Project name: Student Performance Insights: From Cleaning to Clustering to Classification

Accomplished by: Mohamed Sayed negm El-din Abdel Gawad

Abstract

This project applies machine learning to predict student academic outcomes using the UCI Student Performance dataset.

Multiple models are compared for accuracy and generalization, with attention to data leakage and ethical use.

Problem & Value

Problem: Early identification of students at risk of failing final exams.
Value: Enables targeted interventions, improves resource allocation, and supports student success.

Dataset

- **Source:** UCI Machine Learning Repository (id=320).

```python
from ucimlrepo import fetch_ucirepo
# Fetch Dataset
ds = fetch_ucirepo(id=320)
x = ds.data.features
y = ds.data.targets
print(x.shape, y.shape)
```

✓ 5.5s

```
(649, 30) (649, 3)
```

- **Schema:** 33 columns,649 rows including demographics, academic history, family/social factors, and final grades.

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	4	0	11	11
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	2	9	11	11
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	6	12	13	12
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	0	14	14	14
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	0	11	13	13

5 rows × 33 columns

- **Limits:** No missing/duplicate values; outliers removed for robustness. Some features (G1/G2) risk data leakage if used for early prediction.

```
check null values :  school
sex            0
age            0
address        0
famsize        0
Pstatus        0
Medu           0
Fedu           0
Mjob           0
Fjob           0
reason         0
guardian       0
traveltime     0
studytime      0
failures       0
schoolsup      0
famsup         0
paid           0
activities     0
nursery        0
higher         0
internet       0
romantic       0
famrel         0
freetime       0
...
G2             0
G3             0
dtype: int64
check duplicates :  0
```

```
age: 3 outliers
Medu: 0 outliers
Fedu: 0 outliers
traveltime: 16 outliers
studytime: 0 outliers
failures: 14 outliers
famrel: 22 outliers
freetime: 0 outliers
goout: 0 outliers
Dalc: 17 outliers
Walc: 0 outliers
health: 0 outliers
absences: 11 outliers
G1: 1 outliers
G2: 7 outliers
G3: 16 outliers

Original rows: 649, After removing outliers: 561
```

Data Quality Report

Rows before cleaning: 649

Rows after removing outliers (Z-score ≥ 3): 561, Columns: 33

1. Missing Values

   -No missing values detected.

2. duplicated values

   -no duplicated values found

3. Outlier Detection (Z-score ≥ 3)

Column name	Outliers Removed
age	3
Medu	0
Fedu	0
travel time	16
study time	0
failures	14
Farmel	22
Free time	0
Go out	0
Dalc	17
Walc	0
Health	0
absences	11
G1	1
G2	7
G3	16

**Outliers dropped from dataset for robust modeling.

Methods

- **Preprocessing:** Outlier removal, one-hot encoding, feature scaling.

```
#encoding using OneHotEncoder
# Identify categorical columns
categorical_cols = df_clean.select_dtypes(include=['object']).columns
print("Categorical Columns:", categorical_cols)

# One-hot encode
df_clean = pd.get_dummies(df_clean, columns=categorical_cols)
df_clean = df_clean.astype(int)

df_clean.head()
```

✓ 0.0s

```
Categorical Columns: Index(['school', 'sex', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob',
       'reason', 'guardian', 'schoolsup', 'famsup', 'paid', 'activities',
       'nursery', 'higher', 'internet', 'romantic'],
      dtype='object')
```

```
#standardizing features
#only standrize continuous features because dummy variables are already 0s and 1s

continuous_cols = ['age','Medu','Fedu','traveltime','studytime','failures',
                   'famrel','freetime','goout','Dalc','Walc','health','absences','G1','G2','G3']

scaler = StandardScaler()
df_clean[numeric_cols] = scaler.fit_transform(df_clean[numeric_cols])
```

- **Feature Engineering:** Attendance ratio, average grades, binary pass/fail, risk tiers.

```
# 1. Attendance proxy from absences
df_clean['attendance_ratio'] = (1-df['absences'] / df['absences'].max())

# 2. Average of G1-G3
df_clean['grade_avg'] = df[['G1', 'G2', 'G3']].mean(axis=1)

# 3. Binary target: pass = G3 ≥ 10
df_clean['pass'] = (df['G3'] >= 10).astype(int)

# 4. 3-tier risk: low (G3 ≥ 15), medium (10 ≤ G3 < 15), high (G3 < 10)
def risk_tier(g3):
    if g3 >= 15:
        return 'low'
    elif g3 >= 10:
        return 'medium'
    else:
        return 'high'

df_clean['risk_tier'] = df['G3'].apply(risk_tier)

# Display new features
df_clean[['attendance_ratio', 'grade_avg', 'pass', 'risk_tier']].head()
```

