Student Dropout Predictor

Romerico David

May 15, 2025

Contents

1 Introduction and Motivation

For this project, we used the *Predict Students' Dropout and Academic Success* dataset from UC Irvine's Machine Learning Repository. This dataset was constructed by merging several disjoint databases maintained by a higher-education institution, and it contains records from 4,424 undergraduate students enrolled in programs such as agronomy, design, education, nursing, journalism, management, social service, and technology-related disciplines.

Each student is described by 36 features encompassing academic background, demographic attributes, and socio-economic factors available at the time of enrollment. Additionally, it includes academic-performance data collected at the end of the first and second semesters. These features span a range of data types, including real-valued, categorical, and integer values.

The primary objective of this project is to develop a Support Vector Machine (SVM) model to classify students into one of two categories: those who have dropped out and those who have either graduated or are still enrolled. This reduces the original three-class problem into a binary classification task.

Because of the dataset's potential to effectively predict student dropout and its personal relevance to us as college students, we thought it was worthwhile to explore this set of data that can be used to support early-intervention strategies in educational settings.

2 Data Preprocessing

2.1 Plan for Preprocessing

Given that the dataset contains **36 features** and **4,424 instances**, a significant amount of data preprocessing was required to ensure the model could be trained effectively due to the large discrepancy between the number of features and instances.

Support Vector Machines (SVMs) work by identifying a decision boundary that maximizes the margin between classes. This process relies heavily on dot products and distance calculations, which are sensitive to the magnitudes of individual feature values. If features are on vastly different scales, those with larger numerical ranges can disproportionately influence the model's decision boundary. So, it is essential to scale all features to a common range.

Categorical and nominal features are particularly difficult to scale consistently, and naïve one-hot encoding can introduce unintended ordinal relationships or other unintended issues. To address these challenges, we applied the following preprocessing steps:

1. **Removed categorical and nominal features** due to the difficulty of scaling them meaningfully and their incompatibility with SVMs (without complex encoding). This step reduced noise and ensured all remaining features were ordinal and numerically meaningful.

2. **Assessed the target class distribution** by analyzing the proportion of students who dropped out versus those who did not. An imbalanced class distribution can lead to a skewed model.

3. **Statistically analyzed** the remaining features, calculating the minimum, maximum, mean, and median values to identify potential outliers and understand the spread of each variable.

4. **Scaled all ordinal features** to ensure they all share a consistent range.

2.2 Implementation Details and Results

The following screenshots provide quick snippets of the original raw dataset, which notably contains no missing values.

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Previous qualification (grade)	Nacionality	Mother's qualification	Father's qualification	Mother's occupation	Father's occupation
0	1	17	5	171	1	1	122.0	1	19	12	5	9
1	1	15	1	9254	1	1	160.0	1	1	3	3	3
2	1	1	5	9070	1	1	122.0	1	37	37	9	9
3	1	17	2	9773	1	1	122.0	1	38	37	5	3
4	2	39	1	8014	0	1	100.0	1	37	38	9	9

Figure 1: Snippet of the raw dataset (part 1).

Admission grade	Displaced	Educational special needs	Debtor	Tuition fees up to date	Gender	Scholarship holder	Age at enrollment	International	Curricular units 1st sem (credited)	Curricular units 1st sem (enrolled)	Curricular units 1st sem (evaluations)	Curricular units 1st sem (approved)	Curricular units 1st sem (grade)
127.3	1	0	0	1	1	0	20	0	0	0	0	0	0.000000
142.5	1	0	0	0	1	0	19	0	0	6	6	6	14.000000
124.8	1	0	0	0	1	0	19	0	0	6	0	0	0.000000
119.6	1	0	0	1	0	0	20	0	0	6	8	6	13.428571
141.5	0	0	0	1	0	0	45	0	0	6	9	5	12.333333

Figure 2: Snippet of the raw dataset (part 2).

