数据探索与分析

具体代码和数据见Git hub

介绍数据集和任务

数据集:本次实验选取的数据集为Kaggle网站上具有572个赞的数据集。学生学业成功因素数据集。该数据集包含 2,392 名美国高中生的全面信息，详细介绍了他们的人口统计、学习习惯、父母参与、课外活动和学业成绩。

任务: 我计划基于此数据集进行一个分类任务，即预测学生的成绩类别（Grade Class）。目标变量为学生的成绩类别，依据GPA将成绩分为五个类别：A、B、C、D、不及格。观测变量包括家长参与度、人口统计信息、学习习惯、课外活动参与情况等。

观测变量如下

1. Parental Involvement : 家长的支持程度
    - 0: None
    - 1: Low
    - 2: Moderate
    - 3: High
    - 4: Very High
2. Demographic Details 统计的学生详细信息
    - Age(年龄)

    在[15-18]区间
    - Gender(性别)
        - 0 : Male
        - 1 : Female
    - Ethnicity(种族)
        - 0: Caucasian
        - 1: African American
        - 2: Asian
        - 3: Other
    - ParentalEducation(父母受教育程度)
        - 0: None
        - 1: High School
        - 2: Some College
        - 3: Bachelor's
        - 4: Higher
3. Study Habits(学习习惯)

1. StudyTimeWeekly

每周的学习时长（小时），0-20

2. Absences

学年期间的缺勤次数 0-30

3. 辅导

是否在辅导，0 表示NO，1表示是

4. Extracurricular Activities(课外活动)

- Extracurricular

是否参加课外活动，0表示不参与 1表示参与

- Sports

是否参加体育运动，0表示不参与 1表示参与

- Music

是否参加音乐活动，0表示不参加 1表示参加

- Volunteering

是否参加志愿活动 0表示不参加 1表示参加

5. Academic Performance(学习表现)

GPA 平均绩点

2.0-4.0

目标变量如下：

1. Grade Class

依据GPA对学生成绩进行分类

- 0：'A'（GPA >= 3.5）
- 1：'B'（3.0 <= GPA < 3.5）
- 2：'C'（2.5 <= GPA < 3.0）
- 3：'D'（2.0 <= GPA < 2.5）
- 4：'不及格'（GPA < 2.0）

数据预处理

- 数据读取：首先，读取并检查数据中的缺失值，结果显示数据没有缺失值。

```
df = pd.read_csv('./data/Student_performance_data.csv')
df.isnull().sum()
```