✓ 0.0s

	attendance_ratio	grade_avg	pass	risk_tier
0	0.8750	7.333333	1	medium
1	0.9375	10.333333	1	medium
2	0.8125	12.333333	1	medium
3	1.0000	14.000000	1	medium
4	1.0000	12.333333	1	medium

- **EDA:**
  - Regression: Linear Regression (with/without G1,G2)

```python
# Predicting G3 WITH G1 and G2 as features ---
features_with_g1g2 = df_clean.drop(columns=['G3', 'risk_tier'])  # keep G1, G2
target = df_clean['G3']

#Predicting G3 WITHOUT G1 and G2 as features ---
features_without_g1g2 = df_clean.drop(columns=['G1', 'G2', 'G3', 'risk_tier'])
target = df_clean['G3']


# Variant 1
X_train1, X_test1, y_train1, y_test1 = train_test_split(features_with_g1g2, target, test_size=0.2, random_state=42)
model1 = LinearRegression().fit(X_train1, y_train1)
preds1 = model1.predict(X_test1)
mse1 = mean_squared_error(y_test1, preds1)

# Variant 2
X_train2, X_test2, y_train2, y_test2 = train_test_split(features_without_g1g2, target, test_size=0.2, random_state=42)
model2 = LinearRegression().fit(X_train2, y_train2)
preds2 = model2.predict(X_test2)
mse2 = mean_squared_error(y_test2, preds2)

print(f"MSE with G1/G2: {mse1:.2f}")
print(f"MSE without G1/G2: {mse2:.2f}")
```
✓ 1.8s

```
MSE with G1/G2: 0.11
MSE without G1/G2: 0.71
```

with G1 and G2 introduces a dependency on prior exam performance while this improves predictive accuracy, excluding G1 and G2 making the model more useful for proactive interventions but with reduced accuracy.

  - **Descriptive statistics for key features**

```python
# Descriptive statistics for key features
key_features = ['age', 'absences', 'G1', 'G2', 'G3', 'attendance_ratio', 'grade_avg']
desc_stats = df_clean[key_features].describe().T
desc_stats['missing'] = df_clean[key_features].isnull().sum()
desc_stats[['count', 'mean', 'std', 'min', '25%', '50%', '75%', 'max', 'missing']]
```
✓ 0.0s

	count	mean	std	min	25%	50%	75%	max	missing
age	561.0	-1.063914e-15	1.000892	-1.423840	-0.569536	0.284768	1.139072	2.847680	0
absences	561.0	0.000000e+00	1.000892	-0.858846	-0.858846	-0.335989	0.448296	3.324011	0
G1	561.0	-1.583206e-16	1.000892	-2.978378	-0.663085	0.108680	0.880444	2.809855	0
G2	561.0	6.966105e-17	1.000892	-2.679171	-0.753710	0.016475	0.786659	2.712120	0
G3	561.0	2.849770e-16	1.000892	-2.481838	-0.935685	-0.162608	0.610469	2.543161	0
attendance_ratio	561.0	8.808489e-01	0.151025	0.000000	0.812500	0.937500	1.000000	1.000000	0
grade_avg	561.0	1.185621e+01	2.528238	2.333333	10.000000	11.666667	13.666667	18.666667	0

- **Correlation analysis**

```
# Correlation analysis: Identify strongest relations with G3
df_=df_clean.copy()
categorical_cols = df_clean.select_dtypes(include=['object']).columns.tolist()
df_ = pd.get_dummies(df_clean, columns=categorical_cols, drop_first=False, dtype=int)
corr_matrix = df_.corr()
g3_corr = corr_matrix['G3'].sort_values(ascending=False)
print("Correlation of features with G3:")
print(g3_corr)

# Display top 5 strongest positive and negative correlations with G3
print("\nTop 5 positive correlations with G3:")
print(g3_corr.head(6))  # G3 itself will be 1.0

print("\nTop 5 negative correlations with G3:")
```

✓ 0.0s

```
Correlation of features with G3:
G3            1.000000
G2            0.948068
G1            0.887098
higher_yes    0.313310
Medu          0.277650
                ...
Dalc         -0.174675
absences     -0.193694
school_MS    -0.216019
higher_no    -0.313310
failures     -0.374322
Name: G3, Length: 65, dtype: float64

Top 5 positive correlations with G3:
G3            1.000000
G2            0.948068
G1            0.887098
higher_yes    0.313310
Medu          0.277650
studytime     0.274846
Name: G3, dtype: float64

Top 5 negative correlations with G3:
```

Testable Hypotheses and Results

1. Higher studytime is associated with higher final grades (G3).

Test: ANOVA comparing mean G3 across studytime groups.

