Student Performance Analysis Project

Table of Contents

Introduction

This project analyzes student performance datato understand what influences academic scores. The dataset includes demographics, parental education, lunch types, test prep, and test scores in math, reading, and writing ✍.

#Code Explanation

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, classification_report,
recall_score, f1_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

- Essential libraries for data manipulation, visualization, and machine learning
- Includes preprocessing tools for data preparationist item learning

---

LOAD DATA SET

- `Loads the student performance dataset`
- Displays first and last 5 rows by default

```
# Load the dataset
df = pd.read_csv('/content/StudentsPerformance.csv')
df
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 1000,\n \"fields\": [\n   {\n   \"column\": \"gender\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"male\",\n   \"female\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"race/ethnicity\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 5,\n   \"samples\": [\n   \"group C\",\n   \"group E\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"parental level of education\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 6,\n   \"samples\": [\n   \"bachelor's degree\",\n   \"some college\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"lunch\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"free/reduced\",\n   \"standard\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"test preparation course\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"completed\",\n   \"none\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"math score\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 15,\n   \"min\": 0,\n   \"max\": 100,\n   \"num_unique_values\": 81,\n   \"samples\": [\n   55,\n   72\n   ],\n   \"semantic_type\": \"\",\n

\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"reading score\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 14,\n   \"min\": 17,\n   \"max\": 100,\n   \"num_unique_values\": 72,\n   \"samples\": [\n   78,\n   23\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"writing score\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 15,\n   \"min\": 10,\n   \"max\": 100,\n   \"num_unique_values\": 77,\n   \"samples\": [\n   75,\n   76\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"df"}

========== 1. BASIC INSPECTION ==========

- Displays the first 5 rows of the dataset.
- Helps quickly verify the data format, column names, and initial values.

```
df.head()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 1000,\n \"fields\": [\n   {\n   \"column\": \"gender\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"male\",\n   \"female\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"race/ethnicity\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 5,\n   \"samples\": [\n   \"group C\",\n   \"group E\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"parental level of education\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 6,\n   \"samples\": [\n   \"bachelor's degree\",\n   \"some college\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"lunch\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"free/reduced\",\n   \"standard\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"test preparation course\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"completed\",\n   \"none\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\

```
n    },\n    {\n      \"column\": \"math score\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
15,\n        \"min\": 0,\n        \"max\": 100,\n
\"num_unique_values\": 81,\n        \"samples\": [\n          55,\n
72\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"reading score\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 14,\n        \"min\": 17,\n
\"max\": 100,\n        \"num_unique_values\": 72,\n
\"samples\": [\n          78,\n          23\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"writing score\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
15,\n        \"min\": 10,\n        \"max\": 100,\n
\"num_unique_values\": 77,\n        \"samples\": [\n          75,\n
76\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


- Displays the last 5 rows of the dataset.
- Useful for checking if the dataset is complete and consistent till the end.

```
df.tail()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 5,\n  \"fields\": [\n
{\n      \"column\": \"gender\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"male\",\n          \"female\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"race/ethnicity\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n          \"group
E\",\n          \"group C\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"parental level of education\",\n      \"properties\": {\
n        \"dtype\": \"string\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"master's degree\",\n          \"high
school\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"lunch\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 2,\n        \"samples\": [\n
\"free/reduced\",\n          \"standard\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"test preparation course\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
\"none\",\n          \"completed\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

n   },\n   {\n   \"column\": \"math score\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 11,\n   \"min\": 59,\n   \"max\": 88,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   62,\n   77\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"reading score\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 16,\n   \"min\": 55,\n   \"max\": 99,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   55,\n   86\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"writing score\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 15,\n   \"min\": 55,\n   \"max\": 95,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   55,\n   86\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

- Returns a tuple (rows, columns), here (1000, 8).
- Confirms the dataset has 1000 records and 8 features.

```
df.shape
```

```
(1000, 8)
```

- Provides information about data types, non-null counts, and memory usage.
- Useful for identifying missing values and understanding data structure.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64
 6   reading score                1000 non-null   int64
 7   writing score                1000 non-null   int64
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

- Gives summary statistics (count, mean, std, min, max, quartiles) for numerical columns.
- Helps understand the range and distribution of scores.

```
df.describe()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8,\n  \"fields\": [\n
{\n      \"column\": \"math score\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 335.8676421540409,\n
\"min\": 0.0,\n        \"max\": 1000.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          66.089,\n
66.0,\n          1000.0\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"reading score\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 334.2004716262942,\n
\"min\": 14.60019193725222,\n        \"max\": 1000.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          69.169,\n
70.0,\n          1000.0\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"writing score\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 334.8025670597152,\n
\"min\": 10.0,\n        \"max\": 1000.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          68.054,\n
69.0,\n          1000.0\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```


- Provides summary stats (count, unique, top, freq) for categorical columns.
- Helps identify the most common categories and category count

