# modeling-student-performance

October 12, 2024

## 1 Modeling Student Performance: Using Multiple Linear Regression to Predict Exam Scores

Our selected dataset is Student Performance Factors.

All of the scripts and data for this project can be found on our Git Repository.

[252]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from plotnine import *
from matplotlib import gridspec

from formatting.plot_settings import colors

import pandas as pd
import numpy as np

import itertools
from itertools import combinations

import statsmodels.api as sm
import statsmodels.formula.api as smf
from scipy import stats

from statsmodels.stats.anova import anova_lm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.graphics.regressionplots import influence_plot
from statsmodels.stats.outliers_influence import OLSInfluence

from sklearn.linear_model import Lasso
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

%config InlineBackend.figure_format = 'retina'
```

### 1.1 Student Performance Factors Dataset Overview

The "Student Performance Factors" dataset contains 19 variables that may influence students' exam scores. It is designed to help researchers analyze the potential impact of these factors on student performance. The dataset includes information such as study time (`Hours_Studied`), attendance (`Attendance`), parental involvement (`Parental_Involvement`), access to resources (`Access_to_Resources`), and participation in extracurricular activities (`Extracurricular_Activities`). Additionally, it covers socioeconomic and background data such as family income (`Family_Income`), motivation level (`Motivation_Level`), tutoring sessions (`Tutoring_Sessions`), school type (`School_Type`), sleep hours (`Sleep_Hours`), and parental education level (`Parental_Education_Level`).

#### 1.1.1 Purpose and Applications

Researchers can use this dataset to build regression models for predicting exam scores (`Exam_Score`) and to identify significant factors affecting student academic performance. The dataset's potential applications include:

- Supporting educational decision-making
- Assisting in policy formulation
- Optimizing the allocation of educational resources

Ultimately, the goal is to better understand and improve the key factors influencing student success, thereby enabling educators and policymakers to provide more targeted support.

#### 1.1.2 Variable Descriptions

1. **Hours_Studied**: Daily study hours.

2. **Attendance**: Attendance rate (percentage).

3. **Parental_Involvement**: Parent involvement (Low, Medium, High).

4. **Access_to_Resources**: Resource accessibility (Low, Medium, High).

5. **Extracurricular_Activities**: Participation in extracurricular activities.

6. **Sleep_Hours**: Daily sleep hours.

7. **Previous_Scores**: Prior exam scores.

8. **Motivation_Level**: Motivation level (Low, Medium, High).

9. **Internet_Access**: Internet access.

10. **Tutoring_Sessions**: Number of tutoring sessions.

11. **Family_Income**: Family income level (Low, Medium, High).

12. **Teacher_Quality**: Teacher quality (Low, Medium, High).

13. **School_Type**: School type (Public or Private).

14. **Peer_Influence**: Peer influence (Positive, Neutral, Negative).

15. **Physical_Activity**: Weekly physical activity hours.

16. **Learning_Disabilities**: Presence of learning disabilities.

17. **Parental_Education_Level**: Parents' education level (High School, College, Postgraduate).

18. **Distance_from_Home**: Distance from home to school (Near, Moderate, Far).

19. **Gender**: Student gender (Male or Female).

20. **Exam_Score**: Academic performance indicator (exam score).

#### 1.1.3 Research Questions

1. **Which factors are the most significant predictors of students' exam scores?**
2. **How do parental involvement, access to resources, and socioeconomic factors impact student performance?**
3. **What is the combined effect of study habits, peer influence, and tutoring sessions on exam outcomes?**
4. **Does school type or teacher quality significantly influence exam scores?**

#### 1.1.4 Methods Used in the Analysis

1. **Exploratory Data Analysis**: Initial analysis includes correlation calculations to understand the relationships between predictors and exam scores.

2. **Multiple Linear Regression**: Regression models are built using significant predictors such as attendance, hours studied, and previous scores. The model is validated using metrics like adjusted R-squared, p-values, and F-statistics.

3. **ANOVA (Types I, II, and III)**: Variance analysis is conducted to understand the contribution of each predictor to the total variance in exam scores.

4. **Model Evaluation**: The model's prediction capability is visualized through plots of actual vs. predicted exam scores, residuals distribution, and summary statistics.

[253]:
```
students = pd.read_csv("data/StudentPerformanceFactors.csv")
```

## 1.2 EDA

### 1.2.1 First sniff of the data

[254]:
```
print(f"Our dataset has {students.shape[1]} variables and {students.shape[0]}␣
↪records")
```

```
Our dataset has 20 variables and 6607 records
```

[255]:
```
students.head()
```

[255]:
```
   Hours_Studied  Attendance Parental_Involvement Access_to_Resources  \
0             23          84                  Low                High
1             19          64                  Low              Medium
2             24          98               Medium              Medium
3             29          89                  Low              Medium
4             19          92               Medium              Medium

  Extracurricular_Activities  Sleep_Hours  Previous_Scores Motivation_Level  \
0                         No            7               73              Low
1                         No            8               59              Low
2                        Yes            7               91           Medium
3                        Yes            8               98           Medium
4                        Yes            6               65           Medium

  Internet_Access  Tutoring_Sessions Family_Income Teacher_Quality  \
0             Yes                  0           Low          Medium
1             Yes                  2        Medium          Medium
2             Yes                  2        Medium          Medium
3             Yes                  1        Medium          Medium
4             Yes                  3        Medium            High

  School_Type Peer_Influence  Physical_Activity Learning_Disabilities  \
0      Public       Positive                  3                    No
1      Public       Negative                  4                    No
2      Public        Neutral                  4                    No
3      Public       Negative                  4                    No
4      Public        Neutral                  4                    No

  Parental_Education_Level Distance_from_Home  Gender  Exam_Score
0              High School               Near    Male          67
1                  College           Moderate  Female          61
2             Postgraduate               Near    Male          74
3              High School           Moderate    Male          71
4                  College               Near  Female          70
```

[256]:
```
students.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 6607 entries, 0 to 6606
Data columns (total 20 columns):
 #   Column                      Non-Null Count  Dtype 
---  ------                      --------------  ----- 
 0   Hours_Studied               6607 non-null   int64 
 1   Attendance                  6607 non-null   int64 
 2   Parental_Involvement        6607 non-null   object
 3   Access_to_Resources         6607 non-null   object
 4   Extracurricular_Activities  6607 non-null   object
 5   Sleep_Hours                 6607 non-null   int64 
 6   Previous_Scores             6607 non-null   int64 
 7   Motivation_Level            6607 non-null   object
 8   Internet_Access             6607 non-null   object
 9   Tutoring_Sessions           6607 non-null   int64 
 10  Family_Income               6607 non-null   object
 11  Teacher_Quality             6529 non-null   object
 12  School_Type                 6607 non-null   object
 13  Peer_Influence              6607 non-null   object
 14  Physical_Activity           6607 non-null   int64 
 15  Learning_Disabilities       6607 non-null   object
 16  Parental_Education_Level    6517 non-null   object
 17  Distance_from_Home          6540 non-null   object
 18  Gender                      6607 non-null   object
 19  Exam_Score                  6607 non-null   int64 
dtypes: int64(7), object(13)
memory usage: 1.0+ MB
```

### 1.2.2 Variable types

[257]:
```
object_cols = []
numeric_cols = []
for colname in students.columns:
    type = students[colname].dtype
    if type == "int64":
        numeric_cols.append(colname)
    elif type == "object":
        object_cols.append(colname)
```

[258]:
```
print(f"Object columns: {object_cols}")
print(f"Numeric columns: {numeric_cols}")
```