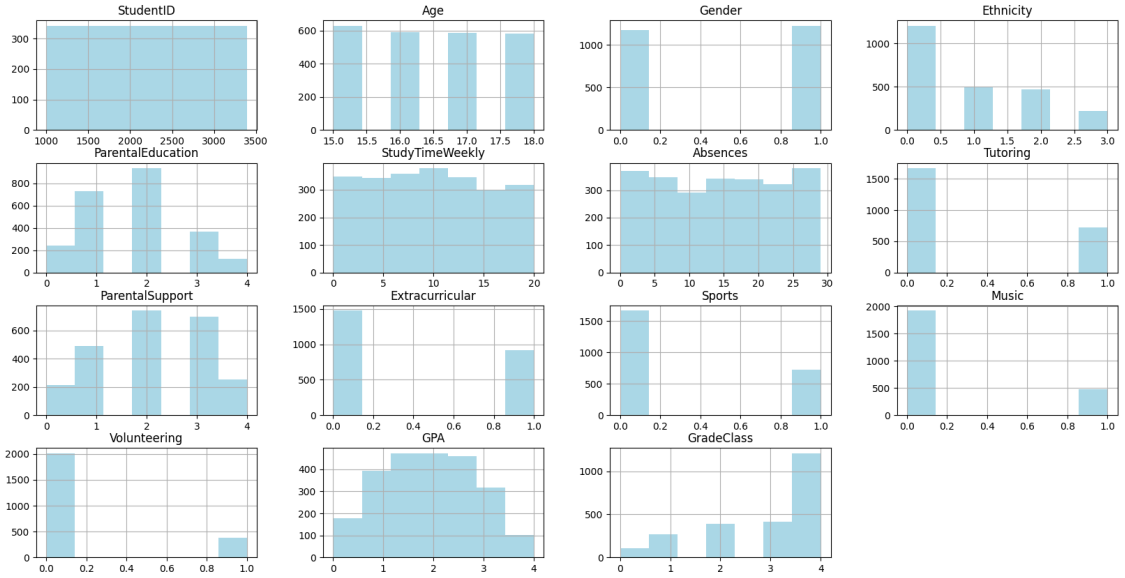
对数据进行读取，然后查看数据中的Nna缺失值进行统计。

StudentID	0
Age	0
Gender	0
Ethnicity	0
ParentalEducation	0
StudyTimeWeekly	0
Absences	0
Tutoring	0
ParentalSupport	0
Extracurricular	0

结果显示，没有缺失值。

```
df.hist(figsize=(20,10),bins=7, color='lightblue')
```

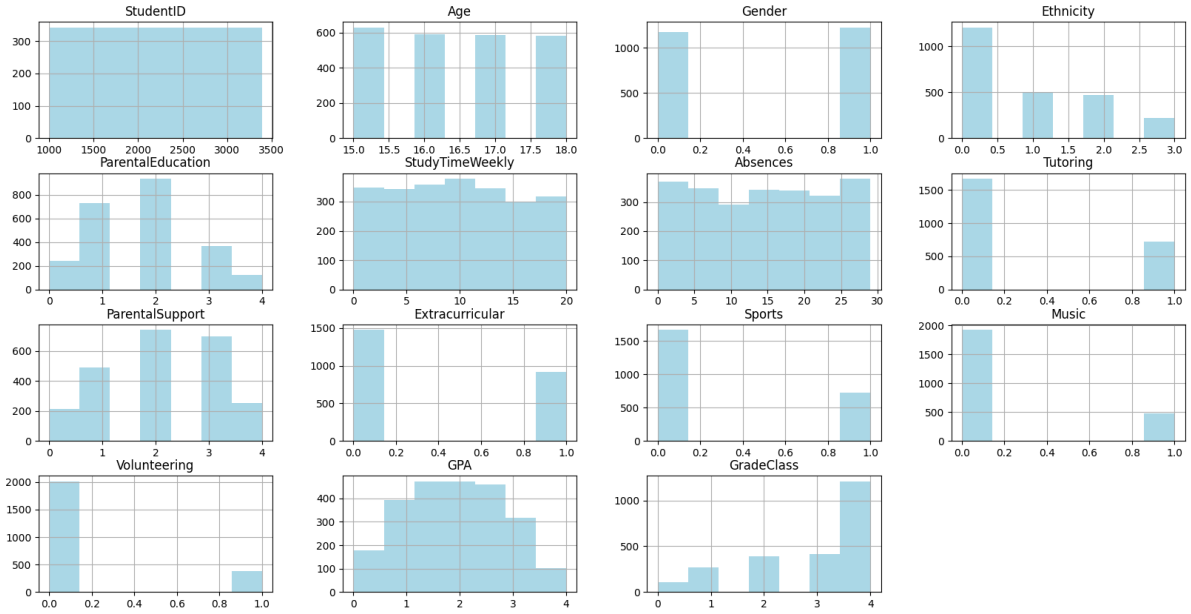
对数据进行可视化 查看异常值



从直方图中可以看出，并没有偏离定义的数值。

数据探索

查看数据



从以上直方图中可以看出

1. 对于`StudentID`仅仅是一个作为统计学生的ID并不存在什么分布。因此它对于目标变量`Grade Class`不存在什么影响。
2. 对于`Age`变量，存在4个不同的年龄，考虑到随着年龄的提升，其心智可能更加成熟，选择把它做为**分类特征。**
3. 对于`Gender`变量，仅仅是男与女的性别之分，对目标变量不存在什么影响
4. 对于`Ethnicity`变量，存在4个种族，且白人偏多，对目标变量也不存在什么影响
5. 对于`ParentalEducation`变量，父母的受教育程度可能对孩子的学习环境产生影响，因此把它作为**分类特征**
6. 对于`StudyTimeWeekly`变量，每周的学习时间对于一个普通人来说，的确会影响其学习成绩。把它作为**分类特征**
7. 对于`Absences`变量，每个月的缺勤次数，代表该学生不太注重课堂，也会影响其学习成绩。把它作为**分类特征。**
8. 对于`Tutoring`变量，对于孩子的辅导，也可能影响其学习成绩。把它作为**分类特征。**
9. 对于`ParentalSupport`变量，父母对于孩子的支持程度，从侧面也可以影响孩子的学习成绩。把它作为**分类特征。**
10. 对于`Extracurricular`变量，孩子参加课外活动导致可能没时间学习也可以影响孩子的学习成绩。把它作为**分类特征。**
11. 对于`Sports`变量，孩子参加体育运动也可以影响孩子的学习成绩。把它作为**分类特征。**
12. 对于`Music`变量，孩子参加音乐活动也可以影响孩子的学习成绩。把它作为**分类特征。**
13. 对于`Volunteering`变量，孩子参加志愿活动也可以影响孩子的学习成绩。把它作为**分类特征。**

