Cleaning of Dataset, Visualization and Model Prediction

Annet Jacob 22122011

Aleena Joby 22122303

Abhidev SP 23122001

Gokul Manoj 23122117

In [1]:
```
# Attributes Information:

# school - student's school ('GP', 'MS', 'LVA', 'SLA').
# sex - student's sex (binary: 'F' - female or 'M' - male).
# age - student's age (numeric: from 15 to 22).
# address - student's home address type (binary: 'U' - urban or 'R' - rural).
# famsize - family size (binary: 'LE3' - less than or equal to 3 or 'GT3' - gr
# Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A
# Medu - mother's education (numeric: 0 - none, 1 - primary education (4th gra
# Fedu - father's education (numeric: 0 - none, 1 - primary education (4th gra
# Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'servi
# Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'servi
# reason - reason to choose this school (nominal: close to 'home', school 'rep
# guardian - student's guardian (nominal: 'mother', 'father' or 'other').
# traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30
# studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 -
# failures - number of past class failures (numeric: n if 0 <= n < 3, else 3).
# schoolsup - extra educational support (binary: yes or no).
# famsup - family educational support (binary: yes or no).
# paid - extra paid classes within the course subject (Math or Portuguese) (bi
# activities - extra-curricular activities (binary: yes or no).
# nursery - attended nursery school (binary: yes or no).
# higher - wants to take higher education (binary: yes or no).
# internet - Internet access at home (binary: yes or no).
# romantic - with a romantic relationship (binary: yes or no).
# famrel - quality of family relationships (numeric: from 1 - very bad to 5 -
# freetime - free time after school (numeric: from 1 - very low to 5 - very hi
# goout - going out with friends (numeric: from 1 - very low to 5 - very high)
# Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very h
# Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very h
# health - current health status (numeric: from 1 - very bad to 5 - very good)
# absences - number of school absences (numeric: from 0 to 93).
# Grades which are related with the course subject:
# G1 - first period grade (numeric: from 0 to 20)
# G2 - second period grade (numeric: from 0 to 20)
# G3 - final grade (numeric: from 0 to 20, Output Target)
```

In [17]:
```
#Importing of packages

import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import numpy as np
import warnings
import pickle
import pydotplus

from sklearn.linear_model import (
    LinearRegression,
    Ridge,
    Lasso,
    LogisticRegression,
    SGDClassifier,
    BayesianRidge,
)
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn import tree
from six import StringIO
from IPython.display import Image
from sklearn.ensemble import (
    RandomForestRegressor,
    GradientBoostingRegressor,
    BaggingClassifier,
    GradientBoostingClassifier,
    RandomForestClassifier,
)
from sklearn.svm import LinearSVC
from numpy.ma.core import sqrt
from sklearn.model_selection import GridSearchCV, cross_val_score, train_test_
from numpy.polynomial.polynomial import polyfit
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    mean_squared_error,
    r2_score,
    mean_absolute_error,
    confusion_matrix
)
# Suppress FutureWarnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [18]:
```
df = pd.read_csv('StudentInfo.csv')
```

In [19]:
```
df.drop(df[df['G3'] < 1].index, inplace = True)
```

In [66]:
```
df.columns
```

Out[66]:
```
Index(['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu',
       'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime',
       'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery',
       'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc',
       'Walc', 'health', 'absences', 'G1', 'G2', 'G3'],
      dtype='object')
```

One Hot encoding

In [67]:
```
#df_ohe = pd.get_dummies(df,columns = ['sex'],drop_first=False)
```

In [69]:
```
#df_ohe.columns
```

In [70]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Assuming df is your DataFrame

# Identify categorical columns
categorical_columns = df.select_dtypes(include=['object']).columns

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Loop through each categorical column and label encode
for column in categorical_columns:
    df[column + '_encoded'] = label_encoder.fit_transform(df[column])

# Display the DataFrame with label-encoded columns
print(df)
```

```
     school sex  age address famsize Pstatus  Medu  Fedu      Mjob      Fjob
\
0        GP   F   18       U     GT3       A     4     4   at_home   teacher
1        GP   F   17       U     GT3       T     1     1   at_home     other
2        GP   F   15       U     LE3       T     1     1   at_home     other
3        GP   F   15       U     GT3       T     4     2    health  services
4        GP   F   16       U     GT3       T     3     3     other     other
...     ...  ..  ...     ...     ...     ...   ...   ...       ...       ...
1039    SLA   F   19       R     GT3       T     2     3  services     other
1040    SLA   F   18       U     LE3       T     3     1   teacher  services
1041    SLA   F   18       U     GT3       T     1     1     other     other
1042    SLA   M   17       U     LE3       T     3     1  services  services
1043    SLA   M   18       R     LE3       T     3     2  services     other

      ... reason_encoded guardian_encoded  schoolsup_encoded  famsup_encoded
\
0     ...              0                1                  1               0
1     ...              0                0                  0               1
2     ...              2                1                  1               0
3     ...              1                1                  0               1
4     ...              1                0                  0               1
...   ...            ...              ...                ...             ...
1039  ...              0                1                  0               0
1040  ...              0                1                  0               1
1041  ...              0                1                  0               0
1042  ...              0                1                  0               0
1043  ...              0                1                  0               0

      paid_encoded activities_encoded nursery_encoded higher_encoded  \
0                0                  0               1              1
1                0                  0               0              1
2                1                  0               1              1
3                1                  1               1              1
4                1                  0               1              1
...            ...                ...             ...            ...
1039             0                  1               0              1
1040             0                  0               1              1
1041             0                  1               1              1
1042             0                  0               0              1
1043             0                  0               0              1

     internet_encoded romantic_encoded
0                   0                 0
1                   1                 0
2                   1                 0
3                   1                 1
4                   0                 0
...               ...               ...
1039                1                 0
1040                1                 0
1041                0                 0
1042                1                 0
1043                1                 0

[991 rows x 50 columns]
```