```
Object columns: ['Parental_Involvement', 'Access_to_Resources',
'Extracurricular_Activities', 'Motivation_Level', 'Internet_Access',
'Family_Income', 'Teacher_Quality', 'School_Type', 'Peer_Influence',
'Learning_Disabilities', 'Parental_Education_Level', 'Distance_from_Home',
'Gender']
Numeric columns: ['Hours_Studied', 'Attendance', 'Sleep_Hours',
'Previous_Scores', 'Tutoring_Sessions', 'Physical_Activity', 'Exam_Score']
```

[259]:
```
unique_values = students.nunique()
unique_values
```

[259]:
```
Hours_Studied                 41
Attendance                    41
Parental_Involvement           3
Access_to_Resources            3
Extracurricular_Activities     2
Sleep_Hours                    7
Previous_Scores               51
Motivation_Level               3
Internet_Access                2
Tutoring_Sessions              9
Family_Income                  3
Teacher_Quality                3
School_Type                    2
Peer_Influence                 3
Physical_Activity              7
Learning_Disabilities          2
Parental_Education_Level       3
Distance_from_Home             3
Gender                         2
Exam_Score                    45
dtype: int64
```

Let's check that all of our object variables can be turned into categorical:

[260]:
```
students[object_cols].nunique()
```

[260]:
```
Parental_Involvement          3
Access_to_Resources           3
Extracurricular_Activities    2
Motivation_Level              3
Internet_Access               2
Family_Income                 3
Teacher_Quality               3
School_Type                   2
Peer_Influence                3
Learning_Disabilities         2
Parental_Education_Level      3
Distance_from_Home            3
Gender                        2
dtype: int64
```

Since all of them are not actually continuous or discrete values with a lot of unique values, let's go ahead and turn them info categorical variables.

[261]:
```
students[object_cols] = students[object_cols].astype('category')
```

Let's check our resulting column types and redefine our lists.

[262]:
```
students.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6607 entries, 0 to 6606
Data columns (total 20 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Hours_Studied               6607 non-null   int64
 1   Attendance                  6607 non-null   int64
 2   Parental_Involvement        6607 non-null   category
 3   Access_to_Resources         6607 non-null   category
 4   Extracurricular_Activities  6607 non-null   category
 5   Sleep_Hours                 6607 non-null   int64
 6   Previous_Scores             6607 non-null   int64
 7   Motivation_Level            6607 non-null   category
 8   Internet_Access             6607 non-null   category
 9   Tutoring_Sessions           6607 non-null   int64
 10  Family_Income               6607 non-null   category
 11  Teacher_Quality             6529 non-null   category
 12  School_Type                 6607 non-null   category
 13  Peer_Influence              6607 non-null   category
 14  Physical_Activity           6607 non-null   int64
 15  Learning_Disabilities       6607 non-null   category
 16  Parental_Education_Level    6517 non-null   category
 17  Distance_from_Home          6540 non-null   category
 18  Gender                      6607 non-null   category
 19  Exam_Score                  6607 non-null   int64
dtypes: category(13), int64(7)
memory usage: 447.0 KB
```

[263]:
```
categorical_variables = []
numerical_variables = []
for colname in students.columns:
    type = students[colname].dtype
    if type == "int64":
        numerical_variables.append(colname)
    elif type == "category":
        categorical_variables.append(colname)
```

[264]:
```
print(f"Categorical variables: {categorical_variables}")
print(f"Numerical variables: {numerical_variables}")
```

```
Categorical variables: ['Parental_Involvement', 'Access_to_Resources',
'Extracurricular_Activities', 'Motivation_Level', 'Internet_Access',
'Family_Income', 'Teacher_Quality', 'School_Type', 'Peer_Influence',
'Learning_Disabilities', 'Parental_Education_Level', 'Distance_from_Home',
'Gender']
```

```
Numerical variables: ['Hours_Studied', 'Attendance', 'Sleep_Hours',
'Previous_Scores', 'Tutoring_Sessions', 'Physical_Activity', 'Exam_Score']
```

### 1.2.3 Missing values

Let's take a closer look at the missing values in our dataset.

[265]:
```
students.isnull().sum()
```

[265]:
```
Hours_Studied                  0
Attendance                     0
Parental_Involvement           0
Access_to_Resources            0
Extracurricular_Activities     0
Sleep_Hours                    0
Previous_Scores                0
Motivation_Level               0
Internet_Access                0
Tutoring_Sessions              0
Family_Income                  0
Teacher_Quality               78
School_Type                    0
Peer_Influence                 0
Physical_Activity              0
Learning_Disabilities          0
Parental_Education_Level      90
Distance_from_Home            67
Gender                         0
Exam_Score                     0
dtype: int64
```

Let's check the actual frequency of our missing values based on the total amount of records we have.

[266]:
```
students.isnull().sum()/len(students)*100
```

[266]:
```
Hours_Studied                 0.000000
Attendance                    0.000000
Parental_Involvement          0.000000
Access_to_Resources           0.000000
Extracurricular_Activities    0.000000
Sleep_Hours                   0.000000
Previous_Scores               0.000000
Motivation_Level              0.000000
Internet_Access               0.000000
Tutoring_Sessions             0.000000
Family_Income                 0.000000
Teacher_Quality               1.180566
School_Type                   0.000000
Peer_Influence                0.000000
```

```
Physical_Activity             0.000000
Learning_Disabilities         0.000000
Parental_Education_Level      1.362192
Distance_from_Home            1.014076
Gender                        0.000000
Exam_Score                    0.000000
dtype: float64
```

The percentage of missing values we have is very very low, and since we have a considerable number of records as it is, we've decided to **drop the records with missing data**

[267]:
```python
students_adjusted = students.dropna()
students_adjusted.isnull().sum()
```

[267]:
```
Hours_Studied                 0
Attendance                    0
Parental_Involvement          0
Access_to_Resources           0
Extracurricular_Activities    0
Sleep_Hours                   0
Previous_Scores               0
Motivation_Level              0
Internet_Access               0
Tutoring_Sessions             0
Family_Income                 0
Teacher_Quality               0
School_Type                   0
Peer_Influence                0
Physical_Activity             0
Learning_Disabilities         0
Parental_Education_Level      0
Distance_from_Home            0
Gender                        0
Exam_Score                    0
dtype: int64
```

[268]:
```python
print(f"The original dataset, students, has {students.shape[1]} variables and␣
↪{students.shape[0]} records")
print(f"We had a total of {students.shape[0]-students_adjusted.shape[0]} records␣
↪with missing data.")
print(f"Our resulting dataset, adjusted_students, has {students_adjusted.
↪shape[1]} variables and {students_adjusted.shape[0]} records")
```

```
The original dataset, students, has 20 variables and 6607 records
We had a total of 229 records with missing data.
Our resulting dataset, adjusted_students, has 20 variables and 6378 records
```

### 1.2.4 First visualizations

We want to visualize our variables, and therefore will use a number of different plots. We will approach numerical and categorical variables differently.

[269]:
```
print(f"Categorical variables: {categorical_variables}")
print(f"Numerical variables: {numerical_variables}")
```

```
Categorical variables: ['Parental_Involvement', 'Access_to_Resources',
'Extracurricular_Activities', 'Motivation_Level', 'Internet_Access',
'Family_Income', 'Teacher_Quality', 'School_Type', 'Peer_Influence',
'Learning_Disabilities', 'Parental_Education_Level', 'Distance_from_Home',
'Gender']
Numerical variables: ['Hours_Studied', 'Attendance', 'Sleep_Hours',
'Previous_Scores', 'Tutoring_Sessions', 'Physical_Activity', 'Exam_Score']
```

#### Categorical variables

#### Bar plots

[270]:
```
fig = plt.figure(figsize=(18, 18))
gs = gridspec.GridSpec(nrows=4, ncols=4)

ax00 = fig.add_subplot(gs[0, 0])
ax01 = fig.add_subplot(gs[0, 1])
ax02 = fig.add_subplot(gs[0, 2])
ax03 = fig.add_subplot(gs[0, 3])
ax10 = fig.add_subplot(gs[1, 0])
ax11 = fig.add_subplot(gs[1, 1])
ax12 = fig.add_subplot(gs[1, 2])
ax13 = fig.add_subplot(gs[1, 3])
ax20 = fig.add_subplot(gs[2, 0])
ax21 = fig.add_subplot(gs[2, 1])
ax22 = fig.add_subplot(gs[2, 2])
ax23 = fig.add_subplot(gs[2, 3])
ax30 = fig.add_subplot(gs[3, 0])

ax = [ax00, ax01, ax02, ax03, ax10, ax11, ax12, ax13, ax20, ax21, ax22, ax23,␣
 ↪ax30]

for i, colname in enumerate(categorical_variables):

    group = students.groupby(colname, observed=False).size()

    ax[i].bar(group.index.astype(str), group, color=colors['blue'])

    ax[i].set_ylabel('Frequency')
    ax[i].set_title(f"Frequency for each\n{colname} group")
```

```
plt.tight_layout()
plt.show()
```

modeling-student-performance_files/modeling-student-performa

**Confusion tables**

[271]:
```
parent_invol_resource_access_crossed = pd.
 ↪crosstab(students['Parental_Involvement'],students['Access_to_Resources'])
print(parent_invol_resource_access_crossed)
```

```
Access_to_Resources   High  Low  Medium
Parental_Involvement
High                   568  413     927
Low                    414  231     692
Medium                 993  669    1700
```

Since students who have medium/high parental involvement also have medium/high access to resources, both categories aren't necessary for the model and since access to resources matters more for our model, we'll be dropping parental involvement.

[272]:
```
peer_influence_income_crossed = pd.
 ↪crosstab(students['Peer_Influence'],students['Family_Income'])
print(peer_influence_income_crossed)
```

```
Family_Income   High   Low  Medium
Peer_Influence
Negative         251   577     549
Neutral          493  1038    1061
Positive         525  1057    1056
```

Since students who have medium/high family income also have neutral/positive peer relationships, both categories aren't necessary for the model and we've chosen to keep family income as it's a variable of interest.

[273]:
```
distance_motivation_crossed = pd.crosstab(students['Distance_from_Home'],␣
 ↪students['Motivation_Level'])
print(distance_motivation_crossed)
```