```
df.describe(include=['object'])
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 4,\n  \"fields\": [\n
{\n      \"column\": \"gender\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 4,\n
\"samples\": [\n          2,\n          \"518\",\n          \"1000\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"race/ethnicity\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          5,\n
\"319\",\n          \"1000\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"parental level of education\",\n      \"properties\": {\
n        \"dtype\": \"string\",\n        \"num_unique_values\": 4,\n
\"samples\": [\n          6,\n          \"226\",\n          \"1000\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"lunch\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 4,\n
```

\"samples\": [\n   2,\n   \"645\",\n   \"1000\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"test preparation course\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 4,\n   \"samples\": [\n   2,\n   \"642\",\n   \"1000\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

========== 2. DATA CLEANING ==========

- Checks for missing values (none found)
- Demonstrates handling missing values (though not needed here)

```
#Finding the missing values
df.isnull().sum()

gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64

df.fillna(df.median(numeric_only=True), inplace=True)

df
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 1000,\n \"fields\": [\n   {\n   \"column\": \"gender\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.4999259204381489,\n   \"min\": 0.0,\n   \"max\": 1.0,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1.0,\n   0.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"race/ethnicity\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 5,\n   \"samples\": [\n   \"group C\",\n   \"group E\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"parental level of education\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 6,\n   \"samples\": [\n   \"bachelor's degree\",\n   \"some college\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"lunch\",\n   \"properties\": {\n   \"dtype\": \"category\",\

```
n        \"num_unique_values\": 2,\n        \"samples\": [\n
\"free/reduced\",\n          \"standard\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"test preparation course\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
\"completed\",\n          \"none\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"math score\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.15163080096009438,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 81,\n        \"samples\": [\n
0.5499999999999999,\n          0.72\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"reading score\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.17590592695484603,\n        \"min\": 0.0,\n        \"max\":
1.0000000000000002,\n        \"num_unique_values\": 72,\n
\"samples\": [\n          0.7349397590361446,\n
0.07228915662650603\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"writing score\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.16884063345410744,\n        \"min\":
0.0,\n        \"max\": 1.0,\n        \"num_unique_values\": 77,\n
\"samples\": [\n          0.7222222222222222,\n
0.7333333333333333\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


- Converts categorical variables to numerical values
- Enables mathematical operations on all features

```
df['gender'] = LabelEncoder().fit_transform(df['gender'])
df['race/ethnicity'] =
LabelEncoder().fit_transform(df['race/ethnicity'])
df['parental level of education'] =
LabelEncoder().fit_transform(df['parental level of education'])
df['lunch'] = LabelEncoder().fit_transform(df['lunch'])
df['test preparation course'] = LabelEncoder().fit_transform(df['test
preparation course'])
df['math score'] = LabelEncoder().fit_transform(df['math score'])
df['reading score'] = LabelEncoder().fit_transform(df['reading
score'])
df['writing score'] = LabelEncoder().fit_transform(df['writing
score'])
```

======STANDARDIZATION====

- Transforms data to have mean=0 and variance=1
- Useful for algorithms assuming normal distribution

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
normalized_data = scaler.fit_transform(df)
print(normalized_data)
```

```
[[0.         0.25       0.2        ... 0.61971831 0.65789474 0.25
]
 [0.         0.5        0.8        ... 0.87323944 0.84210526 0.5
]
 [0.         0.25       0.6        ... 0.94366197 0.90789474 0.25
]
 ...
 [0.         0.5        0.4        ... 0.6056338  0.53947368 0.5
]
 [0.         0.75       0.8        ... 0.70422535 0.69736842 0.75
]
 [0.         0.75       0.8        ... 0.81690141 0.81578947 0.75
]]
```

==== NORMALIZATION USING MINMAXSCALER ====

- Scales all features to range [0,1]
- Preserves relationships while enabling comparison

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
standardized_data = scaler.fit_transform(df)
print(standardized_data)
```

```
[[-0.96462528 -1.01504393 -0.81264039 ...  0.19394105  0.3921326
  -1.01504393]
 [-0.96462528 -0.15044092  0.82795259 ...  1.44607125  1.323248
  -0.15044092]
 [-0.96462528 -1.01504393  0.28108826 ...  1.79388519  1.65578922
  -1.01504393]
 ...
 [-0.96462528 -0.15044092 -0.26577606 ...  0.12437827 -0.20644159
  -0.15044092]
 [-0.96462528  0.71416208  0.82795259 ...  0.61131778  0.59165733
   0.71416208]
```

```
[-0.96462528  0.71416208  0.82795259 ...  1.16782009  1.19023152
   0.71416208]]
```

====EXPLORATORY DATA ANALYSIS====

===FINDING CORRELATION===

- `Calculates pairwise correlations between variables`
- Helps identify relationships between features