Curricular units 1st sem (without evaluations)	Curricular units 2nd sem (credited)	Curricular units 2nd sem (enrolled)	Curricular units 2nd sem (evaluations)	Curricular units 2nd sem (approved)	Curricular units 2nd sem (grade)	Curricular units 2nd sem (without evaluations)	Unemployment rate	Inflation rate	GDP	Target
0	0	0	0	0	0.000000	0	10.8	1.4	1.74	Dropout
0	0	6	6	6	13.666667	0	13.9	-0.3	0.79	Graduate
0	0	6	0	0	0.000000	0	10.8	1.4	1.74	Dropout
0	0	6	10	5	12.400000	0	9.4	-0.8	-3.12	Graduate
0	0	6	6	6	13.000000	0	13.9	-0.3	0.79	Graduate

Figure 3: Snippet of the raw dataset (part 3).

To begin describing the implementation in concrete terms, we analyzed the dataset and identified that **18 out of the original 36 features were categorical or nominal**. They were:

- Marital status

- Application mode
- Application order
- Course
- Daytime/evening attendance
- Previous qualification
- Nacionality
- Mother's qualification
- Father's qualification
- Mother's occupation
- Father's occupation
- Displaced
- Educational special needs
- Debtor
- Tuition fees up to date
- Gender
- Scholarship holder
- International

These features were therefore removed.

Next, we examined the distribution of dropout versus non-dropout labels to verify that the labels had a reasonably balanced distribution. We found that approximately **67.88%** of students were classified as non-dropouts (either enrolled or graduated), while **32.12%** were labeled as dropouts. This level of class imbalance was considered acceptable for proceeding with model training without requiring additional rebalancing techniques.

Thus, we transformed the target labels for binary classification by assigning a value of `1` to students labeled as *Enrolled* or *Graduate* and a value of `-1` to those labeled as *Dropout*.

Next, we statistically analyzed the instances across each remaining (ordinal) feature to determine the most appropriate scaling technique. Specifically, we examined the **maximum**, **minimum**, **mean**, and **median** values to assess the distribution of values and detect potential outliers. Below are our notable findings:

1. **Age at Enrollment**:
    - Maximum: 70.00

- Mean: 23.27
- Median: 20.00

The maximum age is significantly higher than both the mean and median, indicating a strong right-skew and the presence of high-value outliers.

2. **Curricular Units 1st Semester (Evaluations)**:
   - Maximum: 45.00
   - Mean: 8.30
   - Median: 8.00
3. **Curricular Units 2nd Semester (Evaluations)**:
   - Maximum: 33.00
   - Mean: 8.06
   - Median: 8.00

   In both semesters, the maximum number of evaluations is considerably higher than the mean.
4. **Inflation Rate**:
   - Maximum: 3.70
   - Mean: 1.23
   - Median: 1.40

   The inflation rate has a somewhat elevated maximum relative to the mean and median, but the variation is modest.
5. **GDP**:
   - Minimum: -4.06
   - Mean: 0.002
   - Median: 0.32

   The minimum GDP value is notably lower than the average range, indicating a potential low-end outlier.

While these features contain outliers, we chose not to remove them, as they appear to be *contextually valid* within the domain of the dataset.

Given that the ordinal features in our dataset generally have known and bounded domains, we decided to apply the `MinMaxScaler` from the `scikit-learn` package. This transformation rescales each feature individually to the range $[0, 1]$ based on the formula:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}\,.$$

As a result, the dataset was fully transformed and normalized to the $[0, 1]$ range. The output of this preprocessing step is shown in the screenshots below:

	Previous qualification (grade)	Admission grade	Age at enrollment	Curricular units 1st sem (credited)	Curricular units 1st sem (enrolled)	Curricular units 1st sem (evaluations)	Curricular units 1st sem (approved)	Curricular units 1st sem (grade)	Curricular units 1st sem (without evaluations)	Curricular units 2nd sem (credited)	Curricular units 2nd sem (enrolled)	Curricular units 2nd sem (evaluations)
0	0.284211	0.340000	0.056604	0.0	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.00000	0.000000
1	0.684211	0.500000	0.037736	0.0	0.230769	0.133333	0.230769	0.741722	0.0	0.0	0.26087	0.181818
2	0.284211	0.313684	0.037736	0.0	0.230769	0.000000	0.000000	0.000000	0.0	0.0	0.26087	0.000000
3	0.284211	0.258947	0.056604	0.0	0.230769	0.177778	0.230769	0.711447	0.0	0.0	0.26087	0.303030
4	0.052632	0.489474	0.528302	0.0	0.230769	0.200000	0.192308	0.653422	0.0	0.0	0.26087	0.181818

Figure 4: Scaled dataset after preprocessing (part 1).

Curricular units 2nd sem (approved)	Curricular units 2nd sem (grade)	Curricular units 2nd sem (without evaluations)	Unemployment rate	Inflation rate	GDP	Target	y_labels
0.00	0.000000	0.0	0.372093	0.488889	0.766182	Dropout	-1
0.30	0.735897	0.0	0.732558	0.111111	0.640687	Graduate	1
0.00	0.000000	0.0	0.372093	0.488889	0.766182	Dropout	-1
0.25	0.667692	0.0	0.209302	0.000000	0.124174	Graduate	1
0.30	0.700000	0.0	0.732558	0.111111	0.640687	Graduate	1

Figure 5: Scaled dataset after preprocessing (part 2).

3 Linear SVM using Stochastic Gradient Descent

3.1 Model Overview

We implemented a **soft-margin linear SVM** with *stochastic gradient descent* (SGD). For a training set $\{(x_i, y_i)\}_{i=1}^{n}$ with labels $y_i \in \{-1, +1\}$, the hinge-loss optimization problem is

$$\min_{w,b} \frac{1}{2}\|w\|^2 \; + \; C\sum_{i=1}^{n} \max\bigl(0,\, 1 - y_i(w \cdot x_i + b)\bigr),$$

where $C > 0$ controls the trade-off between margin size and hinge-loss violations (i.e. slack variables). Because hinge loss and the ℓ_2 penalty are both convex, the SGD procedure will converge to a minimizer under a diminishing step size.

3.2 Training and Testing the Model

a) **Data split**: The processed dataset was divided into 80% training and 20% testing.

b) **Intercept term**: A column of ones was appended to x so that the intercept b is learned as an additional weight.

c) **Update rule**: At each iteration t, we sampled one instance (x_i, y_i) and updated the running parameter

$$\theta \leftarrow \begin{cases} \theta + y_i\, x_i & \text{if } y_i(w \cdot x_i) < 1, \\ \theta & \text{otherwise,} \end{cases}$$

where $w_t = \dfrac{\theta}{2Ct}$. The final weight vector is the time-average $w = \dfrac{1}{T}\displaystyle\sum_{t=1}^{T} w_t$.

d) Regularization Term $C = 0.01$.

e) $T = 1{,}000$ iterations were (randomly) used to converge on the 3.5k training points.

```python
# Train a soft-margin linear SVM via SGD using hinge loss and regularization term C
def train_linear_svm(X, y, C=0.01, T=1000):
    """
    Runs T iterations of SGD on the hinge loss SVM objective.
    Returns the averaged weight vector w_bar.
    """
    n, d = X.shape
    theta = np.zeros(d)
    w_sum = np.zeros(d)

    for t in range(1, T + 1):
        # form the current predictor
        w_t = theta / (2 * C * t)

        # uniformly pick one point at random
        i = np.random.randint(n)

        if y[i] * (w_t.dot(X[i])) < 1:
            # only on violations do we update theta
            theta = theta + y[i] * X[i]

        # otherwise theta stays the same

        w_sum += w_t

    # return the average of all w_t's
    return w_sum / T
```

Figure 6: Python implementation of the SGD-based linear SVM.

