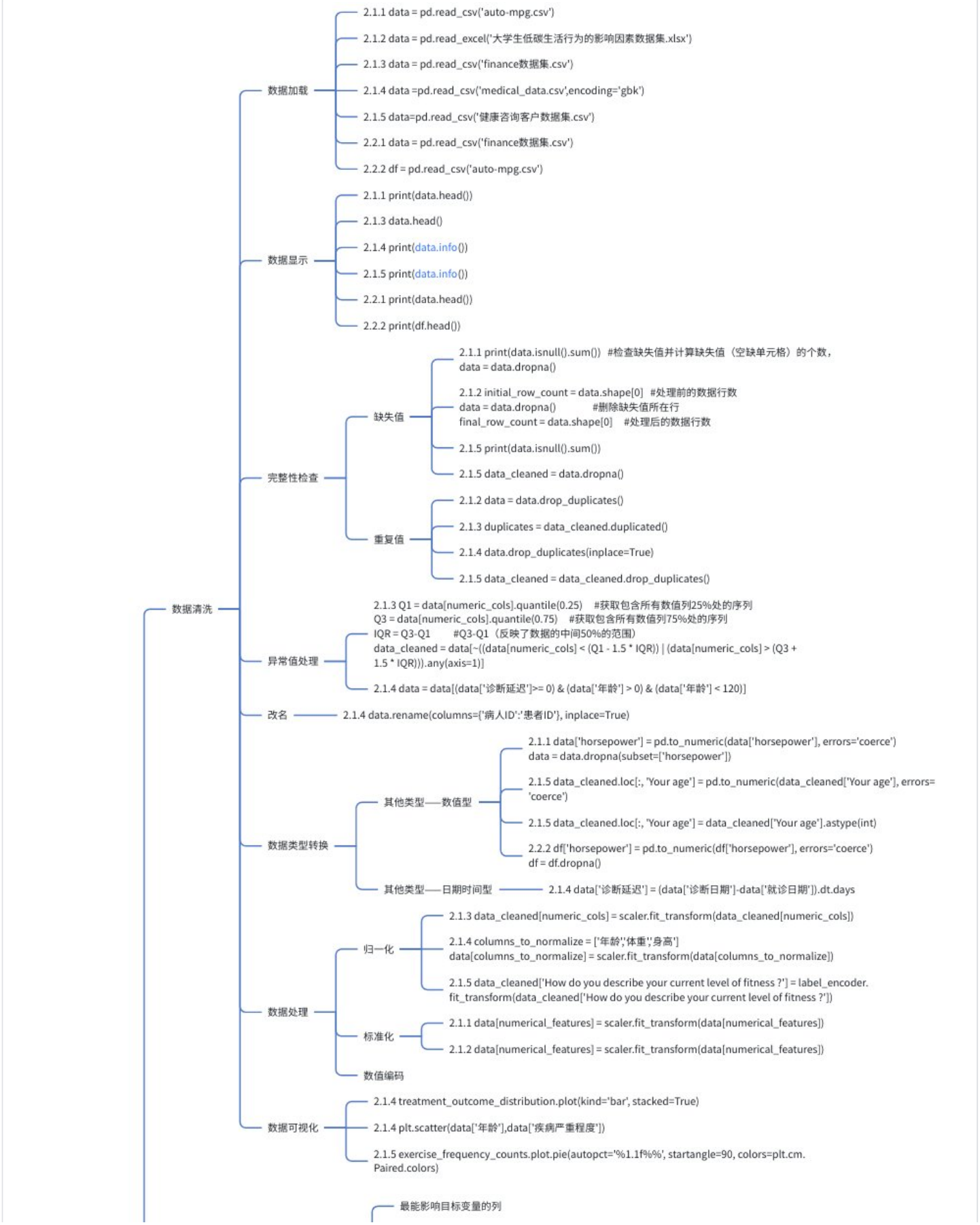
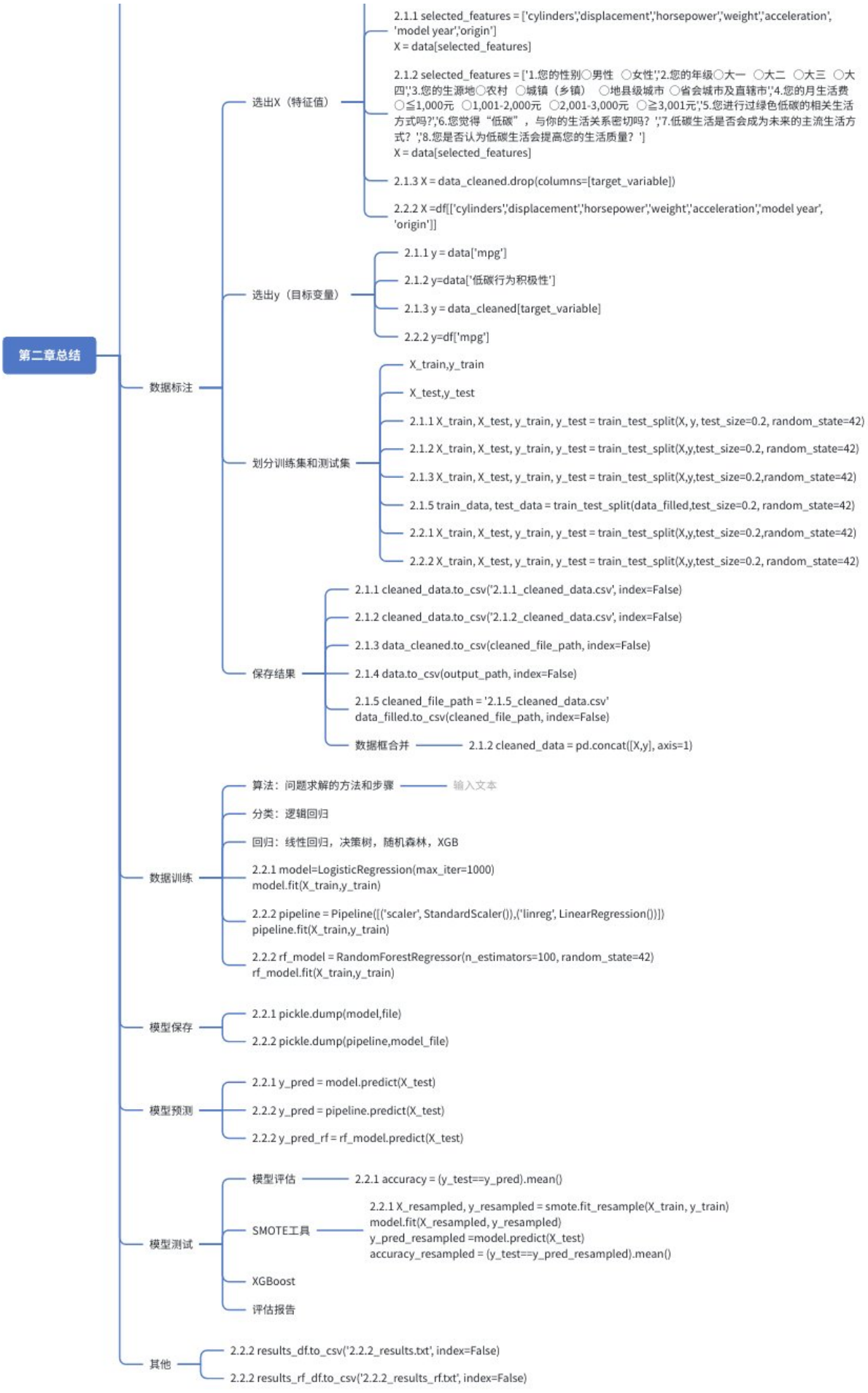
第二章总结


数据清洗
数据加载
2.1.1 data = pd.read_csv('auto-mpg.csv')
2.1.2 data = pd.read_excel('大学生低碳生活行为的影响因素数据集.xlsx')
2.1.3 data = pd.read_csv('finance数据集.csv')
2.1.4 data =pd.read_csv('medical_data.csv',encoding='gbk')
2.1.5 data=pd.read_csv('健康咨询客户数据集.csv')
2.2.1 data = pd.read_csv('finance数据集.csv')
2.2.2 df = pd.read_csv('auto-mpg.csv')
数据显示
2.1.1 print(data.head())
2.1.3 data.head()
2.1.4 print(data.info())
2.1.5 print(data.info())
2.2.1 print(data.head())
2.2.2 print(df.head())
完整性检查
缺失值
2.1.1 print(data.isnull().sum()) #检查缺失值并计算缺失值（空缺单元格）的个数，
data = data.dropna()
2.1.2 initial_row_count = data.shape[0] #处理前的数据行数
data = data.dropna() #删除缺失值所在行
final_row_count = data.shape[0] #处理后的数据行数
2.1.5 print(data.isnull().sum())
2.1.5 data_cleaned = data.dropna()
重复值
2.1.2 data = data.drop_duplicates()
2.1.3 duplicates = data_cleaned.duplicated()
2.1.4 data.drop_duplicates(inplace=True)
2.1.5 data_cleaned = data_cleaned.drop_duplicates()
异常值处理
2.1.3 Q1 = data[numeric_cols].quantile(0.25) #获取包含所有数值列25%处的序列
Q3 = data[numeric_cols].quantile(0.75) #获取包含所有数值列75%处的序列
