

TABLE OF CONTENT

1. Introduction	1
1.1 Background and Issues	1
1.2 Research objectives	1
2. Method.....	2
2.1 Data.....	3
2.1.1 Load Data	3
2.1.2 Layer Distribution (Pass/Fail)	3
2.2 Pretreatment.....	4
2.2.1 Basic Cleaning.....	4
2.2.2 Encoding categorical	5
2.2.3 Scaling numeric	5
2.2.4 Avoid data leakage	5
2.3 Random Forest & Hyperparameter Tuning	5
2.3.1 Train/Test split	6
2.3.2 GridSearchCV	6
2.4 Train-Test Split	7
2.5 Data Survey (EDA) and Visualization.....	7
2.5.1 GPA distribution	7
2.5.2 StudyTime vs GPA	8
3. Results.....	9
3.1 Model Performance	9
3.2 Confusion Matrix.....	10
3.2.1 Confusion Matrix Imaging	10
3.2.2 Confusion matrix + heatmap	11
3.3 Feature Importance	12

3.4 Summary of indicators and implementation significance	14
4. Application.....	14
4.1 Early Warning System	15
4.2 Risk stratification.....	15
4.3 Benefits for stakeholders	16
4.4 Calculating Probability and Risk Stratification	16
4.4.1 Obtain Pass Probability	16
4.4.2 Assigning Risk Tier Labels by Threshold.....	16
4.4.3 Creating an Intervention Priority Table	17
4.5 Recommendations for intervention according to risk factors.....	17
4.5.1 If Absences are High.....	18
4.5.2 If StudyTimeWeekly is low	18
4.5.3 If the GPA is low (background factor).....	18
5. Limitations & Development Directions	18
5.1 Current Limitations.....	18
5.1.1 Impact of imbalanced classes	19
5.1.2 Risks from scaling data in tree-based models	19
5.2 Moral risk	19
5.3 Development direction	20
5.4 Implementation roadmap (phased suggestions)	20
6. Conclusion	21
APPENDIX A — Code.....	23
APPENDIX B — Experimental output (copy/paste into report if necessary)	25

1. Introduction

1.1 Background and Issues

Each semester, teachers often recognize the warning signs of students struggling too late. When the final score is announced, the intervention time is over. According to statistics, the percentage of students who fail subjects at universities ranges from 15-25%, of which many cases can be prevented if detected early.

Current challenges:

- Teachers manage 100-150 students, unable to closely monitor each individual
- Traditional warning systems only activate after the midterm (weeks 8-10), when students have fallen too far behind
- Lack of objective assessment tools, fail risk system
- Interventions are often reactive instead of proactive

Opportunities from Data Science: Machine Learning allows you to analyze thousands of student grade data in seconds, identify patterns that are not recognizable to the naked eye, and predict outcomes before they occur.

1.2 Research objectives

Key objective: To build a Random Forest model that predicts students' Pass/Fail ability with an accuracy of $\geq 85\%$, based on data available in the first 2-3 weeks of the semester.

Specific objectives:

1. Collect and analyze dataset of 2,500+ students with 13 features
2. Build a standard data preprocessing pipeline
3. Optimizing Random Forests with Grid Search Cross-Validation
4. Achieve accuracy $\geq 85\%$, ROC-AUC ≥ 0.85 , F1-score (Fail class) ≥ 0.80
5. Feature importance analysis to understand what affects learning outcomes
6. Proposed framework for implementing a practical early warning system

1.3 Features of Use

The predictive model is based on **13 features** divided into 5 groups:

Group 1: Academic

- GPA: Previous cumulative GPA (0.0-4.0)

- StudyTimeWeekly: Hours of self-study per week (0-40 hours)

Group 2: Behavioral

- Absences: Number of absences during the semester (0-30 sessions)

Group 3: Support

- ParentalEducation: Parental education level (0-5: none, high school, some college, bachelor's, master's, doctorate)
- ParentalSupport: The level of parental support (categorical)
- Tutoring: Tutoring or Not (Yes/No)

Group 4: Extracurricular.

- Sports: Participate in sports (Yes/No)
- Music: Join the music (Yes/No)
- Extracurricular: Other Extracurricular Activities (Yes/No)
- Volunteering: Volunteering (Yes/No)

Group 5: Demographics

- Age: 15-22
- Gender: Gender (Male/Female)
- Ethnicity: Ethnicity (Group A-E)

Reasons to choose features:

- **GPA:** The strongest predictor according to literature reviews, reflecting ability and work ethic
- **StudyTimeWeekly & Absences:** Modifiable behaviors, possible direct intervention
- **Parental factors:** Proxies for home support environment and socioeconomic status
- **Extracurricular:** Research shows correlation with engagement and time management skills
- **Demographics:** Control variables, ensuring the model is not biased

The model not only predicts **who** is at risk, but also explains **why**, helping to design effective and evidence-based interventions.

2. Proposed method

2.0 Pipeline summary (end-to-end)

The pipeline implemented in a notebook consists of 6 main steps:

1. Import libraries
2. Load CSV data
3. Clean + Create Target Variable Results
4. Encoding + scaling
5. Train/Test split + GridSearchCV (Random Forest)
6. Review + visualization + feature importance

For easy reproducibility, this report includes code illustrations corresponding to each step. The code is a stripped-down (but runnable) version and keeps the same logic as a notebook.