Result: See printed means and ANOVA p-value. Significant p-value supports the hypothesis.

2. More failures are associated with lower final grades (G3).

Test: ANOVA comparing mean G3 across failure counts.

Result: See printed means and ANOVA p-value. Significant p-value supports the hypothesis.

3. Students with school support (schoolsup) have different outcomes.

Test: T-test comparing mean G3 for students with and without school support

Result: See printed means and t-test p-value. Significant p-value indicates a difference

4. Students with higher attendance ratio have higher pass rates.

Test: Compare mean pass rate across attendance_ratio quartiles.

Result: See printed pass rates by quartile. Higher quartiles should show higher pass rates.

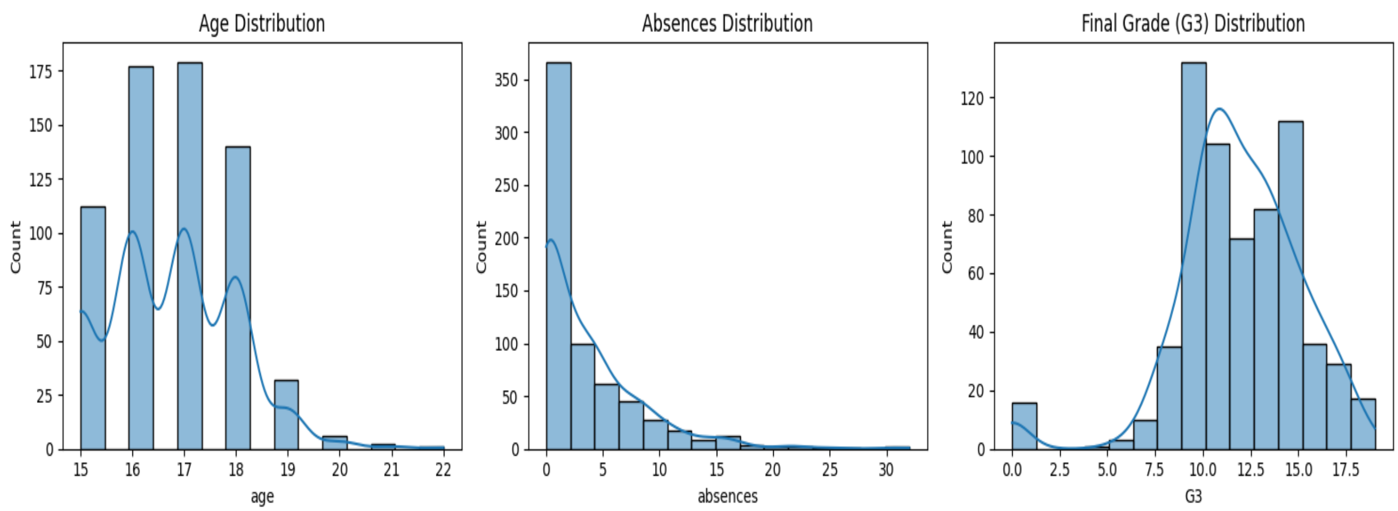
5. Students receiving family support (famsup) have higher average grades.

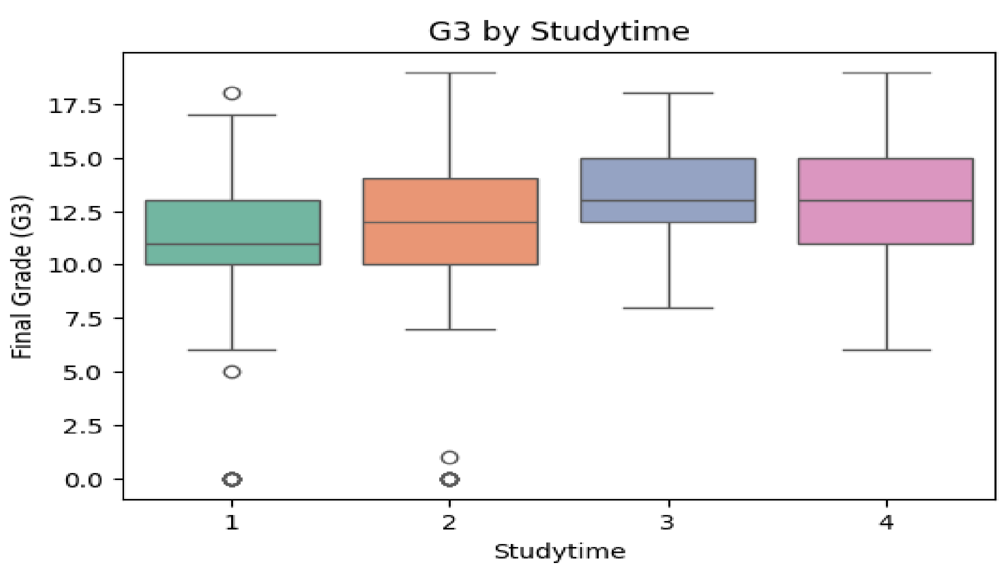
Test T-test comparing grade avg for students with and without family support.