从上面分析可以看出，除了学生的性别、种族，其余特征均有可能会影响学生的成绩。

在上面的直方图中可以看出，

1. 数值型变量：`StudyTimeWeekly`，`Absences`，`GPA`
2. 分类变量：`Age`，`Gender`，`Ehnicity`，`ParentalEducation`，`Tutoring`，`ParentalSupport`，`Extracurricular`，`Sports`，`Music`，`Volunteering`，`GradeClass`

```
categoric_columns = ['Age', 'Gender', 'Ethnicity',
'ParentalEducation', 'Tutoring', 'ParentalSupport', 'Extracurricular',
'Sports', 'Music', 'Volunteering', 'GradeClass']
numeric_columns = ['StudyTimeWeekly', 'Absences', 'GPA']
```

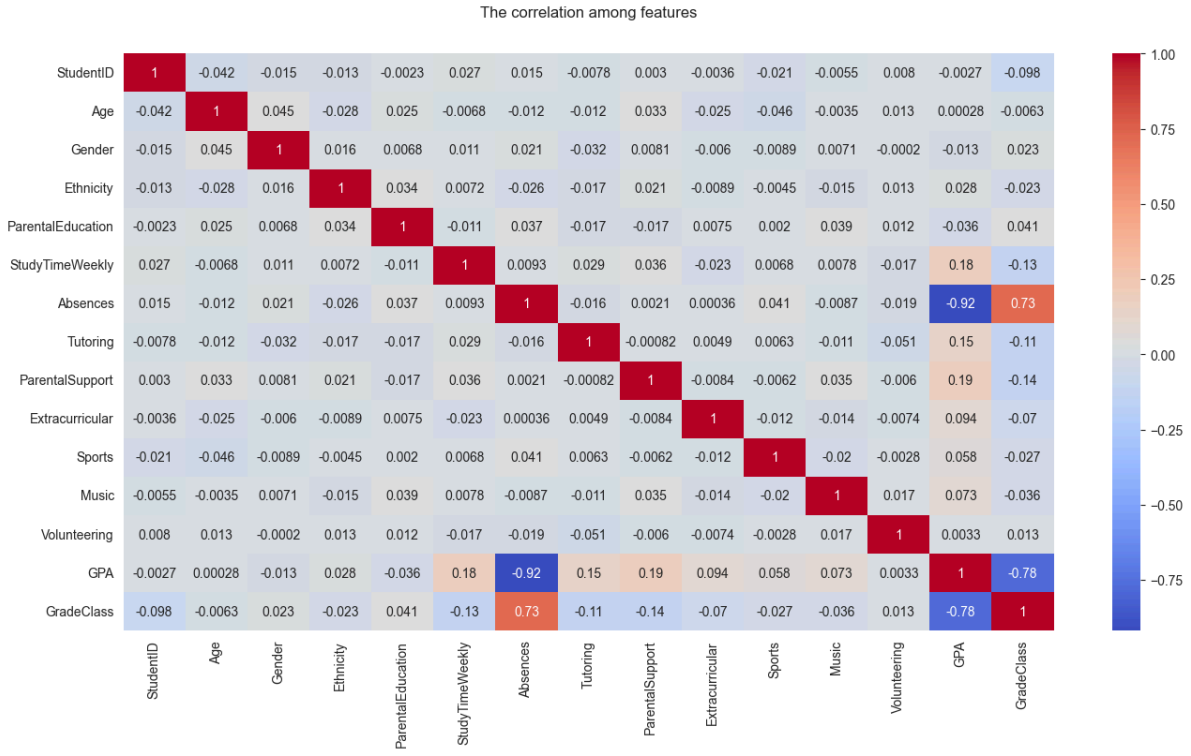
> 将Age、Gender、Ethnicity、ParentalEducation、Tutoring等分类变量进行标签编码，转换为数值型变量

```
df = df.copy()
for column in df[categoric_columns]:
df[column] = label_encoder.fit_transform(df[column])
```