```
df_corr = df.corr()
df_corr
```

{"summary":"{\n \"name\": \"df_corr\",\n \"rows\": 9,\n \"fields\": [\n   {\n   \"column\": \"gender\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.3770936634102204,\n   \"min\": -0.30402375114881797,\n   \"max\": 1.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   -0.0015019243791521083,\n   0.16695459657464118,\n   1.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"race/ethnicity\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.41709563002306654,\n   \"min\": -0.03194564692573896,\n   \"max\": 1.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   1.0,\n   0.2171598878447463,\n   -0.0015019243791521083\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"parental level of education\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.34721366526040887,\n   \"min\": -0.0828728906507834,\n   \"max\": 1.0,\n   \"num_unique_values\": 9,\n   \"samples\": [\n   -0.0828728906507834,\n   -0.03194564692573896,\n   -0.06652543308558444\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"lunch\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.31871894840094844,\n   \"min\": 0.006319907050482893,\n   \"max\": 1.0,\n   \"num_unique_values\": 9,\n   \"samples\": [\n   0.24461008998653624,\n   0.0465625895477168,\n   0.3509452611612282\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"test preparation course\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.3847373166946836,\n   \"min\": -0.31363349993845463,\n   \"max\": 1.0,\n   \"num_unique_values\": 9,\n   \"samples\": [\n   -0.31363349993845463,\n   -0.017508038014672662,\n   -0.17749134236860675\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\":

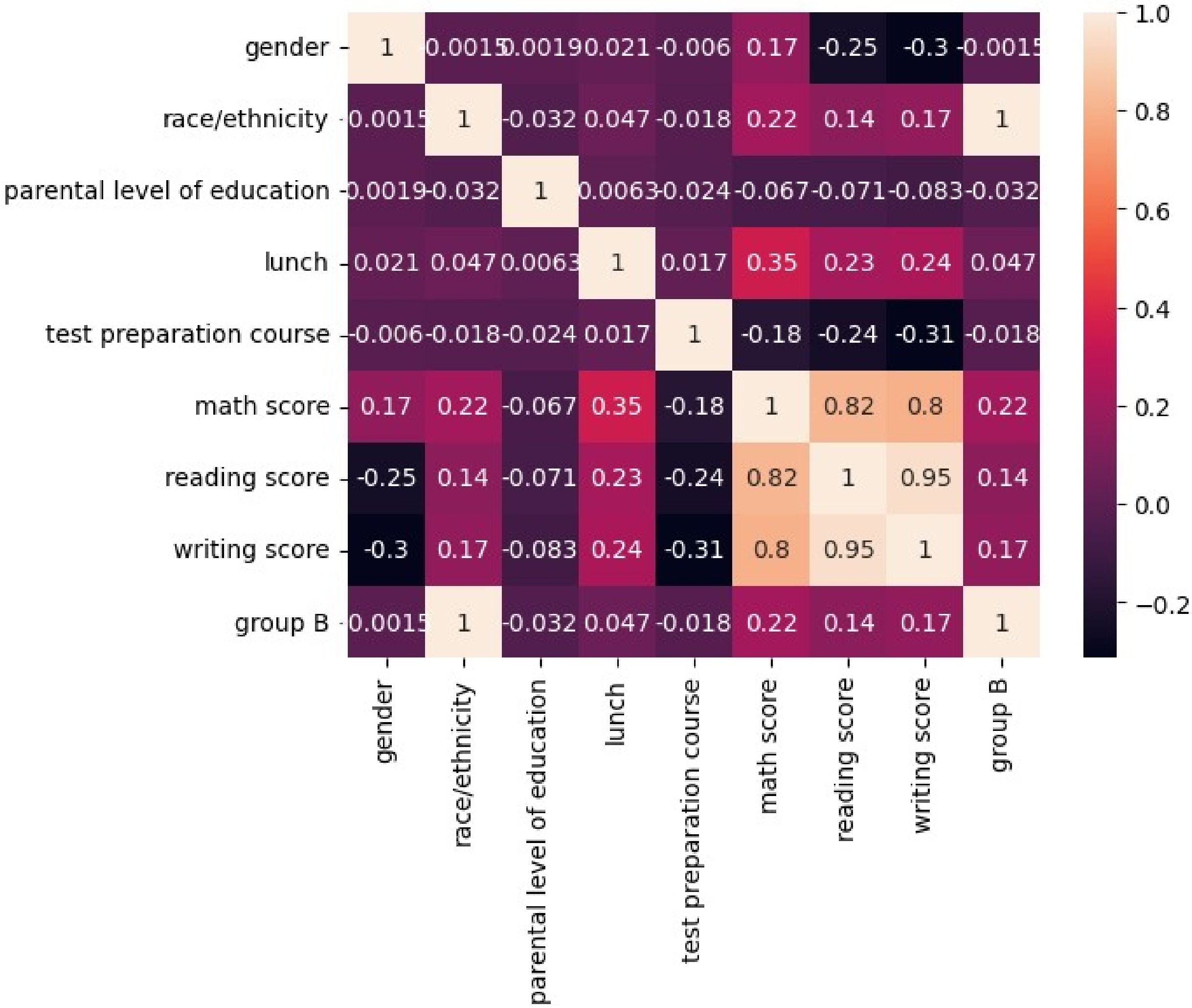
```
\"math score\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.4121644646256709,\n        \"min\": -
0.17749134236860675,\n        \"max\": 1.0,\n
\"num_unique_values\": 9,\n        \"samples\": [\n
0.8002879265277941,\n          0.2171598878447463,\n          1.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"reading score\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.4957323163730022,\n        \"min\": -0.24685526095874422,\n
\"max\": 1.0,\n        \"num_unique_values\": 9,\n        \"samples\":
[\n          0.9540393552910628,\n          0.14465352731961204,\n
0.8153996772328951\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"writing score\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.5117088492704792,\n        \"min\": -
0.31363349993845463,\n        \"max\": 1.0,\n
\"num_unique_values\": 9,\n        \"samples\": [\n          1.0,\n
0.1654312629227181,\n          0.8002879265277941\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"group B\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.41709563002306654,\n        \"min\": -0.03194564692573897,\n
\"max\": 1.0,\n        \"num_unique_values\": 8,\n        \"samples\":
[\n          1.0,\n          0.21715988784474632,\n          -
0.0015019243791521083\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df_corr"}
```


===VISUALIZE DISTRIBUTIONS USING HEATMAP===

- `Visualizes correlation matrix`
- Color intensity shows strength of relationships

