# Predicting Student Success with Machine Learning

CSCA 5622: Intro to Machine Learning Supervised Learning

PHPO 4876: Philipp Adrian Pohlmann

University of Colorado Boulder

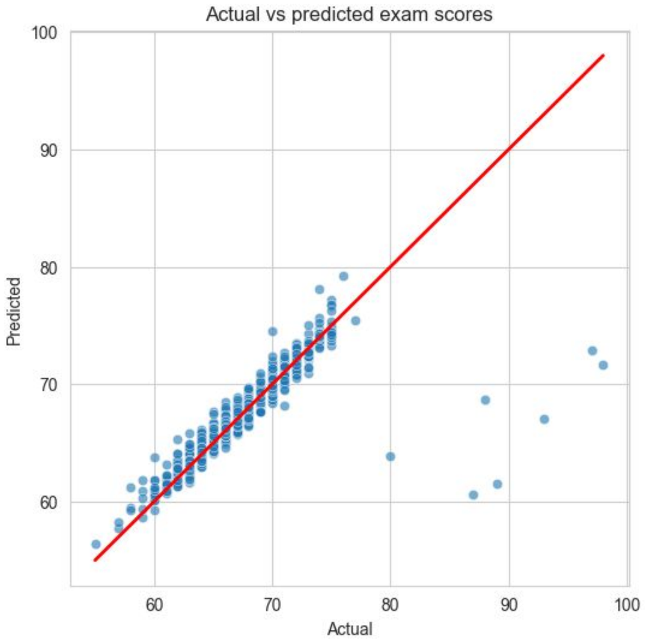
# **The Problem:** Early Identification of At-Risk Students

**The Challenge:**

- Universities have limited resources to support struggling students
- Need to identify students before it's too late
- Traditional methods rely on intuition, not data

**Research Approach:**

- Predict exam scores using behavioral and demographic factors
- Compare multiple ML approaches
- Identify highest impact intervention points



# **The Dataset:** Student Performance Factors

**<u>Student Performance Factors</u>**

**Behavioral:**

- Hours studied, Attendance (%), Sleep hours, Tutoring sessions

**Demographic:**

- Family income, Parental education, Distance from home, Gender

**School-Related:**

- Teacher quality, Access to resources, School type, Peer influence

**Target: Exam Score (range: 55-101 points)**

```
Dataset loaded successfully
Shape: 6,607 rows × 20 columns
```

| | Hours_Studied | Attendance | Parental_Involvement | Access_to_Resources |
|---|---|---|---|---|
| **0** | 23 | 84 | Low | High |
| **1** | 19 | 64 | Low | Medium |
| **2** | 24 | 98 | Medium | Medium |
| **3** | 29 | 89 | Low | Medium |
| **4** | 19 | 92 | Medium | Medium |

# Machine Learning Models

## Data Preprocessing

✓ Missing values: <2% (filled)
✓ Categorical encoding: 13 text variables
✓ Feature scaling: Standard Scaler normalization
✓ 80/20 train / test split

**Models Tested (Simple → Complex):**

- Linear Regression (baseline)
- Decision Tree & Random Forest
- Gradient Boosting
- Support Vector Regression (SVR)
- Polynomial Regression (2nd degree)

```python
# Define models to compare
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(random_state=42),
    "Random Forest": RandomForestRegressor(n_estimators=100, random_state
    "Gradient Boosting": GradientBoostingRegressor(random_state=42),
    "SVR": SVR(kernel='rbf')
}

results = {}

# Train and evaluate models
for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

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

    results[name] = {"RMSE": rmse, "r2": r2, "MAE": mae}
    print(f"{name} -> RMSE: {rmse:.3f}, r2: {r2:.3f}, MAE: {mae:.3f}")

# Convert results
results_df = pd.DataFrame(results).T
results_df
```

# **Model Results:** Why simple models win

**Performance Comparison**

🏆 **Winner:**

- Linear Regression: $R^2 = 0.77$ | RMSE = 1.80 | MAE = 0.45

**Close seconds:**

- SVR: $R^2 = 0.76$
- Polynomial Regression: $R^2 = 0.75$

**Underperformers:**

- Random Forest: $R^2 = 0.65$
- Decision Tree: $R^2 = 0.00$ (failed to generalize)

