36106-25AU-AT2-25589351-experiment-2

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 -:--:--
                       0.4/1.6
MB 12.4 MB/s eta 0:00:01
                       1.6/1.6 MB
23.5 MB/s eta 0:00:01
                       1.6/1.6 MB
23.5 MB/s eta 0:00:01
                       1.6/1.6 MB 13.2
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.metrics import recall_score, f1_score, classification_report,␣
↪confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import GridSearchCV
```

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 = """

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:

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

[7]:
```
# Do not modify this code
experiment_id = "2"
print_tile(size="h1", key='experiment_id', value=experiment_id)
```

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

[8]:
```
print("Experiment ID:", experiment_id)
```

```
Experiment ID: 2
```

[9]:
```
# <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 predicted using a Decision Tree classifier trained␣
 ↪on behavioral, academic, and demographic features.
This experiment aims to test whether a tree-based model can provide␣
 ↪interpretable decision paths while maintaining competitive performance.

The hypothesis is that DecisionTreeClassifier can produce meaningful␣
 ↪predictions with a simpler model structure and higher interpretability.
It will perform better than SVC, as it is less sensitive to irrelevant features␣
 ↪and does not rely on a smaller, highly optimized feature set.
This makes it a strong candidate for supporting early identification of at-risk␣
 ↪students while offering better transparency.
"""
```

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

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

[10]:
```
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 predicted using a Decision Tree classifier trained on behavioral, academic, and demographic features.
This experiment aims to test whether a tree-based model can provide interpretable decision paths while maintaining competitive performance.

The hypothesis is that DecisionTreeClassifier can produce meaningful predictions with a simpler model structure and higher interpretability.
It will perform better than SVC, as it is less sensitive to irrelevant features and does not rely on a smaller, highly optimized feature set.
This makes it a strong candidate for supporting early identification of at-risk students while offering better transparency.

[11]:
```
# <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 Decision Tree Classifier is expected to outperform both the baseline and␣
 ↪SVC models, with a weighted F1 score around 80 and recall near 85.
Its ability to handle the full feature set and ignore irrelevant features may␣
 ↪lead to better overall performance.

Different hyperparameter settings (e.g., max_depth, min_samples_split,␣
 ↪criterion) will be tested to optimize recall,
especially for the 'Poor' and 'average' category. The model's interpretability␣
 ↪is also a key benefit for practical use.

Possible scenarios include:
- Strong improvement over SVC and baseline, showing good feature interaction␣
 ↪and accuracy.
- Similar performance to SVC but with better interpretability.
- Weaker results due to overfitting or imbalance, indicating a need for␣
 ↪ensemble methods like Random Forest or Extra Trees.
"""
```

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

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

[12]:
```
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 Decision Tree Classifier is expected to outperform both the baseline and SVC
models, with a weighted F1 score around 80 and recall near 85.
Its ability to handle the full feature set and ignore irrelevant features may
lead to better overall performance.

Different hyperparameter settings (e.g., max_depth, min_samples_split,
criterion) will be tested to optimize recall,
especially for the 'Poor' and 'average' category. The model's interpretability
is also a key benefit for practical use.

Possible scenarios include:
- Strong improvement over SVC and baseline, showing good feature interaction and
accuracy.
- Similar performance to SVC but with better interpretability.
- Weaker results due to overfitting or imbalance, indicating a need for ensemble
methods like Random Forest or Extra Trees.
```

1.4 C. Data Understanding

[ ]:
```
# Do not modify this code
# Load training data
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()
```

[13]:
```
# <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 attributes offer a detailed perspective on the student's academic␣
 ↪conditions, behaviors, and personal context,
and were selected to highlight the most relevant factors affecting student␣
 ↪performance.
The selected features are well-suited for training a Decision Tree model, as it␣
 ↪can handle a large number of variables
and identify meaningful patterns.
Less relevant features were removed in Experiment 0 based on their low impact␣
 ↪on the target variable.
"""
```

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

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

[14]:
```
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 attributes offer a detailed perspective on the student's academic
conditions, behaviors, and personal context,
and were selected to highlight the most relevant factors affecting student
performance.
The selected features are well-suited for training a Decision Tree model, as it
can handle a large number of variables
and identify meaningful patterns.
Less relevant features were removed in Experiment 0 based on their low impact on
the target variable.
```