IQR = Q3-Q1 #Q3-Q1（反映了数据的中间50%的范围）
data_cleaned = data[~((data[numeric_cols] < (Q1 - 1.5 * IQR)) | (data[numeric_cols] > (Q3 + 1.5 * IQR))).any(axis=1)]
2.1.4 data = data[(data['诊断延迟']>= 0) & (data['年龄'] > 0) & (data['年龄'] < 120)]
改名
2.1.4 data.rename(columns={'病人ID':'患者ID'}, inplace=True)
数据类型转换
其他类型——数值型
2.1.1 data['horsepower'] = pd.to_numeric(data['horsepower'], errors='coerce')
data = data.dropna(subset=['horsepower'])
2.1.5 data_cleaned.loc[:, 'Your age'] = pd.to_numeric(data_cleaned['Your age'], errors='coerce')
2.1.5 data_cleaned.loc[:, 'Your age'] = data_cleaned['Your age'].astype(int)
2.2.2 df['horsepower'] = pd.to_numeric(df['horsepower'], errors='coerce')
df = df.dropna()
其他类型——日期时间型
2.1.4 data['诊断延迟'] = (data['诊断日期']-data['就诊日期']).dt.days
数据处理
归一化
2.1.3 data_cleaned[numeric_cols] = scaler.fit_transform(data_cleaned[numeric_cols])
2.1.4 columns_to_normalize = ['年龄','体重','身高']
data[columns_to_normalize] = scaler.fit_transform(data[columns_to_normalize])
2.1.5 data_cleaned['How do you describe your current level of fitness ?'] = label_encoder.fit_transform(data_cleaned['How do you describe your current level of fitness ?'])