```python
import seaborn as sns
sns.heatmap(df_corr, annot = True)
```

```
<Axes: >
```



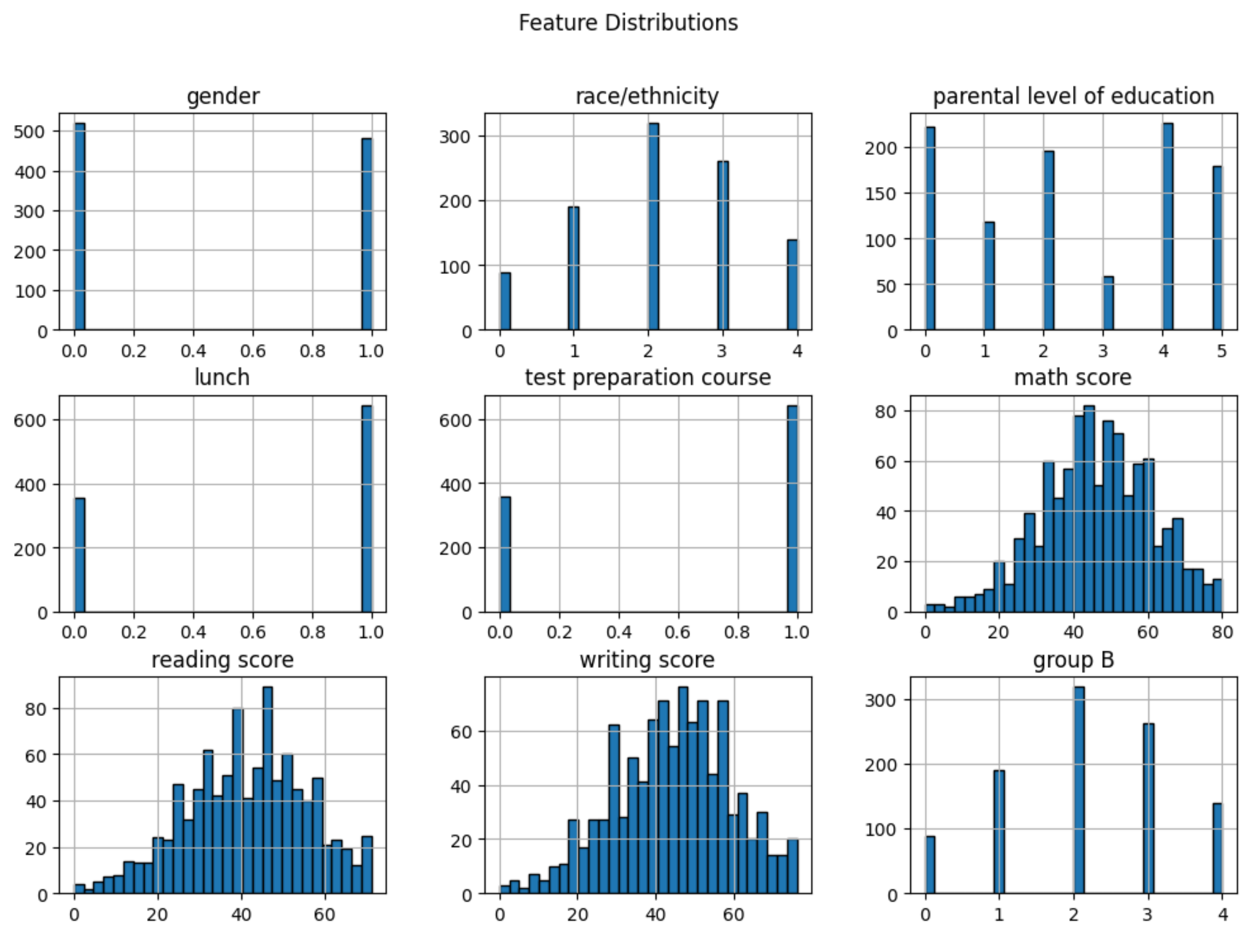
===**HISTOGRAM**===

- Helps visualize the distribution of numerical features (math, reading, writing scores).

- Identifies skewness, outliers, and central tendency of the data

- Aids in selecting appropriate data preprocessing or modeling techniques based on distribution.

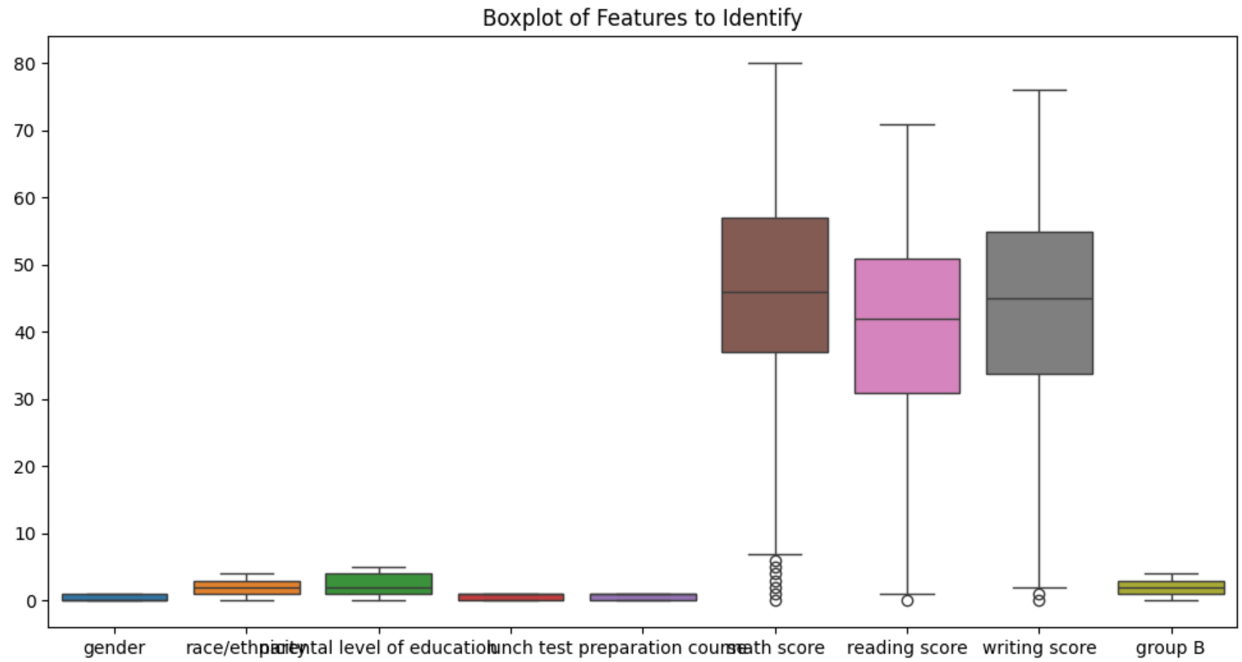
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 6))
df.hist(bins=30, figsize=(12, 8), edgecolor='black')
plt.suptitle("Feature Distributions")
plt.show()

<Figure size 1200x600 with 0 Axes>
```



===**BOX PLOT**===

- Shows the **median**, **quartiles (Q1 & Q3)**, and **outliers** for each feature.
- Helps in identifying:
  - **Skewness** in the data
  - **Spread/dispersion** of scores
  - Any **extreme values (outliers)** that may affect analysis or modeling

```
plt.figure(figsize = (12, 6))
sns.boxplot(data = df)
plt.title("Boxplot of Features to Identify")
plt.show()
```