1.6 G. Train Machine Learning Model

1.6.1 G.1 Import Algorithm

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

[15]:
```
# <Student to fill this section>
algorithm_selection_explanations = """
Provide some explanations on why you believe this algorithm is a good fit

Decision Tree Classifier is a good fit for this experiment because it can model␣
 ↪complex, non-linear relationships
and handle a wide range of numerical features effectively. It is robust to␣
 ↪irrelevant or less important features
and can naturally capture interactions between variables.

Another key advantage is its high interpretability, which allows for clear␣
 ↪explanations of prediction outcomes,
an important factor when identifying at-risk students in an educational setting.
Its ability to handle a large feature set also makes it well-suited for this␣
 ↪dataset.
"""
```

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

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

[16]:
```
print("Algorithm Selection Explanations:", algorithm_selection_explanations)
```

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

Decision Tree Classifier is a good fit for this experiment because it can model
complex, non-linear relationships
and handle a wide range of numerical features effectively. It is robust to
irrelevant or less important features
and can naturally capture interactions between variables.

Another key advantage is its high interpretability, which allows for clear
explanations of prediction outcomes,
an important factor when identifying at-risk students in an educational setting.
Its ability to handle a large feature set also makes it well-suited for this
dataset.
```

1.6.2 G.2 Set Hyperparameters

[ ]:
```
# <Student to fill this section>
param_grid = {
    'max_depth': [5, 10, 15, 20],
    'min_samples_split': [2, 5, 10, 20],
    'criterion': ['gini', 'entropy'],
    'class_weight': ['balanced']
}
```

[17]:
```
# <Student to fill this section>
hyperparameters_selection_explanations = """
Explain why you are tuning these hyperparameters

These hyperparameters are tuned to control the complexity and performance of␣
 ↪the Decision Tree.
'max_depth' limits how deep the tree can grow to avoid overfitting, while␣
 ↪'min_samples_split' controls the minimum number of samples required to split␣
 ↪a node.
'criterion' determines how the quality of a split is measured ('gini' or␣
 ↪'entropy'),
and 'class_weight' is set to 'balanced' to handle class imbalance in the target␣
 ↪variable.
Tuning these values helps improve the model's generalization and recall,␣
 ↪especially for the less frequent classes.
"""
```

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

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

[18]:
```
print("Hyperparameters Selection Explanations:",␣
 ↪hyperparameters_selection_explanations)
```

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

These hyperparameters are tuned to control the complexity and performance of the
Decision Tree.
'max_depth' limits how deep the tree can grow to avoid overfitting, while
'min_samples_split' controls the minimum number of samples required to split a
node.
'criterion' determines how the quality of a split is measured ('gini' or
'entropy'),
and 'class_weight' is set to 'balanced' to handle class imbalance in the target
variable.
```

Tuning these values helps improve the model's generalization and recall, especially for the less frequent classes.

1.6.3 G.3 Fit Model

[ ]:
```
# <Student to fill this section>
dt_model = DecisionTreeClassifier(random_state=42)

# Grid Search
grid_search_dt = GridSearchCV(dt_model, param_grid, cv=5, scoring='f1_weighted')
grid_search_dt.fit(X_train, y_train)

print("Best parameters:", grid_search_dt.best_params_)
print("Best Weighted F1 Score:", round(grid_search_dt.best_score_, 4))

results_df = pd.DataFrame(grid_search_dt.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 parameters: {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth':
10, 'min_samples_split': 2}
Best Weighted F1 Score: 0.953
```

[ ]:

	mean_test_score	std_test_score	\
0	0.953042	0.025906	
1	0.953042	0.025906	
2	0.953042	0.025906	
3	0.950629	0.026646	
4	0.950629	0.026646	
5	0.950629	0.026646	
6	0.950448	0.027379	
7	0.950448	0.027379	
8	0.950448	0.027379	
9	0.947829	0.023447	
10	0.947829	0.023447	
11	0.947829	0.023447	
12	0.947689	0.030166	
13	0.947689	0.030166	
14	0.947689	0.030166	

```
15         0.946206        0.029152
16         0.946206        0.029152
17         0.946206        0.029152
18         0.939043        0.030069
19         0.939043        0.030069
20         0.939043        0.030069
21         0.923299        0.032232
22         0.922274        0.031130
23         0.922274        0.031130
24         0.922274        0.031130
25         0.920494        0.035559
26         0.920324        0.035165
27         0.909341        0.027225
28         0.876274        0.036238
29         0.875974        0.036194
30         0.874965        0.036876
31         0.867134        0.037043

                                                params
0       {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 10,
'min_samples_split': 2}
1       {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 15,
'min_samples_split': 2}
2       {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 20,
'min_samples_split': 2}
3    {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 10,
'min_samples_split': 5}
4    {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 15,
'min_samples_split': 5}
5    {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 20,
'min_samples_split': 5}
6       {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 15,
'min_samples_split': 5}
7       {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 10,
'min_samples_split': 5}
8       {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 20,
'min_samples_split': 5}
9    {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 10,
'min_samples_split': 2}
10   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 20,
'min_samples_split': 2}
11   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 15,
'min_samples_split': 2}
12  {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 20,
'min_samples_split': 10}
13  {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 10,
'min_samples_split': 10}
```

```
14  {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 15,
'min_samples_split': 10}
15     {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 10,
'min_samples_split': 10}
16     {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 20,
'min_samples_split': 10}
17     {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 15,
'min_samples_split': 10}
18  {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 15,
'min_samples_split': 20}
19  {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 10,
'min_samples_split': 20}
20  {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 20,
'min_samples_split': 20}
21    {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 5,
'min_samples_split': 5}
22     {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 10,
'min_samples_split': 20}
23     {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 15,
'min_samples_split': 20}
24     {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 20,
'min_samples_split': 20}
25   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 5,
'min_samples_split': 10}
26    {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 5,
'min_samples_split': 2}
27   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 5,
'min_samples_split': 20}
28       {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 5,
'min_samples_split': 5}
29       {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 5,
'min_samples_split': 2}
30      {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 5,
'min_samples_split': 10}
31      {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 5,
'min_samples_split': 20}
```

[ ]:
```
dt_model = DecisionTreeClassifier(class_weight= 'balanced', criterion=␣
 ↪'entropy', max_depth= 5, min_samples_split= 20, random_state=42)
dt_model.fit(X_train, y_train)

cv_scores = cross_val_score(dt_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): [0.87497796 0.89667138
```

```
0.90319018 0.91457209 0.95729316]
Mean CV Weighted F1 Score: 0.9093409522725183
```

1.6.4 G.4 Model Technical Performance

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

y_preds = dt_model.predict(X_val)

print("DecisionTree 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))
```

```
DecisionTree Evaluation:


Recall_score: 0.9356435643564357

f1_score: 0.936442971062295

Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.93      0.95       103
           1       0.98      0.92      0.95        51
           2       0.82      0.94      0.88        35
           3       0.93      1.00      0.96        13

    accuracy                           0.94       202
   macro avg       0.92      0.95      0.93       202
weighted avg       0.94      0.94      0.94       202


Confusion Matrix:
 [[96  1  5  1]
 [ 2 47  2  0]
 [ 2  0 33  0]
 [ 0  0  0 13]]
```

[19]:
```
# <Student to fill this section>
model_performance_explanations = """
Provide some explanations on model performance

The Decision Tree model achieved a weighted F1 score of 0.90 in␣
 ↪cross-validation and 0.94 on the validation set, with a recall of 0.94.
This indicates strong and consistent performance across data splits.
```

```
GridSearchCV was used to efficiently explore multiple hyperparameter␣
 ↪combinations for the Decision Tree, as manual tuning would be impractical␣
 ↪due to the many possible hyperparameter combinations.
Although the highest-scoring configuration was identified, a slightly␣
 ↪lower-performing set (around 90% of the best score) was selected to reduce␣
 ↪the risk of overfitting and improve generalization to unseen data.

The model shows excellent recall for the 'Poor' class (label 0), correctly␣
 ↪identifying 96 out of 103 students, and strong recall for the 'Average'␣
 ↪class (label 1),
identifying 47 out of 51 students, which is critical for early intervention.

Weighted F1 score was used due to class imbalance, ensuring balanced␣
 ↪performance across all categories.
The confusion matrix shows minimal misclassifications.