In [75]:
```
# Assuming df is your original DataFrame with encoded columns
# Example: df = pd.DataFrame({'Category': ['A', 'B', 'A', 'C', 'B']})
# (Assuming you have already applied label encoding)

# Filter columns ending with '_encoded'
encoded_columns = [col for col in df.columns if col.endswith('_encoded')]

# Create a new DataFrame with only the encoded columns
df_ohe = df[encoded_columns]

# Display the new DataFrame
print(df_ohe)
```

```
      school_encoded  sex_encoded  address_encoded  famsize_encoded  \
0                  0            0                1                0
1                  0            0                1                0
2                  0            0                1                1
3                  0            0                1                0
4                  0            0                1                0
...              ...          ...              ...              ...
1039               3            0                0                0
1040               3            0                1                1
1041               3            0                1                0
1042               3            1                1                1
1043               3            1                0                1

      Pstatus_encoded  Mjob_encoded  Fjob_encoded  reason_encoded  \
0                   0             0             4               0
1                   1             0             2               0
2                   1             0             2               2
3                   1             1             3               1
4                   1             2             2               1
...               ...           ...           ...             ...
1039                1             3             2               0
1040                1             4             3               0
1041                1             2             2               0
1042                1             3             3               0
1043                1             3             2               0

      guardian_encoded  schoolsup_encoded  famsup_encoded  paid_encoded  \
0                    1                  1               0             0
1                    0                  0               1             0
2                    1                  1               0             1
3                    1                  0               1             1
4                    0                  0               1             1
...                ...                ...             ...           ...
1039                 1                  0               0             0
1040                 1                  0               1             0
1041                 1                  0               0             0
1042                 1                  0               0             0
1043                 1                  0               0             0

      activities_encoded  nursery_encoded  higher_encoded  internet_encoded
\
0                      0                1               1                 0
1                      0                0               1                 1
2                      0                1               1                 1
3                      1                1               1                 1
4                      0                1               1                 0
...                  ...              ...             ...               ...
1039                   1                0               1                 1
1040                   0                1               1                 1
1041                   1                1               1                 0
1042                   0                0               1                 1
1043                   0                0               1                 1

      romantic_encoded
0                    0
1                    0
2                    0
```

```
3                    1
4                    0
...                ...
1039                 0
1040                 0
1041                 0
1042                 0
1043                 0

[991 rows x 17 columns]
```

In [80]:
```
import pandas as pd

# Assuming df is your original DataFrame with numerical columns
# Example: df = pd.DataFrame({'Numeric1': [1, 2, 3], 'Numeric2': [4, 5, 6]})

# Assuming df_ohe is your DataFrame with encoded columns
# Example: df_ohe = pd.DataFrame({'Category_encoded': [0, 1, 0]})

# Identify numerical columns in the original DataFrame
numerical_columns = df.select_dtypes(include=['number']).columns

# Concatenate numerical columns from df to df_ohe
df_ohe = pd.concat([df_ohe, df[numerical_columns]], axis=1)

# Display the updated DataFrame
print(df_ohe)
```

```
      school  sex  address  famsize  Pstatus  Mjob  Fjob  reason  guardian
\
0          0    0        1        0        0     0     4       0         1
1          0    0        1        0        1     0     2       0         0
2          0    0        1        1        1     0     2       2         1
3          0    0        1        0        1     1     3       1         1
4          0    0        1        0        1     2     2       1         0
...      ...  ...      ...      ...      ...   ...   ...     ...       ...
1039       3    0        0        0        1     3     2       0         1
1040       3    0        1        1        1     4     3       0         1
1041       3    0        1        0        1     2     2       0         1
1042       3    1        1        1        1     3     3       0         1
1043       3    1        0        1        1     3     2       0         1

      schoolsup  ...  reason_encoded  guardian_encoded  schoolsup_encoded  \
0             1  ...               0                 1                  1
1             0  ...               0                 0                  0
2             1  ...               2                 1                  1
3             0  ...               1                 1                  0
4             0  ...               1                 0                  0
...         ...  ...             ...               ...                ...
1039          0  ...               0                 1                  0
1040          0  ...               0                 1                  0
1041          0  ...               0                 1                  0
1042          0  ...               0                 1                  0
1043          0  ...               0                 1                  0

      famsup_encoded  paid_encoded  activities_encoded  nursery_encoded  \
0                  0             0                   0                1
1                  1             0                   0                0
2                  0             1                   0                1
3                  1             1                   1                1
4                  1             1                   0                1
...              ...           ...                 ...              ...
1039               0             0                   1                0
1040               1             0                   0                1
1041               0             0                   1                1
1042               0             0                   0                0
1043               0             0                   0                0

      higher_encoded  internet_encoded  romantic_encoded
0                  1                 0                 0
1                  1                 1                 0
2                  1                 1                 0
3                  1                 1                 1
4                  1                 0                 0
...              ...               ...               ...
1039               1                 1                 0
1040               1                 1                 0
1041               1                 0                 0
1042               1                 1                 0
1043               1                 1                 0

[991 rows x 50 columns]
```