===**MODEL TRAINING & EVALUATION**===

- Prepares the **feature matrix (X)** and **target vector (y)** for training a machine learning model.
- Focuses on specific factors that may impact performance (like race, lunch type, test prep, and writing score) to predict a target score.

```
X = df.iloc[:, [1, 4, 5, 7]].values
y = df.iloc[:, 8].values

X

array([[ 1,  1, 52, 50],
       [ 2,  0, 49, 64],
       [ 1,  1, 70, 69],
       ...,
       [ 2,  0, 39, 41],
       [ 3,  0, 48, 53],
       [ 3,  1, 57, 62]])

y

array([1, 2, 1, 0, 2, 1, 1, 1, 3, 1, 2, 3, 1, 0, 0, 2, 2, 1, 2, 2, 3,
1,
       3, 2, 3, 0, 1, 2, 2, 3, 3, 1, 4, 3, 4, 4, 3, 3, 3, 1, 2, 2, 1,
1,
       4, 1, 0, 2, 3, 2, 4, 4, 2, 3, 2, 2, 4, 3, 3, 2, 4, 0, 0, 2, 3,
1,
       3, 2, 1, 2, 3, 3, 0, 2, 2, 1, 4, 0, 3, 4, 1, 1, 0, 4, 3, 2, 2,
3,
       0, 3, 2, 2, 2, 2, 1, 2, 1, 4, 3, 3, 1, 3, 3, 1, 2, 2, 3, 4, 1,
```

```
1,
        3, 2, 0, 3, 4, 2, 1, 3, 3, 2, 2, 1, 2, 3, 4, 1, 1, 3, 3, 0, 3,
2,
        4, 2, 3, 2, 1, 4, 2, 3, 3, 2, 4, 0, 3, 2, 1, 2, 3, 4, 0, 0, 1,
3,
        3, 2, 4, 1, 1, 3, 1, 4, 1, 2, 4, 2, 2, 1, 1, 2, 0, 4, 3, 2, 2,
2,
        1, 2, 1, 3, 2, 2, 4, 3, 2, 2, 4, 3, 1, 2, 4, 3, 1, 3, 2, 3, 2,
4,
        1, 1, 2, 3, 2, 1, 2, 3, 4, 4, 1, 1, 3, 2, 2, 2, 4, 1, 4, 2, 1,
1,
        3, 1, 2, 3, 1, 4, 2, 3, 0, 2, 3, 2, 1, 4, 2, 3, 3, 3, 1, 2, 3,
4,
        3, 4, 3, 2, 4, 1, 1, 2, 0, 3, 1, 3, 3, 4, 2, 2, 1, 2, 2, 2, 2,
4,
        3, 3, 2, 3, 3, 4, 2, 2, 3, 3, 1, 2, 2, 4, 2, 1, 3, 3, 3, 3, 1,
1,
        4, 1, 1, 4, 2, 3, 2, 4, 3, 1, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 4,
2,
        1, 3, 1, 1, 3, 2, 2, 2, 3, 2, 1, 3, 2, 4, 2, 2, 2, 2, 2, 0, 2,
1,
        2, 2, 4, 1, 2, 1, 3, 2, 1, 3, 2, 2, 1, 3, 3, 2, 1, 2, 3, 4, 1,
4,
        2, 2, 2, 1, 0, 2, 3, 3, 1, 1, 2, 3, 2, 0, 2, 2, 0, 3, 4, 2, 3,
3,
        3, 4, 3, 3, 0, 0, 1, 2, 2, 4, 0, 4, 4, 2, 3, 3, 4, 3, 4, 2, 2,
0,
        1, 2, 1, 3, 2, 0, 0, 3, 2, 2, 1, 1, 3, 3, 3, 4, 3, 1, 2, 4, 2,
2,
