36106-25AU-AT2-25589351-experiment-1

April 25, 2025

1 Experiment Notebook

1.1 0. Setup Environment

1.1.1 0.a Install Environment and Mandatory Packages

[1]:
```
# Do not modify this code
!pip install -q utstd

from utstd.folders import *
from utstd.ipyrenders import *

at = AtFolder(
    course_code=36106,
    assignment="AT2",
)
at.run()
```

```
                        0.0/1.6 MB
? eta -:--:--
                       1.1/1.6
MB 33.0 MB/s eta 0:00:01
                       1.6/1.6 MB
22.4 MB/s eta 0:00:01
                       1.6/1.6 MB 14.4
MB/s eta 0:00:00
Mounted at /content/gdrive

You can now save your data files in:
/content/gdrive/MyDrive/36106/assignment/AT2/data
```

1.1.2 0.b Disable Warnings Messages

[ ]:
```
# Do not modify this code
import warnings
warnings.simplefilter(action='ignore')
```

1.1.3 0.c Install Additional Packages

If you are using additional packages, you need to install them here using the command:
`! pip install <package_name>`

[ ]:
```
# <Student to fill this section>
```

1.1.4 0.d Import Packages

[2]:
```
# <Student to fill this section>
import pandas as pd
import altair as alt
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import recall_score, f1_score, classification_report,␣
 ↪confusion_matrix, ConfusionMatrixDisplay
```

1.2 A. Project Description

[3]:
```
# <Student to fill this section>
student_name = "Fatemeh Elyasifar"
student_id = "25589351"
```

[ ]:
```
# Do not modify this code
print_tile(size="h1", key='student_name', value=student_name)
```

```
<IPython.core.display.HTML object>
```

[ ]:
```
# Do not modify this code
print_tile(size="h1", key='student_id', value=student_id)
```

```
<IPython.core.display.HTML object>
```

[4]:
```
print("Student Name:", student_name)
print("Student ID:", student_id)
```

```
Student Name: Fatemeh Elyasifar
Student ID: 25589351
```

[5]:
```
# <Student to fill this section>
business_objective = """
Explain clearly what is the goal of this project for the business. How will the␣
 ↪results be used? What will be the impact of accurate or incorrect results?

The objective is to develop a reliable and interpretable machine learning model␣
 ↪to predict student performance at the end of the semester
using academic, behavioral, and demographic data. The model aims to help␣
 ↪university staff, student support officers, and academic advisors identify␣
 ↪at-risk students (those with poor or average performance),
with a target F1-score and recall above 80%, enabling timely outreach and␣
 ↪efficient allocation of support services.
The focus is on improving recall for underperforming students while maintaining␣
 ↪balanced performance.
The project also identifies key predictors and includes feature tuning to␣
 ↪enhance accuracy and relevance.

Accurate predictions help the university support underperforming students,␣
 ↪improve outcomes, and reduce dropout rates.
Inaccurate results risk missing students in need or misusing limited resources.
Therefore, maintaining a strong balance between precision and recall is␣
 ↪essential to ensure the model is both effective and trustworthy.
High recall is especially important to capture as many at-risk students as␣
 ↪possible.
"""
```

[ ]:
```
# Do not modify this code
print_tile(size="h3", key='business_objective', value=business_objective)
```

```
<IPython.core.display.HTML object>
```

[6]:
```
print("Business Objective:", business_objective)
```

```
Business Objective:
Explain clearly what is the goal of this project for the business. How will the
results be used? What will be the impact of accurate or incorrect results?

The objective is to develop a reliable and interpretable machine learning model
to predict student performance at the end of the semester
using academic, behavioral, and demographic data. The model aims to help
university staff, student support officers, and academic advisors identify at-
risk students (those with poor or average performance),
with a target F1-score and recall above 80%, enabling timely outreach and
efficient allocation of support services.
The focus is on improving recall for underperforming students while maintaining
balanced performance.
The project also identifies key predictors and includes feature tuning to
```

enhance accuracy and relevance.

Accurate predictions help the university support underperforming students,
improve outcomes, and reduce dropout rates.
Inaccurate results risk missing students in need or misusing limited resources.
Therefore, maintaining a strong balance between precision and recall is
essential to ensure the model is both effective and trustworthy.
High recall is especially important to capture as many at-risk students as
possible.