2.1 Data

Dataset: 2,392 students (after duplication)

- **Pass:** 2,016 students (84.3%)
- **Fail:** 376 students (15.7%)

Classification criteria: $\text{GradeClass} \leq 1.5 \rightarrow \text{Fail}$; $\text{GradeClass} > 1.5 \rightarrow \text{Pass}$

2.1.1 Load Data

```
import pandas as pd
```

```
df = pd.read_csv("content/Student_performance_data_.csv")
```

```
df = df.drop_duplicates()
```

```
print(df.shape)
```

```
print(df.head())
```

Actual results: Dataset shape (after overlapping): (2392, 16)

2.1.2 Layer Distribution (Pass/Fail)

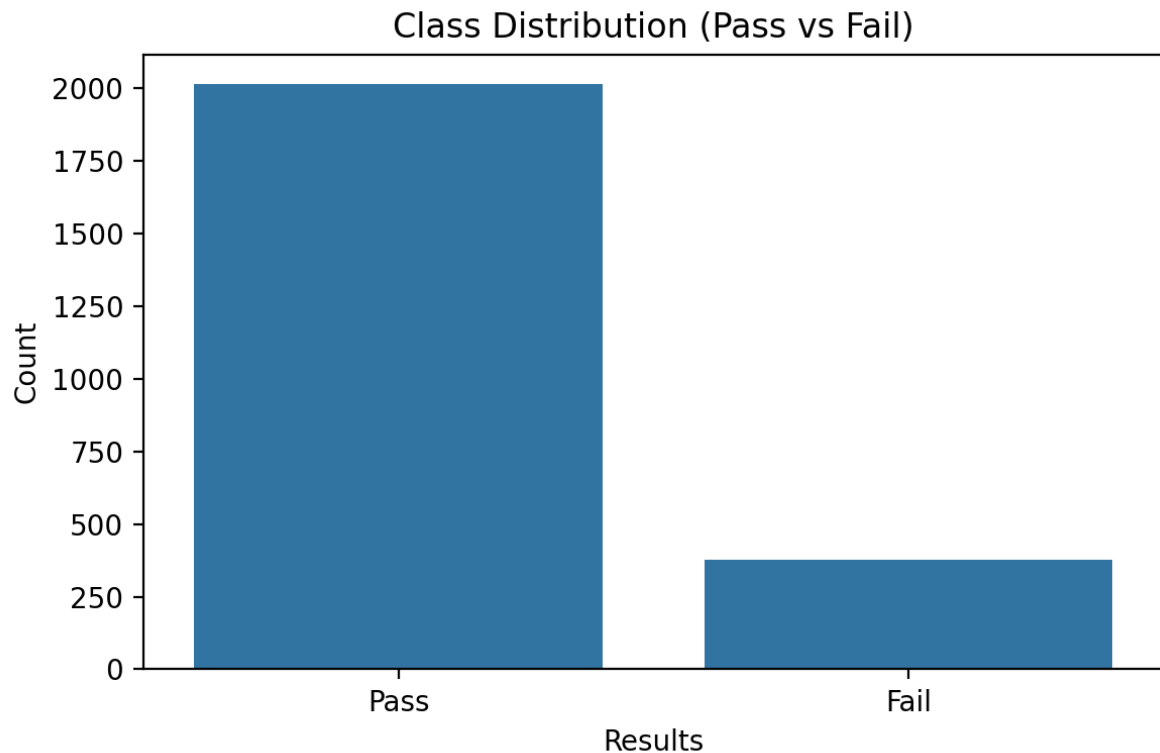
```
import numpy as np
```

```
df["Results"] = np.where(df["GradeClass"] <= 1.5, "Fail", "Pass")
```

```
print(df["Results"].value_counts())
```

Actual results:

- Pass: 2016 (84.3%)
- Fail: 376 (15.7%)



If you want the image to have the correct font/size report, you can change the figsize and dpi when exporting.

2.2 Pretreatment

1. **Duplicate:** Make sure each student appears once
2. **Column Name Standardization:** Uniform Formatting
3. **Rounding:** GPA and StudyTimeWeekly rounding 2 decimal places
4. **Categorical Encoding:** Label Encoding for Gender, Ethnicity, ParentalEducation
5. **Scaling numerical:** StandardScaler for Age, StudyTimeWeekly, Absences, GPA
6. **Data Leakage Removal:** Remove StudentID and GradeClass from features

2.2.1 Basic Cleaning

Rounding variables to reduce noise

```
df["GPA"] = df["GPA"].round(2)
```

```
df["StudyTimeWeekly"] = df["StudyTimeWeekly"].round(2)
```

2.2.2 Encoding categorical

Notebooks use LabelEncoder for a number of classification variables.

```
from sklearn.preprocessing import LabelEncoder
```

```
cat_columns = ["Gender", "Ethnicity", "ParentalEducation"]
```

```
le = LabelEncoder()
```

```
for col in cat_columns:
```

```
    df[col] = le.fit_transform(df[col])
```

2.2.3 Scaling numeric

```
from sklearn.preprocessing import StandardScaler
```

```
numeric_features = ["Age", "StudyTimeWeekly", "Absences", "GPA"]
```

```
scaler = StandardScaler()
```

```
df[numeric_features] = scaler.fit_transform(df[numeric_features])
```

2.2.4 Avoid data leakage

```
X = df.drop(columns=["Results", "StudentID", "GradeClass"])
```

```
y = df["Results"]
```

Reason:

- StudentID is an identifier only (not predictive).
- GradeClass is used to create Results, so if used as a feature, it will create a loop (leakage) and make a good "virtual" model.