```
Motivation_Level    High   Low  Medium
Distance_from_Home
Far                  142   185     331
Moderate             394   611     993
Near                 773  1125    1986
```

Since students with medium/high motivation levels live moderate/near schools, both categories aren't necessary for the model and we've chosen to keep motivation levels as it's a variable of interest.

[274]:
```
students = students_adjusted.drop(columns=['Distance_from_Home',
    'Peer_Influence', 'Parental_Involvement'])
```

[275]:
```
students.sample(5)
```

[275]:
```
      Hours_Studied  Attendance Access_to_Resources  \
4669             20          71                High
3077             23          87                 Low
4616             27          99              Medium
5903             26          63              Medium
5270             28          91              Medium

     Extracurricular_Activities  Sleep_Hours  Previous_Scores  \
4669                        Yes            6               56
3077                         No            7               79
4616                        Yes            5               69
5903                        Yes            5               76
5270                         No            7               90

     Motivation_Level Internet_Access  Tutoring_Sessions Family_Income  \
4669             High             Yes                  5        Medium
3077           Medium             Yes                  2        Medium
4616           Medium             Yes                  2        Medium
5903           Medium             Yes                  3        Medium
5270           Medium             Yes                  2          High

     Teacher_Quality School_Type  Physical_Activity Learning_Disabilities  \
4669          Medium     Private                  5                    No
3077          Medium      Public                  2                    No
4616             Low      Public                  2                    No
5903            High      Public                  5                   Yes
5270             Low     Private                  3                    No

     Parental_Education_Level  Gender  Exam_Score
4669                  College    Male          68
3077              High School    Male          67
4616             Postgraduate    Male          73
5903              High School  Female          66
5270              High School    Male          71
```

### Numerical variables

### Boxplots

[276]:
```
fig = plt.figure(figsize=(14, 14))
gs = gridspec.GridSpec(nrows=3, ncols=3)

ax00 = fig.add_subplot(gs[0, 0])
ax01 = fig.add_subplot(gs[0, 1])
ax02 = fig.add_subplot(gs[0, 2])
ax10 = fig.add_subplot(gs[1, 0])
ax11 = fig.add_subplot(gs[1, 1])
ax12 = fig.add_subplot(gs[1, 2])
ax20 = fig.add_subplot(gs[2, 0])

ax = [ax00, ax01, ax02, ax10, ax11, ax12, ax20]

for i, colname in enumerate(numerical_variables):

    ax[i].boxplot(students[colname],
                  patch_artist=True,
                   boxprops=dict(facecolor=colors['custom_blues'][0]),
                  capprops=dict(color=colors['custom_blues'][4]),
                  medianprops=dict(color='black', linewidth=2))

    ax[i].set_title(f"Distribution for {colname}")

plt.tight_layout()
plt.show()
```

modeling-student-performance_files/modeling-student-performa

**Pairwise plot**

[277]:
```
sns.pairplot(students[numerical_variables], diag_kind='hist', plot_kws={'s': 15,␣
↪'color': colors['blue']})
```

[277]: <seaborn.axisgrid.PairGrid at 0x28d05faa0>

modeling-student-performance_files/modeling-student-performa

**Correlation heatmap**

[278]:
```
student_corr = students[numeric_cols].corr()
```

[279]:
```
plt.figure()
sns.heatmap(student_corr, annot=True, cmap='Blues', fmt='.2f', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```

modeling-student-performance_files/modeling-student-performa

## 1.3 Our hypotheses

Since we were able to see some linear relationship between a few of our numerical variables and `Exam_Score`, we have decided to pick this variable as our response variable.

The correlation heatmap also supports our choice of response variable, where we can see correlation between exam scores and the rest of the variables.

In the context of our problem, modeling student performance, we think choosing exam scores as our response variable is a good way to ascertain how students perform, and which factors influence the way they score in tests.

We have initially seen linearity in the scatterplots between exam score and hours studied, attendance, previous scores and tutoring sessions. We'll try to assess wether or not these numerical variables are statistically significant when predicting exam scores.

In regards to our remaining categorical variables after discarding based on confusion tables, we'll check for significance for all of them and try to ascertain which variables are best to include in our model with the goal of predicting exam scores.

As previously stated, the correlation heatmap seems to indicate we don't have severe multicollinearity. Further checks need to be done to ensure this.

We have to consider the fact that Exam Scores are clustered around 65 out of 100, and we have what we could consider some outliers in the higher ranges. Based on this, we think we may encounter that the high scores have a different relationship to the predictors compared to the majority of the other scores, which are around the 65%.

**Our response variable will be: Exam_Score**

### 1.4 Initial Model

[280]:
```
X = students.drop('Exam_Score', axis=1)
y = students['Exam_Score']
```

[281]:
```
categorical_cols = X.select_dtypes(include=['object', 'category']).columns
numerical_cols = X.select_dtypes(include=['int64', 'float64']).columns
```

[282]:
```
le = LabelEncoder()
for col in categorical_cols:
    X[col] = le.fit_transform(X[col])

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


results = pd.DataFrame(columns=[
    'Predictor', 'Correlation', 'P-value', 'R-squared', 'Adj_R-squared', 'VIF',
    'T-statistic', 'F-statistic', 'AIC', 'BIC',
    'ANOVA_Type1_F', 'ANOVA_Type1_P',
    'ANOVA_Type2_F', 'ANOVA_Type2_P',
    'ANOVA_Type3_F', 'ANOVA_Type3_P'
])
```

[283]:
```
for i, predictor in enumerate(X.columns):
    X_i = sm.add_constant(X_scaled[:, i])
    model = sm.OLS(y, X_i).fit()

    p_value = model.pvalues.iloc[1]
    rsq = model.rsquared
    adj_rsq = model.rsquared_adj
    t_statistic = model.tvalues.iloc[1]
    f_statistic = model.fvalue
    vif = variance_inflation_factor(X_scaled, i)
    aic = model.aic
    bic = model.bic

    correlation = pd.Series(X_scaled[:, i]).corr(y)

    formula = f'Exam_Score ~ {predictor}'
    model_anova = smf.ols(formula, data=students_adjusted).fit()

    anova_type1 = anova_lm(model_anova, typ=1)
    anova_type1_f = anova_type1['F'].iloc[0]
    anova_type1_p = anova_type1['PR(>F)'].iloc[0]

    anova_type2 = anova_lm(model_anova, typ=2)
    anova_type2_f = anova_type2['F'].iloc[0]
```

```
    anova_type2_p = anova_type2['PR(>F)'].iloc[0]

    anova_type3 = anova_lm(model_anova, typ=3)
    anova_type3_f = anova_type3['F'].iloc[0]
    anova_type3_p = anova_type3['PR(>F)'].iloc[0]

    current_results = pd.DataFrame({
        'Predictor': [predictor],
        'Correlation': [correlation],  # Correlation in the second column
        'P-value': [p_value],
        'R-squared': [rsq],
        'Adj_R-squared': [adj_rsq],
        'VIF': [vif],
        'T-statistic': [t_statistic],
        'F-statistic': [f_statistic],
        'AIC': [aic],
        'BIC': [bic],
        'ANOVA_Type1_F': [anova_type1_f],
        'ANOVA_Type1_P': [anova_type1_p],
        'ANOVA_Type2_F': [anova_type2_f],
        'ANOVA_Type2_P': [anova_type2_p],
        'ANOVA_Type3_F': [anova_type3_f],
        'ANOVA_Type3_P': [anova_type3_p]
    })

    current_results = current_results.dropna(how='all', axis=1)
    results = pd.concat([results, current_results], ignore_index=True)

results_sorted = results.sort_values(by='Adj_R-squared', ascending=False)
results_sorted
```

```
/var/folders/bn/8l9zg4092yg1r262fl2zn9nw0000gn/T/ipykernel_4079/3609911391.py:51
: FutureWarning: The behavior of DataFrame concatenation with empty or all-NA
entries is deprecated. In a future version, this will no longer exclude empty or
all-NA columns when determining the result dtypes. To retain the old behavior,
exclude the relevant entries before the concat operation.
```

