE)：{mse}\n')
    report_file.write(f'决定系数(R^2)：{r2}\n')

# 运用工具分析算法中错误案例产生的原因并进行纠正
# 这里以XGBoost为例进行错误案例分析
xgb_model = xgb.XGBRegressor(n_estimators=100, random_state=42)  # 正确代码：xgb.XGBRegressor
xgb_model.fit(X_train, y_train)  # 正确代码：fit
y_pred_xgb = xgb_model.predict(X_test)  # 正确代码：predict

results_df_xgb = pd.DataFrame(y_pred_xgb, columns=['预测结果'])
results_df_xgb.to_csv('2.2.3_results_xgb.txt', index=False)

with open('2.2.3_report_xgb.txt', 'w') as xgb_report_file:
    xgb_report_file.write(f'XGBoost训练集得分：{xgb_model.score(X_train, y_train)}\n')
    xgb_report_file.write(f'XGBoost测试集得分：{xgb_model.score(X_test, y_test)}\n')  # 正确代码：score
    xgb_report_file.write(f'XGBoost均方误差(MSE)：{mean_squared_error(y_test, y_pred_xgb)}\n')  # 正确代码：mean_squared_error
    xgb_report_file.write(f'XGBoost决定系数(R^2)：{r2_score(y_test, y_pred_xgb)}\n')  # 正确代码：r2_score
```

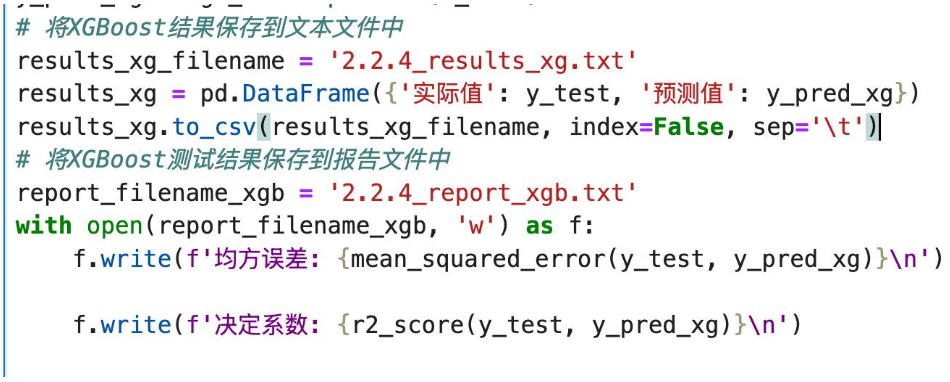
2.2.4



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import joblib
from xgboost import XGBRegressor
# 加载数据集
data = pd.read_excel(file_path)
# 显示数据集的前五行
print(data.head())
# 删除不必要的列并处理分类变量
data_cleaned = data.drop(columns=['序号', '所用时间'])
data_cleaned = pd.get_dummies(data_cleaned, drop_first=True)
# 定义目标变量和特征
target = '5.您进行过绿色低碳的相关生活方式吗?'
# features = data_cleaned.drop(columns=[target])
features = data_cleaned.drop(columns=[target])
# 定义自变量因变量
X = features
y = data_cleaned[target]
# 将数据拆分为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# 训练线性回归模型
model = LinearRegression()
model.fit(X_train, y_train)
```

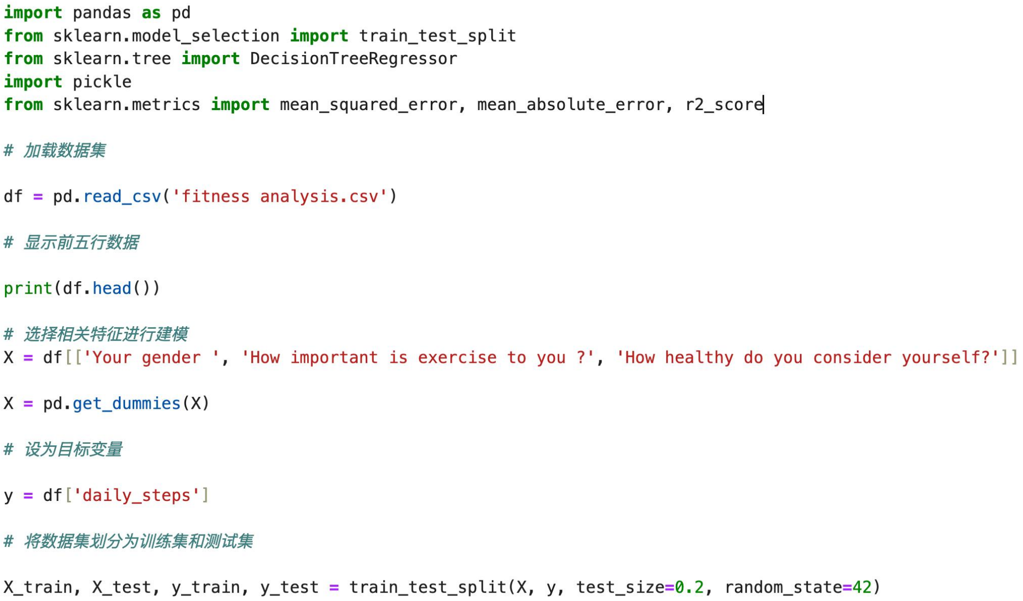


```
# 保存训练好的模型
model_filename = '2.2.4_model.pkl'
joblib.dump(model, model_filename)
# 进行预测
y_pred = model.predict(X_test)
# 将结果保存到文本文件中
results = pd.DataFrame({'实际值': y_test, '预测值': y_pred})
results_filename = '2.2.4_results.txt'
results.to_csv(results_filename, index=False, sep='\t')
# 将测试结果保存到报告文件中
report_filename = '2.2.4_report.txt'
with open(report_filename, 'w') as f:
    f.write(f'均方误差: {mean_squared_error(y_test, y_pred)}\n')
    f.write(f'决定系数: {r2_score(y_test, y_pred)}\n')
# 分析并纠正错误（示例：使用XGBoost）
# 训练XGBoost模型
xgb_model = XGBRegressor(
    n_estimators=1000,
    learning_rate=0.05,
    max_depth=5,
    subsample=0.8,
    colsample_bytree=0.8
)
xgb_model.fit(X_train, y_train)
# 使用XGBoost模型进行预测
y_pred_xg = xgb_model.predict(X_test)
```