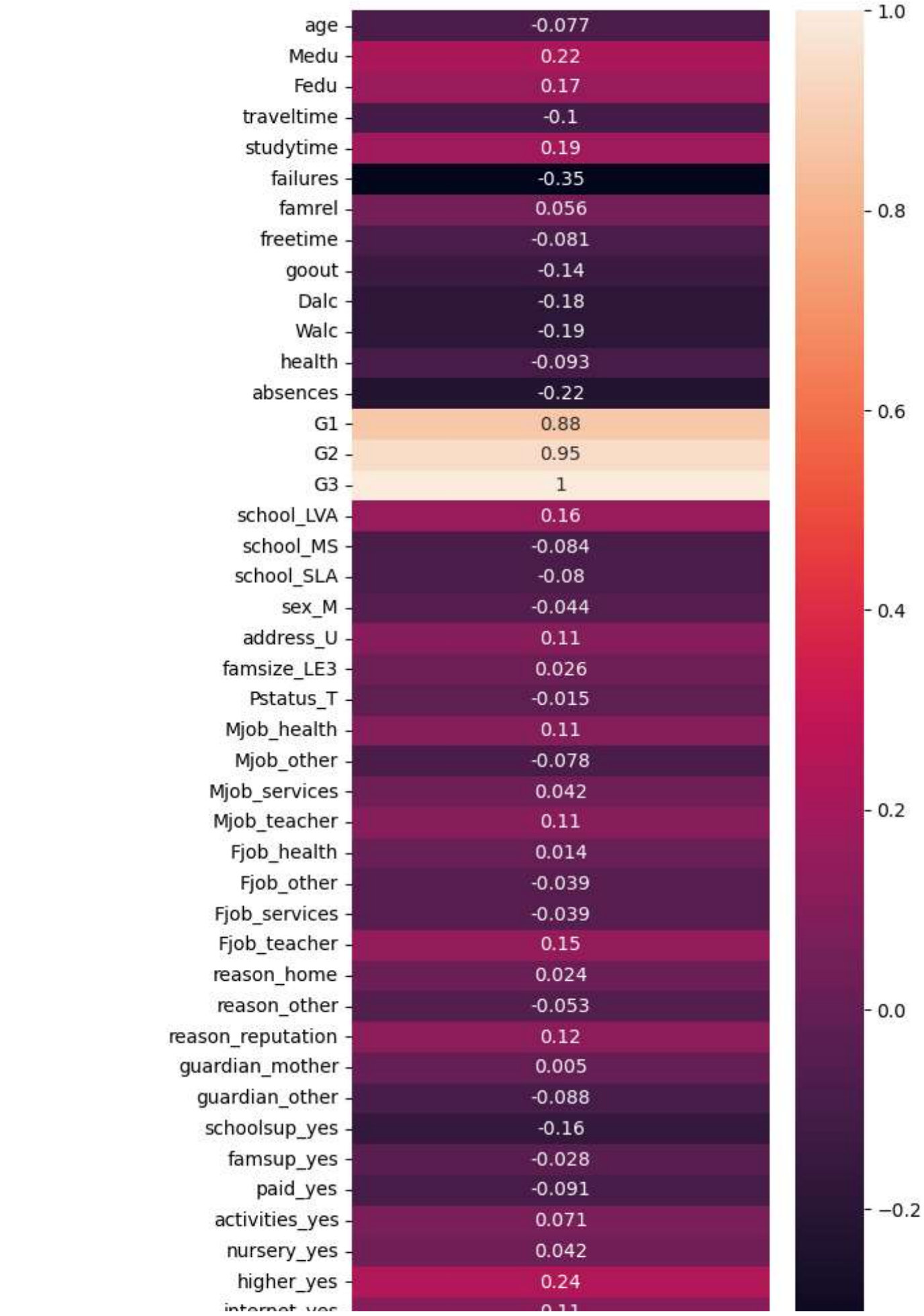
In [81]:
```
print(df_ohe.dtypes)
```

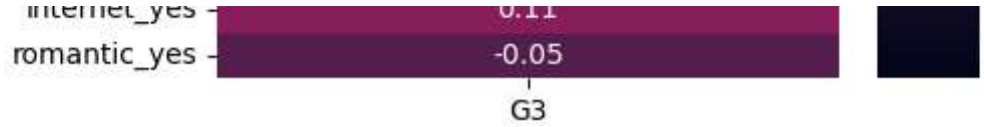
```
school                int64
sex                   int64
address               int64
famsize               int64
Pstatus               int64
Mjob                  int64
Fjob                  int64
reason                int64
guardian              int64
schoolsup             int64
famsup                int64
paid                  int64
activities            int64
nursery               int64
higher                int64
internet              int64
romantic              int64
age                   int64
Medu                  int64
Fedu                  int64
traveltime            int64
studytime             int64
failures              int64
famrel                int64
freetime              int64
goout                 int64
Dalc                  int64
Walc                  int64
health                int64
absences              int64
G1                    int64
G2                    int64
G3                    int64
school_encoded        int64
sex_encoded           int64
address_encoded       int64
famsize_encoded       int64
Pstatus_encoded       int64
Mjob_encoded          int64
Fjob_encoded          int64
reason_encoded        int64
guardian_encoded      int64
schoolsup_encoded     int64
famsup_encoded        int64
paid_encoded          int64
activities_encoded    int64
nursery_encoded       int64
higher_encoded        int64
internet_encoded      int64
romantic_encoded      int64
dtype: object
```

In [77]:
```
# Assuming df_ohe is your DataFrame with encoded columns
# Example: df_ohe = df[['Category_encoded']]

# Remove "encoded" from column names
df_ohe.columns = df_ohe.columns.str.replace('_encoded', '')

# Display the DataFrame with updated column names
print(df_ohe)
```

```
      school  sex  address  famsize  Pstatus  Mjob  Fjob  reason  guardian
\
0          0    0        1        0        0     0     4       0         1
1          0    0        1        0        1     0     2       0         0
2          0    0        1        1        1     0     2       2         1
3          0    0        1        0        1     1     3       1         1
4          0    0        1        0        1     2     2       1         0
...      ...  ...      ...      ...      ...   ...   ...     ...       ...
1039       3    0        0        0        1     3     2       0         1
1040       3    0        1        1        1     4     3       0         1
1041       3    0        1        0        1     2     2       0         1
1042       3    1        1        1        1     3     3       0         1
1043       3    1        0        1        1     3     2       0         1

      schoolsup  famsup  paid  activities  nursery  higher  internet  romant
ic
0             1       0     0           0        1       1         0
0
1             0       1     0           0        0       1         1
0
2             1       0     1           0        1       1         1
0
3             0       1     1           1        1       1         1
1
4             0       1     1           0        1       1         0
0
...         ...     ...   ...         ...      ...     ...       ...
...
1039          0       0     0           1        0       1         1
0
1040          0       1     0           0        1       1         1
0
1041          0       0     0           1        1       1         0
0
1042          0       0     0           0        0       1         1
0
1043          0       0     0           0        0       1         1
0

[991 rows x 17 columns]
```

In [21]:
```
# Calculate the correlation matrix for all columns
correlation_matrix = df_ohe.corr()

# Extract the correlation values for the 'G3' column
correlation_with_G3 = correlation_matrix['G3']

# Create a heatmap of the correlation values
plt.figure(figsize=(5, 13))
sns.heatmap(correlation_with_G3.to_frame(), annot=True, cbar=True)
plt.show()
```


age -0.077
Medu 0.22
Fedu 0.17
traveltime -0.1
studytime 0.19
failures -0.35
famrel 0.056
freetime -0.081
goout -0.14
Dalc -0.18
Walc -0.19
health -0.093
absences -0.22
G1 0.88
G2 0.95
G3 1
school_LVA 0.16
school_MS -0.084
school_SLA -0.08
sex_M -0.044
address_U 0.11
famsize_LE3 0.026
Pstatus_T -0.015
Mjob_health 0.11
Mjob_other -0.078
Mjob_services 0.042
Mjob_teacher 0.11
Fjob_health 0.014
Fjob_other -0.039
Fjob_services -0.039
Fjob_teacher 0.15
reason_home 0.024
reason_other -0.053
reason_reputation 0.12
guardian_mother 0.005
guardian_other -0.088
schoolsup_yes -0.16
famsup_yes -0.028
paid_yes -0.091
activities_yes 0.071
nursery_yes 0.042
higher_yes 0.24
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