1.3 B. Experiment Description

[8]:
```
# Do not modify this code
experiment_id = "1"
print_tile(size="h1", key='experiment_id', value=experiment_id)
```

<IPython.core.display.HTML object>

[9]:
```
print("Experiment ID:", experiment_id)
```

Experiment ID: 1

[11]:
```
# <Student to fill this section>
experiment_hypothesis = """
Present the hypothesis you want to test, the question you want to answer or the␣
 ↪insight you are seeking.
Explain the reasons why you think it is worthwhile considering it

Student performance can be effectively predicted using a Support Vector␣
 ↪Classifier (SVC)
trained on a dataset comprising behavioral, academic, and demographic features.
SVC, a parametric model, is expected to capture complex patterns in student␣
 ↪data and distinguish between different performance categories
due to its ability to model non-linear relationships and its sensitivity to␣
 ↪feature scaling.
It serves as the first trained model to benchmark predictive performance before␣
 ↪introducing ensemble methods or advanced feature engineering.
"""
```

[ ]:
```
# Do not modify this code
print_tile(size="h3", key='experiment_hypothesis', value=experiment_hypothesis)
```

<IPython.core.display.HTML object>

[12]:
```
print("Experiment Hypothesis:", experiment_hypothesis)
```

Experiment Hypothesis:
Present the hypothesis you want to test, the question you want to answer or the insight you are seeking.
Explain the reasons why you think it is worthwhile considering it

Student performance can be effectively predicted using a Support Vector Classifier (SVC)
trained on a dataset comprising behavioral, academic, and demographic features.
SVC, a parametric model, is expected to capture complex patterns in student data and distinguish between different performance categories
due to its ability to model non-linear relationships and its sensitivity to feature scaling.
It serves as the first trained model to benchmark predictive performance before introducing ensemble methods or advanced feature engineering.

[13]:
```
# <Student to fill this section>
experiment_expectations = """
Detail what will be the expected outcome of the experiment. If possible, estimate the goal you are expecting.
List the possible scenarios resulting from this experiment.

The SVC model is expected to perform better than the baseline DummyClassifier, with an anticipated recall score of around 60 and a weighted F1 score of approximately 65.
Multiple models will be trained using different combinations of C, kernel, and gamma hyperparameters to evaluate their impact on performance.
The experiment will compare weighted F1 scores across these configurations to identify the most effective setup.
Based on the results, recommendations may be made for potential improvements in data preprocessing, such as alternative scaling methods or additional feature transformations,
to enhance model performance in future experiments.

Possible scenarios include:
- SVC significantly outperforms the baseline, indicating that the current features and preprocessing are effective.
- SVC shows only minimal enhancement, suggesting the need for further feature engineering or model tuning.
- SVC underperforms, indicating possible issues with class imbalance, feature noise, or the need for a different modeling approach.
"""
```

[ ]:
```
# Do not modify this code
print_tile(size="h3", key='experiment_expectations', value=experiment_expectations)
```

```
<IPython.core.display.HTML object>
```

[14]:
```
print("Experiment Expectations:", experiment_expectations)
```

```
Experiment Expectations:
Detail what will be the expected outcome of the experiment. If possible,
estimate the goal you are expecting.
List the possible scenarios resulting from this experiment.

The SVC model is expected to perform better than the baseline DummyClassifier,
with an anticipated recall score of around 60 and a weighted F1 score of
approximately 65.
Multiple models will be trained using different combinations of C, kernel, and
gamma hyperparameters to evaluate their impact on performance.
The experiment will compare weighted F1 scores across these configurations to
identify the most effective setup.
Based on the results, recommendations may be made for potential improvements in
data preprocessing, such as alternative scaling methods or additional feature
transformations,
to enhance model performance in future experiments.

Possible scenarios include:
- SVC significantly outperforms the baseline, indicating that the current
features and preprocessing are effective.
- SVC shows only minimal enhancement, suggesting the need for further feature
engineering or model tuning.
- SVC underperforms, indicating possible issues with class imbalance, feature
noise, or the need for a different modeling approach.
```