派生变量: 考虑了创建派生变量，例如将StudyTimeWeekly按时间段分为不同等级。

相关性分析

```
#相关性分析
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(16, 8))
sns.heatmap(df.corr(), annot = True, cmap = "coolwarm")
plt.title('The correlation among features',y= 1.05)
plt.show()
```



从热力图可以看出，与`GradeClass`最相关的是`Absences`,其次是`ParentalSupport`,`StudyTimeWeekly`,`Tutoring`,`StudentID`,`Extracurricular`,`ParentaEducation`,`Music`,`Sports`,`Gender`,`Ethnicity`,`Volunteering`, `age`

最为相关的是 1. 缺勤 2. 父母的支持 3. 学习时间 4. 辅导课 5.学生号码 6.课外活动 7. 父母的受教育程度 8.参加音乐活动 9. 参加体育活动 10.性别 11.种族 12.参加志愿活动 13.年龄

其中 `GPA` 平均绩点对于GradeClass没有太大的关系，因为GPA仅仅是反应学生的成绩，而GPA可以对GradeClass产生影响，但是目前的问题是学生的表现对于学生的成绩有那些影响，如果选择GPA，反而会对目标变量产生影响。

`5.学生号码`，这些对成绩没有影响，仅仅是为了来统计学生数量，选择将该进行删除。

`13.年龄` 对于Gradeclass的影响最小

```
df.drop(columns=[ 'GPA', 'StudentID', 'Age'])
```

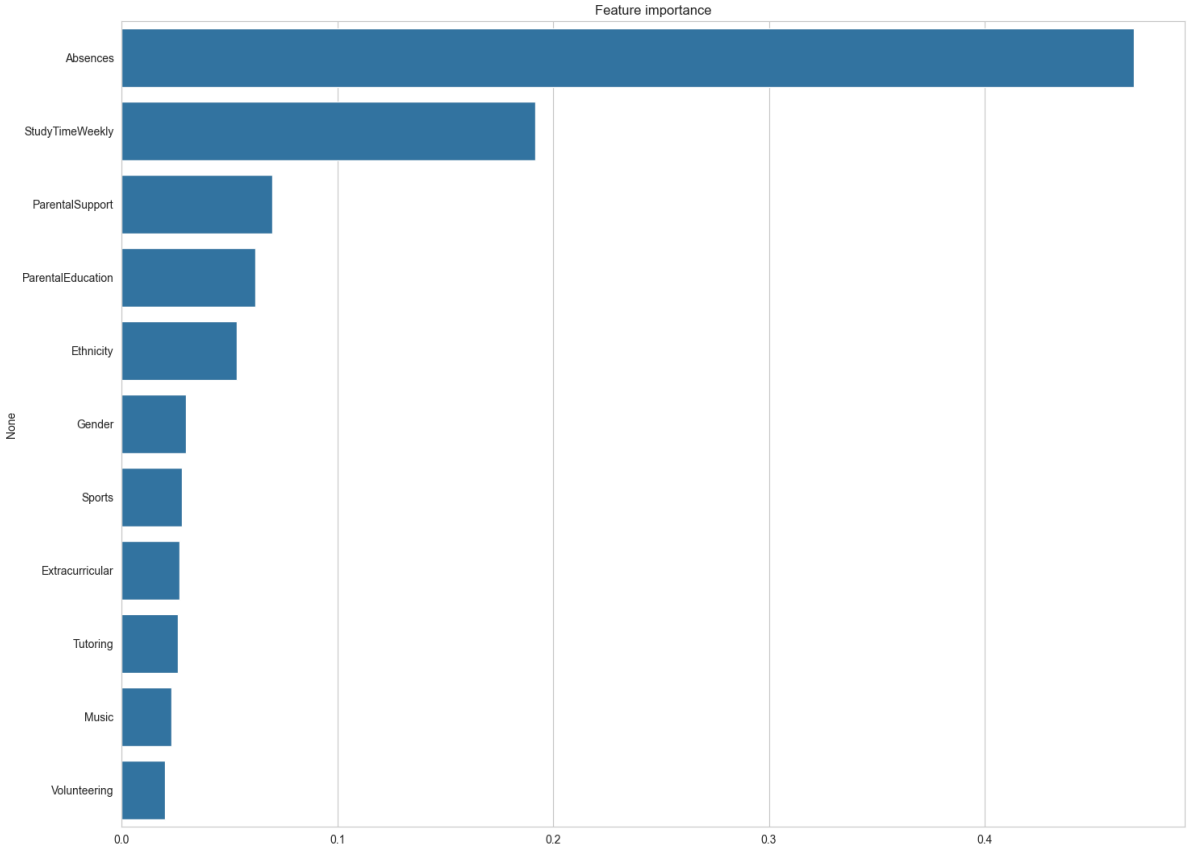
特征工程

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(random_state = 42)
clf = clf.fit(X, y)

fimp = pd.Series(data=clf.feature_importances_,
index=X.columns).sort_values(ascending=False)
plt.figure(figsize=(17,13))
plt.title("Feature importance")
ax = sns.barplot(y=fimp.index, x=fimp.values, orient='h')
```

通过以上代码进行特征重要性分析



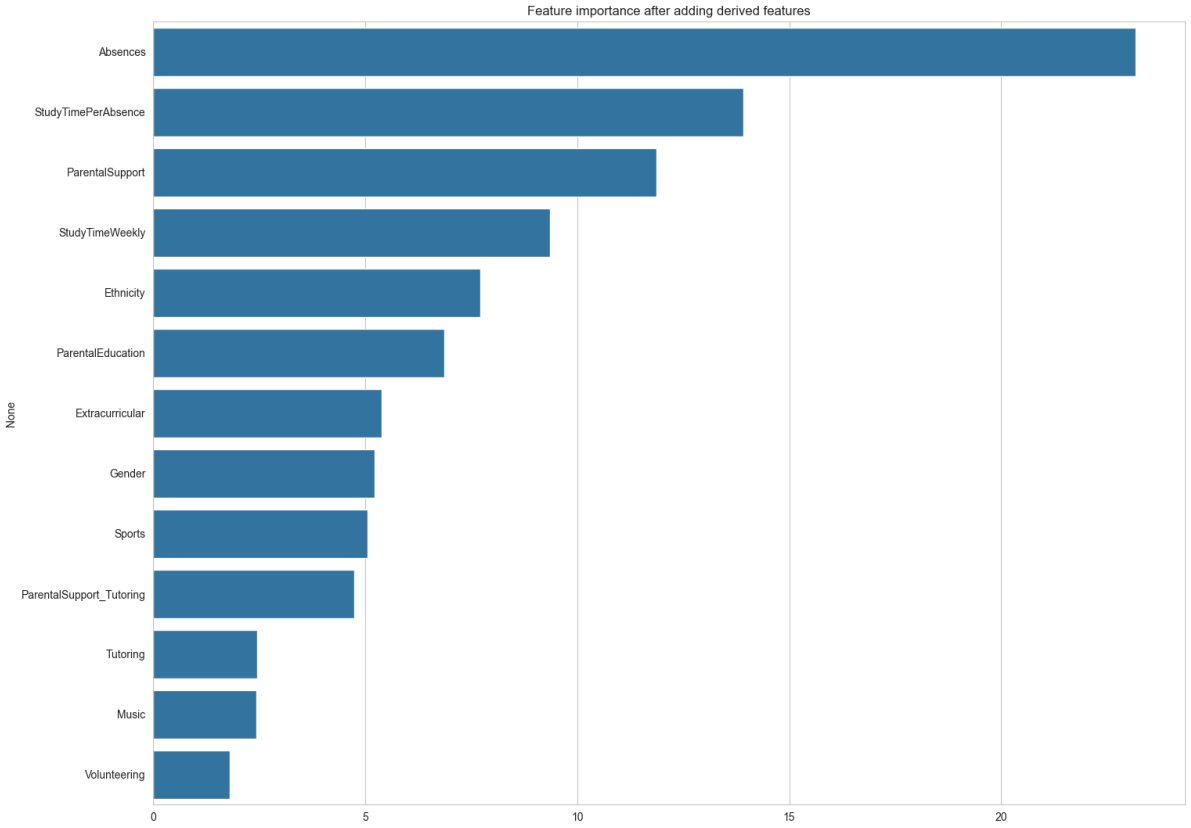
设计派生变量