The trained linear SVM produced the following coefficients and intercept:

Feature	Coefficient
Intercept	−0.5773
Previous qualification (grade)	−0.0590
Admission grade	−0.0077
Age at enrollment	−0.5452
Curricular units 1st sem (credited)	−0.0174
Curricular units 1st sem (enrolled)	−0.2587
Curricular units 1st sem (evaluations)	−0.2724
Curricular units 1st sem (approved)	0.5804
Curricular units 1st sem (grade)	0.8073
Curricular units 1st sem (without eval.)	−0.0482
Curricular units 2nd sem (credited)	0.0207
Curricular units 2nd sem (enrolled)	−0.3253
Curricular units 2nd sem (evaluations)	−0.0777
Curricular units 2nd sem (approved)	1.0302
Curricular units 2nd sem (grade)	1.9480
Curricular units 2nd sem (without eval.)	−0.1016
Unemployment rate	−0.0159
Inflation rate	0.0068
GDP	−0.0267

Table 1: Learned feature coefficients from the trained linear SVM.

Using the learned coefficients and intercept, the model achieved an accuracy of **82.0%** on the test dataset.

3.3 Identifying Significant Features via 5-Fold Cross-Validation

While a soft-margin linear SVM addresses non-linearly separable data, it is inherently limited to finding a linear decision boundary in the original feature space. Given the structure of our dataset—comprising complex, interrelated academic and demographic factors—it is reasonable to assume that the true boundary between dropout and non-dropout cases is non-linear.

To address this, we turn to a very powerful class of models: **kernel-based SVMs** trained using **dual optimization**. These models operate by implicitly mapping the original feature vectors into a higher-dimensional space via a kernel function (e.g., polynomial or Gaussian), where linear separation is more likely to be achievable.

This approach allows the SVM to model more complex relationships between features without explicitly computing the high-dimensional transformation, thanks to the kernel trick.

Nevertheless, to maximize our use of the linear SVM, we leveraged it as a tool for **statistical feature analysis**. We can analyze the learned coefficients to identify the most significant features that contribute to the classification performance.

To do this, we employed a **five-fold cross-validation** procedure using `KFold` from the `scikit-learn` library. We explicitly set `shuffle=False` to ensure that the fold partitions

remained fixed across runs. This helped maintain consistency in how each feature contributed to the model.

Cross-validation was conducted on the 80% training set, with each fold splitting the data into 60% for training and 20% for validation. In each fold, the linear SVM was trained on the 60% subset and evaluated on the corresponding 20% validation set. We recorded both the learned weight vector and validation accuracy in each iteration. The average cross-validation performance was:

$$\text{CV accuracy} = 0.805 \pm 0.004\,.$$

We evaluated several statistical metrics from the cross-validation, which produced the following results:

	feature	mean_w	std_w	t_stat	p_value
12	Curricular units 2nd sem (approved)	1.187810	0.103578	25.642668	0.000014
18	bias	-0.597130	0.061461	-21.724668	0.000027
13	Curricular units 2nd sem (grade)	1.856012	0.191950	21.621134	0.000027
2	Age at enrollment	-0.787903	0.120257	-14.650351	0.000126
6	Curricular units 1st sem (approved)	0.671046	0.119692	12.536371	0.000233
7	Curricular units 1st sem (grade)	0.921117	0.208184	9.893572	0.000586
15	Unemployment rate	-0.195260	0.055855	-7.816890	0.001446
10	Curricular units 2nd sem (enrolled)	-0.160337	0.077423	-4.630743	0.009802
4	Curricular units 1st sem (enrolled)	-0.129548	0.071758	-4.036904	0.015644
1	Admission grade	0.158843	0.115114	3.085490	0.036735
16	Inflation rate	-0.127070	0.099765	-2.848072	0.046486
5	Curricular units 1st sem (evaluations)	-0.143837	0.124103	-2.591617	0.060581
14	Curricular units 2nd sem (without evaluations)	-0.043278	0.084940	-1.139313	0.318169
8	Curricular units 1st sem (without evaluations)	-0.018786	0.055359	-0.758804	0.490225
11	Curricular units 2nd sem (evaluations)	-0.047160	0.147164	-0.716568	0.513252
3	Curricular units 1st sem (credited)	-0.035458	0.145563	-0.544687	0.614913
9	Curricular units 2nd sem (credited)	-0.019186	0.093032	-0.461141	0.668657
17	GDP	-0.023902	0.180467	-0.296159	0.781848
0	Previous qualification (grade)	-0.008185	0.113540	-0.161199	0.879750

