# 36106-25AU-AT2-25589351-experiment-3

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 -:--:--
                      0.3/1.6
MB 8.4 MB/s eta 0:00:01
                      1.6/1.6 MB
22.9 MB/s eta 0:00:01
                      1.6/1.6 MB 18.0
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
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]:
```python
# <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.
"""
```

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

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

[6]:
```python
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 = "3"
print_tile(size="h1", key='experiment_id', value=experiment_id)
```

<IPython.core.display.HTML object>

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

Experiment ID: 3

[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 effectively predicted using an Extra Trees␣
  ↪classifier trained on behavioral, academic, and demographic features.
The Decision trees (Extra Trees) can outperform single tree models by reducing␣
  ↪variance and improving generalization.

The hypothesis is that ExtraTreesClassifier will deliver more stable and␣
  ↪accurate predictions compared to the DecisionTreeClassifier,
because it is using multiple randomized trees to reduce overfitting and enhance␣
  ↪performance.
It will maintain interpretability while offering higher robustness, making it a␣
  ↪promising model for consistently identifying at-risk students.
"""
```

[ ]:
```
# 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 effectively predicted using an Extra Trees classifier trained on behavioral, academic, and demographic features.
The Decision trees (Extra Trees) can outperform single tree models by reducing variance and improving generalization.

The hypothesis is that ExtraTreesClassifier will deliver more stable and accurate predictions compared to the DecisionTreeClassifier,
because it is using multiple randomized trees to reduce overfitting and enhance performance.
It will maintain interpretability while offering higher robustness, making it a promising model for consistently identifying at-risk students.

[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 Extra Trees Classifier is expected to outperform the baseline, Decision␣
 ↪Tree, and SVC models, with a weighted F1 score above 80 and recall close to␣
 ↪or above 85.
Because it is using multiple randomized trees, it is likely to deliver more␣
 ↪stable predictions, reduce overfitting, and capture complex feature␣
 ↪interactions.

Various hyperparameters (e.g., n_estimators, max_depth, min_samples_leaf) will␣
 ↪be tuned to maximize performance, particularly for identifying students in␣
 ↪the 'Poor' and 'Average' category.
While the model may be less interpretable than a single Decision Tree, it is␣
 ↪expected to provide better generalization and consistency.

Possible scenarios include:
- Significant performance gain over Decision Tree and SVC, validating the␣
 ↪benefit of ensemble methods.
- Moderate improvement with higher stability and less variance, supporting its␣
 ↪use in real-world applications.
- Minimal gains or overfitting if not tuned properly, indicating a need for␣
 ↪advanced sampling or feature selection techniques.
"""
```

```
[ ]: # 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 Extra Trees Classifier is expected to outperform the baseline, Decision
Tree, and SVC models, with a weighted F1 score above 80 and recall close to or
above 85.
Because it is using multiple randomized trees, it is likely to deliver more
stable predictions, reduce overfitting, and capture complex feature
interactions.

Various hyperparameters (e.g., n_estimators, max_depth, min_samples_leaf) will
be tuned to maximize performance, particularly for identifying students in the
'Poor' and 'Average' category.
While the model may be less interpretable than a single Decision Tree, it is
expected to provide better generalization and consistency.

Possible scenarios include:
- Significant performance gain over Decision Tree and SVC, validating the
benefit of ensemble methods.
- Moderate improvement with higher stability and less variance, supporting its
use in real-world applications.
- Minimal gains or overfitting if not tuned properly, indicating a need for
advanced sampling or feature selection techniques.
```

---

### 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 features capture key aspects of a student's academics, behavior, and␣
 ↪background.
They are well-suited for Decision Tree models, which can handle many variables␣
 ↪and uncover important 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 features capture key aspects of a student's academics, behavior, and
background.
They are well-suited for Decision Tree models, which can handle many variables
and uncover important patterns.
Less relevant features were removed in Experiment 0 based on their low impact on
the target variable.
```

---

## 1.6 E. Data Preparation

### 1.6.1 E.1 Data Transformation

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

# Apply SMOTE only on training set
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
```

[ ]:
```
print(X_train.shape)
print(X_train_resampled.shape)
```

```
(706, 125)
(1408, 125)
```

[15]:
```
# <Student to fill this section>
data_transformation_1_explanations = """
Provide some explanations on why you believe it is important to perform this␣
↪data transformation and its impacts

SMOTE was applied to the training set to address class imbalance across all␣
↪classes, including multiple minority classes.
By generating synthetic examples for each underrepresented class, it helps the␣
↪model learn more equally from all categories,
improving recall and reducing bias. This enhances the model's ability to␣
↪identify students at different risk levels.
After resampling, the training set now contains 1,408 samples.
"""
```

[ ]:
```
# Do not modify this code
print_tile(size="h3", key='data_transformation_1_explanations',␣
↪value=data_transformation_1_explanations)
```

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

[16]:
```
print("Data Transformation 1 Explanations:", data_transformation_1_explanations)
```

```
Data Transformation 1 Explanations:
Provide some explanations on why you believe it is important to perform this
data transformation and its impacts

SMOTE was applied to the training set to address class imbalance across all
classes, including multiple minority classes.
By generating synthetic examples for each underrepresented class, it helps the
model learn more equally from all categories,
improving recall and reducing bias. This enhances the model's ability to
```