```python
# 设计派生变量
df['StudyTimePerAbsence'] = df['StudyTimeWeekly'] / (df['Absences'] + 1)  # 避免除以零
df['ParentalSupport_Tutoring'] = df['ParentalSupport'] * df['Tutoring']

# 重新进行相关性分析和特征重要性分析
X = df.drop(columns=['GradeClass', 'GPA', 'StudentID', 'Age'])
clf.fit(X, y)
fimp = pd.Series(data=clf.feature_importances_, index=X.columns).sort_values(ascending=False)

plt.figure(figsize=(17, 13))
plt.title("Feature importance after adding derived features")
sns.barplot(y=fimp.index, x=fimp.values, orient='h')
plt.show()
```

> **StudyTimePerAbsence**: 这是每周学习时间除以缺勤次数的派生变量。此变量的目的是将学习时间与缺勤次数结合起来，避免仅考虑单一因素可能导致的偏差。通过加入1来避免除零错误。
>
> **ParentalSupport_Tutoring**: 这是家长支持和辅导的乘积，目的是评估家长支持和辅导的协同效应。



通过对比两张图新的派生变量`StudyTimePerAbsence`的特征重要性排名第二，超过了原始的`StudyTimeWeekly`，这表明结合学习时间和缺勤次数后得出的新特征对于预测学生成绩更具解释力。且特征重要性的排序发生了一些变化，`StudyTimeWeekly`的重要性下降，而`StudyTimePerAbsence`和`ParentalSupport_Tutoring`的重要性上升，说明派生特征捕捉到了原始特征未能充分表达的信息。通过引入派生变量，模型可能变得更加复杂，但同时也可能提高了模型的预测性能，因为这些派生变量能够更好地解释学生成绩的波动。

划分数据集

```
from sklearn.model_selection import  train_test_split

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

构建模型

```
import pandas as pd  # 导入Pandas库，用于数据处理
from sklearn.model_selection import train_test_split  # 导入train_test_split函数，用于将数据集划分为训练集和测试集
from sklearn.linear_model import LogisticRegression  # 导入LogisticRegression模型
from sklearn.neighbors import KNeighborsClassifier  # 导入KNeighborsClassifier模型
from sklearn.svm import SVC  # 导入SVC模型
from sklearn.tree import DecisionTreeClassifier  # 导入DecisionTreeClassifier模型
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier  # 导入集成模型
from sklearn.naive_bayes import GaussianNB  # 导入GaussianNB模型
from xgboost import XGBClassifier  # 导入XGBClassifier模型
from lightgbm import LGBMClassifier  # 导入LGBMClassifier模型
from catboost import CatBoostClassifier  # 导入CatBoostClassifier模型
import plotly.express as px  # 导入Plotly库，用于数据可视化


# 将数据集划分为训练集和测试集（70%训练，30%测试）
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 定义一个字典，其中包含所有要训练和评估的分类模型
classification_models = {
    "Logistic Regression": LogisticRegression(),  # 逻辑回归
    "K-Nearest Neighbors": KNeighborsClassifier(),  # k近邻分类器
    "Support Vector Machine": SVC(),  # 支持向量机
    "Decision Tree": DecisionTreeClassifier(),  # 决策树
    "Random Forest": RandomForestClassifier(),  # 随机森林
    "Gradient Boosting": GradientBoostingClassifier(),  # 梯度提升
    "AdaBoost": AdaBoostClassifier(),  # 自适应增强
    "Gaussian Naive Bayes": GaussianNB(),  # 高斯朴素贝叶斯
    "XGBoost": XGBClassifier(),  # XGBoost
    "CatBoost": CatBoostClassifier(silent=True),  # CatBoost（静默模式）
}

model_names = []  # 用于存储模型名称的列表
accuracies = []  # 用于存储模型准确率的列表

# 训练并评估每个模型
for name, clf in classification_models.items():
    clf.fit(X_train, y_train)  # 在训练数据上训练模型
    score = clf.score(X_test, y_test)  # 在测试数据上评估模型准确率
    model_names.append(name)  # 将模型名称添加到列表中
```

```
    accuracies.append(score)  # 将模型准确率添加到列表中
    print(f"{name} accuracy: {score:.2f}")  # 输出模型名称及其准确率

# 创建一个包含模型名称和对应准确率的数据框
df_models = pd.DataFrame({'Model': model_names, 'Accuracy': accuracies})

# 使用Plotly绘制模型准确率的柱状图
fig = px.bar(df_models, x='Model', y='Accuracy', title='Model Accuracies')
 # 设置x轴为模型名称，y轴为准确率，并设置标题
fig.show()  # 显示图表