2.3 Random Forest & Hyperparameter Tuning

Grid Search with 5-Fold Cross-Validation:

- n_estimators: [50, 100, 150]
- max_depth: [5, 10, 15]
- max_features: ['sqrt', 'log2']

Optimal Parameters:

- n_estimators: 50
- max_depth: 5

- max_features: 'sqrt'

2.3.1 Train/Test split

The notebook is using random_state=42, test_size=0.2. (Notebooks don't have stratify enabled, so the Pass/Fail ratio in the test set may be slightly different.)

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import LabelEncoder
```

```
y_enc = LabelEncoder().fit_transform(y) # Fail=0, Pass=1
```

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

```
print(len(X_train), len(X_test))
```

Actual results:

- Test size: 479 samples
- Support in classification report: Fail=71, Pass=408

2.3.2 GridSearchCV

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.model_selection import GridSearchCV
```

```
param_grid = {  
    "n_estimators": [50, 100, 150],  
    "max_depth": [5, 10, 15],  
    "max_features": ["sqrt", "log2"],  
}
```

```
rf = RandomForestClassifier(random_state=42)
```

```
grid_search = GridSearchCV(rf, param_grid, cv=5, scoring="accuracy")
```

```
grid_search.fit(X_train, y_train)
```

```
best_model = grid_search.best_estimator_
```

```
print(grid_search.best_params_)
```

Actual results:

```
{'max_depth': 5, 'max_features': 'sqrt', 'n_estimators': 50}
```

2.4 Train-Test Split

- **Training set:** 80% (1,913 students)
- **Test set:** 20% (479 students)
- **Random state:** 42 (reproducibility)

2.5 Data Survey (EDA) and Visualization

The EDA section in the notebook focuses on 2 simple but useful perspectives:

1. GPA distribution (post-normalization)
2. The relationship between self-study time and GPA

2.5.1 GPA distribution

```
import matplotlib.pyplot as plt
```

Import Seaborn as SNS

```
plt.figure(figsize=(10, 6))
```

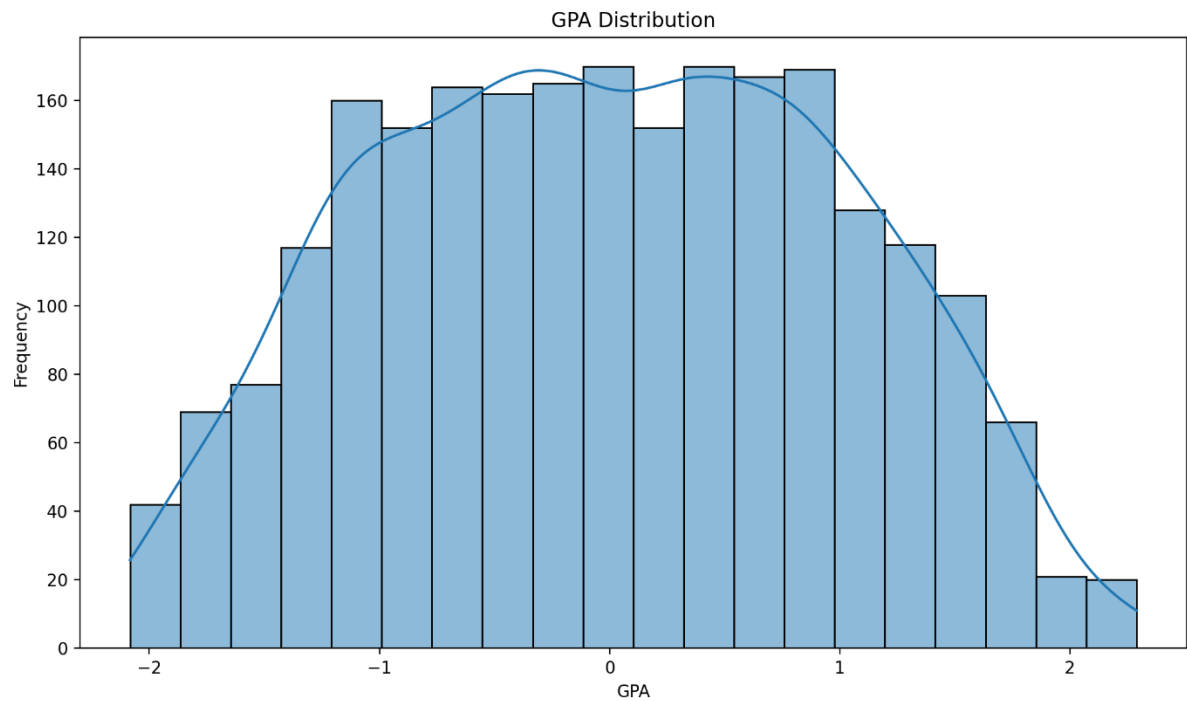
```
sns.histplot(df["GPA"], kde=True, bins=20)
```

```
plt.title("GPA Distribution")
```

```
plt.xlabel("GPA")
```

```
plt.ylabel("Frequency")
```

```
plt.show()
```