标准化
2.1.1 data[numerical_features] = scaler.fit_transform(data[numerical_features])
2.1.2 data[numerical_features] = scaler.fit_transform(data[numerical_features])
数值编码
数据可视化
2.1.4 treatment_outcome_distribution.plot(kind='bar', stacked=True)
2.1.4 plt.scatter(data['年龄'],data['疾病严重程度'])
2.1.5 exercise_frequency_counts.plot.pie(autopct='%1.1f%%', startangle=90, colors=plt.cm.Paired.colors)
最能影响目标变量的列


第二章总结
数据标注
选出X（特征值）
2.1.1 selected_features = ['cylinders','displacement','horsepower','weight','acceleration', 'model year','origin']
X = data[selected_features]
2.1.2 selected_features = ['1.您的性别○男性 ○女性','2.您的年级○大一 ○大二 ○大三 ○大四','3.您的生源地○农村 ○城镇（乡镇） ○地县级城市 ○省会城市及直辖市','4.您的月生活费○≦1,000元 ○1,001-2,000元 ○2,001-3,000元 ○≧3,001元','5.您进行过绿色低碳的相关生活方式吗?','6.您觉得“低碳”，与你的生活关系密切吗？','7.低碳生活是否会成为未来的主流生活方式？','8.您是否认为低碳生活会提高您的生活质量？']
X = data[selected_features]
2.1.3 X = data_cleaned.drop(columns=[target_variable])
2.2.2 X =df[['cylinders','displacement','horsepower','weight','acceleration','model year', 'origin']]
选出y（目标变量）
2.1.1 y = data['mpg']
2.1.2 y=data['低碳行为积极性']
2.1.3 y = data_cleaned[target_variable]
2.2.2 y=df['mpg']
划分训练集和测试集
X_train,y_train
X_test,y_test
2.1.1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
2.1.2 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=42)
2.1.3 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=42)
2.1.5 train_data, test_data = train_test_split(data_filled,test_size=0.2, random_state=42)
2.2.1 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=42)
2.2.2 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=42)
保存结果
2.1.1 cleaned_data.to_csv('2.1.1_cleaned_data.csv', index=False)
2.1.2 cleaned_data.to_csv('2.1.2_cleaned_data.csv', index=False)
2.1.3 data_cleaned.to_csv(cleaned_file_path, index=False)
2.1.4 data.to_csv(output_path, index=False)
2.1.5 cleaned_file_path = '2.1.5_cleaned_data.csv'
data_filled.to_csv(cleaned_file_path, index=False)
数据框合并
2.1.2 cleaned_data = pd.concat([X,y], axis=1)
数据训练
算法：问题求解的方法和步骤
输入文本
分类：逻辑回归
回归：线性回归，决策树，随机森林，XGB
2.2.1 model=LogisticRegression(max_iter=1000)
model.fit(X_train,y_train)
2.2.2 pipeline = Pipeline([('scaler', StandardScaler()),('linreg', LinearRegression())])
pipeline.fit(X_train,y_train)
2.2.2 rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train,y_train)
模型保存
2.2.1 pickle.dump(model,file)
2.2.2 pickle.dump(pipeline,model_file)
模型预测
2.2.1 y_pred = model.predict(X_test)
2.2.2 y_pred = pipeline.predict(X_test)
2.2.2 y_pred_rf = rf_model.predict(X_test)
模型测试
模型评估
2.2.1 accuracy = (y_test==y_pred).mean()
SMOTE工具
2.2.1 X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
model.fit(X_resampled, y_resampled)
y_pred_resampled =model.predict(X_test)
accuracy_resampled = (y_test==y_pred_resampled).mean()
XGBoost
评估报告
其他
2.2.2 results_df.to_csv('2.2.2_results.txt', index=False)
2.2.2 results_rf_df.to_csv('2.2.2_results_rf.txt', index=False)