```

之后构建模型 构建了一个包含模型的字典

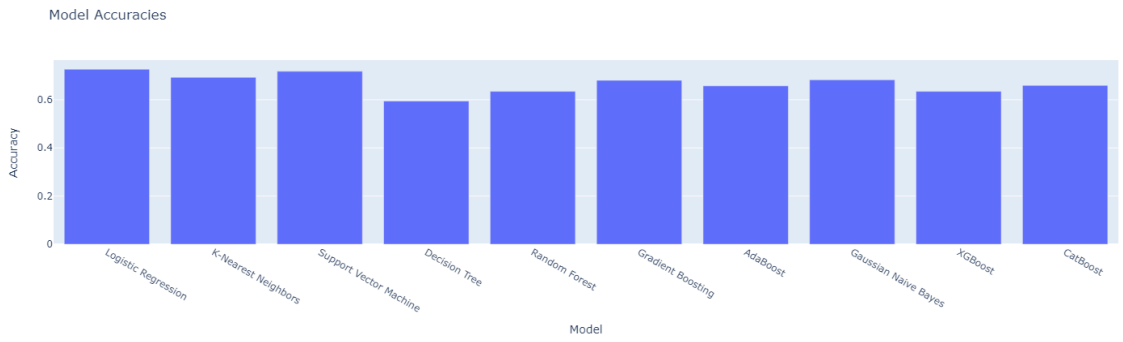
包括 Logistic_Regression 逻辑回归、K-Nearest Neighbors K近邻算法、Support Vector Machine 支持向量机、Decision Tree 决策树、Random Forest 随机森林、Gradient Boosting 梯度提升、AdaBoost 迭代算法、Gaussian Naive Bayes 高斯朴素贝叶斯、XGBoost 极端梯度提升、CatBoost 分类提升

然后对模型进行训练，构建出一个dataFrame来画出柱状图来看出那一个模型的分类精度最高。

评估模型

最终展示结果如下

> Logistic Regression accuracy: 0.73
> K-Nearest Neighbors accuracy: 0.69
> Support Vector Machine accuracy: 0.72
> Decision Tree accuracy: 0.58
> Random Forest accuracy: 0.64
> Gradient Boosting accuracy: 0.68
> AdaBoost accuracy: 0.66
> Gaussian Naive Bayes accuracy: 0.68
> XGBoost accuracy: 0.63
> CatBoost accuracy: 0.66