        3, 4, 2, 3, 3, 0, 1, 2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 3, 3, 2,
3,
        2, 3, 0, 1, 0, 2, 3, 2, 1, 1, 2, 4, 2, 2, 2, 2, 3, 3, 4, 1, 2,
1,
        4, 2, 0, 2, 3, 0, 0, 2, 2, 2, 2, 3, 1, 3, 4, 3, 3, 4, 1, 3, 2,
0,
        1, 2, 3, 2, 1, 0, 0, 2, 2, 2, 1, 3, 2, 3, 1, 4, 3, 1, 2, 4, 3,
1,
        0, 1, 2, 2, 3, 0, 3, 1, 1, 2, 3, 4, 3, 1, 3, 2, 3, 2, 2, 4, 2,
2,
        3, 2, 2, 2, 4, 4, 1, 2, 2, 3, 4, 0, 2, 3, 2, 3, 3, 4, 0, 2, 2,
2,
        2, 1, 1, 3, 4, 2, 2, 2, 1, 3, 3, 2, 2, 3, 1, 1, 4, 3, 1, 3, 1,
0,
        2, 2, 4, 0, 0, 1, 1, 3, 3, 4, 3, 3, 3, 2, 0, 2, 2, 0, 2, 0, 4,
4,
        2, 2, 1, 0, 3, 3, 3, 2, 4, 3, 3, 2, 2, 2, 4, 1, 3, 2, 2, 2, 0,
2,
        4, 3, 3, 2, 2, 1, 2, 0, 4, 3, 1, 3, 3, 2, 3, 1, 1, 2, 3, 0, 1,
3,
```

```
       4, 3, 3, 3, 1, 4, 1, 1, 3, 4, 1, 3, 2, 0, 3, 0, 1, 1, 2, 3, 3, 3,
       2, 2, 3, 2, 3, 2, 2, 1, 2, 3, 2, 3, 2, 2, 3, 1, 4, 2, 3, 3, 3, 1,
       1, 2, 1, 4, 4, 3, 0, 4, 2, 4, 2, 3, 2, 3, 2, 0, 3, 2, 4, 1, 0, 3,
       1, 0, 3, 2, 3, 3, 2, 4, 3, 3, 1, 1, 2, 2, 2, 4, 2, 3, 1, 2, 1, 4,
       4, 4, 3, 2, 1, 0, 2, 3, 4, 2, 2, 1, 3, 2, 3, 0, 2, 2, 1, 3, 3, 2,
       2, 1, 3, 4, 2, 2, 2, 4, 3, 4, 3, 1, 2, 3, 3, 1, 3, 1, 2, 1, 3, 0,
       1, 3, 1, 2, 1, 1, 1, 2, 0, 4, 3, 1, 1, 2, 2, 1, 4, 1, 2, 2, 1, 3,
       3, 4, 1, 4, 3, 4, 4, 2, 2, 2, 4, 1, 2, 0, 3, 4, 2, 1, 0, 0, 2, 4,
       2, 1, 0, 3, 1, 2, 0, 3, 4, 1, 2, 2, 2, 2, 3, 1, 0, 2, 0, 1, 1, 2,
       4, 0, 1, 2, 3, 2, 1, 1, 3, 4, 2, 3, 2, 3, 2, 0, 4, 4, 2, 1, 1, 2,
       1, 2, 2, 4, 3, 2, 2, 3, 2, 1, 4, 2, 1, 2, 1, 4, 2, 2, 3, 2, 3, 3,
       2, 4, 1, 3, 4, 2, 4, 2, 3, 3, 4, 4, 0, 3, 4, 4, 1, 1, 3, 3, 3, 2,
       0, 3, 3, 3, 1, 3, 2, 4, 3, 0, 2, 2, 1, 4, 4, 2, 2, 1, 3, 2, 3, 1,
       3, 4, 4, 3, 4, 2, 2, 3, 3, 2, 2, 3, 0, 4, 3, 3, 2, 3, 2, 0, 1, 2,
       1, 3, 1, 4, 4, 3, 4, 2, 2, 4, 2, 3, 3, 2, 0, 3, 4, 2, 3, 3, 0, 2,
       4, 1, 3, 2, 0, 3, 0, 2, 1, 2, 3, 2, 1, 3, 1, 0, 2, 0, 2, 4, 0, 3,
       4, 1, 3, 3, 0, 4, 2, 2, 3, 3])
```