**Key Finding: Simplest model outperformed complex ensembles**

| | Model | RMSE | r2 | MAE |
|---|---|---|---|---|
| 0 | Linear regression | 1.80 | 0.77 | 0.45 |
| 1 | SVR | 1.84 | 0.76 | 0.51 |
| 2 | Polynomial regression | 1.89 | 0.75 | 0.64 |
| 3 | Gradient boosting | 1.90 | 0.74 | 0.69 |
| 4 | Tuned random forest | 2.18 | 0.67 | 1.14 |
| 5 | Random forest | 2.23 | 0.65 | 1.18 |
| 6 | Decision tree | 3.77 | 0.00 | 1.89 |

# What Drives Student Success?

<u>**Top Predictors of Student Success**</u>

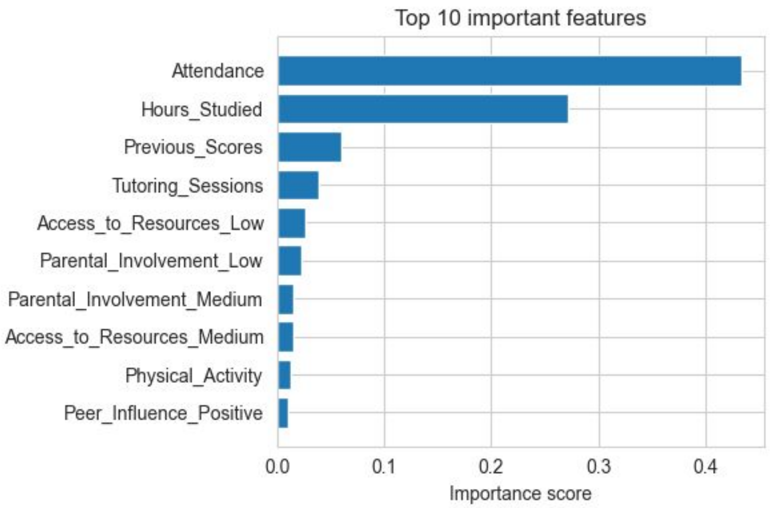
**#1: Attendance (~40% of model importance)** *Showing up is the single biggest factor*

**#2: Hours Studied (~30% importance)** *Consistent study time drives results*

**#3: Previous Scores (~10% importance)** *Prior achievement remains predictive*

**Surprisingly Low Impact:**

- **Family income, Access to resources, Teacher quality**
  - Note: These factors likely affect behavior (attendance, study habits) but aren't direct predictors once you control for behavior



# **Model Validation:** Does The Model Work?

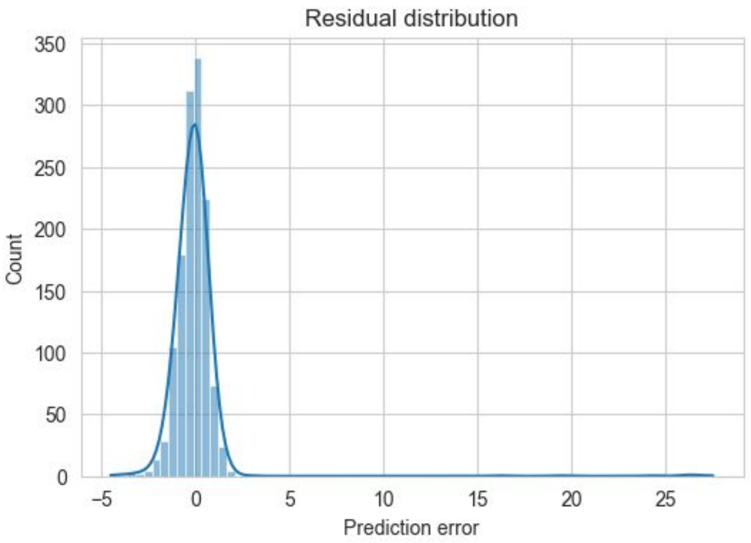
**<u>How can we know this works?</u>**

✓ Predictions track actual scores closely (see plot)

✓ Residuals normally distributed around zero

✓ Test $R^2$ (0.77) > Training $R^2$ (0.72), i.e. no overfitting, model generalizes well

**Avg. prediction error: ±1.8 points which is Accurate enough for early intervention decisions**



# Prediction Quality in Practice

<u>**Real-World Application**</u>

- Model identifies students likely to score <65
- Flag for early intervention

**Offers:** tutoring, study skills workshops, attendance monitoring

**Example Predictions:**

Student A: Predicted 68 → Actual 67 ✓

Student B: Predicted 73 → Actual 75 ✓

Student C: Predicted 59 → Actual 61 ⚠️ (flagged correctly)