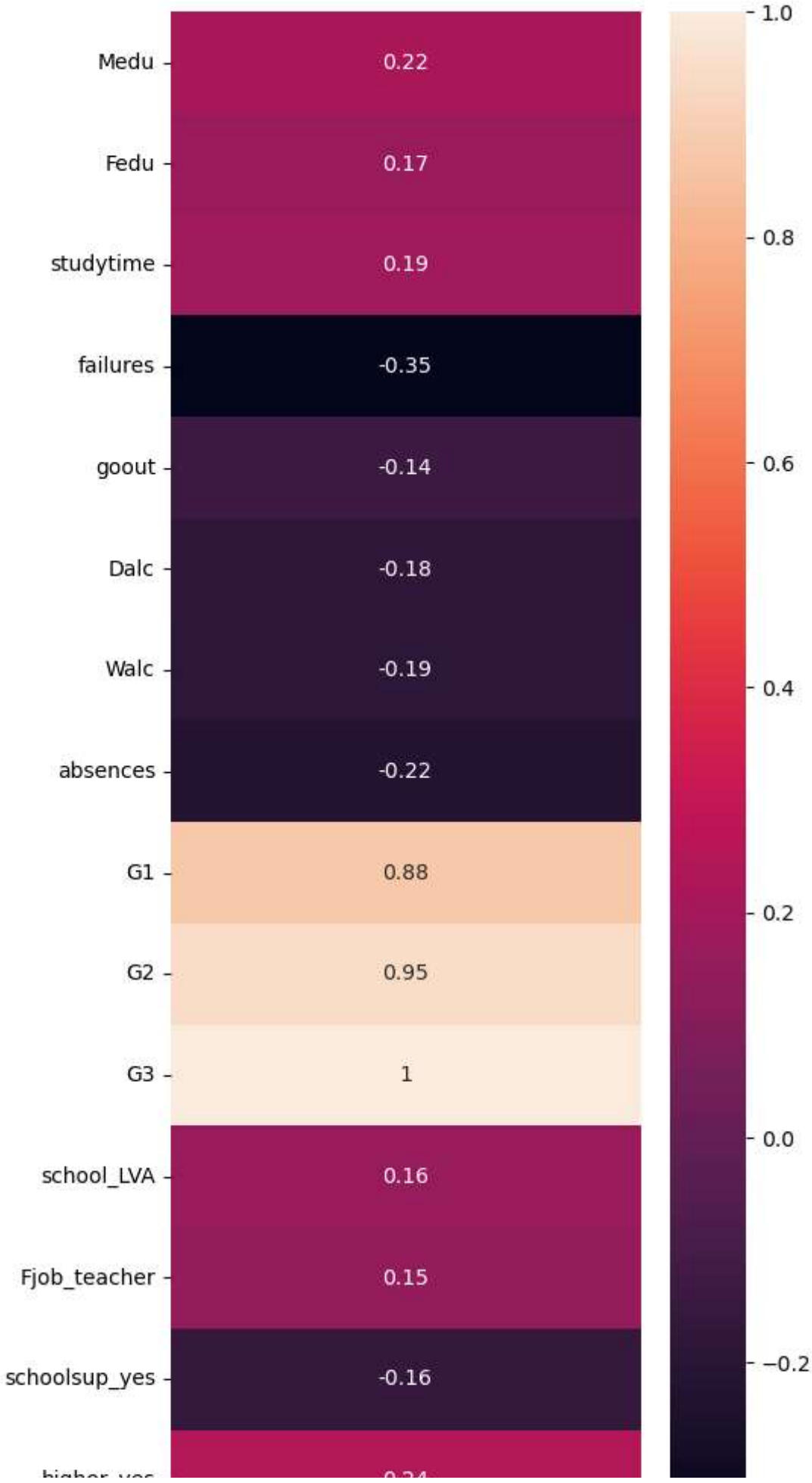
In [22]:
```
THRESHOLD = 0.13
```

In [23]:
```
G3_corr = df_ohe.corr()["G3"]
```

In [24]:
```
df_ohe_after_drop_features = df_ohe.copy()
```

In [25]:
```
for key, value in G3_corr.iteritems():
  if abs(value) < THRESHOLD:
    df_ohe_after_drop_features.drop(columns= key, inplace=True)
```

In [26]:
```
# Calculate the correlation matrix for all columns
correlation_matrix = df_ohe_after_drop_features.corr()

# Extract the correlation values for the 'G3' column
correlation_with_G3 = correlation_matrix['G3']

# Create a heatmap of the correlation values
plt.figure(figsize=(5, 13))
sns.heatmap(correlation_with_G3.to_frame(), annot=True, cbar=True)
plt.show()
```


Medu
0.22
Fedu
0.17
studytime
0.19
failures
-0.35
goout
-0.14
Dalc
-0.18
Walc
-0.19
absences
-0.22
G1
0.88
G2
0.95
G3
1
school_LVA
0.16
Fjob_teacher
0.15
schoolsup_yes
-0.16
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

In [27]:
```
X = df_ohe_after_drop_features.drop('G3',axis = 1)
y = df_ohe_after_drop_features['G3']
```

In [28]:
```
df_ohe_after_drop_features.head()
```

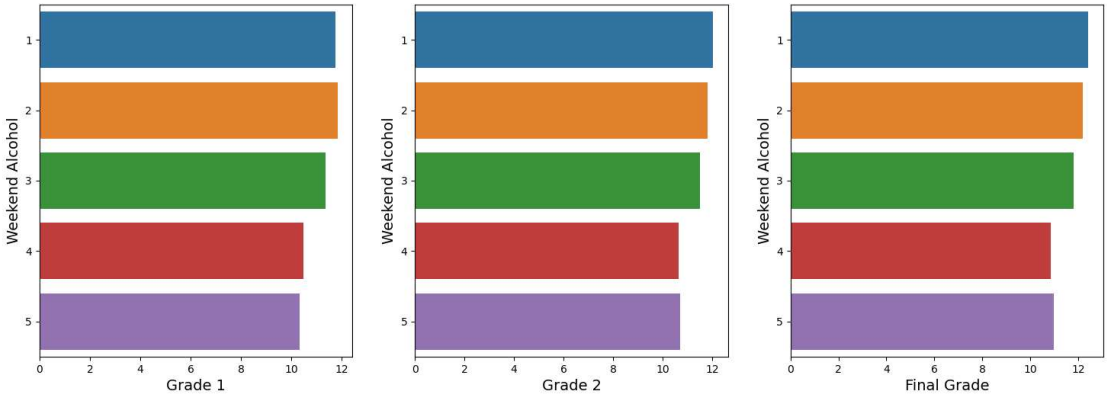
Out[28]:

	Medu	Fedu	studytime	failures	goout	Dalc	Walc	absences	G1	G2	G3	school_LVA	Fj
0	4	4	2	0	4	1	1	6	5	6	6	0	
1	1	1	2	0	3	1	1	4	5	5	6	0	
2	1	1	2	3	2	2	3	10	7	8	10	0	
3	4	2	3	0	2	1	1	2	15	14	15	0	
4	3	3	2	0	2	1	2	4	6	10	10	0	

Effect of alchohol (weekend)

In [87]:
```
# Create a 1x3 grid of subplots
fig, axs = plt.subplots(1, 3, figsize=(18, 6))
bars = [
    "0 - Very Low",
    "1 - Low",
    "2 - Moderate",
    "3 - High",
    "4 - Very High"
]


# Create the first horizontal bar plot
b1 = sns.barplot(x='G1', y='Walc', data=df_ohe, ci=None, ax=axs[0], orient='h
b1.set_ylabel('Weekend Alcohol', fontsize=14)
b1.set_xlabel('Grade 1', fontsize=14)
b1.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF

# Create the second horizontal bar plot
b2 = sns.barplot(x='G2', y='Walc', data=df_ohe, ci=None, ax=axs[1], orient='h
b2.set_ylabel('Weekend Alcohol', fontsize=14)
b2.set_xlabel('Grade 2', fontsize=14)
b2.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF


# Create the third horizontal bar plot
b3 = sns.barplot(x='G3', y='Walc', data=df_ohe, ci=None, ax=axs[2], orient='h
b3.set_ylabel('Weekend Alcohol', fontsize=14)
b3.set_xlabel('Final Grade', fontsize=14)
b3.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF

# Adjust layout
plt.tight_layout()

# Show the plots
plt.show()
```