1.4 C. Data Understanding

[ ]:
```
# Do not modify this code
try:
  X_train = pd.read_csv(at.folder_path / 'X_train.csv')
  y_train = pd.read_csv(at.folder_path / 'y_train.csv')

  X_val = pd.read_csv(at.folder_path / 'X_val.csv')
  y_val = pd.read_csv(at.folder_path / 'y_val.csv')

  X_test = pd.read_csv(at.folder_path / 'X_test.csv')
  y_test = pd.read_csv(at.folder_path / 'y_test.csv')
except Exception as e:
  print(e)
```

1.5 D. Feature Selection

[ ]:
```
# <Student to fill this section>

features_list = X_train.columns.tolist()
```

[15]:
```
# <Student to fill this section>
feature_selection_explanations = """
Provide a rationale on why you are selected these features but also why you␣
 ↪decided to remove other ones

These features provide a comprehensive view of each student's academic,␣
 ↪behavioral, and demographic profile,
and were chosen to capture the key factors that influence performance outcomes.
They are expected to improve the accuracy of student performance predictions.

Features with high cardinality or weak correlation with the target were removed,
to reduce noise and improve model efficiency without sacrificing predictive␣
 ↪value.
"""
```

[ ]:
```
# Do not modify this code
print_tile(size="h3", key='feature_selection_explanations',␣
 ↪value=feature_selection_explanations)
```

```
<IPython.core.display.HTML object>
```

[17]:
```
print("Feature Selection Explanations:", feature_selection_explanations)
```

```
Feature Selection Explanations:
Provide a rationale on why you are selected these features but also why you
decided to remove other ones

These features provide a comprehensive view of each student's academic,
behavioral, and demographic profile,
and were chosen to capture the key factors that influence performance outcomes.
They are expected to improve the accuracy of student performance predictions.

Features with high cardinality or weak correlation with the target were removed,
to reduce noise and improve model efficiency without sacrificing predictive
value.
```

1.6 G. Train Machine Learning Model

1.6.1 G.1 Import Algorithm

[ ]:
```
# <Student to fill this section>
from sklearn.svm import SVC
```

[18]:
```
# <Student to fill this section>
algorithm_selection_explanations = """
Provide some explanations on why you believe this algorithm is a good fit

Support Vector Classifier (SVC) is a good choice for this experiment because it␣
 ↪can effectively model non-linear relationships using kernel functions.
It works well when the data is scaled and is able to separate classes that are␣
 ↪not linearly distributed.

SVC also has useful settings like C, kernel, and gamma that allow us to␣
 ↪fine-tune the model for better results.
This makes it a solid first algorithm to try before testing more complex models␣
 ↪like ensembles.
"""
```

[ ]:
```
# Do not modify this code
print_tile(size="h3", key='algorithm_selection_explanations',␣
 ↪value=algorithm_selection_explanations)
```

```
<IPython.core.display.HTML object>
```

[19]:
```
print("Algorithm Selection Explanations:", algorithm_selection_explanations)
```

```
Algorithm Selection Explanations:
Provide some explanations on why you believe this algorithm is a good fit

Support Vector Classifier (SVC) is a good choice for this experiment because it
can effectively model non-linear relationships using kernel functions.
It works well when the data is scaled and is able to separate classes that are
not linearly distributed.

SVC also has useful settings like C, kernel, and gamma that allow us to fine-
tune the model for better results.
This makes it a solid first algorithm to try before testing more complex models
like ensembles.
```

1.6.2 G.2 Set Hyperparameters

[ ]:
```
param_grid = {
    'C': [0.1, 5, 1, 10],
    'kernel': ['linear','sigmoid', 'rbf'],
    'gamma': ['scale', 'auto'],
    'class_weight': ['balanced']
}
```

[20]:
```
# <Student to fill this section>
hyperparameters_selection_explanations = """
Explain why you are tuning these hyperparameters

These hyperparameters are tuned to find the best combination for improving␣
 ↪model performance.
'C' controls the regularization strength, 'kernel' changes how the decision␣
 ↪boundary is shaped,
and 'gamma' affects how far the influence of a training point reaches.
'class_weight' is set to 'balanced' to help the model handle class imbalance in␣
 ↪the target variable.
Tuning these values helps the SVC model generalize better to unseen data.
"""
```