[283]:

```
                     Predictor  Correlation        P-value  R-squared  \
1                   Attendance     0.003104   0.000000e+00   0.336700
0                Hours_Studied     0.003507  4.524802e-308   0.198118
5              Previous_Scores     0.019833   1.121433e-44   0.030375
8            Tutoring_Sessions     0.002661   2.102216e-36   0.024595
2          Access_to_Resources     0.000353   1.446812e-12   0.007830
13         Learning_Disabilities    -0.005272   1.921126e-11   0.007041
3   Extracurricular_Activities     0.013779   4.646554e-07   0.003977
10             Teacher_Quality    -0.010520   2.191872e-06   0.003510
7              Internet_Access    -0.000532   4.412828e-05   0.002614
```

| | | | | |
|---|---|---|---|---|
| 14 | Parental_Education_Level | 0.012913 | 1.112637e-03 | 0.001666 |
| 12 | Physical_Activity | -0.011554 | 4.461342e-02 | 0.000632 |
| 9 | Family_Income | 0.008112 | 7.047761e-02 | 0.000513 |
| 6 | Motivation_Level | -0.012348 | 1.561947e-01 | 0.000315 |
| 4 | Sleep_Hours | 0.010594 | 1.703173e-01 | 0.000295 |
| 11 | School_Type | -0.012893 | 3.854988e-01 | 0.000118 |
| 15 | Gender | 0.003920 | 6.936955e-01 | 0.000024 |

| | Adj_R-squared | VIF | T-statistic | F-statistic | AIC | \ |
|---|---|---|---|---|---|---|
| 1 | 0.336596 | 1.002926 | 56.890623 | 3236.542960 | 32891.664424 | |
| 0 | 0.197992 | 1.001827 | 39.689943 | 1575.291562 | 34101.790605 | |
| 5 | 0.030223 | 1.003285 | 14.132771 | 199.735203 | 35313.278862 | |
| 8 | 0.024442 | 1.001307 | 12.679697 | 160.774705 | 35351.180193 | |
| 2 | 0.007675 | 1.003326 | -7.093709 | 50.320703 | 35459.872237 | |
| 13 | 0.006885 | 1.002033 | -6.724033 | 45.212618 | 35464.943930 | |
| 3 | 0.003821 | 1.000808 | 5.045584 | 25.457916 | 35484.595934 | |
| 10 | 0.003354 | 1.000999 | -4.739229 | 22.460292 | 35487.583273 | |
| 7 | 0.002457 | 1.002028 | 4.087563 | 16.708173 | 35493.319577 | |
| 14 | 0.001509 | 1.002298 | 3.261866 | 10.639768 | 35499.376896 | |
| 12 | 0.000476 | 1.004972 | 2.008678 | 4.034786 | 35505.976356 | |
| 9 | 0.000356 | 1.003210 | -1.809132 | 3.272958 | 35506.737986 | |
| 6 | 0.000159 | 1.002270 | -1.418154 | 2.011161 | 35507.999656 | |
| 4 | 0.000138 | 1.001995 | -1.371340 | 1.880573 | 35508.130246 | |
| 11 | -0.000039 | 1.002499 | -0.867869 | 0.753196 | 35509.257743 | |
| 15 | -0.000133 | 1.001120 | -0.393863 | 0.155128 | 35509.855956 | |

| | BIC | ANOVA_Type1_F | ANOVA_Type1_P | ANOVA_Type2_F | ANOVA_Type2_P | \ |
|---|---|---|---|---|---|---|
| 1 | 32905.185643 | 3236.542960 | 0.000000e+00 | 3236.542960 | 0.000000e+00 | |
| 0 | 34115.311825 | 1575.291562 | 4.524802e-308 | 1575.291562 | 4.524802e-308 | |
| 5 | 35326.800082 | 199.735203 | 1.121433e-44 | 199.735203 | 1.121433e-44 | |
| 8 | 35364.701412 | 160.774705 | 2.102216e-36 | 160.774705 | 2.102216e-36 | |
| 2 | 35473.393457 | 92.416360 | 2.722170e-40 | 92.416360 | 2.722170e-40 | |
| 13 | 35478.465150 | 45.212618 | 1.921126e-11 | 45.212618 | 1.921126e-11 | |
| 3 | 35498.117154 | 25.457916 | 4.646554e-07 | 25.457916 | 4.646554e-07 | |
| 10 | 35501.104493 | 18.597490 | 8.844704e-09 | 18.597490 | 8.844704e-09 | |
| 7 | 35506.840797 | 16.708173 | 4.412828e-05 | 16.708173 | 4.412828e-05 | |
| 14 | 35512.898116 | 35.947108 | 2.990498e-16 | 35.947108 | 2.990498e-16 | |
| 12 | 35519.497576 | 4.034786 | 4.461342e-02 | 4.034786 | 4.461342e-02 | |
| 9 | 35520.259206 | 28.782492 | 3.597672e-13 | 28.782492 | 3.597672e-13 | |
| 6 | 35521.520876 | 25.535515 | 9.000204e-12 | 25.535515 | 9.000204e-12 | |
| 4 | 35521.651465 | 1.880573 | 1.703173e-01 | 1.880573 | 1.703173e-01 | |
| 11 | 35522.778962 | 0.753196 | 3.854988e-01 | 0.753196 | 3.854988e-01 | |
| 15 | 35523.377176 | 0.155128 | 6.936955e-01 | 0.155128 | 6.936955e-01 | |

| | ANOVA_Type3_F | ANOVA_Type3_P |
|---|---|---|
| 1 | 3.398701e+04 | 0.0 |
| 0 | 1.613659e+05 | 0.0 |

| | | |
|---|---|---|
| 5 | 6.180827e+04 | 0.0 |
| 8 | 7.646139e+05 | 0.0 |
| 2 | 5.916635e+05 | 0.0 |
| 13 | 1.702976e+06 | 0.0 |
| 3 | 7.550905e+05 | 0.0 |
| 10 | 5.727176e+05 | 0.0 |
| 7 | 1.405650e+05 | 0.0 |
| 14 | 5.805053e+05 | 0.0 |
| 12 | 1.997981e+05 | 0.0 |
| 9 | 3.727122e+05 | 0.0 |
| 6 | 3.854527e+05 | 0.0 |
| 4 | 7.932091e+04 | 0.0 |
| 11 | 5.749512e+05 | 0.0 |
| 15 | 7.945274e+05 | 0.0 |

After looking at the $R_a^2$ of exam score regressed onto each individual predictor, we're going to naively choose the 6 predictors with the strongest $R_a^2$ values to model.

[284]:
```
selected_predictors = [
    'Attendance',
    'Hours_Studied',
    'Previous_Scores',
    'Tutoring_Sessions',
    'Access_to_Resources',
    'Learning_Disabilities'
]

categorical_predictors = ['Access_to_Resources', 'Learning_Disabilities']

X = students[selected_predictors]
y = students['Exam_Score']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)

le = LabelEncoder()
for col in categorical_predictors:
    X_train[col] = le.fit_transform(X_train[col])
    X_test[col] = le.transform(X_test[col])

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

X_train_scaled = sm.add_constant(X_train_scaled)
model = sm.OLS(y_train, X_train_scaled).fit()
```

```
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             Exam_Score   R-squared:                       0.596
Model:                            OLS   Adj. R-squared:                  0.595
Method:                 Least Squares   F-statistic:                     1095.
Date:                Sat, 12 Oct 2024   Prob (F-statistic):               0.00
Time:                        22:32:07   Log-Likelihood:                -10405.
No. Observations:                4464   AIC:                         2.082e+04
Df Residuals:                    4457   BIC:                         2.087e+04
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         67.2581      0.037   1803.964      0.000      67.185      67.331
x1             2.2692      0.037     60.806      0.000       2.196       2.342
x2             1.7525      0.037     46.965      0.000       1.679       1.826
x3             0.6812      0.037     18.257      0.000       0.608       0.754
x4             0.5998      0.037     16.081      0.000       0.527       0.673
x5            -0.3533      0.037     -9.473      0.000      -0.426      -0.280
x6            -0.2544      0.037     -6.819      0.000      -0.327      -0.181
==============================================================================
Omnibus:                     5584.026   Durbin-Watson:                   2.007
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           879042.429
Skew:                           6.811   Prob(JB):                         0.00
Kurtosis:                      70.383   Cond. No.                         1.06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Since the model has a moderate $R_a^2$ value, we want to look at the residuals to ensure that none of the model assumptions were violated.

[285]:
```
X_test_scaled = sm.add_constant(X_test_scaled)
y_pred = model.predict(X_test_scaled)
```

[286]:
```
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, color=colors['blue'], alpha=.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],␣
 ↪color=colors['red'], lw=2)
plt.xlabel('Actual Exam Scores')
plt.ylabel('Predicted Exam Scores')
plt.title('Actual vs Predicted Exam Scores')
plt.grid(True)
```

```
plt.show()
```

modeling-student-performance_files/modeling-student-performa

There are some points that don't align with the rest of the distribution and might be influential points.

[287]:
```
fig, ax = plt.subplots(figsize=(10,7))
influence_plot(model, ax=ax, criterion="cooks")
plt.show()
```

modeling-student-performance_files/modeling-student-performa

There are a significant number of influential points so we're going to do some transformations on the data.

### 1.5 Box-Cox Transformation

[288]:
```
le = LabelEncoder()
for col in categorical_predictors:
    X.loc[:, col] = le.fit_transform(X[col])

X_boxcox = X.apply(lambda x: stats.boxcox(x + 1)[0])
y_boxcox, fitted_lambda = stats.boxcox(y + 1)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_boxcox)

X_const = sm.add_constant(X_scaled)
model_boxcox = sm.OLS(y_boxcox, X_const).fit()
print("Box-Cox Transformation Model Summary")
print(f"Lambda used for Box-Cox Transformation: {fitted_lambda}")
print(model_boxcox.summary())
```