- Ensures that all features are on the **same scale**, which is important for many machine learning algorithms.
- Prevents features with larger values from dominating the model.
- Especially useful for models that are **distance-based** (like SVM, KNN) or involve **gradient descent** (like logistic/linear regression).

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

===**SPLITTING DATA INTO TRAIN (80%) AND TEST (20%)**===

- Trains the model on one portion (80%) of the data.
- Tests the model's **accuracy and generalization** on the remaining unseen data (20%).
- Prevents **overfitting** by validating model performance outside the training set.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size =
0.80, test_size = 0.20, random_state = 0)
```

===**LOGISTIC** REGRESSION===

- To **build a predictive model** using training data.
- Once trained, the model can be used to make predictions on new (test) data.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)

LogisticRegression()
```

- To **evaluate the model's performance** on test data.
- Compares `y_pred` (predicted outputs) with `y_test` (actual outputs) to measure accuracy, error, etc.

```
y_pred = model.predict(X_test)

y_pred

array([3, 2, 2, 3, 2, 2, 2, 2, 4, 3, 2, 1, 2, 2, 1, 2, 1, 2, 0, 0, 1,
3,
       1, 1, 3, 2, 1, 1, 2, 2, 1, 2, 0, 1, 3, 2, 3, 2, 2, 3, 3, 2, 0,
3,
       3, 2, 2, 2, 3, 2, 2, 1, 2, 2, 3, 3, 3, 2, 4, 3, 3, 4, 4, 2, 1,
2,
       3, 2, 1, 0, 3, 3, 3, 1, 4, 3, 3, 3, 2, 1, 2, 3, 4, 1, 4, 3, 3,
1,
       3, 2, 2, 1, 1, 1, 4, 2, 1, 1, 2, 1, 1, 4, 1, 0, 2, 1, 2, 2, 3,
4,
       4, 3, 3, 2, 4, 4, 4, 2, 0, 1, 3, 3, 2, 1, 4, 3, 3, 3, 3, 4, 3,
3,
       1, 3, 2, 4, 3, 3, 2, 3, 3, 2, 3, 2, 2, 1, 0, 0, 2, 4, 2, 3, 3,
4,
       2, 4, 3, 2, 2, 0, 3, 2, 0, 2, 3, 0, 2, 2, 4, 3, 2, 4, 2, 2, 2,
4,
       3, 1, 4, 3, 1, 3, 0, 2, 3, 3, 2, 4, 2, 2, 1, 0, 2, 1, 0, 3, 1,
1,
       0, 0])
```