[ ]:
```
# Do not modify this code
print_tile(size="h3", key='hyperparameters_selection_explanations',␣
 ↪value=hyperparameters_selection_explanations)
```

```
<IPython.core.display.HTML object>
```

[21]:
```
print("Hyperparameters Selection Explanations:",␣
 ↪hyperparameters_selection_explanations)
```

```
Hyperparameters Selection Explanations:
Explain why you are tuning these hyperparameters

These hyperparameters are tuned to find the best combination for improving model
performance.
'C' controls the regularization strength, 'kernel' changes how the decision
boundary is shaped,
and 'gamma' affects how far the influence of a training point reaches.
'class_weight' is set to 'balanced' to help the model handle class imbalance in
the target variable.
Tuning these values helps the SVC model generalize better to unseen data.
```

1.6.3 G.3 Fit Model

[ ]:
```
# <Student to fill this section>
svc = SVC(random_state=42)
grid_search_svc = GridSearchCV(svc, param_grid, cv=5, scoring='f1_weighted')
grid_search_svc.fit(X_train, y_train)

print("Best Params:", grid_search_svc.best_params_)
print("Best Weighted F1 Score:", round(grid_search_svc.best_score_, 4))

results_df = pd.DataFrame(grid_search_svc.cv_results_)
results_df = results_df[[
    'mean_test_score', 'std_test_score', 'params'
]]
results_df = results_df.sort_values(by='mean_test_score', ascending=False).
 ↪reset_index(drop=True)

# Display full params in the results DataFrame
pd.set_option('display.max_colwidth', None)

results_df
```

```
Best Params: {'C': 0.1, 'class_weight': 'balanced', 'gamma': 'scale', 'kernel':
'linear'}
Best Weighted F1 Score: 0.7268
```

[ ]:

	mean_test_score	std_test_score	\
0	0.726800	0.026077	
1	0.726800	0.026077	
2	0.719979	0.044072	
3	0.718157	0.042938	
4	0.715469	0.049569	
5	0.704867	0.039580	
6	0.703629	0.024703	
7	0.703629	0.024703	
8	0.702974	0.028509	
9	0.702974	0.028509	
10	0.700942	0.045177	
11	0.700446	0.028874	
12	0.700446	0.028874	
13	0.697475	0.037176	
14	0.693304	0.029897	
15	0.689202	0.037271	
16	0.689080	0.029356	
17	0.688381	0.030233	
18	0.669634	0.029099	
19	0.669226	0.028734	
20	0.619943	0.046761	

```
21         0.593614        0.045036
22         0.584735        0.070282
23         0.574475        0.068487

params
0    {'C': 0.1, 'class_weight': 'balanced', 'gamma': 'scale', 'kernel':
'linear'}
1     {'C': 0.1, 'class_weight': 'balanced', 'gamma': 'auto', 'kernel':
'linear'}
2     {'C': 10, 'class_weight': 'balanced', 'gamma': 'auto', 'kernel':
'sigmoid'}
3     {'C': 5, 'class_weight': 'balanced', 'gamma': 'scale', 'kernel':
'sigmoid'}
4      {'C': 5, 'class_weight': 'balanced', 'gamma': 'auto', 'kernel':
'sigmoid'}
5    {'C': 10, 'class_weight': 'balanced', 'gamma': 'scale', 'kernel':
'sigmoid'}
6       {'C': 5, 'class_weight': 'balanced', 'gamma': 'auto', 'kernel':
'linear'}
7      {'C': 5, 'class_weight': 'balanced', 'gamma': 'scale', 'kernel':
'linear'}
8     {'C': 10, 'class_weight': 'balanced', 'gamma': 'scale', 'kernel':
'linear'}
9      {'C': 10, 'class_weight': 'balanced', 'gamma': 'auto', 'kernel':
'linear'}
10    {'C': 1, 'class_weight': 'balanced', 'gamma': 'scale', 'kernel':
'sigmoid'}
11     {'C': 1, 'class_weight': 'balanced', 'gamma': 'scale', 'kernel':
'linear'}
12      {'C': 1, 'class_weight': 'balanced', 'gamma': 'auto', 'kernel':
'linear'}
13     {'C': 1, 'class_weight': 'balanced', 'gamma': 'auto', 'kernel':
'sigmoid'}
14        {'C': 10, 'class_weight': 'balanced', 'gamma': 'auto', 'kernel':
'rbf'}
15        {'C': 5, 'class_weight': 'balanced', 'gamma': 'scale', 'kernel':
'rbf'}
16       {'C': 10, 'class_weight': 'balanced', 'gamma': 'scale', 'kernel':
'rbf'}
17         {'C': 5, 'class_weight': 'balanced', 'gamma': 'auto', 'kernel':
'rbf'}
18        {'C': 1, 'class_weight': 'balanced', 'gamma': 'scale', 'kernel':
'rbf'}
19         {'C': 1, 'class_weight': 'balanced', 'gamma': 'auto', 'kernel':
'rbf'}
20  {'C': 0.1, 'class_weight': 'balanced', 'gamma': 'scale', 'kernel':
'sigmoid'}
```

```
21   {'C': 0.1, 'class_weight': 'balanced', 'gamma': 'auto', 'kernel':
'sigmoid'}
22      {'C': 0.1, 'class_weight': 'balanced', 'gamma': 'scale', 'kernel':
'rbf'}
23       {'C': 0.1, 'class_weight': 'balanced', 'gamma': 'auto', 'kernel':
'rbf'}
```