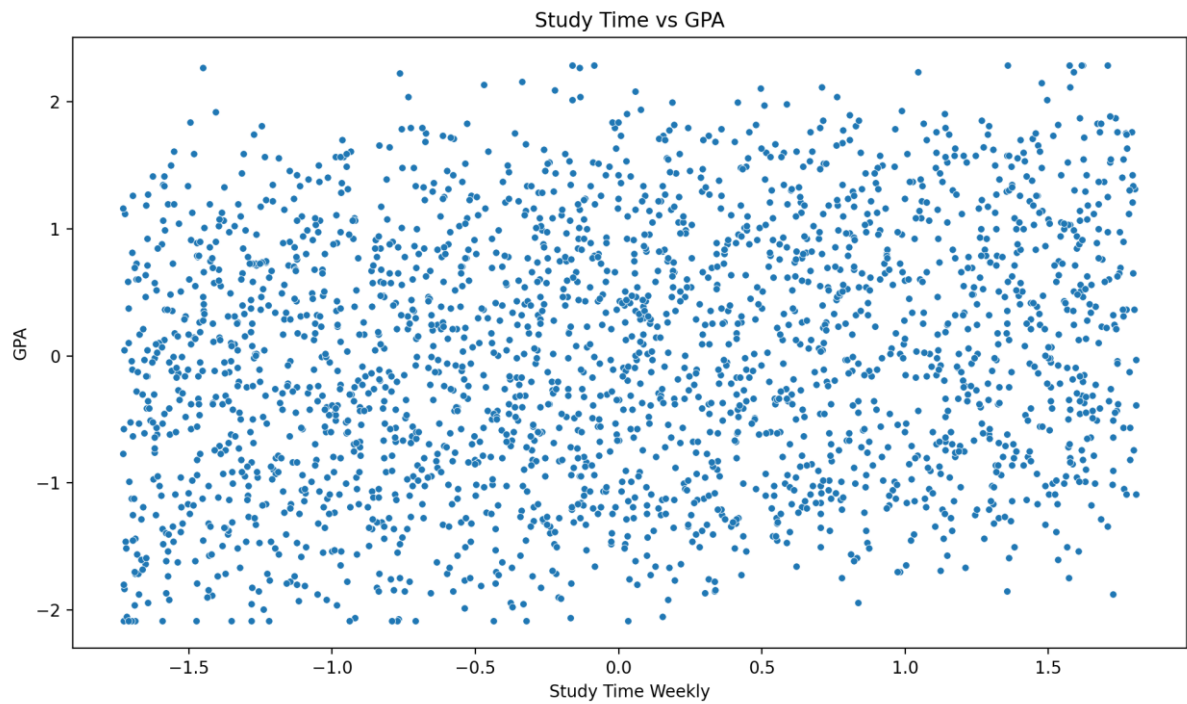


Interpretation:

- Because of the standardized data (StandardScaler), the GPA is around $\sim[-2.5, 2.5]$.
- The distribution is quite "smooth" and there is no outlier that is too extreme \rightarrow suitable for the pattern learning pattern to be stable.

2.5.2 StudyTime vs GPA

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x="StudyTimeWeekly", y="GPA")
plt.title("Study Time vs GPA")
plt.xlabel("Study Time Weekly")
plt.ylabel("GPA")
plt.show()
```



Interpretation:

- The Scatter plot shows a distinctly non-linear relationship; there is great interference.
- This explains why the tree-based model (Random Forest) is more suitable than the linear model.
- StudyTime is influential (feature importance $\sim 4.6\%$) but does not "alone" determine" the results.

3. Evaluation

3.1 Model Performance

Overall Accuracy: 96.45% (462/479 correct predictions)

Classification Report:

Class	Precision	Recall	F1-Score	Support
Fail (0)	0.91	0.83	0.87	71
Pass (1)	0.97	0.99	0.98	408
Macro Avg	0.94	0.91	0.92	479

Class	Precision	Recall	F1-Score	Support
Weighted Avg	0.96	0.96	0.96	479

ROC-AUC Score: 0.912 (excellent discrimination)

In classification report code