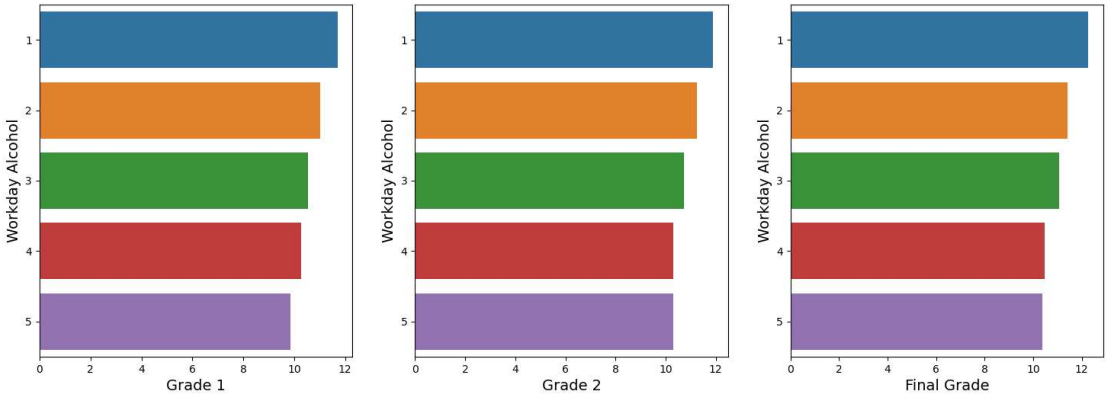
```
/var/folders/my/gyxc13cs3l9cxl2m7_zp6sdw0000gn/T/ipykernel_12248/486987469.p
y:32: UserWarning: Tight layout not applied. tight_layout cannot make axes w
idth small enough to accommodate all axes decorations
  plt.tight_layout()
```



0 - Very Low
1 - Low
2 - Moderate
3 - High
4 - Very High

In [51]:
```
# Create a 1x3 grid of subplots
fig, axs = plt.subplots(1, 3, figsize=(18, 6))
bars = [
    "0 - Very Low",
    "1 - Low",
    "2 - Moderate",
    "3 - High",
    "4 - Very High"
]


# Create the first horizontal bar plot
b1 = sns.barplot(x='G1', y='Dalc', data=df_ohe, ci=None, ax=axs[0], orient='h
b1.set_ylabel('Workday Alcohol', fontsize=14)
b1.set_xlabel('Grade 1', fontsize=14)
b1.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF

# Create the second horizontal bar plot
b2 = sns.barplot(x='G2', y='Dalc', data=df_ohe, ci=None, ax=axs[1], orient='h
b2.set_ylabel('Workday Alcohol', fontsize=14)
b2.set_xlabel('Grade 2', fontsize=14)
b2.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF


# Create the third horizontal bar plot
b3 = sns.barplot(x='G3', y='Dalc', data=df_ohe, ci=None, ax=axs[2], orient='h
b3.set_ylabel('Workday Alcohol', fontsize=14)
b3.set_xlabel('Final Grade', fontsize=14)
b3.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF

# Adjust layout
plt.tight_layout()

# Show the plots
plt.show()
```

```
/var/folders/my/gyxc13cs3l9cxl2m7_zp6sdw0000gn/T/ipykernel_12248/2557201993.
py:32: UserWarning: Tight layout not applied. tight_layout cannot make axes
width small enough to accommodate all axes decorations
  plt.tight_layout()
```

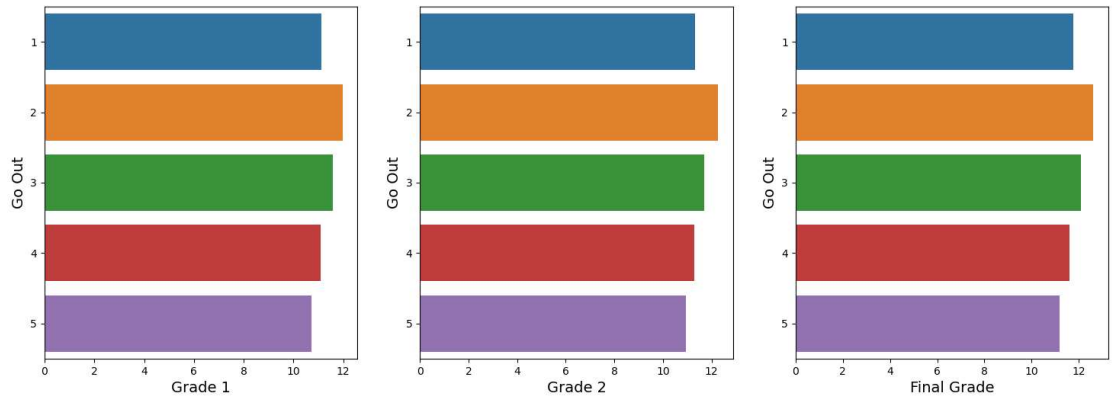


0 - Very Low
1 - Low
2 - Moderate
3 - High
4 - Very High

Effect of Social life on Quality of education

In [82]:
```
# Create a 1x3 grid of subplots
fig, axs = plt.subplots(1, 3, figsize=(18, 6))
bars = [
    "0 - Very Low",
    "1 - Low",
    "2 - Moderate",
    "3 - High",
    "4 - Very High"
]


# Create the first horizontal bar plot
b1 = sns.barplot(x='G1', y='goout', data=df_ohe, ci=None, ax=axs[0], orient='
b1.set_ylabel('Go Out', fontsize=14)
b1.set_xlabel('Grade 1', fontsize=14)
b1.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF

# Create the second horizontal bar plot
b2 = sns.barplot(x=df['G2'], y=df['goout'], data=df, ci=None, ax=axs[1], orie
b2.set_ylabel('Go Out', fontsize=14)
b2.set_xlabel('Grade 2', fontsize=14)
b2.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF


# Create the third horizontal bar plot
b3 = sns.barplot(x='G3', y='goout', data=df_ohe, ci=None, ax=axs[2], orient='
b3.set_ylabel('Go Out', fontsize=14)
b3.set_xlabel('Final Grade', fontsize=14)
b3.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF

# Adjust layout
plt.tight_layout()

# Show the plots
plt.show()
```

```
/var/folders/my/gyxc13cs3l9cxl2m7_zp6sdw0000gn/T/ipykernel_12248/2249051947.
py:32: UserWarning: Tight layout not applied. tight_layout cannot make axes
width small enough to accommodate all axes decorations
  plt.tight_layout()
```