[ ]:
```
svc_model = SVC(C= 0.1, class_weight= 'balanced', gamma= 'scale', kernel=␣
 ↪'linear', random_state=42)
svc_model.fit(X_train, y_train)

cv_scores = cross_val_score(svc_model, X_train, y_train, cv=5,␣
 ↪scoring='f1_weighted')
print("Cross-Validated Weighted F1 Scores (Train Set):", cv_scores)
print("Mean CV Weighted F1 Score:", cv_scores.mean())
```

```
Cross-Validated Weighted F1 Scores (Train Set): [0.71364555 0.68891871
0.75046186 0.71978049 0.76119368]
Mean CV Weighted F1 Score: 0.7268000582466871
```

1.6.4 G.4 Model Technical Performance

[ ]:
```
# <Student to fill this section>
y_preds = svc_model.predict(X_val)

print("SVC Evaluation:\n")
print("\nRecall_score:", recall_score(y_val, y_preds, average='weighted'))
print("\nf1_score:", f1_score(y_val, y_preds, average='weighted'))
print("\nClassification Report:\n", classification_report(y_val, y_preds))
print("\nConfusion Matrix:\n", confusion_matrix(y_val, y_preds))
```

```
SVC Evaluation:


Recall_score: 0.7227722772277227

f1_score: 0.7285000825464085

Classification Report:
               precision    recall  f1-score   support

           0       0.92      0.85      0.88       103
           1       0.63      0.53      0.57        51
           2       0.51      0.69      0.59        35
           3       0.44      0.54      0.48        13

    accuracy                           0.72       202
   macro avg       0.62      0.65      0.63       202
```

```
weighted avg       0.74      0.72      0.73       202


Confusion Matrix:
 [[88  9  5  1]
 [ 8 27 13  3]
 [ 0  6 24  5]
 [ 0  1  5  7]]
```

[22]:
```
# <Student to fill this section>
model_performance_explanations = """
Provide some explanations on model performance

The SVC model achieved a weighted F1 score of 0.73 during cross-validation and␣
 ↪maintained similar performance on the validation set, with a weighted F1␣
 ↪score of 0.73 and a recall of 0.72.
These results indicate consistent generalization across training and validation␣
 ↪data.

GridSearchCV was used to efficiently find the optimal hyperparameters, as␣
 ↪manually tuning them would be time-consuming.
The best parameters were selected based on the highest F1 score to maximize␣
 ↪model performance.

Recall is a key metric for this task, as the goal is to identify students who␣
 ↪are at risk of poor performance and may require intervention.
The model shows strong recall for the 'Poor' performance class (label 0),␣
 ↪achieving a recall of 0.85, which is the most critical group to identify.
However, since the target classes are imbalanced, the weighted recall score of␣
 ↪0.72 provides a more balanced view of overall model performance across all␣
 ↪categories.
F1 score is also considered as the primary performance metric due to class␣
 ↪imbalance, helping balance both precision and recall in evaluation.

Based on the confusion matrix, out of 103 students in the 'Poor' category, 88␣
 ↪were correctly identified, which aligns with the high recall of 0.85 for␣
 ↪this class.
Most misclassifications occurred between neighbouring performance levels such␣
 ↪as 'Average' and 'Good', which is expected given the similarities between␣
 ↪those categories.
This confirms that the model can not be reliable in flagging the␣
 ↪lowest-performing students, which is critical for early intervention.

Overall, the model demonstrates moderate predictive ability and stability,␣
 ↪making it suitable as a first trained model for identifying at-risk students.
"""
```