```
Box-Cox Transformation Model Summary
```

```
Lambda used for Box-Cox Transformation: -2.6806950740503064
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.719
Model:                            OLS   Adj. R-squared:                  0.719
Method:                 Least Squares   F-statistic:                     2718.
Date:                Sat, 12 Oct 2024   Prob (F-statistic):               0.00
Time:                        22:32:23   Log-Likelihood:                 85842.
No. Observations:                6378   AIC:                        -1.717e+05
Df Residuals:                    6371   BIC:                        -1.716e+05
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3730   4.33e-09   8.62e+07      0.000       0.373       0.373
x1          4.152e-07   4.33e-09     95.841      0.000    4.07e-07    4.24e-07
x2          3.166e-07   4.33e-09     73.077      0.000    3.08e-07    3.25e-07
x3           1.24e-07   4.33e-09     28.624      0.000    1.16e-07    1.33e-07
x4          1.099e-07   4.33e-09     25.381      0.000    1.01e-07    1.18e-07
x5         -5.522e-08   4.33e-09    -12.752      0.000   -6.37e-08   -4.67e-08
x6         -5.117e-08   4.33e-09    -11.813      0.000   -5.97e-08   -4.27e-08
==============================================================================
Omnibus:                     4676.971   Durbin-Watson:                   1.986
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           183625.672
Skew:                           3.079   Prob(JB):                         0.00
Kurtosis:                      28.555   Cond. No.                         1.04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

```
/var/folders/bn/8l9zg4092yg1r262fl2zn9nw0000gn/T/ipykernel_4079/2231151566.py:3:
FutureWarning: Setting an item of incompatible dtype is deprecated and will
raise in a future error of pandas. Value '[0 2 2 ... 1 0 1]' has dtype
incompatible with category, please explicitly cast to a compatible dtype first.
/var/folders/bn/8l9zg4092yg1r262fl2zn9nw0000gn/T/ipykernel_4079/2231151566.py:3:
FutureWarning: Setting an item of incompatible dtype is deprecated and will
raise in a future error of pandas. Value '[0 0 0 ... 0 0 0]' has dtype
incompatible with category, please explicitly cast to a compatible dtype first.
```

The $R^2$ has improved so now we'll see how the residuals have changed.

[289]:
```
y_pred_boxcox = model_boxcox.predict(X_const)
residuals = y_boxcox - y_pred_boxcox
```

[290]:

```python
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_pred_boxcox, y=residuals, color=colors['blue'])
plt.axhline(0, color=colors['red'], linestyle='--')
plt.title('Residuals vs Predicted Values')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.grid()
plt.show()
```

modeling-student-performance_files/modeling-student-performa

There still seems to be a significant amount of data that's falling outside the expected bounds of the data so we'll try including more predictors and splitting the data into multiple models.

## 1.6 Splitting the Data Into 2 Models

### 1.6.1 Set Up

[291]:

```python
high_scores = students[students['Exam_Score'] >= 80].reset_index().drop(columns=['index'])
low_scores = students[students['Exam_Score'] < 80].reset_index().drop(columns=['index'])
print(f"There are {len(low_scores)} rows within the low scores dataset.")
print(f"There are {len(high_scores)} rows within the high scores dataset.")
```

```
There are 6330 rows within the low scores dataset.
There are 48 rows within the high scores dataset.
```

After creating two new datasets, we're going to use those datasets to make training and test datasets for each model.

[292]:

```python
train_high, test_high = train_test_split(high_scores, test_size=0.2, random_state=42)
train_low, test_low = train_test_split(low_scores, test_size=0.2, random_state=42)
print(f"There are {len(train_low)} rows within the low scores training dataset.")
print(f"There are {len(train_high)} rows within the high scores training dataset.")
```

```
There are 5064 rows within the low scores training dataset.
There are 38 rows within the high scores training dataset.
```

[293]:
```
X_train_high = train_high.drop('Exam_Score', axis= 1)
y_train_high = train_high['Exam_Score']

X_test_high = test_high.drop('Exam_Score', axis= 1)
y_test_high = test_high['Exam_Score']

X_train_low = train_low.drop('Exam_Score', axis= 1)
y_train_low = train_low['Exam_Score']

X_test_low = test_low.drop('Exam_Score', axis= 1)
y_test_low = test_low['Exam_Score']
```

[294]:
```
categorical_cols1 = X_train_high.select_dtypes(include=['object', 'category']).
 ↪columns
numerical_cols1 = X_train_high.select_dtypes(include=['int64', 'float64']).
 ↪columns

categorical_cols2 = X_test_high.select_dtypes(include=['object', 'category']).
 ↪columns
numerical_cols2 = X_test_high.select_dtypes(include=['int64', 'float64']).columns

categorical_cols3 = X_train_low.select_dtypes(include=['object', 'category']).
 ↪columns
numerical_cols3 = X_train_low.select_dtypes(include=['int64', 'float64']).columns

categorical_cols4 = X_test_low.select_dtypes(include=['object', 'category']).
 ↪columns
numerical_cols4 = X_test_low.select_dtypes(include=['int64', 'float64']).columns
```

[295]:
```
preprocessor_train_high = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols1),
        ('cat', OneHotEncoder(drop='first'), categorical_cols1)  # drop='first'␣
 ↪avoids dummy variable trap
    ]
)

preprocessor_test_high = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols2),
        ('cat', OneHotEncoder(drop='first'), categorical_cols2)  # drop='first'␣
 ↪avoids dummy variable trap
    ]
)
preprocessor_train_low = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols3),
```

```
        ('cat', OneHotEncoder(drop='first'), categorical_cols3)  # drop='first'␣
↪avoids dummy variable trap
    ]
)

preprocessor_test_low = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols4),
        ('cat', OneHotEncoder(drop='first'), categorical_cols4)  # drop='first'␣
↪avoids dummy variable trap
    ]
)
```

### 1.7 Low Model

Students who scored 80 or lower on their exam.

[296]:
```
lasso_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor_train_low),
    ('lasso', Lasso(alpha=0.1))  # You can adjust alpha for regularization
])

lasso_pipeline.fit(X_train_low, y_train_low)
```

[296]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num', StandardScaler(),
                                                  Index(['Hours_Studied',
'Attendance', 'Sleep_Hours', 'Previous_Scores',
       'Tutoring_Sessions', 'Physical_Activity'],
      dtype='object')),
                                                 ('cat',
                                                  OneHotEncoder(drop='first'),
                                                  Index(['Access_to_Resources',
'Extracurricular_Activities', 'Motivation_Level',
       'Internet_Access', 'Family_Income', 'Teacher_Quality', 'School_Type',
       'Learning_Disabilities', 'Parental_Education_Level', 'Gender'],
      dtype='object'))])),
                ('lasso', Lasso(alpha=0.1))])
```

[297]:
```
encoded_categorical_names = lasso_pipeline.named_steps['preprocessor'] \
    .transformers_[1][1].get_feature_names_out(categorical_cols1)

all_feature_names = list(numerical_cols1) + list(encoded_categorical_names)

lasso_coefficients = lasso_pipeline.named_steps['lasso'].coef_
```

```python
selected_features = [name for name, coef in zip(all_feature_names,
 ↪lasso_coefficients) if coef != 0]

print(f'Selected predictors: {selected_features}')
```

```
Selected predictors: ['Hours_Studied', 'Attendance', 'Previous_Scores',
'Tutoring_Sessions', 'Physical_Activity', 'Access_to_Resources_Low',
'Access_to_Resources_Medium', 'Extracurricular_Activities_Yes',
'Motivation_Level_Low', 'Family_Income_Low', 'Parental_Education_Level_High
School', 'Parental_Education_Level_Postgraduate']
```

We let the lasso modeling tool pick the predictors to use. We then dropped all other predictors from the model.

[298]:
```python
x_lasso_train_low = X_train_low.drop(columns=['Sleep_Hours',
                                               'Internet_Access',
                                               'Teacher_Quality',
                                               'School_Type',
                                               'Learning_Disabilities',
                                               'Gender'])
y_lasso_train = y_train_low
formula = 'Exam_Score ~ ' + ' + '.join(x_lasso_train_low.columns)
```

### 1.7.1 Summarizing the model with the low score dataset:

[299]:
```python
lasso_model_low = smf.ols(formula= formula, data = train_low).fit()
lasso_model_low.summary()
```

[299]:

| | | | |
|---|---|---|---|
| **Dep. Variable:** | Exam_Score | **R-squared:** | 0.898 |
| **Model:** | OLS | **Adj. R-squared:** | 0.898 |
| **Method:** | Least Squares | **F-statistic:** | 3183. |
| **Date:** | Sat, 12 Oct 2024 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 22:32:24 | **Log-Likelihood:** | -7540.9 |
| **No. Observations:** | 5064 | **AIC:** | 1.511e+04 |
| **Df Residuals:** | 5049 | **BIC:** | 1.521e+04 |
| **Df Model:** | 14 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P> abs(t) | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 42.0979 | 0.158 | 266.026 | 0.000 | 41.788 | 42.408 |
| **Access_to_Resources[T.Low]** | -1.9372 | 0.044 | -44.274 | 0.000 | -2.023 | -1.851 |
| **Access_to_Resources[T.Medium]** | -0.9681 | 0.035 | -27.774 | 0.000 | -1.036 | -0.900 |
| **Extracurricular_Activities[T.Yes]** | 0.4688 | 0.031 | 15.212 | 0.000 | 0.408 | 0.529 |
| **Motivation_Level[T.Low]** | -0.9550 | 0.044 | -21.845 | 0.000 | -1.041 | -0.869 |
| **Motivation_Level[T.Medium]** | -0.4403 | 0.040 | -11.040 | 0.000 | -0.519 | -0.362 |
| **Family_Income[T.Low]** | -1.0166 | 0.042 | -24.447 | 0.000 | -1.098 | -0.935 |
| **Family_Income[T.Medium]** | -0.4775 | 0.042 | -11.470 | 0.000 | -0.559 | -0.396 |
| **Parental_Education_Level[T.High School]** | -0.4723 | 0.035 | -13.593 | 0.000 | -0.540 | -0.404 |
| **Parental_Education_Level[T.Postgraduate]** | 0.5296 | 0.043 | 12.201 | 0.000 | 0.444 | 0.615 |
| **Hours_Studied** | 0.2976 | 0.003 | 117.166 | 0.000 | 0.293 | 0.303 |
| **Attendance** | 0.1991 | 0.001 | 152.018 | 0.000 | 0.196 | 0.202 |
| **Previous_Scores** | 0.0470 | 0.001 | 44.784 | 0.000 | 0.045 | 0.049 |
| **Tutoring_Sessions** | 0.4998 | 0.012 | 40.552 | 0.000 | 0.476 | 0.524 |
| **Physical_Activity** | 0.2205 | 0.015 | 14.948 | 0.000 | 0.192 | 0.249 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 872.171 | **Durbin-Watson:** | 1.966 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 13229.088 |
| **Skew:** | 0.352 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 10.887 | **Cond. No.** | 1.19e+03 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.19e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

[300]:
```
data = pd.DataFrame({
    'Fitted_Values': lasso_model_low.fittedvalues,
    'Residuals': lasso_model_low.resid
})