可以看出SVC 支持向量机的模型性能最好。

测试模型

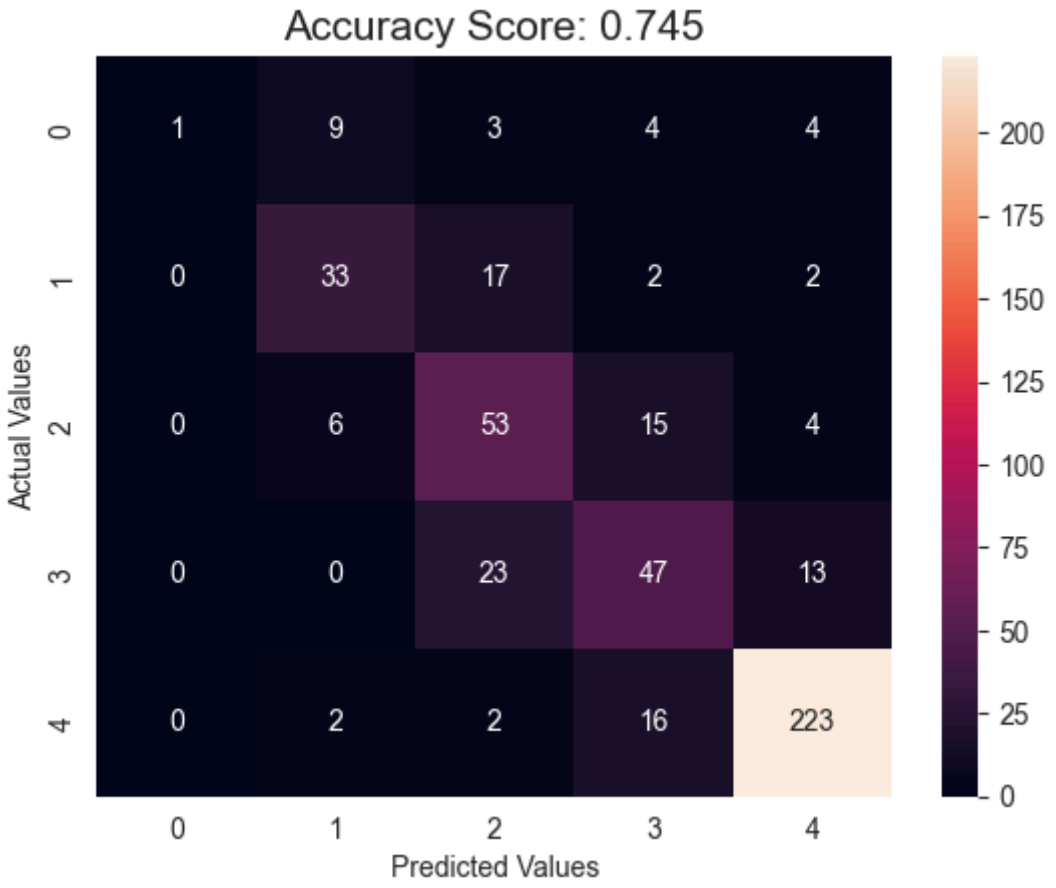
之后使用测试集来测试模型的泛化性能。

```
from sklearn.metrics import confusion_matrix, accuracy_score
# 初始化模型
best_model.fit(X_train, y_train)
model_score = best_model.score(X_test, y_test)
```

```
y_pred = best_model.predict(X_test)

# 绘制混淆矩阵
score = round(accuracy_score(y_test, y_pred), 3)
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt=".0f")
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.title('Accuracy Score: {0}'.format(score), size=15)
plt.show()
```

最终的混淆矩阵图



从上图可以看出，模型的性能精度达0.745.

实验总结

1. 数据集介绍和任务描述

数据集：本次实验选取了一个包含2,392名美国高中生信息的数据集，涵盖人口统计、学习习惯、父母参与、课外活动和学业成绩。

任务：基于该数据集进行分类任务，目标是预测学生的成绩类别（Grade Class），依据GPA将成绩分为五个类别：A、B、C、D、不及格。

2. 数据探索与预处理

数据探索:

- 数据中无缺失值，经过可视化检查未发现异常值。
- 将观测变量分为数值型变量和分类变量。

数据预处理:

- 对分类变量进行标签编码。
- 创建两个派生变量:
  - **StudyTimePerAbsence**: 每周学习时间除以缺勤次数，避免除零。
  - **ParentalSupport_Tutoring**: 家长支持与辅导的乘积。

相关性分析:

- 通过相关性热力图确定了与成绩类别最相关的特征，包括缺勤次数、学习时间、家长支持等。

3. 模型选择与评估

模型训练:

- 选择了多种分类算法，包括逻辑回归、支持向量机、随机森林等。
- 使用训练集和测试集对模型进行训练和评估。

模型评估:

- 支持向量机的准确率最高，为74.5%。
- 通过混淆矩阵进一步验证模型的性能。

4. 结果

- 派生变量提升了模型的预测性能。
- 支持向量机在本次任务中表现最佳，适合用于学生成绩分类预测。