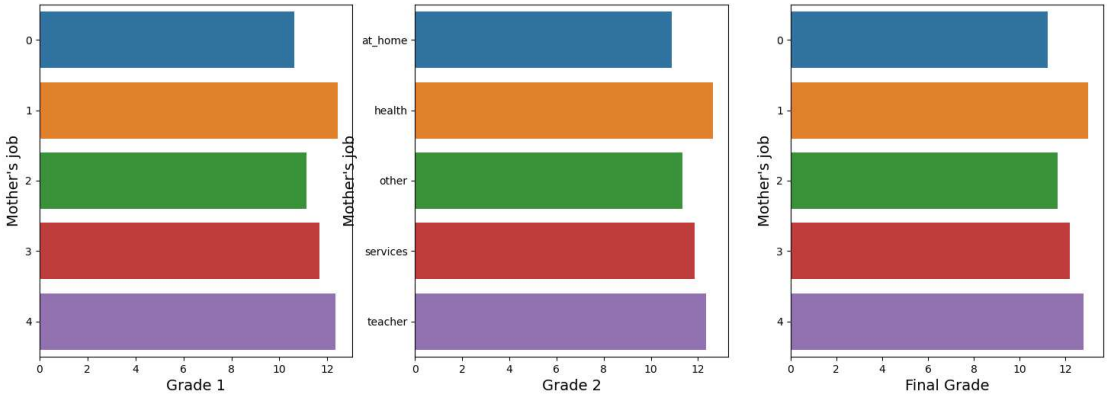
0 - Very Low
1 - Low
2 - Moderate
3 - High
4 - Very High

Effect of parents jobs on the quality of education of students

Mother's job

In [85]:
```
# Create a 1x3 grid of subplots
fig, axs = plt.subplots(1, 3, figsize=(18, 6))
bars = [
    "0 - Very Low",
    "1 - Low",
    "2 - Moderate",
    "3 - High",
    "4 - Very High"
]


# Create the first horizontal bar plot
b1 = sns.barplot(x='G1', y='Mjob', data=df_ohe, ci=None, ax=axs[0], orient='h
b1.set_ylabel("Mother's job", fontsize=14)
b1.set_xlabel('Grade 1', fontsize=14)
b1.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF

# Create the second horizontal bar plot
b2 = sns.barplot(x=df['G2'], y=df['Mjob'], data=df, ci=None, ax=axs[1], orient
b2.set_ylabel("Mother's job", fontsize=14)
b2.set_xlabel('Grade 2', fontsize=14)
b2.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF


# Create the third horizontal bar plot
b3 = sns.barplot(x='G3', y='Mjob', data=df_ohe, ci=None, ax=axs[2], orient='h
b3.set_ylabel("Mother's job", fontsize=14)
b3.set_xlabel('Final Grade', fontsize=14)
b3.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF

# Adjust layout
plt.tight_layout()

# Show the plots
plt.show()
```

```
/var/folders/my/gyxc13cs3l9cxl2m7_zp6sdw0000gn/T/ipykernel_12248/2095971424.
py:32: UserWarning: Tight layout not applied. tight_layout cannot make axes
width small enough to accommodate all axes decorations
  plt.tight_layout()
```

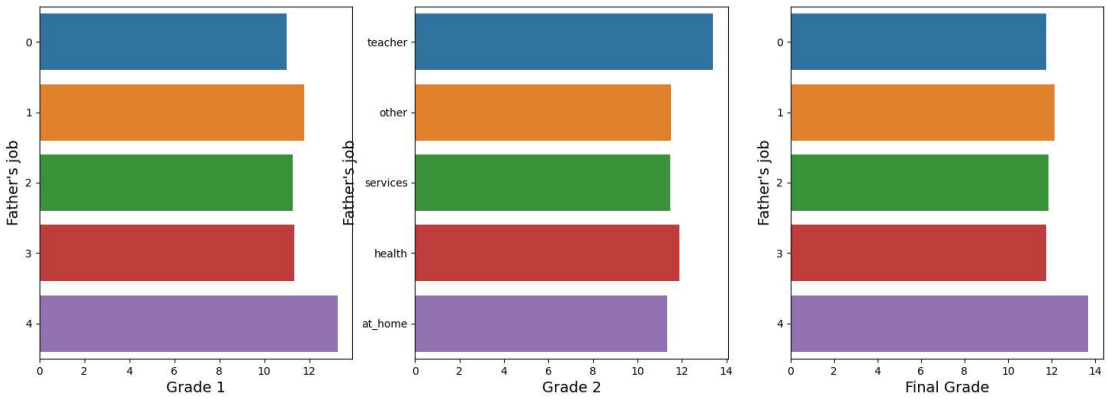


0 - Very Low
1 - Low
2 - Moderate
3 - High
4 - Very High

Father's job

In [86]:
```
# Create a 1x3 grid of subplots
fig, axs = plt.subplots(1, 3, figsize=(18, 6))
bars = [
    "0 - Very Low",
    "1 - Low",
    "2 - Moderate",
    "3 - High",
    "4 - Very High"
]


# Create the first horizontal bar plot
b1 = sns.barplot(x='G1', y='Fjob', data=df_ohe, ci=None, ax=axs[0], orient='h
b1.set_ylabel("Father's job", fontsize=14)
b1.set_xlabel('Grade 1', fontsize=14)
b1.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF

# Create the second horizontal bar plot
b2 = sns.barplot(x=df['G2'], y=df['Fjob'], data=df, ci=None, ax=axs[1], orient
b2.set_ylabel("Father's job", fontsize=14)
b2.set_xlabel('Grade 2', fontsize=14)
b2.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF


# Create the third horizontal bar plot
b3 = sns.barplot(x='G3', y='Fjob', data=df_ohe, ci=None, ax=axs[2], orient='h
b3.set_ylabel("Father's job", fontsize=14)
b3.set_xlabel('Final Grade', fontsize=14)
b3.text(0.05, -0.35, '\n'.join(bars), fontsize=12, transform=plt.gcf().transF

# Adjust layout
plt.tight_layout()

# Show the plots
plt.show()
```

```
/var/folders/my/gyxc13cs3l9cxl2m7_zp6sdw0000gn/T/ipykernel_12248/260497475.p
y:32: UserWarning: Tight layout not applied. tight_layout cannot make axes w
idth small enough to accommodate all axes decorations
  plt.tight_layout()
```



0 - Very Low
1 - Low
2 - Moderate
3 - High
4 - Very High

In [29]:
```
X_all_features_except_G3 = df_ohe.drop('G3',axis = 1)
y_G3 = df_ohe ['G3']
```

In [30]:
```
def train_regression_model(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, s

    model1 = LinearRegression()
    model2 = BayesianRidge()
    model3 = RandomForestRegressor()
    model4 = GradientBoostingRegressor()
    model5 = DecisionTreeRegressor()
    model6 = Ridge()
    model7 = Lasso()

    models = [model1, model2, model3, model4, model5, model6, model7 ]
    model_name_list = ['LinearRegression', 'BayesianRidge', 'RandomForestRegre
           'DecisionTreeRegressor', 'Ridge', 'Lasso']

    # Dataframe for results
    results = pd.DataFrame(columns=['MAE', 'RMSE', 'RMSE by cross validation'

    for i, model in enumerate(models):
        # Train the model
        model.fit(X_train, y_train)

        # Make predictions on the test set
        y_test_pred = model.predict(X_test)

        # Calculate evaluation metrics
        mse = mean_squared_error(y_test, y_test_pred)
        mae = mean_absolute_error(y_test, y_test_pred)
        rmse = np.sqrt(mse)
        r_squared = r2_score(y_test, y_test_pred)

        # Cross-validation
        scores = cross_val_score(model, X_test, y_test, scoring='neg_mean_squa
        rmse_cross_val = np.sqrt(-scores.mean())

        model_name = model_name_list[i]
        results.loc[model_name, :] = [mae, rmse, rmse_cross_val ,mse,r_squared

    return results
```