Figure 7: Statistical analysis of feature weights across fixed five-fold cross-validation.

Based on these, features with a p-value less than 0.05 were considered **statistically significant** and retained for further modeling. Conversely, features with p-values greater than or equal to 0.05 were excluded from the dataset.

Significant Features ($p < 0.05$):

- Curricular units 2nd sem (approved)
- Curricular units 2nd sem (grade)
- Age at enrollment
- Curricular units 1st sem (approved)
- Curricular units 1st sem (grade)
- Unemployment rate
- Curricular units 2nd sem (enrolled)
- Curricular units 1st sem (enrolled)
- Admission grade
- Inflation rate

Non-Significant Features ($p \geq 0.05$):

- Curricular units 1st sem (evaluations)
- Curricular units 2nd sem (without evaluations)
- Curricular units 1st sem (without evaluations)
- Curricular units 2nd sem (evaluations)
- Curricular units 1st sem (credited)
- Curricular units 2nd sem (credited)
- GDP
- Previous qualification (grade)

The non-significant features were removed from the dataset, resulting in a refined feature set. The following represents the updated dataset to be used for our kernel-based SVM model:

	Admission grade	Age at enrollment	Curricular units 1st sem (enrolled)	Curricular units 1st sem (approved)	Curricular units 1st sem (grade)	Curricular units 2nd sem (enrolled)	Curricular units 2nd sem (approved)	Curricular units 2nd sem (grade)	Unemployment rate	Inflation rate	Target	y_labels
0	0.340000	0.056604	0.000000	0.000000	0.000000	0.00000	0.00	0.000000	0.372093	0.488889	Dropout	-1
1	0.500000	0.037736	0.230769	0.230769	0.741722	0.26087	0.30	0.735897	0.732558	0.111111	Graduate	1
2	0.313684	0.037736	0.230769	0.000000	0.000000	0.26087	0.00	0.000000	0.372093	0.488889	Dropout	-1
3	0.258947	0.056604	0.230769	0.230769	0.711447	0.26087	0.25	0.667692	0.209302	0.000000	Graduate	1
4	0.489474	0.528302	0.230769	0.192308	0.653422	0.26087	0.30	0.700000	0.732558	0.111111	Graduate	1

Figure 8: Updated dataset containing only the most significant features.

4 Kernel-based Support Vector Machine

4.1 Model Description and Rationale

To capture non-linear relationships that a soft-margin linear SVM cannot model, we solve the dual optimization problem:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{n} \alpha_i \; - \; \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \, \alpha_j \, y_i \, y_j \, (x_i \cdot x_j) \\ \text{subject to} \quad & \alpha_i \geq 0 \quad (i = 1, \ldots, n), \\ & \sum_{i=1}^{n} \alpha_i \, y_i = 0. \end{aligned}$$

Because this formulation depends only on the inner product $x_i \cdot x_j$, we can replace it with a kernel function $K(x_i, x_j)$. In our experiments, we use both the polynomial and Gaussian kernels. The regularization constant C still governs the trade-off between margin width and hinge-loss violations, and the dual variables α_i are found via stochastic gradient descent. This "kernel trick" lets us learn a non-linear decision boundary in the original feature space without ever explicitly computing a high-dimensional feature map.