- To **compare actual vs predicted values** for the entire dataset.
- Helps with **visual analysis** of how well your model is performing.

```
y_pred1=model.predict(X)

df['Prediction']=y_pred1

df
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1000,\n  \"fields\":
[\n    {\n      \"column\": \"gender\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"race/ethnicity\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 4,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          2,\n          4\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"parental level of education\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 1,\n        \"min\":
0,\n        \"max\": 5,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          1,\n          4\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"lunch\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\":
0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"test preparation course\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"math score\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 14,\n        \"min\": 0,\n
\"max\": 80,\n        \"num_unique_values\": 81,\n        \"samples\":
[\n          35,\n          52\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"reading score\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 14,\n        \"min\": 0,\n
\"max\": 71,\n        \"num_unique_values\": 72,\n        \"samples\":
[\n          50,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"writing score\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 15,\n        \"min\": 0,\n
\"max\": 76,\n        \"num_unique_values\": 77,\n        \"samples\":
[\n          51,\n          52\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"group B\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 4,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          2,\n          4\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Prediction\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 4,\n        \"num_unique_values\": 5,\n        \"samples\":
```

```
[\n          2,\n          4\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


===CONFUSION MATRIX===

- Understand model **performance** at a deeper level than accuracy alone.
- Helps answer:
  - Are there more **false positives** or **false negatives**?
  - Is the model biased toward a particular class?

```
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix, precision_score
cm = confusion_matrix(y_test, y_pred)

print(cm)

[[17  0  0  0  0]
 [ 0 36  0  0  0]
 [ 0  0 65  0  0]
 [ 0  0  0 57  0]
 [ 0  0  0  0 25]]
```

EVALUATING PERFORMANCE USING ACCURACY, PRECISION, RECALL, and F1-SCORE

- When your target labels (y) are **multiclass or imbalanced**, `'weighted'` gives a more accurate evaluation.
- Other options: `'macro'`, `'micro'`, `'binary'` (binary only).

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

Accuracy: 1.0

precision = precision_score(y_test, y_pred, average='weighted') #
Changed to 'weighted'
print("Precision:", precision)

Precision: 1.0

from sklearn.metrics import recall_score
recall = recall_score(y_test, y_pred, average='weighted') # Add
average='weighted'
print("Recall:", recall)

Recall: 1.0
```

===**CLASSIFICATION REPORT**===

- Gives a **comprehensive view** of model performance across all classes.
- Especially helpful for **imbalanced datasets** where accuracy alone might be misleading.

```python
print("Classification Report:\n", classification_report(y_test, 
y_pred))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      1.00      1.00        36
           2       1.00      1.00      1.00        65
           3       1.00      1.00      1.00        57
           4       1.00      1.00      1.00        25

    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200
```

Dataset Overview

Contains **1000** records with **8** features:

- **Categorical:** gender, race/ethnicity, parental education, lunch, test prep
- **Numerical:** math, reading, writing scores

Key Stats:

- 518 females (51.8%) | 482 males (48.2%)
- Average Scores: Math - 66.09 | Reading - 69.17 | Writing - 68.05
- Most common ethnicity: Group C (31.9%)
- Top parental education: "Some college" (22.6%)

===CRISP-DM Model Implementation===

Business Understanding Understanding factors affecting student performance can help educational institutions:

- Develop targeted intervention programs
- Allocate resources effectively
- Improve overall academic outcomes

Data Understanding

Initial exploration revealed:

- No missing values
- Scores range from 0-100
- Potential correlations between variables

Data Preparation
Key steps included:

- Handling missing values (none found)
- Encoding categorical variables
- Normalization and standardization
- Feature engineering

Modeling

- Potential models: Linear regression, logistic regression
- Target variables: Test scores
- Features: Demographic and preparation factors

Evaluation (To be implemented):

- Model accuracy metrics
- Feature importance analysis
- Validation techniques

Deployment (Planned):

- Predictive model for student performance
- Dashboard for educators

Project Goals

- Identify key factors affecting student performance
- Explore relationships between demographics and scores
- Prepare data for modeling
- Visualize patterns and correlations

Conclusion

- The dataset was thoroughly cleaned and prepared for analysis
- Strong correlations exist between test scores (0.8–0.95)
- Demographic factors show moderate correlations with performance
- Lunch type (standard vs free/reduced) shows notable correlation (0.35) with math scores
- The data is now ready for more advanced predictive modeling .

Future Scope

- Implement predictive models for score prediction
- Cluster analysis to identify student groups
- Deeper feature engineering (e.g., combined scores)
- Interactive visualization dashboard
- Integration with institutional data systems

Real Life Implementation

- **Early intervention system**: Flag at-risk students based on performance trends
- **Resource allocation**: Direct academic support where it's needed most
- **Curriculum development**: Tailor teaching methods to match student needs
- **Scholarship programs**: Identify deserving candidates needing financial support
- **Data-driven decisions**: Empower educators with actionable insights for better planning

References

- *Scikit-learn documentation*
- *Pandas user guide*
- *Seaborn visualization examples*

- *CRISP-DM methodology papers*
- *Educational research on performance factors*