**Accuracy enables proactive support, not just reactive**

| Student ID | Predicted | Actual | Error | Flag? |
|---|---|---|---|---|
| mjk2847 | 68 | 67 | +1 | No |
| tes5392 | 73 | 75 | -2 | No |
| rcb1039 | 59 | 61 | -2 | **Yes** |
| daf4158 | 62 | 64 | -2 | **Yes** |
| lmw8914 | 81 | 79 | +2 | No |
| jsk2763 | 56 | 58 | -2 | **Yes** |

# Hyperparameter Tuning

**Could tuning improve Random Forest?**

- Grid Search: Tested 120 parameter combinations
- Best configuration: max_depth=20, n_estimators=200 w/ cross-validation

**Result:**

- Tuned RF: $R^2 = 0.67$ wich was much worse than Linear Regression

**Takeaway: Data structure is inherently linear and adding complexity doesn't capture additional signal**

```python
# Evaluate tuned model
best_rf = grid_search.best_estimator_
y_test_pred_tuned = best_rf.predict(X_test_scaled)

rmse_tuned = np.sqrt(mean_squared_error(y_test, y_test_pred_tuned))
r2_tuned = r2_score(y_test, y_test_pred_tuned)
mae_tuned = mean_absolute_error(y_test, y_test_pred_tuned)

print("Tuned Random Forest performance:")
print(f"Test RMSE: {rmse_tuned:.3f}")
print(f"Test r2: {r2_tuned:.3f}")
print(f"Test MAE: {mae_tuned:.3f}")
```

```
Tuned Random Forest performance:
Test RMSE: 2.175
Test r2: 0.665
Test MAE: 1.137
```

# **Looking Forward:** AI Assisted Learning

**<u>Beyond Traditional Factors</u>**

- Current model explains 77% of variance but learning is evolving rapidly

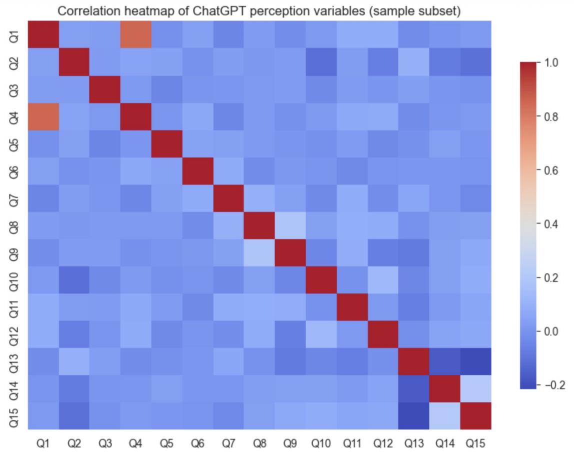
**Exploratory Dataset:**

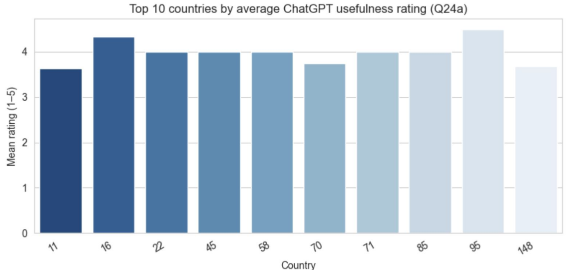
- ChatGPT Usage in Education: 23,218 students in 109 countries
- Survey on AI tooling usage and learning perceptions

**Future Predictors:**

- AI usage frequency for problem-solving, perceived learning efficiency with AI tooling, or confidence in AI-assisted work

**Challenge: Different student populations and would require data from the same cohort**



# Key Takeaways

**#1 Simple models can outperform complex ones**

- Linear Regression ($R^2$=0.77) beat all ensemble methods
- Always start with an interpretable baseline
- Add complexity only if it adds value

**#2 Behavior matters most for student success**

- Attendance and study hours are highest-leverage factors
- Focus interventions here for maximum impact

**#3 Traditional factors explain 77%, but learning is evolving**

- AI-assisted learning may add new predictive signals
- Future models should integrate both traditional + emerging behaviors

# Thank You!

University of Colorado **Boulder**