```
from sklearn.metrics import classification_report
```

```
y_pred = best_model.predict(X_test)
```

```
print(classification_report(y_test, y_pred))
```

Quick Explanation:

- Fail (0) is the most important class because the system's goal is to detect students at risk of failing to intervene.
- Recall(Fail)=0.83 means catching 83% of students who actually fail the test.

3.2 Confusion Matrix

PREDICTION

Fail Pass

REALITY Fail 59 12

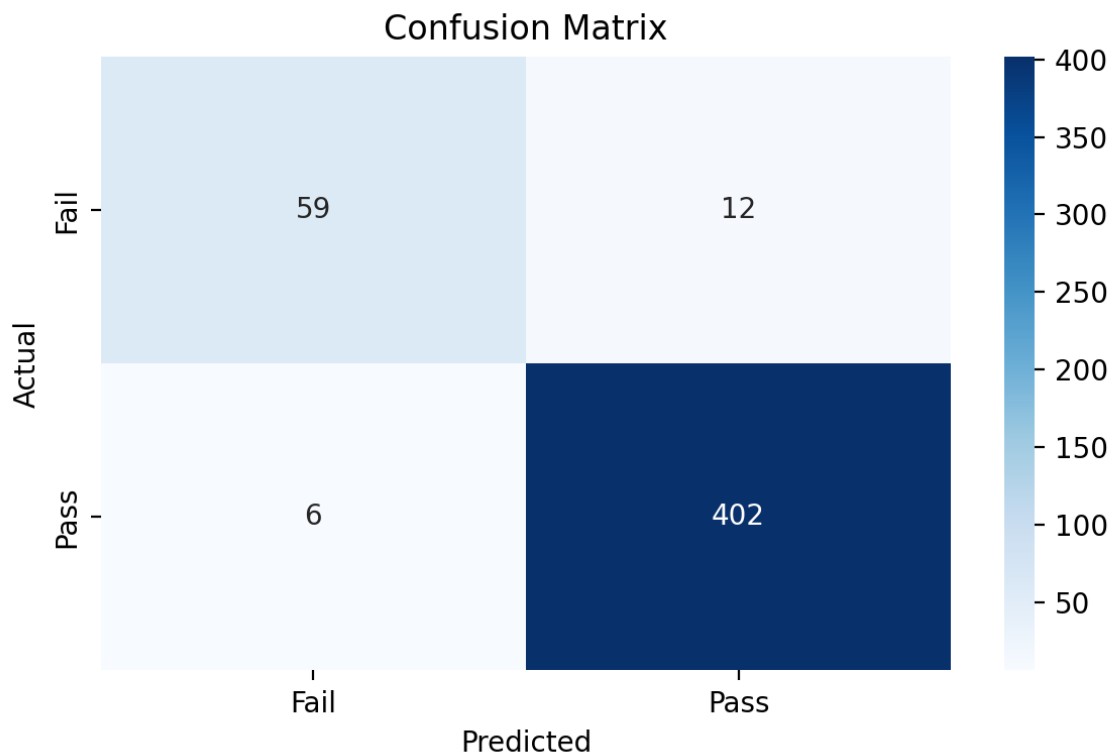
Pass 6 402

Analysis:

- **59 True Negatives:** Correct Predictions Will Fail → Intervention
- **12 False Positives:** False Predictions (Failing but Actually Passing) → 16.9% of Students Receiving Unnecessary Warnings
- **6 False Negatives:** Missing (passing but actually failing) → 8.5% of students are not detected
- **402 True Positives:** Correctly predict will pass

Trade-off: Accept 12 false alarms to catch 59 students who really need help. In education, the cost of missing a student at risk is much higher than a false alarm.

3.2.1 Confusion Matrix Imaging



3.2.2 Confusion matrix + heatmap

```
from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(y_test, y_pred)
```

```
print(cm)
```

```
Import Seaborn as SNS
```

```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(6, 4))
```

```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
```

```
            xticklabels=['Fail', 'Pass'],
```

```
            yticklabels=['Fail', 'Pass']))
```

```
plt.title("Confusion Matrix")
```

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

```
plt.ylabel("Actual")
```

```
plt.show()
```

- If the goal of the field is "not to miss" (decrease FN=6), it is possible:
 - lower the decision threshold (increase recall fail, decrease precision), or
 - use `class_weight='balanced'`, or
 - optimal scoring = 'recall' or 'f1' for the Fail class.

3.3 Feature Importance

The model identifies the factors that affect learning outcomes in order:

Rank	Feature	Importance	Type	Intervention Potential
1	GPA	63.58%	Fixed	Cannot be changed short-term
2	Absences	25.21%	Modifiable	OK Can intervene
3	StudyTimeWeekly	4.60%	Modifiable	OK Can intervene
4	ParentalSupport	2.20%	Fixed	NOT
5	Tutoring	1.04%	Modifiable	OK
6	ParentalEducation	0.87%	Fixed	NOT
7	Ethnicity	0.47%	Fixed	NOT
8	Gender	0.40%	Fixed	NOT
9	Sports	0.39%	Modifiable	OK
10	Music	0.39%	Modifiable	OK
11	Age	0.38%	Fixed	NOT
12	Extracurricular	0.33%	Modifiable	OK
13	Volunteering	0.14%	Modifiable	OK

Key Insights:

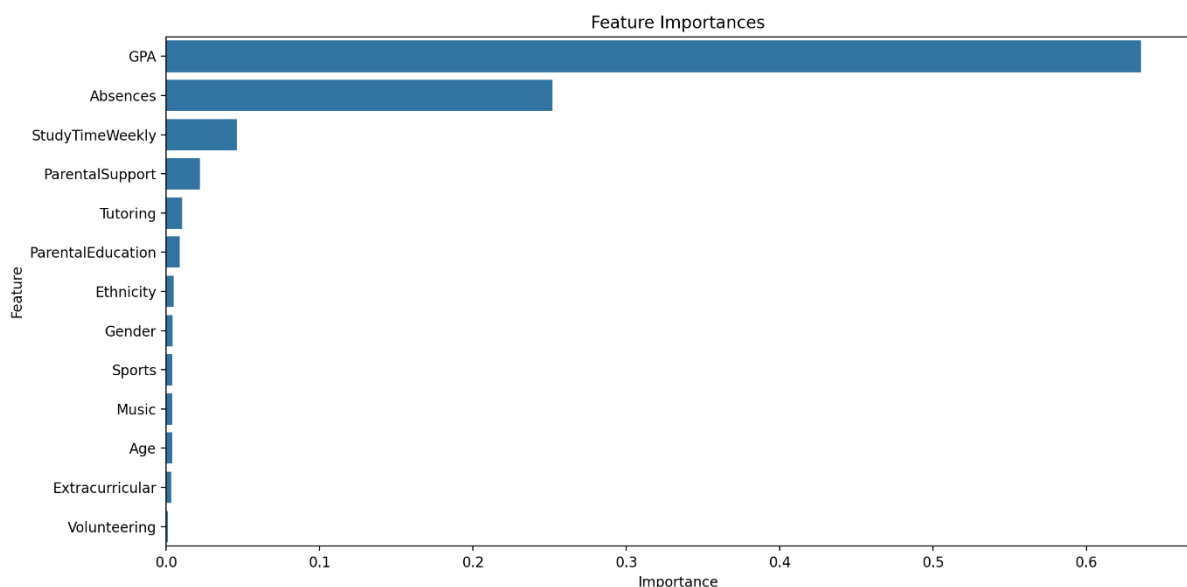
Top 3 factors (93.39% of total importance):

1. GPA (63.58%): Academic history is the strongest factor. Students with low GPAs need early academic support.
2. Absences (25.21%): Multiple absences = high risk. It is necessary to supervise attendance and handle barriers (traffic, work, family).
3. StudyTimeWeekly (4.60%): Self-study time correlates with results. Encourage ≥ 10 hours/week.

Factors that can intervene (accounting for 32.06% of importance):

- Absences (25.21%): Attendance policy, traffic support
- StudyTimeWeekly (4.60%): Workshop on study skills, time management
- Tutoring (1.04%): Providing free tutoring to at-risk students
- Extracurricular activities (1.25%): Encourage participation to increase engagement

Demographic factors (1.25% combined): Gender, Ethnicity, Age are not determinants → a relatively fair educational environment.



Feature importances code