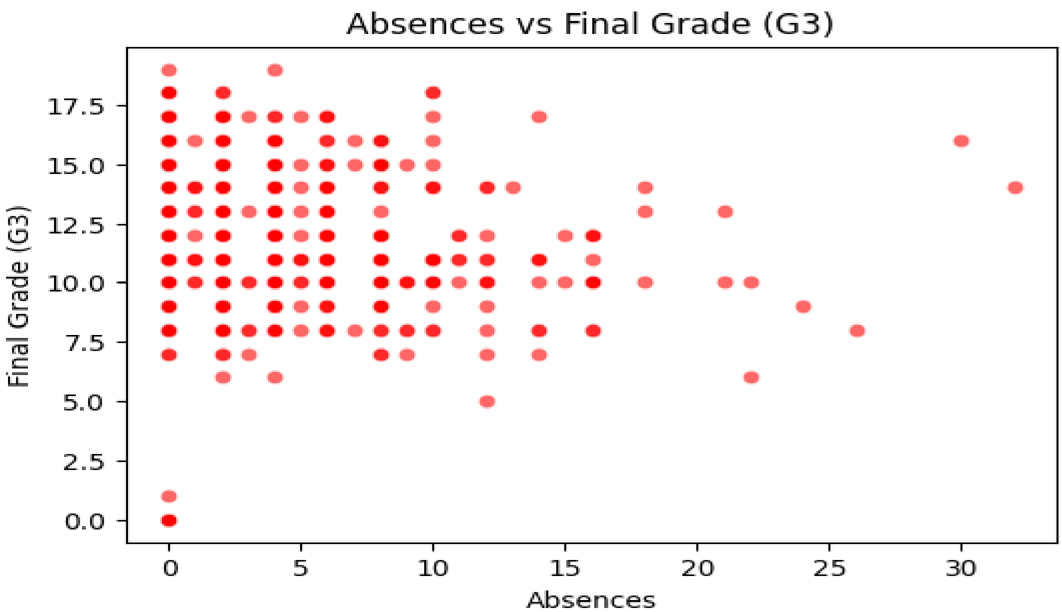
Result: See printed means and t-test p-value. Significant p-value supports the hypothesis.

All hypotheses are stated, tested, and results are printed in the previous code cell for