[ ]:
```
# Do not modify this code
print_tile(size="h3", key='model_performance_explanations',␣
↪value=model_performance_explanations)
```

```
<IPython.core.display.HTML object>
```

[23]:
```
print("Model Performance Explanations:", model_performance_explanations)
```

```
Model Performance Explanations:
Provide some explanations on model performance

The SVC model achieved a weighted F1 score of 0.73 during cross-validation and
maintained similar performance on the validation set, with a weighted F1 score
of 0.73 and a recall of 0.72.
These results indicate consistent generalization across training and validation
data.

GridSearchCV was used to efficiently find the optimal hyperparameters, as
manually tuning them would be time-consuming.
The best parameters were selected based on the highest F1 score to maximize
model performance.

Recall is a key metric for this task, as the goal is to identify students who
are at risk of poor performance and may require intervention.
The model shows strong recall for the 'Poor' performance class (label 0),
achieving a recall of 0.85, which is the most critical group to identify.
However, since the target classes are imbalanced, the weighted recall score of
0.72 provides a more balanced view of overall model performance across all
categories.
F1 score is also considered as the primary performance metric due to class
imbalance, helping balance both precision and recall in evaluation.

Based on the confusion matrix, out of 103 students in the 'Poor' category, 88
were correctly identified, which aligns with the high recall of 0.85 for this
class.
Most misclassifications occurred between neighbouring performance levels such as
'Average' and 'Good', which is expected given the similarities between those
categories.
This confirms that the model can not be reliable in flagging the lowest-
performing students, which is critical for early intervention.

Overall, the model demonstrates moderate predictive ability and stability,
making it suitable as a first trained model for identifying at-risk students.
```

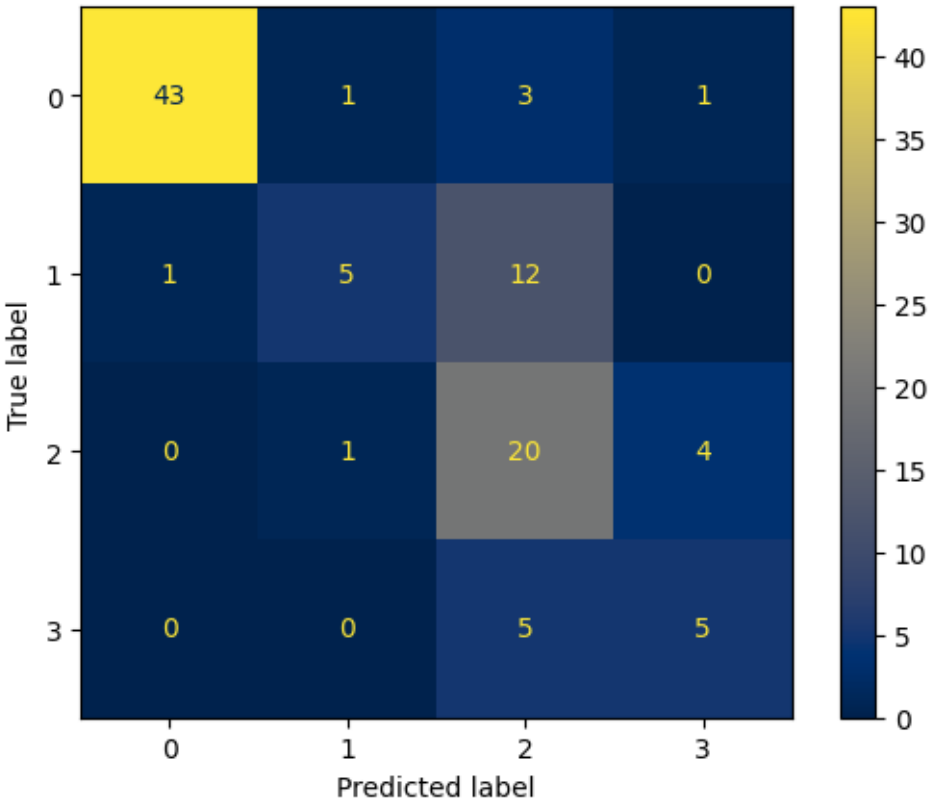
1.6.5 G.5 Business Impact from Current Model Performance