We considered two kernels:

- **Polynomial Kernel**: $K_{\text{poly}}(x, x') = (\gamma \, (x \cdot x') + r)^d$
- **Gaussian Kernel**: $K_{\text{rbf}}(x, x') = \exp[-\gamma \|x - x'\|^2]$

4.2 Training Procedure

The dataset containing only statistically significant features was split into **80 % training** and **20 % test** sets using `train_test_split` from `scikit-learn`.

We implemented the training procedure for the kernel SVM using **stochastic gradient descent**. This method updates the dual variables α_i iteratively based on hinge-loss violations:

- At each iteration t, randomly select a data point (x_j, y_j),
- Compute the margin violation using the precomputed kernel Gram matrix,
- If the point violates the margin (i.e. $y_j f(x_j) < 1$), set $\beta_j \leftarrow \beta_j + y_j$,
- Compute $\alpha^{(t)} = \beta/(2Ct)$ and average across $T = 1000$ iterations to obtain the final dual solution $\bar{\alpha}$.

```python
def train_kernel_svm_sgd(X, y, kernel, T=1000, C=1.0, **kernel_params):
    """
    Dual-SGD for hinge-loss SVM using a precomputed Gram matrix.
    Returns the averaged alpha coefficients.
    """
    n = X.shape[0]
    # precompute Gram matrix
    K = kernel(X, X, **kernel_params)

    # betas accumulate raw counts; we'll convert to alphas each step
    beta = np.zeros(n)
    alpha_sum = np.zeros(n)

    for t in range(1, T + 1):
        # compute current alphas from beta
        alpha = beta / (2 * C * t)

        # uniformly pick one example at random
        j = np.random.randint(n)

        # decision function margin for j
        margin_j = y[j] * np.dot(alpha * y, K[:, j])

        # if margin_j < 1, we incur hinge loss → update beta_j
        if margin_j < 1:
            beta[j] += y[j]

        alpha_sum += alpha

    # return the averaged alpha over all iterations
    return alpha_sum / T
```

Figure 9: Implementation of the training procedure for dual-SGD SVM.

4.3 Tuning Hyperparameters

We conducted a **grid search** with five-fold cross validation over combinations of kernel hyperparameters—specifically, the polynomial degree d, kernel coefficient γ, offset r, and regularization constant C—using `ParameterGrid` from `scikit-learn`.

Kernel	**Grid**
Polynomial	$d \in \{2, 3, 4\},\ \gamma \in \{10^{-4}, \ldots, 1\},\ r \in \{0, 0.5, 1, 2\},\ C \in \{10^{-4}, \ldots, 10\}$
Gaussian	$\gamma \in \{10^{-5}, \ldots, 3\},\ C \in \{10^{-4}, \ldots, 10\}$

Table 2: Hyperparameter grid used for polynomial and RBF kernel SVMs.

This procedure yielded the following optimal configurations:

- **Polynomial kernel:**
  Best parameters {C = 0.001, degree = 3, $\gamma = 0.1$, $r = 0$},

- **Gaussian kernel:**
  Best parameters {C = 0.1, $\gamma = 3$},

4.4 Model Training and Testing Evaluation

We then refitted the full training set with the optimal parameters (Polynomial: $\{d = 3, \gamma = 0.01, r = 1, C = 0.3\}$; Gaussian: $\{\gamma = 0.03, C = 3\}$).

The test-set performance of our two kernel SVM models is summarized in Table 3. Four standard metrics are reported:

- **Accuracy**: fraction of all students whose outcome (dropout vs. non-dropout) was predicted correctly.
- **Log-Loss**: negative log-likelihood of the true labels under the model's predicted probabilities (lower is better).
- **Precision**: among students predicted to drop out, the fraction who actually did.
- **Recall**: among students who truly dropped out, the fraction correctly identified.