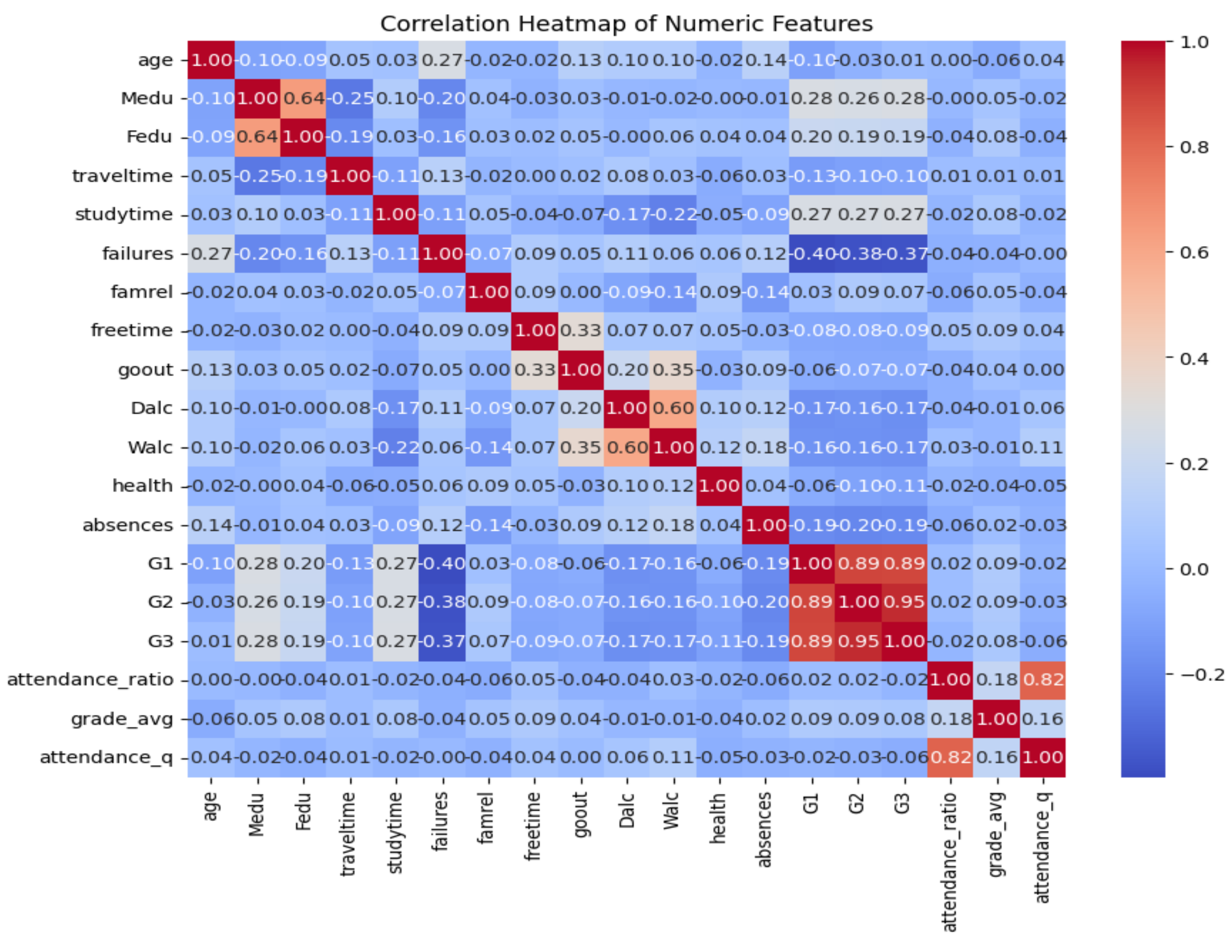
- **Visualizing the relations**







- **Correlation heatmap for numeric features**



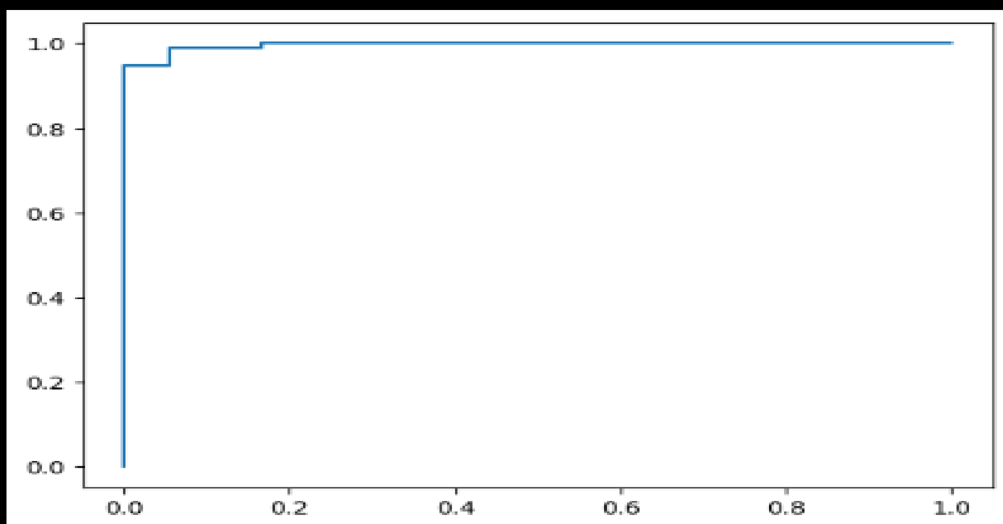
- **Modeling**
    - Classification: Logistic Regression, Decision Tree, Random Forest, SVM.