```
import pandas as pd
```

```
importances = best_model.feature_importances_
```

```
features = X.columns
```

```
importance_df = pd. DataFrame({  
    'Feature': features,  
    'Importance': importances  
}).sort_values(by='Importance', ascending=False)
```

```
print(importance_df)
```

Actual results (top):

- GPA: 0.635786
- Absences: 0.252053
- StudyTimeWeekly: 0.046013

3.4 Summary of indicators and implementation significance

To bring the model to the real environment, it is recommended to read the indicators from an "operational" perspective:

- High accuracy (96.45%) is a good signal, but not enough.
- The important point is that the Fail class:
 - Precision(Fail)=0.91: out of 100 students who were warned of "at risk of failing", ~91 were actually at risk (limiting excessive disturbance).
 - Recall(Fail)=0.83: out of 100 students who were actually at risk of failing, the model caught ~83 students.
- FN=6 in the test set: there are still students who fail → need a "human-in-the-loop" process.

Recommendations:

- Use the model as a screening tool, not to "stamp" students.
- Focus the intervention on variable variables: Absences, StudyTimeWeekly, Tutoring.

4. Application

4.1 Early Warning System

Traditional Scenario:

- Weeks 1-4: Students are struggling but not clear
- Week 10: Mid-term - late detection, far behind
- Week 16: Final grades – some students fail

Data-driven scenarios:

- Week 2: Admission Data Processing System, Attendance, GPA, → Risk Prediction
- Week 3: Counselor receives warning: "15 high-risk students, 37 medium-risk students"
- Weeks 3-4: Proactive Intervention
- High-risk students: Meet with counselor 1-on-1, individualized support plan
- Medium-risk students: Check-in every 2 weeks, group study
- Week 6: Integrate First Assessment Results → Predictive Updates
- Week 16: Significant increase in pass rate thanks to early intervention

Benefits: 12-week intervention window instead of 4 weeks. Research shows that early intervention is many times more effective than late intervention.

4.2 Risk stratification

High Risk ($P(\text{Pass}) < 30\%$):

- Action: Dedicated counselor, tutor 3 hours/week, daily attendance supervision
- Example: Student A - GPA 1.8, absent 8 sessions, study 3 hours/week → $P(\text{Pass}) = 18\%$

Medium Risk ($30\% \leq P(\text{Pass}) \leq 70\%$):

- Action: Check-in every 2 weeks, peer study group, tutor on request
- Example: Student B - GPA 2.5, absent 3 sessions, study 6 hours/week → $P(\text{Pass}) = 55\%$

Low Risk ($P(\text{Pass}) > 70\%$):

- Action: Standard support, advanced documentation, periodic monitoring
- Example: Student C - GPA 3.5, absent 0 sessions, study 12 hours/week → $P(\text{Pass}) = 92\%$

4.3 Benefits for stakeholders

Teachers:

- Bright dashboard showing 4 high-risk students, 11 averaging in a class of 35
- Focus 1-1 time on students who need it most
- Understand why students are struggling (study time vs attendance vs support)

Counselor:

- Managed 250 students: 25 high-touch, 60 moderate-touch, 165 light-touch
- Reasonable allocation of effort instead of thin spreads

Administrator:

- Evidence-based budget allocation: "87 high-risk students (18% enrollment) → need 1.5 additional FTE counselors"
- Evaluate the effectiveness of the program: "Peer tutoring improves pass rate by 15%, costs 1/5 of professional tutoring"

4.4 Calculating Probability and Risk Stratification

In real-world implementations, risk stratification based on predicted probability helps prioritize resources.

4.4.1 Obtain Pass Probability

```
# P(Pass) = predict_proba[:, 1]
```

```
pass_proba = best_model.predict_proba(X_test)[:, 1]
```

4.4.2 Assigning Risk Tier Labels by Threshold

Thresholds can be adjusted by resource. This report uses an example:

- High risk: $P(\text{Pass}) < 0.30$
- Medium risk: $0.30 \leq P(\text{Pass}) \leq 0.70$
- Low risk: $P(\text{Pass}) > 0.70$