residual_plot = (
    ggplot(data, aes(x='Fitted_Values', y='Residuals')) +
    geom_point(color = "#838ceb") +  # Scatter plot of points
    geom_smooth(method='loess', color='grey', se=False, linetype='solid') +  #␣
↪LOWESS line
    geom_hline(yintercept=0, linetype='dashed', color='red') +  # Horizontal␣
↪line at y=0
    labs(x='Fitted Values', y='Residuals', title='Linearity Check: Fitted Values␣
↪vs Residuals') +
    theme_minimal()  # Set the figure size
)

print(residual_plot)
```

```
/var/folders/bn/8l9zg4092yg1r262fl2zn9nw0000gn/T/ipykernel_4079/3866888096.py:15
: FutureWarning: Using print(plot) to draw and show the plot figure is
deprecated and will be removed in a future version. Use plot.show().
```

modeling-student-performance_files/modeling-student-performa

After plotting the fitted values against the residual values, most of the data falls in the expected region which a few outliers visible.

[301]:
```
low_residuals = lasso_model_low.resid

quantiles = np.percentile(low_residuals, [i for i in range(0, 101)])
theoretical_quantiles = np.percentile(np.random.normal(0, 1, 1000), [i for i in␣
 ↪range(0, 101)])

qq_data = pd.DataFrame({
    'Theoretical Quantiles': theoretical_quantiles,
    'Sample Quantiles': quantiles
})

gg = (ggplot(qq_data, aes(x='Theoretical Quantiles', y='Sample Quantiles')) +
      geom_point(color = "#838ceb") +
      geom_abline(slope=1, intercept=0, color='#373f8a') +
      labs(title='Normality Check: Q-Q Plot of Residuals',
           x='Theoretical Quantiles',
           y='Sample Quantiles') +
      theme_minimal())

print(gg)
```

```
/var/folders/bn/8l9zg4092yg1r262fl2zn9nw0000gn/T/ipykernel_4079/2675193875.py:19
: FutureWarning: Using print(plot) to draw and show the plot figure is
deprecated and will be removed in a future version. Use plot.show().
```

modeling-student-performance_files/modeling-student-performa

After doing the QQ plot of the residuals, most of the data falls on the line which a few outliers visible.

[302]:
```
predictors_to_drop = ['Access_to_Resources', 'Extracurricular_Activities',
                          'Family_Income', 'Motivation_Level',␣
 ↪'Parental_Education_Level']

train_long = train_low.melt(id_vars='Exam_Score', value_vars=x_lasso_train_low.
 ↪columns,
                             var_name='Predictor', value_name='Value')
train_long_filtered = train_long[~train_long['Predictor'].
 ↪isin(predictors_to_drop)]
```

For the purposes of graphing scatter plots, we temporarily dropped the categorical data.

[303]:
```
gg = (ggplot(train_long_filtered, aes(x='Value', y='Exam_Score')) +
              geom_point(color=colors['purple'], alpha=0.6) +  # Scatter plot␣
 ↪with some transparency
              facet_wrap('~Predictor', nrow=2, scales= 'free') +  # Create a␣
 ↪grid layout with 2 rows
              labs(title='Scatter Plots of Predictors vs. Exam Score',
                   x='Predictor Value',
                   y='Exam Score') +
              theme_minimal() +  # Clean minimal theme
              theme(
                  axis_text_x=element_blank(),
                  panel_grid_major=element_blank(),
                  panel_grid_minor=element_blank(),
                  subplots_adjust={'top': 0.9},
                  panel_spacing = 0.005
              )
)
gg
```

```
/opt/homebrew/anaconda3/lib/python3.12/site-
packages/plotnine/themes/themeable.py:2419: FutureWarning: You no longer need to
use subplots_adjust to make space for the legend or text around the panels. This
paramater will be removed in a future version. You can still use 'plot_margin'
'panel_spacing' for your other spacing needs.
```

modeling-student-performance_files/modeling-student-performa

[304]:
```python
X_encoded = pd.get_dummies(x_lasso_train_low, drop_first=True)
correlation_matrix = X_encoded.corr()
plt.figure()
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='Purples',␣
 ↪square=True, cbar=True)
plt.title('Correlation Matrix Heatmap')
plt.xticks(rotation=45, ha='right')
plt.show()
```

modeling-student-performance_files/modeling-student-performa

All of the graphs look good but there are some clear outliers so let's determine how many influential points are in the low score dataset.

## 1.8 Influential Points in the Low Score Model

[346]:
```python
influence = OLSInfluence(lasso_model_low)

leverage = influence.hat_matrix_diag
cooks_distance = influence.cooks_distance[0]
studentized_residuals = influence.resid_studentized_internal

influence_data = pd.DataFrame({
    'Index': np.arange(len(train_low)),
    'Leverage': leverage,
    'Cook\'s Distance': cooks_distance,
    'Studentized Residual': studentized_residuals
})

influence_data_sorted = influence_data.sort_values(by=['Leverage', 'Cook\'s␣
 ↪Distance', 'Studentized Residual'], ascending=False)

top_3_influence = influence_data_sorted.head(3)

top_3_influence
```

[346]:

| | Index | Leverage | Cook's Distance | Studentized Residual |
|---|---|---|---|---|
| 5611 | 5043 | 0.007464 | 5.039074e-07 | -0.031705 |
| 3731 | 53 | 0.006856 | 4.598461e-04 | -0.999592 |
| 3717 | 1002 | 0.006809 | 5.532396e-04 | 1.100212 |

[327]:
```
influence_data = pd.DataFrame({
    'Index': np.arange(len(train_low)),
    'Leverage': leverage,
    'Cook\'s Distance': cooks_distance,
    'Studentized Residual': studentized_residuals
})

df_melted = influence_data.melt(id_vars='Index',
                    value_vars=['Leverage', 'Cook\'s Distance', 'Studentized␣
 ↪Residual'],
                    var_name='Metric',
                    value_name='Value')

ggplot(df_melted, aes(x='Index', y='Value')) + \
    geom_point(color='#838ceb', alpha=0.6) + \
    facet_wrap('~Metric', scales='free_y', ncol=3) + \
    labs(title='Influence Metrics for All Observations',
         x='Observation Index',
         y='Value') + \
    theme_minimal() + \
    theme(
        figure_size=(15, 6),
        panel_spacing=0.025,
        panel_grid_major=element_blank(),
        panel_grid_minor=element_blank()
    )
```

modeling-student-performance_files/modeling-student-performa

[328]:
```
colors['custom_purples']
```

[328]: ['#c2c7f2', '#959cdb', '#838ceb', '#6973db', '#4a54b5']

[329]:
```
numeric_columns = ['Hours_Studied', 'Attendance', 'Sleep_Hours',␣
 ↪'Previous_Scores', 'Exam_Score']

train_low_melted = train_low[numeric_columns].melt(var_name='Feature',␣
 ↪value_name='Value')
```