Overall, the model outperforms SVC and demonstrates strong predictive ability;
however, since accuracy above 90% is rare in real-world business problems,␣
 ↪there may be a risk of data leakage or overfitting in this case.
So the model may not be reliable and could fail when applied to real-world data.
"""
```

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

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

[20]:
```
print("Model Performance Explanations:", model_performance_explanations)
```

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

The Decision Tree model achieved a weighted F1 score of 0.90 in cross-validation
and 0.94 on the validation set, with a recall of 0.94.
This indicates strong and consistent performance across data splits.

GridSearchCV was used to efficiently explore multiple hyperparameter
combinations for the Decision Tree, as manual tuning would be impractical due to
the many possible hyperparameter combinations.
Although the highest-scoring configuration was identified, a slightly lower-
performing set (around 90% of the best score) was selected to reduce the risk of
overfitting and improve generalization to unseen data.

The model shows excellent recall for the 'Poor' class (label 0), correctly
identifying 96 out of 103 students, and strong recall for the 'Average' class
```

(label 1),
identifying 47 out of 51 students, which is critical for early intervention.

Weighted F1 score was used due to class imbalance, ensuring balanced performance across all categories.
The confusion matrix shows minimal misclassifications.

Overall, the model outperforms SVC and demonstrates strong predictive ability; however, since accuracy above 90% is rare in real-world business problems, there may be a risk of data leakage or overfitting in this case.
So the model may not be reliable and could fail when applied to real-world data.

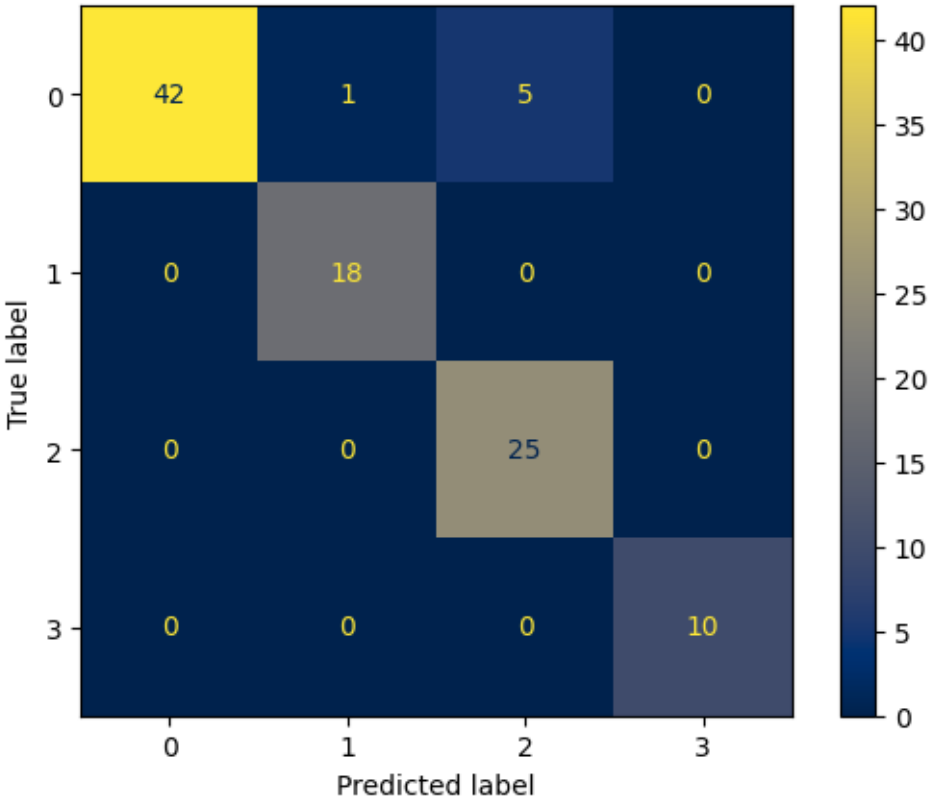
1.6.5 G.5 Business Impact from Current Model Performance

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

y_pred_final = dt_model.predict(X_test)

print("DecisionTree 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))
```

```
DecisionTree Evaluation:


Recall_score: 0.9405940594059405

f1_score: 0.9409978835721411

Classification Report:
               precision    recall  f1-score   support

           0       1.00      0.88      0.93        48
           1       0.95      1.00      0.97        18
           2       0.83      1.00      0.91        25
           3       1.00      1.00      1.00        10

    accuracy                           0.94       101
   macro avg       0.95      0.97      0.95       101
weighted avg       0.95      0.94      0.94       101
```

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

[ ]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x78c4908fd790>



[21]:
```
# <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 Decision Tree model performed very well on unseen test data, with both␣
 ↪weighted F1 score and recall around 94%.
It correctly identified most students across all performance levels, especially␣
 ↪those in the 'Poor' and 'Average' categories, which are key for early␣
 ↪support.
This can help the university take timely action and improve student outcomes.

Although the results are strong, the high accuracy may indicate possible data␣
 ↪leakage or overfitting.
If that's the case, the model might not perform as well on future data.
This could lead to missed opportunities for early intervention, affecting␣
 ↪student success and overall institutional performance.
```

```
So, more testing and review are needed before deploying it in a real-world␣
 ↪setting.
Also, the class distribution remains imbalanced, which may affect the model's␣
 ↪ability to generalize to new student groups.
"""
```

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

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

[22]:
```
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 Decision Tree model performed very well on unseen test data, with both
weighted F1 score and recall around 94%.
It correctly identified most students across all performance levels, especially
those in the 'Poor' and 'Average' categories, which are key for early support.
This can help the university take timely action and improve student outcomes.

Although the results are strong, the high accuracy may indicate possible data
leakage or overfitting.
If that's the case, the model might not perform as well on future data.
This could lead to missed opportunities for early intervention, affecting
student success and overall institutional performance.
So, more testing and review are needed before deploying it in a real-world
setting.
Also, the class distribution remains imbalanced, which may affect the model's
ability to generalize to new student groups.
```

1.7 H. Experiment Outcomes

[23]:
```
# <Student to fill this section>
experiment_outcome = "Hypothesis 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>
```

[24]:
```
print("Experiment Outcome:", experiment_outcome)
```

```
Experiment Outcome: Hypothesis Confirmed
```

[25]:
```
# <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 was: class_weight='balanced',␣
 ↪criterion='entropy', max_depth=5, and min_samples_split=20.
The Decision Tree model confirmed the initial hypothesis by achieving strong␣
 ↪overall performance, outperforming both the baseline and SVC models, and␣
 ↪accurately identifying the majority of at-risk students, including those in␣
 ↪the 'Poor' and 'Average' categories.
However, the unusually high performance raises concerns about possible data␣
 ↪leakage or overfitting, indicating the need for further validation.

Next steps include:

1. **Experiment with ensemble models (e.g., Extra Trees)**
   - Expected gain: Moderate to high (improved performance through multiple␣
 ↪decision trees)
   - Priority: High

2. **Apply advanced sampling techniques (e.g., SMOTE)**
   - Expected gain: Moderate (helps address class imbalance and may reduce␣
 ↪overfitting caused by underrepresented classes if applied correctly on␣
 ↪training data)
   - Priority: Medium

If the model continues to perform well after these validations, it may be a␣
 ↪strong candidate to support early academic intervention in real-world␣
 ↪applications.
"""
```

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

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

[26]:
```
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 was: class_weight='balanced',
criterion='entropy', max_depth=5, and min_samples_split=20.
The Decision Tree model confirmed the initial hypothesis by achieving strong
overall performance, outperforming both the baseline and SVC models, and
accurately identifying the majority of at-risk students, including those in the
'Poor' and 'Average' categories.
However, the unusually high performance raises concerns about possible data
leakage or overfitting, indicating the need for further validation.

Next steps include:

1. **Experiment with ensemble models (e.g., Extra Trees)**
   - Expected gain: Moderate to high (improved performance through multiple
decision trees)
   - Priority: High

2. **Apply advanced sampling techniques (e.g., SMOTE)**
   - Expected gain: Moderate (helps address class imbalance and may reduce
overfitting caused by underrepresented classes if applied correctly on training
data)
   - Priority: Medium

If the model continues to perform well after these validations, it may be a
strong candidate to support early academic intervention in real-world
applications.
```