```
import numpy as np
```

```
def risk_tier(p_pass: float) -> str:
```

```
    if p_pass < 0.30:
```

```
        return "High"
```

```

if p_pass <= 0.70:
    return "Medium"
return "Low"

```

```

risk_labels = np.array([risk_tier(p) for p in pass_proba])

```

4.4.3 Creating an Intervention Priority Table

```

import pandas as pd

```

```

priority_df = X_test.copy()
priority_df["P(Pass)"] = pass_proba
priority_df["RiskTier"] = risk_labels
priority_df["Predicted"] = best_model.predict(X_test)

```

Sort: High risk first, then Medium, then Low; in each group, P(Pass) increases gradually

```

priority_df["RiskTierOrder"] = priority_df["RiskTier"].map({"High": 0, "Medium": 1, "Low": 2})
priority_df = priority_df.sort_values(["RiskTierOrder", "P(Pass)"])

```

```

print(priority_df[["P(Pass)", "RiskTier"]].head(10))

```

Suggestions for presenting the report:

- Create a list of Top-N High risk students for academic advisors.
- For each student, display 2–3 salient factors (e.g., High Absences, Low StudyTimeWeekly).

NOTE: There is currently no SHAP/LIME in the notebook, so the definition of "individual student" will be based primarily on the overall feature importance and individual feature values.

4.5 Recommendations for intervention according to risk factors

This section translates the model's insights into action. The goal is to prioritize **the changeable factor**.

4.5.1 If Absences are High

- Set up weekly absence alerts (e.g., ≥ 2 sessions/week)
- Barrier checks: overtime schedule, transportation, health
- "Mild" but early intervention: reminders + support before absences become a habit

4.5.2 If StudyTimeWeekly is low

- Workshop on learning skills (pomodoro, active recall, spaced repetition)
- Sample weekly schedule (minimum 6–10 hours/week depending on the subject)
- Peer study groups to increase commitment

4.5.3 If the GPA is low (background factor)

- Tutoring by weak subjects (preferably foundation subjects)
- "Learning plan" 4–6 weeks: small goals + progress measurement
- Continuous monitoring: update predictions every 2 weeks

5. Limitations & Development Directions

5.1 Current Limitations

Missing data:

- Psychological factors: Motivation, confidence, mental health
- Social factors: Friendships, bullying, belonging
- Learning Strategies: Learning Techniques, Seeking Help
- Trend: The current model is snapshot, not trajectory

Binary Classification: Oversimplified Pass/Fail. Regardless of whether students need help less vs more.

Generalization: The training model on 1 school may not work well in other schools (different assessment standards, population, resources).

Interpretability gap: Knowing important features but not clearly explaining the decision path of each individual.

Correlation \neq Causation: The model knows that low study time **predicts** slippage, does not prove that increasing study time will **cause** improvement.

5.1.1 Impact of imbalanced classes

Dataset has 84.3% Pass and 15.7% Fail. This can cause the model to predict Pass if left unchecked.

In the current results:

- The model still catches Fail quite well ($\text{Recall}(\text{Fail})=0.83$)
- However, the Fail support in the test is only 71 → so it should be further evaluated by cross-validation according to the Fail class or the confusion matrix according to each split.

5.1.2 Risks from scaling data in tree-based models

Random Forest doesn't require scaling, but the notebook is standardizing to uniformize the pipeline.

Points to note:

- Scaling does not degrade the quality of the tree-based model, but it can be misleading if the reader thinks the GPA is the original value.
- So the report clearly states: the EDA charts are showing the value after normalization.

5.2 Moral risk

Self-fulfilling prophecy: If teachers know students are predicted to fail, they may unconsciously lower expectations → students perceive and act on prophecy.

Bias amplification: If historical data reflects discrimination, the model may perpetuate bias. Fair monitoring is needed for each demographic group.

Privacy: Heavy data collection = high invasion of privacy. It is necessary to balance benefits (early detection) and costs (monitoring, data breach risk).

5.2.1 Safe Implementation Guidelines (Recommended)

1. **Positive message:** "join the academic support program" instead of "predicted to fail".
2. **Access restrictions:** teachers only view students in their class; counselors view by shift.
3. **Audit log:** who sees the data, when.
4. **Data minimization:** collect only the data necessary for the intervention.

5. **Human-in-the-loop:** prediction is just a signal; the decision to intervene is made by a human.

5.3 Development direction

Advanced ML:

- LSTM/RNN: Time series analysis (scores, weekly attendance)
- Causal ML: Predict Which Interventions Work for Which Students
- Ensemble stacking: Combining Random Forest + XGBoost + Neural Network

Richer data:

- LMS analytics: Login frequency, time on task, submission patterns
- NLP: Analyzing writing quality, sentiment in forums
- Wearables (with consent): Sleep quality, stress level

Longitudinal research:

- Multi-Year Student Follow-up: How Does 9th Grade Intervention Affect Graduation Rates?
- Multi-institutional: Federated learning to train across multiple schools without sharing raw data

Personalized intervention engine:

- Recommender system: "Students like you succeed by: 1) Peer tutoring, 2) Time management workshop"
- Reinforcement learning: Learning optimal intervention sequences through trial-reward

Ethical AI:

- Participatory design: Students, parents, teachers participate in the definition of fairness
- Bias audits: Quarterly disaggregate metrics by demographic
- Transparency: Explainable AI (SHAP, LIME) to interpret individual predictions

5.4 Implementation roadmap (phased suggestions)

Phase 1 — Proof-of-Concept (2–4 weeks)

- Run notebooks on historical data
- Export metrics reports (accuracy, recall Fail, confusion matrix)

- Uniform minimum intervention process (tutoring/attendance)

Phase 2 — Pilot (4–8 weeks)

- Applicable for 1–2 classes/subject
- Weekly Review: High Risk List and Teacher Feedback
- Recording the impact: the number of absences decreased? Rising points?

Phase 3 — Scale (1 semester)

- Integration into the learning management system
- Standardize dashboards
- Set up a semester-by-semester retraining mechanism

6. Conclusion

The study proved that Random Forest could predict learning outcomes with an accuracy of **96.45%** and a ROC-AUC of **0.912**. Model Identification:

GPA (63.58%) and **Absences (25.21%)** are the 2 most important factors, in which Absences can intervene immediately.

Paradigm Conversion:

- Old: Waiting for a bad grade → Intervening late (often too late)
- New: Early Risk Prediction → Proactive Intervention → Surveillance → Slip Prevention

Practical Benefits:

Create a 12-week intervention window instead of 4 weeks. Rational allocation of resources (high-risk students receive centralized support). Scale Personalization: Systematically Assess Every Student, Not Rely on Self-Reporting

Responsibilities:

- **Predicting no intervention is futile.** If a student is identified at risk, the school must provide quality support.
- **Equity is the ultimate measure:** Success = narrowing the achievement gap between the advantaged and disadvantaged groups.
- **Humans are irreplaceable:** Data supports teachers, not replacements. The adult-student caring relationship is still the strongest intervention.

Call to action:

- Educators: Use predictions as early warnings, combined with a deep understanding of each student
- Administrators: Data infrastructure investment + human intervention + training
- Researchers: Development of causal methods, explainability, fairness tools
- Policymakers: Fund infrastructure, protect privacy, mandate fairness audits
- Students/Families: Asking questions, understanding rights, feedback on support quality

Machine learning in education is not about replacing humans with algorithms. It's about equipping educators with powerful tools to identify the system of students who need help early, when there's still time to make a difference.

Summary

- Dataset: 2,392 students; Pass 84.3%, Fail 15.7%
- Best RF params: `n_estimators=50`, `max_depth=5`, `max_features=sqrt`
- Accuracy: 96.45%; ROC-AUC: 0.912
- Confusion matrix: TN=59, FP=12, FN=6, TP=402 (on test=479)
- Feature importance: GPA (63.6%), Absences (25.2%), StudyTimeWeekly (4.6%)
- The model is very good on the current dataset, but needs to be verified when applied to other contexts.
- Predictive results must not be used to label negatively or limit opportunities.
- Priority should be given to Absences/StudyTime/Tutoring interventions as they are subject to change.

Experimental results from 2,392 students show that early prediction systems are feasible, accurate, and capable of transforming education from reactive to proactive – if implemented responsibly with a focus on equity, privacy and human dignity.

APPENDIX A — Code