```
# Create the boxplot using plotnine with multiple color shades
ggplot(train_low_melted, aes(x='Feature', y='Value', fill='Feature')) + \
    geom_boxplot(color='#4B0082') + \
    scale_fill_manual(values=colors['custom_purples']) + \
    labs(title='Box Plot of Numeric Features', x='Feature', y='Value') + \
    theme_minimal() + \
    theme(
        figure_size=(12, 6),
        axis_text_x=element_text(rotation=0, hjust=0.5),  # Ensure horizontal␣
↪x-axis labels
        plot_title=element_text(size=14, weight='bold'),
        legend_position='none'
    )
```

modeling-student-performance_files/modeling-student-performa

[330]:
```
influence1 = OLSInfluence(lasso_model_low)
cooks_d1, _ = influence1.cooks_distance

cooks_distance_df = pd.DataFrame({
    'Observation Index': np.arange(len(cooks_d1)),
    'Cook\'s Distance': cooks_d1
})

abs_residuals_low = np.abs(lasso_model_low.resid)

top_two_outliers_idx = abs_residuals_low.nlargest(2).index
top_two_outliers = train_low.loc[top_two_outliers_idx]

cooks_plot = (ggplot(cooks_distance_df, aes(x='Observation Index', y='Cook\'s␣
↪Distance')) +
              geom_point(color="#838ceb", alpha=0.6) +  # Scatter plot for␣
↪Cook's distance
              geom_segment(aes(x=min(cooks_distance_df['Observation Index']),
                               xend=max(cooks_distance_df['Observation Index']),
                               y=0.5, yend=0.5),  # Horizontal line at a␣
↪threshold, set to 0.5
                           color='#373f8a', linetype='dashed') +  # Add a␣
↪threshold line (adjust the y value as needed)
```

```
                labs(title='Cook\'s Distance for Each Observation',
                     x='Observation Index',
                     y='Cook\'s Distance') +
                scale_y_continuous(limits=(0, 0.075)) +  # Set y-axis limits from␣
 ↪0 to 1
                theme_minimal() +  # Clean theme
                theme(axis_text_x=element_text(rotation=90))  # Rotate x-axis␣
 ↪labels for better visibility
               )

print(cooks_plot)
```

```
/var/folders/bn/8l9zg4092yg1r262fl2zn9nw0000gn/T/ipykernel_4079/3930044355.py:28
: FutureWarning: Using print(plot) to draw and show the plot figure is
deprecated and will be removed in a future version. Use plot.show().
/opt/homebrew/anaconda3/lib/python3.12/site-packages/plotnine/layer.py:364:
PlotnineWarning: geom_segment : Removed 5064 rows containing missing values.
```

modeling-student-performance_files/modeling-student-performa

[331]:

```
leverage2 = influence.hat_matrix_diag
student_resid2 = influence.resid_studentized_internal

leverage_df = pd.DataFrame({
    'Leverage': leverage2,
    'Studentized Residuals': student_resid2,
    'Outlier': [False] * len(leverage2)  # Create a column to mark outliers
})

# Create the Leverage vs. Studentized Residuals plot using Plotnine
leverage_plot = (ggplot(leverage_df, aes(x='Leverage', y='Studentized␣
 ↪Residuals')) +
                 geom_point(color="#838ceb", alpha=0.5) +  # Scatter plot with␣
 ↪coloring for outliers
                 geom_hline(yintercept=0, color='#373f8a', linetype='dashed') + ␣
 ↪# Horizontal line at y=0
                 labs(title='Leverage vs Studentized Residuals',
                      x='Leverage',
                      y='Studentized Residuals') +  # Custom color for outliers
```

```
                theme_minimal()  # Clean theme
               )

# Display the leverage vs. studentized residuals plot
print(leverage_plot)
```

```
/var/folders/bn/8l9zg4092yg1r262fl2zn9nw0000gn/T/ipykernel_4079/2111622971.py:21
: FutureWarning: Using print(plot) to draw and show the plot figure is
deprecated and will be removed in a future version. Use plot.show().
```

modeling-student-performance_files/modeling-student-performa

[347]:
```
student_residuals_df = pd.DataFrame({
    'Observation Index': np.arange(len(student_resid2)),
    'Studentized Residuals': np.abs(student_resid2)
})

student_residuals_plot = (ggplot(student_residuals_df, aes(x='Observation␣
↪Index', y='Studentized Residuals')) +
                          geom_segment(aes(x='Observation Index',␣
↪xend='Observation Index',
                                           y=0, yend='Studentized Residuals'),
                                       color='#373f8a') +  # Vertical lines
                          geom_point(color="#838ceb") +  # Points for the␣
↪residuals
                          labs(title='Studentized Residuals for Each␣
↪Observation',
                               x='Observation Index',
                               y='Studentized Residuals') +
                          theme_minimal()
                         )

print(student_residuals_plot)

# print(f"Studentized residual for row 5125: {student_resid2[5125]}")
# print(f"Studentized residual for row 2542: {student_resid2[2542]}")
```

```
/var/folders/bn/8l9zg4092yg1r262fl2zn9nw0000gn/T/ipykernel_4079/5376649.py:17:
FutureWarning: Using print(plot) to draw and show the plot figure is deprecated
and will be removed in a future version. Use plot.show().
```

modeling-student-performance_files/modeling-student-performa

Even though these points are influential, they don't seem to be significantly impacting the data so we chose not to remove any of the data.

### 1.9 Testing the Low Model:

[333]:
```python
# Adding a constant to the test set for the intercept term (if required by statsmodels)
X_test = sm.add_constant(X_test_low)

# Use your previously trained model to predict y_test
y_pred = lasso_model_low.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test_low, y_pred)
rmse = np.sqrt(mse)  # Root Mean Squared Error
mae = mean_absolute_error(y_test_low, y_pred)
r2 = r2_score(y_test_low, y_pred)

# Print the results
print(f'MSE: {mse}')
print(f'RMSE: {rmse}')
print(f'MAE: {mae}')
print(f'R-squared: {r2}')
```

```
MSE: 1.2658230069319085
RMSE: 1.1250879996390988
MAE: 0.8514178758116681
R-squared: 0.8872329372713124
```

[334]:
```python
# Check if y_test_low and y_pred have data
print(f"y_test_low shape: {len(y_test_low)}, y_pred shape: {len(y_pred)}")

# Create a DataFrame from y_test_low and y_pred
results = pd.DataFrame({
    'Actual': y_test_low,
    'Predicted': y_pred
})
```

```python
# Create the scatter plot using Plotnine
gg_plot = (ggplot(results, aes(x='Actual', y='Predicted')) +
           geom_point(alpha=0.5, color=colors['purple']) +  # Scatter plot of actual vs predicted
           geom_abline(intercept=0, slope=1, color=colors['red'], size=1.5) + # Line of perfect fit
           labs(
               x='Actual Exam Score',
               y='Predicted Exam Score',
               title='Actual vs. Predicted Exam Scores'
           ) +
           theme_minimal()  # Set figure size
          )

# Display the plot
print(gg_plot)
```

```
y_test_low shape: 1266, y_pred shape: 1266
```

```
/var/folders/bn/8l9zg4092yg1r262fl2zn9nw0000gn/T/ipykernel_4079/2972994973.py:23
: FutureWarning: Using print(plot) to draw and show the plot figure is
deprecated and will be removed in a future version. Use plot.show().
```

modeling-student-performance_files/modeling-student-performa

We see that all of the residuals fall in line with the line of best fit.

### 1.10 High Model:

The scores in the high model are those over 80.

[335]:
```python
lasso_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor_train_high),
    ('lasso', Lasso(alpha=0.1))  # You can adjust alpha for regularization
])

# Fit the model to the entire dataset
lasso_pipeline.fit(X_train_high, y_train_high)
```

[335]: Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num', StandardScaler(),
                                                  Index(['Hours_Studied',
'Attendance', 'Sleep_Hours', 'Previous_Scores',
       'Tutoring_Sessions', 'Physical_Activity'],
      dtype='object')),
                                                 ('cat',
                                                  OneHotEncoder(drop='first'),
                                                  Index(['Access_to_Resources',
'Extracurricular_Activities', 'Motivation_Level',
       'Internet_Access', 'Family_Income', 'Teacher_Quality', 'School_Type',
       'Learning_Disabilities', 'Parental_Education_Level', 'Gender'],
      dtype='object'))])),
                ('lasso', Lasso(alpha=0.1))])

[336]:
```
encoded_categorical_names = lasso_pipeline.named_steps['preprocessor'] \
    .transformers_[1][1].get_feature_names_out(categorical_cols3)

all_feature_names = list(numerical_cols3) + list(encoded_categorical_names)

lasso_coefficients = lasso_pipeline.named_steps['lasso'].coef_

selected_features = [name for name, coef in zip(all_feature_names,␣
 ↪lasso_coefficients) if coef != 0]