[ ]:
```
# <Student to fill this section>

y_pred_final = svc_model.predict(X_test)

print("SVC Evaluation:\n")
print("\nRecall_score:", recall_score(y_test, y_pred_final, average='weighted'))
print("\nf1_score:", f1_score(y_test, y_pred_final, average='weighted'))
print("\nClassification Report:\n", classification_report(y_test, y_pred_final))
```

```
SVC Evaluation:


Recall_score: 0.7227722772277227

f1_score: 0.7173681247723435

Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.90      0.93        48
           1       0.71      0.28      0.40        18
           2       0.50      0.80      0.62        25
           3       0.50      0.50      0.50        10

    accuracy                           0.72       101
   macro avg       0.67      0.62      0.61       101
weighted avg       0.77      0.72      0.72       101
```

[ ]:
```
ConfusionMatrixDisplay.from_estimator(svc_model, X_test, y_test, cmap='cividis')
```

[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7ce0ad21da90>
```



[24]:
```
# <Student to fill this section>
business_impacts_explanations = """
Interpret the results of the experiments related to the business objective set␣
 ↪earlier. Estimate the impacts of the incorrect results for the business␣
 ↪(some results may have more impact compared to others)

The SVC model performed well on unseen data, with a weighted F1 score of 0.72␣
 ↪and a recall of 0.72.
It correctly identified 90% of students in the 'Poor' category, which is the␣
 ↪most important group to detect for intervention.

However, the model struggled with the 'Average' group, with a recall of only 0.
 ↪28.
These students may also need support, and misclassifying them could lead to␣
 ↪missed opportunities for early help.

Based on the confusion matrix, the model most often confused 'Average' students␣
 ↪with 'Good'.
Misclassifying 'Average' students as 'Good' may delay support, reducing the␣
 ↪impact of early intervention and risking further performance decline.
```

```
To improve business impact, future model enhancements should focus on improving␣
 ↪classification accuracy for the 'Average' category,
as these students often sit on the edge of needing support and are more likely␣
 ↪to benefit from early intervention.
"""
```

[ ]:
```
# Do not modify this code
print_tile(size="h3", key='business_impacts_explanations',␣
 ↪value=business_impacts_explanations)
```

```
<IPython.core.display.HTML object>
```

[25]:
```
print("Business Impacts Explanations:", business_impacts_explanations)
```

```
Business Impacts Explanations:
Interpret the results of the experiments related to the business objective set
earlier. Estimate the impacts of the incorrect results for the business (some
results may have more impact compared to others)

The SVC model performed well on unseen data, with a weighted F1 score of 0.72
and a recall of 0.72.
It correctly identified 90% of students in the 'Poor' category, which is the
most important group to detect for intervention.

However, the model struggled with the 'Average' group, with a recall of only
0.28.
These students may also need support, and misclassifying them could lead to
missed opportunities for early help.