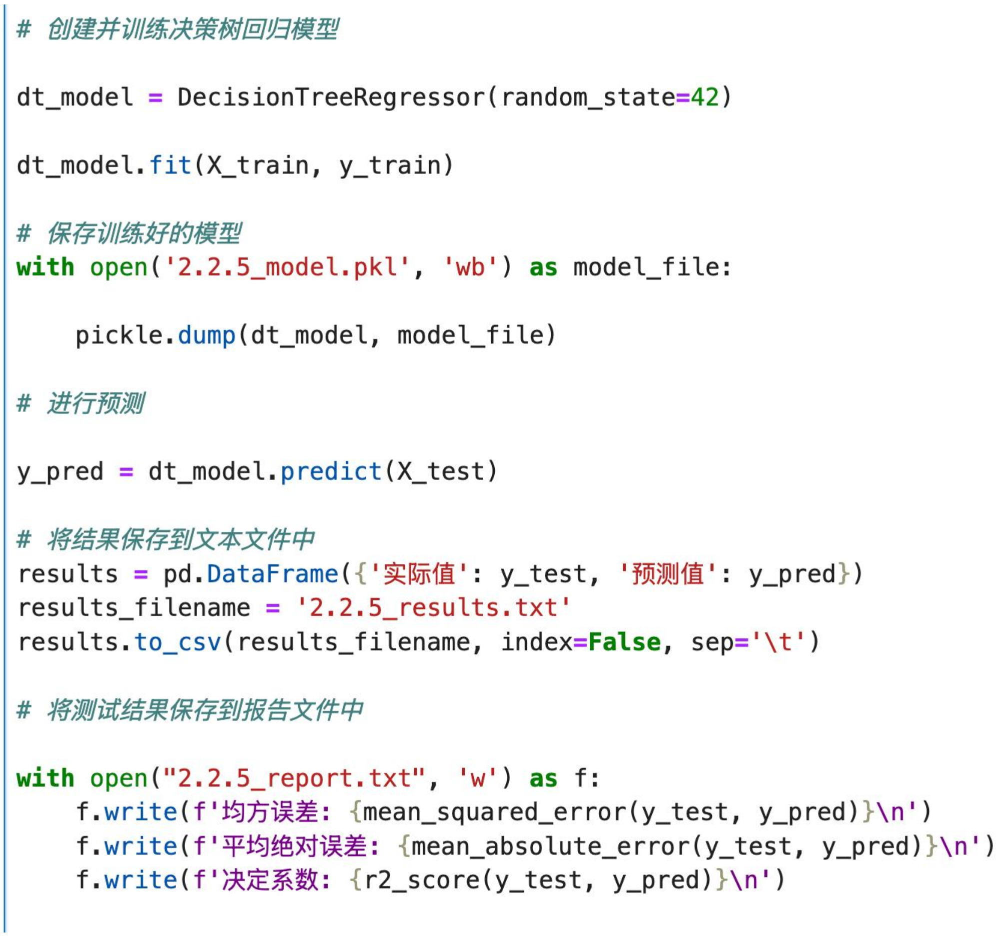


```
# 将XGBoost结果保存到文本文件中
results_xg_filename = '2.2.4_results_xg.txt'
results_xg = pd.DataFrame({'实际值': y_test, '预测值': y_pred_xg})
results_xg.to_csv(results_xg_filename, index=False, sep='\t')
# 将XGBoost测试结果保存到报告文件中
report_filename_xgb = '2.2.4_report_xgb.txt'
with open(report_filename_xgb, 'w') as f:
    f.write(f'均方误差: {mean_squared_error(y_test, y_pred_xg)}\n')

    f.write(f'决定系数: {r2_score(y_test, y_pred_xg)}\n')
```

2.2.5



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
import pickle
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# 加载数据集

df = pd.read_csv('fitness analysis.csv')

# 显示前五行数据

print(df.head())

# 选择相关特征进行建模
X = df[['Your gender ', 'How important is exercise to you ?', 'How healthy do you consider yourself?']]

X = pd.get_dummies(X)

# 设为目标变量

y = df['daily_steps']

# 将数据集划分为训练集和测试集

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```



```
# 创建并训练决策树回归模型

dt_model = DecisionTreeRegressor(random_state=42)

dt_model.fit(X_train, y_train)

# 保存训练好的模型
with open('2.2.5_model.pkl', 'wb') as model_file:

    pickle.dump(dt_model, model_file)

# 进行预测

y_pred = dt_model.predict(X_test)

# 将结果保存到文本文件中
results = pd.DataFrame({'实际值': y_test, '预测值': y_pred})
results_filename = '2.2.5_results.txt'
results.to_csv(results_filename, index=False, sep='\t')

# 将测试结果保存到报告文件中

with open("2.2.5_report.txt", 'w') as f:
    f.write(f'均方误差: {mean_squared_error(y_test, y_pred)}\n')
    f.write(f'平均绝对误差: {mean_absolute_error(y_test, y_pred)}\n')
    f.write(f'决定系数: {r2_score(y_test, y_pred)}\n')
```