print(f'Selected predictors: {selected_features}')
```

Selected predictors: ['Hours_Studied', 'Attendance', 'Sleep_Hours',
'Previous_Scores', 'Tutoring_Sessions', 'Access_to_Resources_Low',
'Access_to_Resources_Medium', 'Internet_Access_Yes', 'Family_Income_Low',
'Family_Income_Medium', 'Teacher_Quality_Low', 'Teacher_Quality_Medium',
'School_Type_Public', 'Parental_Education_Level_High School',
'Parental_Education_Level_Postgraduate', 'Gender_Male']

[350]:
```
x_lasso_train = X_train_high.drop(columns=['Extracurricular_Activities',
                                            'Motivation_Level',
                                            'Physical_Activity',
                                            'Learning_Disabilities'])
y_lasso_train = y_train_high
formula = 'Exam_Score ~ ' + ' + '.join(x_lasso_train.columns)
```

[351]:
```
lasso_model_high = smf.ols(formula= formula, data = train_high).fit()
lasso_model_high.summary()
```

[351]:

| | | | |
|---|---|---|---|
| **Dep. Variable:** | Exam_Score | **R-squared:** | 0.817 |
| **Model:** | OLS | **Adj. R-squared:** | 0.677 |
| **Method:** | Least Squares | **F-statistic:** | 5.845 |
| **Date:** | Sat, 12 Oct 2024 | **Prob (F-statistic):** | 0.000128 |
| **Time:** | 22:38:58 | **Log-Likelihood:** | -93.933 |
| **No. Observations:** | 38 | **AIC:** | 221.9 |
| **Df Residuals:** | 21 | **BIC:** | 249.7 |
| **Df Model:** | 16 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | 61.3570 | 8.689 | 7.061 | 0.000 | 43.287 | 79.427 |
| **Access_to_Resources[T.Low]** | -5.7833 | 2.147 | -2.693 | 0.014 | -10.249 | -1.318 |
| **Access_to_Resources[T.Medium]** | 0.9025 | 2.073 | 0.435 | 0.668 | -3.408 | 5.213 |
| **Internet_Access[T.Yes]** | 2.4094 | 2.284 | 1.055 | 0.303 | -2.340 | 7.159 |
| **Family_Income[T.Low]** | -1.2200 | 2.065 | -0.591 | 0.561 | -5.514 | 3.074 |
| **Family_Income[T.Medium]** | 1.2700 | 1.937 | 0.656 | 0.519 | -2.758 | 5.298 |
| **Teacher_Quality[T.Low]** | -3.5358 | 3.013 | -1.174 | 0.254 | -9.802 | 2.730 |
| **Teacher_Quality[T.Medium]** | -0.6296 | 1.633 | -0.386 | 0.704 | -4.025 | 2.766 |
| **School_Type[T.Public]** | -1.9864 | 2.269 | -0.875 | 0.391 | -6.705 | 2.732 |
| **Parental_Education_Level[T.High School]** | 1.7369 | 1.595 | 1.089 | 0.289 | -1.581 | 5.055 |
| **Parental_Education_Level[T.Postgraduate]** | 1.9619 | 2.174 | 0.902 | 0.377 | -2.560 | 6.484 |
| **Gender[T.Male]** | -0.6569 | 1.600 | -0.411 | 0.686 | -3.984 | 2.671 |
| **Hours_Studied** | 0.1755 | 0.168 | 1.046 | 0.307 | -0.173 | 0.524 |
| **Attendance** | 0.2455 | 0.078 | 3.162 | 0.005 | 0.084 | 0.407 |
| **Sleep_Hours** | -0.6946 | 0.477 | -1.456 | 0.160 | -1.687 | 0.298 |
| **Previous_Scores** | 0.1439 | 0.048 | 3.009 | 0.007 | 0.044 | 0.243 |
| **Tutoring_Sessions** | -0.3698 | 0.636 | -0.581 | 0.567 | -1.692 | 0.953 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 1.452 | **Durbin-Watson:** | 1.866 |
| **Prob(Omnibus):** | 0.484 | **Jarque-Bera (JB):** | 1.252 |
| **Skew:** | 0.429 | **Prob(JB):** | 0.535 |
| **Kurtosis:** | 2.765 | **Cond. No.** | 1.60e+03 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.6e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

[352]:
```
df = pd.DataFrame({
    'Fitted Values': lasso_model_high.fittedvalues,
    'Residuals': lasso_model_high.resid
})

gg = (ggplot(df, aes(x='Fitted Values', y='Residuals')) +
      geom_point(color='green', alpha=0.8) +  # Scatter plot
      geom_hline(yintercept=0, linetype='dashed', color='gray') +  # Horizontal␣
↪line at y=0
      labs(title='Linearity Check: Fitted Values vs Residuals',
```

```
            x='Fitted Values',
            y='Residuals') +
      theme_minimal()  # Clean minimal theme
)

print(gg)
```

/var/folders/bn/8l9zg4092yg1r262fl2zn9nw0000gn/T/ipykernel_4079/2767330584.py:15
: FutureWarning: Using print(plot) to draw and show the plot figure is
deprecated and will be removed in a future version. Use plot.show().

modeling-student-performance_files/modeling-student-performa

[353]:
```
df_resid = pd.DataFrame({
    'Residuals': lasso_model_high.resid
})

gg_qq = (ggplot(df_resid, aes(sample='Residuals')) +
         stat_qq(color='green', alpha=0.8) +  # Q-Q points
         stat_qq_line(color='grey') +  # Line representing normal distribution
         labs(title='Normality Check: Q-Q Plot of Residuals',
              x='Theoretical Quantiles',
              y='Sample Quantiles') +
         theme_minimal()  # Clean minimal theme
)

print(gg_qq)
```

/var/folders/bn/8l9zg4092yg1r262fl2zn9nw0000gn/T/ipykernel_4079/3077312418.py:14
: FutureWarning: Using print(plot) to draw and show the plot figure is
deprecated and will be removed in a future version. Use plot.show().

modeling-student-performance_files/modeling-student-performa

[354]:

```
predictors_to_drop = ['Access_to_Resources', 'Extracurricular_Activities',
                      'Family_Income', 'Motivation_Level', 'Parental_Education_Level']

train_high_long = train_high.melt(id_vars='Exam_Score', value_vars=x_lasso_train.columns,
                          var_name='Predictor', value_name='Value')

train_high_long_filtered = train_high_long[~train_high_long['Predictor'].isin(predictors_to_drop)]
```

[355]:

```
gg_scatter = (ggplot(train_high_long_filtered, aes(x='Value', y='Exam_Score')) +
              geom_point(color='green', alpha=0.6) +  # Scatter plot with some transparency
              facet_wrap('~Predictor', nrow=2, scales= 'free') +  # Create a grid layout with 2 rows
              labs(title='Scatter Plots of Predictors vs. Exam Score',
                   x='Predictor Value',
                   y='Exam Score') +
              theme_minimal() +  # Clean minimal theme
              theme(
                  axis_text_x=element_blank(),
                  panel_grid_major=element_blank(),
                  panel_grid_minor=element_blank(),
                  subplots_adjust={'top': 0.9},
                  panel_spacing = 0.005
              )
)

print(gg_scatter)
```

```
/var/folders/bn/8l9zg4092yg1r262fl2zn9nw0000gn/T/ipykernel_4079/1629869834.py:17
: FutureWarning: Using print(plot) to draw and show the plot figure is
deprecated and will be removed in a future version. Use plot.show().
/opt/homebrew/anaconda3/lib/python3.12/site-
packages/plotnine/themes/themeable.py:2419: FutureWarning: You no longer need to
use subplots_adjust to make space for the legend or text around the panels. This
paramater will be removed in a future version. You can still use 'plot_margin'
'panel_spacing' for your other spacing needs.
```

modeling-student-performance_files/modeling-student-performa

[356]:
```
non_numeric_cols = x_lasso_train.select_dtypes(exclude='number').columns

X_encoded = pd.get_dummies(x_lasso_train, drop_first=True)

correlation_matrix = X_encoded.corr()

plt.figure(figsize=(12, 8))
Greens = sns.light_palette("green", as_cmap=True)
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap=Greens, square=True,␣
↪cbar=True)
plt.title('Correlation Matrix Heatmap')
plt.show()
```

modeling-student-performance_files/modeling-student-performa

[357]:
```
X_test_2 = sm.add_constant(X_test_high)

y_pred = lasso_model_high.predict(X_test_2)

mse = mean_squared_error(y_test_high, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test_high, y_pred)
r2 = r2_score(y_test_high, y_pred)

print(f'MSE: {mse}')
print(f'RMSE: {rmse}')
print(f'MAE: {mae}')
print(f'R-squared: {r2}')
```

```
MSE: 40.244720366728785
RMSE: 6.3438726631868
MAE: 4.884048073870764
R-squared: -0.799853325882325
```

[358]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test_high, y_pred, alpha=0.5, color='green')  # Scatter plot of␣
↪actual vs predicted
```

```
plt.plot([min(y_test_high), max(y_test_high)], [min(y_test_high), max(y_test_high)], color='gray', lw=2)  # Line of perfect fit
plt.xlabel('Actual Exam Score')
plt.ylabel('Predicted Exam Score')
plt.title('Actual vs. Predicted Exam Scores')
plt.show()
```

modeling-student-performance_files/modeling-student-performa