Kernel	**Accuracy**	**Log-Loss**	**Precision**	**Recall**
Polynomial	0.8147	6.6793	0.8073	0.9551
Gaussian	0.8215	6.4349	0.8160	0.9517

Table 3: Test-set metrics for the best polynomial and Gaussian kernel SVMs.

4.5 Ablation Check: Re-adding Dropped Ordinal Features

Purpose. We sought to verify that the earlier feature-selection step did not strip away any useful contributions to the classification prediction and so we refitted both kernel SVMs on the *entire* ordinal feature matrix. The hyperparameter grid search was repeated on this superset after which the models were re-fit and evaluated on the procedure.

Kernel	Accuracy	Log-Loss	Precision	Recall
Polynomial (all)	0.8158	6.6385	0.8102	0.9517
Gaussian (all)	0.8158	6.6385	0.8085	0.9551

Table 4: Performance after re-introducing all ordinal features.

Take-away. The performance is virtually identical to that in Table 3, confirming that the non-significant features did not contribute much to the classification problem. Retaining only the statistically significant features thus simplifies the model without sacrificing performance.

4.6 Final Results and Summary

Kernel-based SVM Interpretation. Both kernel SVM models demonstrate a solid predictive performance on the student dropout task, with the Gaussian kernel slightly outperforming the polynomial kernel overall. The polynomial SVM achieves 81.47% accuracy, 0.8073 precision, and 0.9551 recall, while the Gaussian SVM reaches 82.15% accuracy, 0.8160 precision, and 0.9517 recall. Both models maintain high recall—above 95%—indicating they successfully identify nearly all true dropouts, and their precision values around 80% show a reasonable balance between avoiding false alarms and catching at-risk students. Overall, these results suggest that kernelized SVMs can reliably flag students at risk of dropping out, with the Gaussian kernel offering a slight edge.

Comparison to linear model. Interestingly, the Gaussian kernel's 82.15% accuracy matches our soft-margin linear SVM baseline (82.0%), while the polynomial kernel is slightly below. This suggests that the relationship between the selected features and dropout risk is largely linear–contrary to our initial assumption–and that mapping into a higher-dimensional feature space yields only marginal gains.

Challenges and limitations.

- **Computational cost**: Each full grid search (polynomial plus Gaussian) required approximately 10-15 minutes of CPU time, limiting the size of the hyperparameter grids we could explore (which could greatly explain why we're not seeing larger improvements over the linear SVM).
- **Feature scope**. The ablation study proves that reintroducing the statistically insignificant ordinal features does not improve performance. However, the apparent linearity of the data possibly suggests that removing the categorical and nominal variables may have removed useful information. Future work should either develop effective encodings for those features (despite the challenges encoding has for distance-based algorithms) or to actually explore other models besides SVMs.

Key takeaways.

- The Gaussian kernel SVM achieved 82.15% accuracy and 0.8160 precision—0.68 pp higher precision than the polynomial kernel—demonstrating a modest gain from non-linear mapping.
- The linear SVM baseline matched kernel performance (82.10% vs. 82.15%), indicating that our 10 statistically significant ordinal features capture the bulk of the predictive signal.

- Reintroducing the 8 non-significant ordinal features (e.g. GDP, previous qualification, credited units) did not alter accuracy (0.8158%), confirming that the refined 10-feature set suffices.
- Semester grades dominate feature importance: "Curricular units 2nd sem (grade)" has the largest coefficient (1.9480), followed by "Curricular units 1st sem (grade)" (0.8073), whereas economic factors like inflation (0.0068) and GDP (–0.0267) contribute minimally.
- All SVM variants maintain recall above 95%, effectively flagging at-risk students, though false-positive rates of 15–17% suggest potential value in cost-sensitive threshold tuning.