In [31]:
```
train_regression_model(X, y)
```

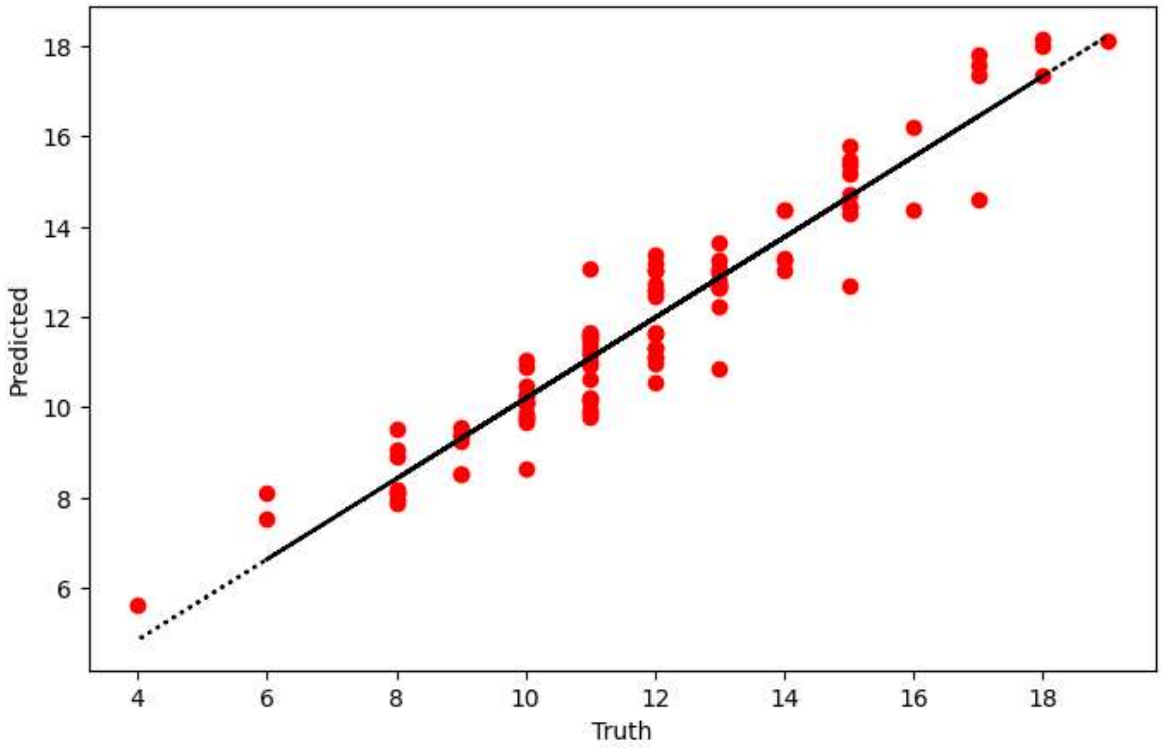
Out[31]:

	MAE	RMSE	RMSE by cross validation	MSE	R^2
LinearRegression	0.672675	0.859612	0.991129	0.738932	0.906293
BayesianRidge	0.672036	0.859694	0.964315	0.739073	0.906276
RandomForestRegressor	0.741	0.916331	1.03639	0.839662	0.89352
GradientBoostingRegressor	0.652198	0.84234	1.133341	0.709536	0.910021
DecisionTreeRegressor	0.95	1.252996	1.590597	1.57	0.800903
Ridge	0.67261	0.85961	0.985111	0.73893	0.906294
Lasso	0.737147	0.97954	1.032126	0.959498	0.878323

In [32]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, shuf

best_model4 = GradientBoostingRegressor()
best_model4.fit(X_train, y_train)

y_test_pred = best_model4.predict(X_test)
```

In [33]:
```
n,m=polyfit(y_test, y_test_pred, 1)
plt.figure(figsize=(8,5))
plt.scatter(x = y_test, y = y_test_pred, c="red")
plt.plot(y_test,  m*(y_test) + n   ,':', c="black")

plt.xlabel("Truth")
plt.ylabel("Predicted")
```

Out[33]:
```
Text(0, 0.5, 'Predicted')
```



In [36]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Assuming y_test contains the true target values
# and y_test_pred contains the predicted values

# Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_test_pred)
print("Mean Absolute Error (MAE):", mae)

# Root Mean Squared Error (RMSE)
rmse = mean_squared_error(y_test, y_test_pred, squared=False)
print("Root Mean Squared Error (RMSE):", rmse)

# R-squared (R^2)
r2 = r2_score(y_test, y_test_pred)
print("R-squared (R^2):", r2)
```

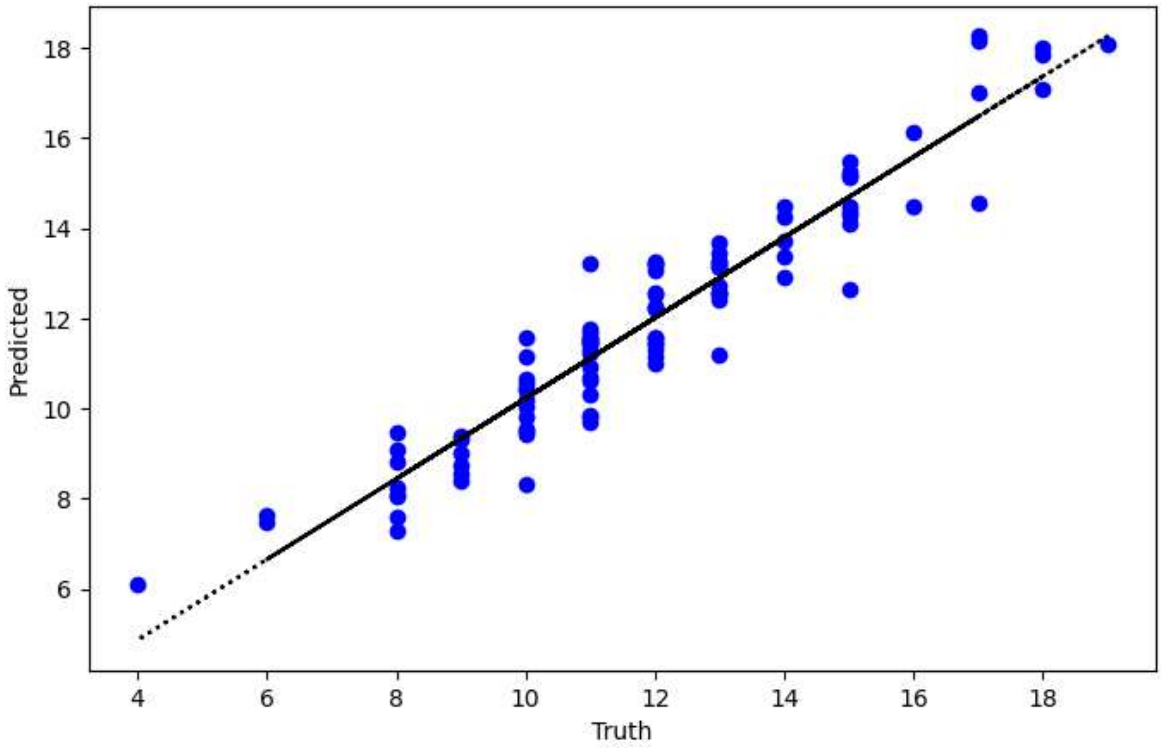
```
Mean Absolute Error (MAE): 0.6534590677286011
Root Mean Squared Error (RMSE): 0.8438385589553595
R-squared (R^2): 0.9097007819849019
```