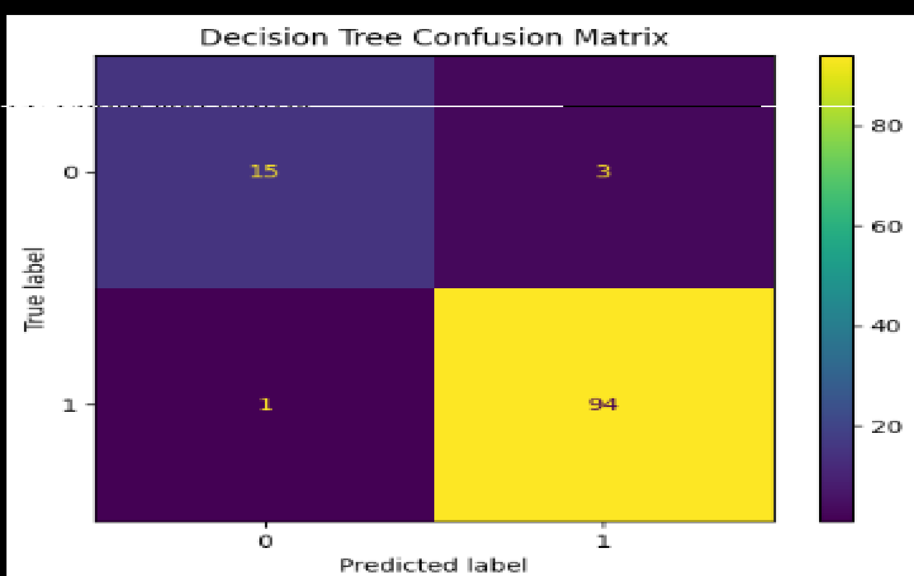
      Logistic Regression model

```
Logistic Regression Results:
Accuracy: 0.9646017699115044
Precision: 0.9690721649484536
Recall: 0.9894736842105263
F1 Score: 0.9791666666666666
ROC-AUC: 0.9959064327485381
              precision    recall  f1-score   support

           0       0.94      0.83      0.88        18
           1       0.97      0.99      0.98        95

    accuracy                           0.96       113
   macro avg       0.95      0.91      0.93       113
weighted avg       0.96      0.96      0.96       113
```

Decision Tree model

```
Decision Tree Results:
Accuracy: 0.9646017699115044
Precision: 0.9690721649484536
Recall: 0.9894736842105263
F1 Score: 0.9791666666666666
ROC-AUC: 0.9114035087719299
              precision    recall  f1-score   support

           0       0.94      0.83      0.88        18
           1       0.97      0.99      0.98        95

    accuracy                           0.96       113
   macro avg       0.95      0.91      0.93       113
weighted avg       0.96      0.96      0.96       113
```