Based on the confusion matrix, the model most often confused 'Average' students
with 'Good'.
Misclassifying 'Average' students as 'Good' may delay support, reducing the
impact of early intervention and risking further performance decline.
To improve business impact, future model enhancements should focus on improving
classification accuracy for the 'Average' category,
as these students often sit on the edge of needing support and are more likely
to benefit from early intervention.
```

1.7 H. Experiment Outcomes

[27]:
```
# <Student to fill this section>
experiment_outcome = "Hypothesis Partially Confirmed" # Either 'Hypothesis␣
 ↪Confirmed', 'Hypothesis Partially Confirmed' or 'Hypothesis Rejected'
```

[ ]:
```
# Do not modify this code
```

```
print_tile(size="h2", key='experiment_outcomes_explanations',␣
 ↪value=experiment_outcome)
```

```
<IPython.core.display.HTML object>
```

[28]:
```
print("Experiment Outcome:", experiment_outcome)
```

```
Experiment Outcome: Hypothesis Partially Confirmed
```

[29]:
```
# <Student to fill this section>
experiment_results_explanations = """
Reflect on the outcome of the experiment and list the new insights you gained␣
 ↪from it. Provide rationale for pursuing more experimentation with the␣
 ↪current approach or call out if you think it is a dead end.
Given the results achieved and the overall objective of the project, list the␣
 ↪potential next steps and experiments. For each of them assess the expected␣
 ↪uplift or gains and rank them accordingly. If the experiment achieved the␣
 ↪required outcome for the business, recommend the steps to deploy this␣
 ↪solution into production.

The best combination of hyperparameters is C = 0.1, class_weight = 'balanced',␣
 ↪gamma = 'scale', kernel = 'linear'.
The SVC model confirmed the initial hypothesis by achieving strong overall␣
 ↪performance, performing better than the baseline DummyClassifier, and␣
 ↪identifying the majority of at-risk students,
with a weighted recall of 0.72 for the 'Poor' class and a weighted F1 score of␣
 ↪0.72. However, the model struggled to clearly separate some categories,
particularly confusing 'Average' students with 'Good'.

This suggests there is still room for improvement, especially in class-level␣
 ↪recall. While the current approach is promising,
further experimentation is needed to improve class separation and ensure more␣
 ↪balanced performance.

Potential next steps:

1. **Experiment with non-parametric models (e.g., Decision Tree)**
   - Expected gain: Moderate to high (improved handling of complex patterns and␣
 ↪class imbalance)
   - Priority: High

Although the scores are relatively high, there are some limitations that need␣
 ↪to be addressed before considering deployment.
These include a low number of training samples (706) and a dataset that may no␣
 ↪longer reflect the most up-to-date student behaviors or academic patterns.
```

```
Further testing with a larger and more current dataset is recommended to␣
 ↪validate the model's performance and reliability before real-world␣
 ↪implementation.
"""
```

[ ]:
```
# Do not modify this code
print_tile(size="h2", key='experiment_results_explanations',␣
 ↪value=experiment_results_explanations)
```

```
<IPython.core.display.HTML object>
```

[30]:
```
print("Experiment Results Explanations:", experiment_results_explanations)
```

```
Experiment Results Explanations:
Reflect on the outcome of the experiment and list the new insights you gained
from it. Provide rationale for pursuing more experimentation with the current
approach or call out if you think it is a dead end.
Given the results achieved and the overall objective of the project, list the
potential next steps and experiments. For each of them assess the expected
uplift or gains and rank them accordingly. If the experiment achieved the
required outcome for the business, recommend the steps to deploy this solution
into production.

The best combination of hyperparameters is C = 0.1, class_weight = 'balanced',
gamma = 'scale', kernel = 'linear'.
The SVC model confirmed the initial hypothesis by achieving strong overall
performance, performing better than the baseline DummyClassifier, and
identifying the majority of at-risk students,
with a weighted recall of 0.72 for the 'Poor' class and a weighted F1 score of
0.72. However, the model struggled to clearly separate some categories,
particularly confusing 'Average' students with 'Good'.

This suggests there is still room for improvement, especially in class-level
recall. While the current approach is promising,
further experimentation is needed to improve class separation and ensure more
balanced performance.

Potential next steps:

1. **Experiment with non-parametric models (e.g., Decision Tree)**
   - Expected gain: Moderate to high (improved handling of complex patterns and
class imbalance)
   - Priority: High

Although the scores are relatively high, there are some limitations that need to
be addressed before considering deployment.
These include a low number of training samples (706) and a dataset that may no
```

longer reflect the most up-to-date student behaviors or academic patterns. Further testing with a larger and more current dataset is recommended to validate the model's performance and reliability before real-world implementation.