In [35]:
```
with open('reg_model.pkl', 'wb') as file:
    pickle.dump(best_model4, file)
```

In [37]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, shuf

best_model2 = BayesianRidge()
best_model2.fit(X_train, y_train)

y_test_pred = best_model2.predict(X_test)
```

In [45]:
```
n,m=polyfit(y_test, y_test_pred, 1)
plt.figure(figsize=(8,5))
plt.scatter(x = y_test, y = y_test_pred, c="blue")
plt.plot(y_test,  m*(y_test) + n   ,':', c="black")

plt.xlabel("Truth")
plt.ylabel("Predicted")
```

Out[45]:
```
Text(0, 0.5, 'Predicted')
```



In [39]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Assuming y_test contains the true target values
# and y_test_pred contains the predicted values

# Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_test_pred)
print("Mean Absolute Error (MAE):", mae)

# Root Mean Squared Error (RMSE)
rmse = mean_squared_error(y_test, y_test_pred, squared=False)
print("Root Mean Squared Error (RMSE):", rmse)

# R-squared (R^2)
r2 = r2_score(y_test, y_test_pred)
print("R-squared (R^2):", r2)
```

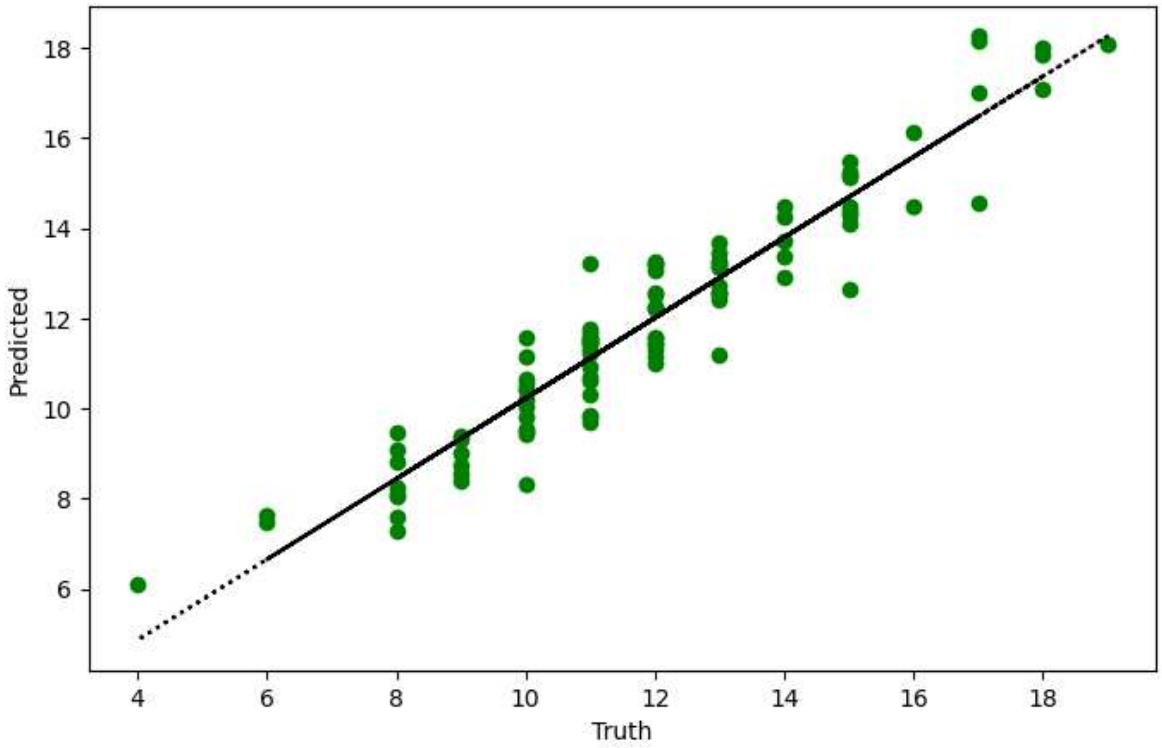
```
Mean Absolute Error (MAE): 0.6720363837531744
Root Mean Squared Error (RMSE): 0.8596936791631052
R-squared (R^2): 0.9062755881615862
```

In [40]:
```
with open('reg_model.pkl', 'wb') as file:
    pickle.dump(best_model2, file)
```

In [41]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, shuf

best_model1 = LinearRegression()
best_model1.fit(X_train, y_train)

y_test_pred = best_model1.predict(X_test)
```

In [46]:
```
n,m=polyfit(y_test, y_test_pred, 1)
plt.figure(figsize=(8,5))
plt.scatter(x = y_test, y = y_test_pred, c="green")
plt.plot(y_test,  m*(y_test) + n   ,':', c="black")

plt.xlabel("Truth")
plt.ylabel("Predicted")
```

Out[46]:
```
Text(0, 0.5, 'Predicted')
```



In [47]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Assuming y_test contains the true target values
# and y_test_pred contains the predicted values

# Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_test_pred)
print("Mean Absolute Error (MAE):", mae)

# Root Mean Squared Error (RMSE)
rmse = mean_squared_error(y_test, y_test_pred, squared=False)
print("Root Mean Squared Error (RMSE):", rmse)

# R-squared (R^2)
r2 = r2_score(y_test, y_test_pred)
print("R-squared (R^2):", r2)
```

```
Mean Absolute Error (MAE): 0.67267522787367
Root Mean Squared Error (RMSE): 0.8596115835808256
R-squared (R^2): 0.9062934875435814
```

In [48]:
```
with open('reg_model.pkl', 'wb') as file:
    pickle.dump(best_model2, file)
```

In [ ]:
```

```