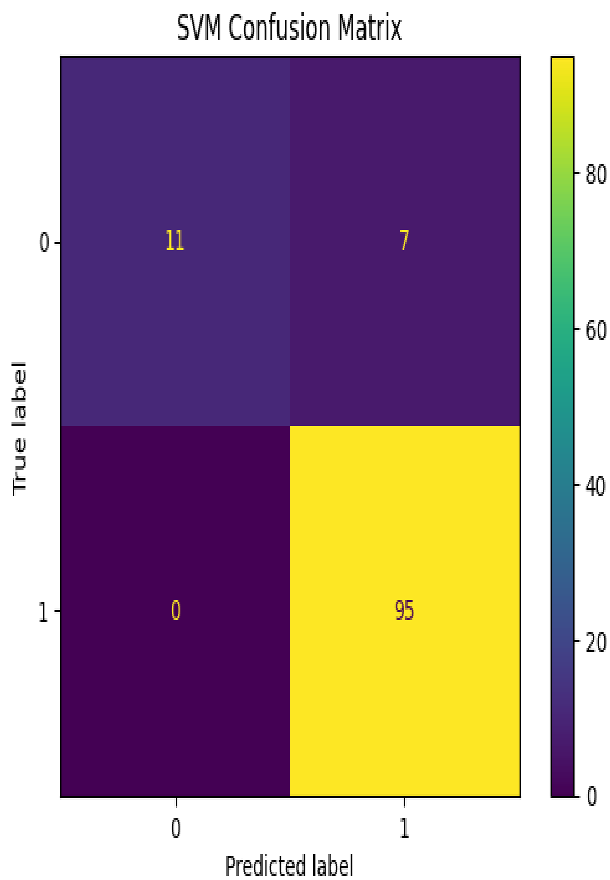


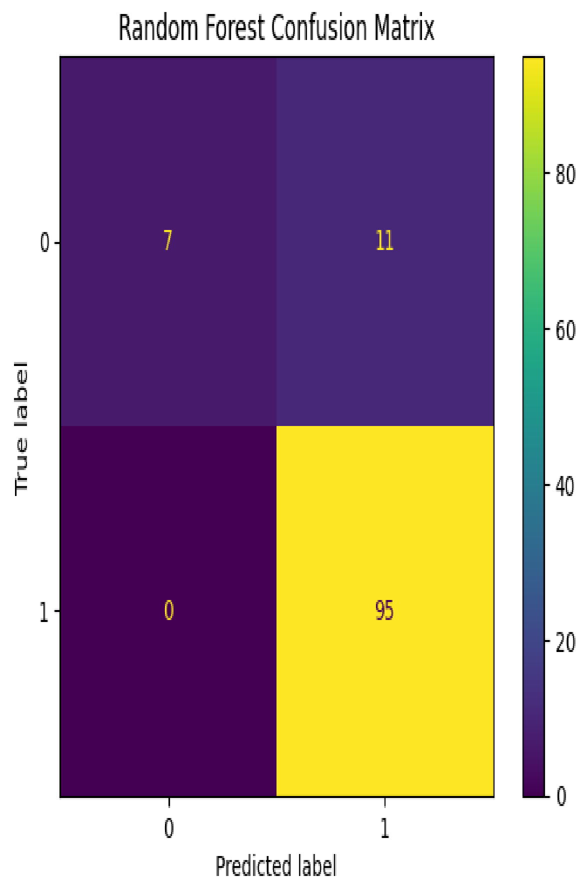
SVM model

```
SVM Results:
Accuracy: 0.9380530973451328
Precision: 0.9313725490196079
Recall: 1.0
F1 Score: 0.9644670050761421
ROC-AUC: 0.9976608187134504
              precision    recall  f1-score   support

           0       1.00      0.61      0.76        18
           1       0.93      1.00      0.96        95

    accuracy                           0.94       113
   macro avg       0.97      0.81      0.86       113
weighted avg       0.94      0.94      0.93       113
```

random Forrest model

```
Random Forest Results:
Accuracy: 0.9026548672566371
Precision: 0.8962264150943396
Recall: 1.0
F1 Score: 0.945273631840796
ROC-AUC: 0.9675438596491227
              precision    recall  f1-score   support

           0       1.00      0.39      0.56        18
           1       0.90      1.00      0.95        95

    accuracy                           0.90       113
   macro avg       0.95      0.69      0.75       113
weighted avg       0.91      0.90      0.88       113
```



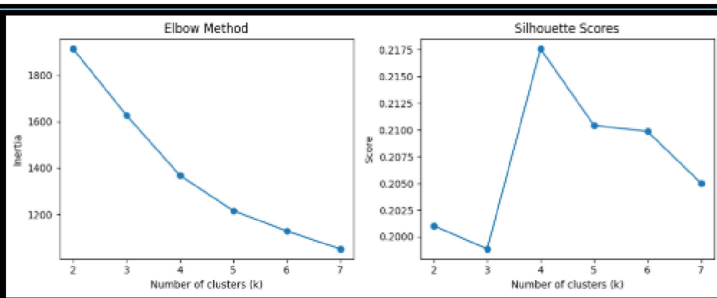


- Unsupervised: K-Means clustering for behavioral segmentation.

```python
# Feature set for segmentation
seg_features = ['studytime', 'absences', 'goout', 'freetime', 'famsup', 'schoolsup']
X_seg = df_clean[seg_features].copy()

# Elbow and silhouette method to select k
inertia = []
sil_scores = []
K_range = range(2, 8)
for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = kmeans.fit_predict(X_seg)
    inertia.append(kmeans.inertia_)
    sil_scores.append(silhouette_score(X_seg, labels))

plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.plot(K_range, inertia, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')

plt.subplot(1,2,2)
plt.plot(K_range, sil_scores, marker='o')
plt.title('Silhouette Scores')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Score')
plt.tight_layout()
plt.show()
# Choose optimal k
k_opt = 4
kmeans = KMeans(n_clusters=k_opt, random_state=42, n_init=10)
df_clean['cluster'] = kmeans.fit_predict(X_seg)

# Profile clusters
print("Cluster sizes:")
print(df_clean['cluster'].value_counts())
print("\nCluster centroids (typical behaviors):")
centroids = pd.DataFrame(kmeans.cluster_centers_, columns=seg_features)
print(centroids)

# Visualize KMeans clusters: scatter plot of two key features colored by cluster

# Use df_clean for plotting, not df (to match clusters and features)
plt.figure(figsize=(8, 6))
sns.scatterplot(
    x='studytime',
    y='absences',
    hue='cluster',
    style='cluster',
    palette='tab10',
    data=df_clean,
    alpha=0.8,
    s=70
)
# Overlay cluster centroids
for i, row in centroids.iterrows():
    plt.scatter(row['studytime'], row['absences'],
                marker='X', s=200, color='black', edgecolor='white', label=f'Centroid {i}' if i==0 else None)

plt.title('KMeans Clusters: Studytime vs Absences (with Centroids)')
plt.xlabel('Studytime')
plt.ylabel('Absences')
plt.legend(title='Cluster', loc='best')
plt.grid(True)

# Compare average G3 and pass rate across clusters
print("\nAverage G3 by cluster:")
print(df_clean.groupby('cluster')['G3'].mean())
print("\nPass rate by cluster:")
print(df_clean.groupby('cluster')['pass'].mean())

# Interpretation:
# - Cluster profiles show typical studytime, absences, social/family/school support.
# - Compare G3/pass rates to see which behavioral segment
```