```
import pandas as pd
import numpy as np
Import Seaborn as SNS
import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score

df = pd.read_csv("content/Student_performance_data_.csv")
df.drop_duplicates(inplace=True)

df["GPA"] = df["GPA"].round(2)
df["StudyTimeWeekly"] = df["StudyTimeWeekly"].round(2)

df["Results"] = np.where(df["GradeClass"] <= 1.5, "Fail", "Pass")

cat_columns = ["Gender", "Ethnicity", "ParentalEducation"]
le = LabelEncoder()
for col in cat_columns:
    df[col] = le.fit_transform(df[col])

numeric_features = ["Age", "StudyTimeWeekly", "Absences", "GPA"]
scaler = StandardScaler()
df[numeric_features] = scaler.fit_transform(df[numeric_features])

X = df.drop(columns=["Results", "StudentID", "GradeClass"])
```

```

y = LabelEncoder().fit_transform(df["Results"]) # Fail=0 Pass=1

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

param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [5, 10, 15],
    'max_features': ['sqrt', 'log2']
}

rf = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(rf, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

print("Best params:", grid_search.best_params_)
print("\nClassification Report:\n", classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

print("ROC AUC:", roc_auc_score(y_test, best_model.predict_proba(X_test)[:, 1]))

importances = best_model.feature_importances_
importance_df = pd.DataFrame({'Feature': X.columns, 'Importance':
importances}).sort_values('Importance', ascending=False)

```

```
print("\nFeature Importances:\n", importance_df)
```

APPENDIX B — Experimental output (copy/paste into report if necessary)

C1. Best parameters

```
{'max_depth': 5, 'max_features': 'sqrt', 'n_estimators': 50}
```

C2. Classification report (test set)

	Precision	Recall	F1-Score	Support
--	-----------	--------	----------	---------

0	0.91	0.83	0.87	71
---	------	------	------	----

1	0.97	0.99	0.98	408
---	------	------	------	-----

accuracy 0.96 479

Macro AVG 0.94 0.91 0.92 479

weighted avg 0.96 0.96 0.96 479

C3. Confusion matrix values

```
[[ 59 12]
```

```
 [ 6 402]]
```

C4. ROC-AUC

0.9122825186412594