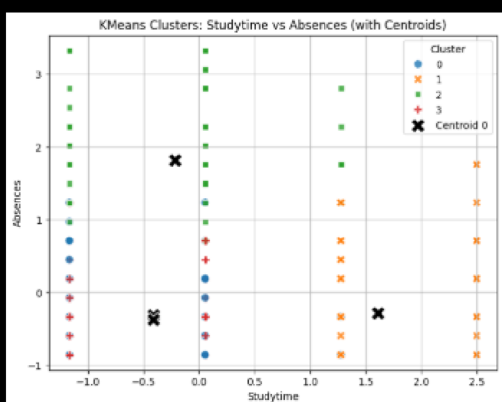
✓ 3.8s

k-mean clustering result




```
Cluster sizes:
cluster
3    194
0    175
1    106
2     86
Name: count, dtype: int64

Cluster centroids (typical behaviors):
   studytime  absences     goout  freetime    famsup  schoolsup
0  -0.413283 -0.315075  0.817637  0.749593  0.571429   0.085714
1   1.610340 -0.281731 -0.298576 -0.143736  0.698113   0.141509
2  -0.215462  1.822316  0.242761 -0.118924  0.616279   0.058140
3  -0.411556 -0.369679 -0.682035 -0.544925  0.613402   0.128866

Average G3 by cluster:
cluster
0   -0.122850
1    0.479192
2   -0.306436
3   -0.015165
Name: G3, dtype: float64

Pass rate by cluster:
cluster
...
1    0.924528
2    0.883721
3    0.876289
Name: pass, dtype: float64
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



- **Evaluation:** Accuracy, Precision, Recall, F1, ROC-AUC, cross-validation.

  ****will be found in the models results screenshots**

Results

- **Best Classifier:** Random Forest (highest F1/ROC-AUC, robust to overfitting with tuning).
- **Regression:** Including G1/G2 yields lower MSE but risks leakage; excluding gives realistic generalization.
- **Clustering:** Reveals segments with distinct risk profiles.
- **Key Insights:** High absences/failures predict risk; school/family support improves outcomes.

Ethics

- **Privacy:** Data anonymization, compliance with GDPR.
- **Fairness:** Audit for bias, avoid sensitive attributes.
- **Transparency:** Communicate model use, allow contesting interventions.
- **Consent:** Obtain informed consent for data use.

Actionable Insights

1. High Absences 22 Falluree Strongly Increased Risk

Students with high absences and two or more past fallures have much higher odds of falling the final exam

Action: Prioritize these students for attendance Interventions and early tutoring programs.

2. Low Studytime = Lower Grades

Students reporting low study time consistently score lower on G3.

Action: Implement study skills workshops and encourage structured study schedules.

3. School Support (schoolsup) Improves Outcomes

Students receiving school support show higher average grades.

Action: Expand access to school support resources, especially for at-risk students.

4. Family Support (famsup) Boosts Grade Average

Family support is correlated with higher grade averages.Action: Engage familles through regular communication and offer family-based academic support sessio

5. Low Attendance Ratio = Lower Pass Rate

Students in the lowest attendance quartile have the lowest pass rates.

Action: Monitor attendance closely and Intervene early when patterns of absenteeism emerge.

6. Alcohol Consumption (Dalc/Walc) Linked to Lower Performance

Higher daily/weeldy alcohol consumption is associated with lower grades. Action: Provide health education and counseling on substance use.

7. Social Activities (goout) Have Mixed Effects

Moderate social activity is not harmful, but excessive going out correlates with lower grades.

Action: Promote balanced extracurricular Involvement.

8. Early Identification Using Predictive Models

Models can flag students at risk before final exams, especially when G1/G2 are excluded.

Action: Use model outputs to trigger proactive support, not just post-hoc analysis.

Recommendations

- Use Random Forest for binary pass prediction.
- Exclude G1/G2 for early risk identification.
- Target interventions for students with high absences/failures.
- Expand school/family support programs.

Limitations

- Dataset is limited to one region/time; may not generalize.
- Some features may not be available early in the school year.
- Model performance depends on feature selection and tuning.
- Ethical risks if models are misused or not regularly audited.