```
identify students at different risk levels.
After resampling, the training set now contains 1,408 samples.
```

---

## 1.7 G. Train Machine Learning Model

### 1.7.1 G.1 Import Algorithm

[ ]:
```python
# <Student to fill this section>
from sklearn.ensemble import ExtraTreesClassifier
```

[17]:
```python
# <Student to fill this section>
algorithm_selection_explanations = """
Provide some explanations on why you believe this algorithm is a good fit

Extra Trees is a good fit for this task because it combines the strength of␣
↪multiple randomized decision trees to improve accuracy and reduce␣
↪overfitting.
It performs well with high-dimensional data, handles non-linear relationships,␣
↪and is robust to noise.
Its ensemble nature also provides better generalization compared to a single␣
↪decision tree, making it suitable for predicting student performance across␣
↪varied feature sets.
"""
```

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

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

[18]:
```python
print("Algorithm Selection Explanations:", algorithm_selection_explanations)
```

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

Extra Trees is a good fit for this task because it combines the strength of
multiple randomized decision trees to improve accuracy and reduce overfitting.
It performs well with high-dimensional data, handles non-linear relationships,
and is robust to noise.
Its ensemble nature also provides better generalization compared to a single
decision tree, making it suitable for predicting student performance across
varied feature sets.
```

### 1.7.2 G.2 Set Hyperparameters

[ ]:
```
# <Student to fill this section>
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [5, 10, 20],
    'min_samples_leaf': [2, 5, 7],
    'criterion': ['gini', 'entropy'],
    'class_weight': ['balanced']
}
```

[19]:
```
# <Student to fill this section>
hyperparameters_selection_explanations = """
Explain why you are tuning these hyperparameters

These hyperparameters are tuned to improve model performance and control␣
↪overfitting.
- 'n_estimators' determines the number of trees in the ensemble, which can␣
↪affect stability and accuracy.
- 'max_depth' controls the depth of each tree, helping balance bias and␣
↪variance.
- 'min_samples_leaf' prevents trees from learning overly specific patterns,␣
↪improving generalization.
- 'criterion' changes the splitting metric, allowing evaluation of both Gini␣
↪and entropy for better splits.
- 'class_weight' is set to 'balanced' to handle class imbalance introduced by␣
↪the original dataset distribution.
Tuning these helps the model generalize better and perform well across all␣
↪student categories.
"""
```

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

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

[20]:
```
print("Hyperparameters Selection Explanations:",␣
↪hyperparameters_selection_explanations)
```

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

These hyperparameters are tuned to improve model performance and control
overfitting.
- 'n_estimators' determines the number of trees in the ensemble, which can
affect stability and accuracy.
```

```
- 'max_depth' controls the depth of each tree, helping balance bias and
variance.
- 'min_samples_leaf' prevents trees from learning overly specific patterns,
improving generalization.
- 'criterion' changes the splitting metric, allowing evaluation of both Gini and
entropy for better splits.
- 'class_weight' is set to 'balanced' to handle class imbalance introduced by
the original dataset distribution.
Tuning these helps the model generalize better and perform well across all
student categories.
```

### 1.7.3 G.3 Fit Model

[ ]:
```python
# <Student to fill this section>
et_model = ExtraTreesClassifier(random_state=42)

grid_search_et = GridSearchCV(
    estimator=et_model,
    param_grid=param_grid,
    cv=5,
    scoring='f1_weighted',
)

grid_search_et.fit(X_train_resampled, y_train_resampled)

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

results_df = pd.DataFrame(grid_search_et.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 Extra Trees parameters: {'class_weight': 'balanced', 'criterion':
'entropy', 'max_depth': 20, 'min_samples_leaf': 2, 'n_estimators': 200}
Best Weighted F1 Score: 0.9166
```

[ ]:
```
   mean_test_score  std_test_score  \
0         0.916605        0.032721
1         0.915096        0.030682
```

```
2           0.914879        0.033196
3           0.912301        0.029170
4           0.886043        0.032322
5           0.883861        0.040147
6           0.879461        0.038034
7           0.874476        0.039890
8           0.871725        0.044932
9           0.866431        0.042623
10          0.864954        0.040550
11          0.863029        0.051211
12          0.854188        0.039097
13          0.850862        0.043134
14          0.849078        0.039683
15          0.848485        0.042791
16          0.846922        0.035607
17          0.846893        0.047721
18          0.846147        0.041542
19          0.843509        0.043706
20          0.839120        0.042126
21          0.831300        0.045500
22          0.828890        0.047135
23          0.828627        0.040520
24          0.791353        0.038294
25          0.790385        0.039180
26          0.788927        0.046564
27          0.788484        0.039178
28          0.787819        0.042378
29          0.785856        0.040176
30          0.785182        0.041584
31          0.782991        0.036942
32          0.780938        0.041053
33          0.780451        0.037190
34          0.780373        0.037077
35          0.775855        0.037874

                               params
0   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 20,
'min_samples_leaf': 2, 'n_estimators': 200}
1      {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 20,
'min_samples_leaf': 2, 'n_estimators': 100}
2      {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 20,
'min_samples_leaf': 2, 'n_estimators': 200}
3   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 20,
'min_samples_leaf': 2, 'n_estimators': 100}
4   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 10,
'min_samples_leaf': 2, 'n_estimators': 200}
5      {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 10,
```

```
'min_samples_leaf': 2, 'n_estimators': 200}
6   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 10,
'min_samples_leaf': 2, 'n_estimators': 100}
7      {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 10,
'min_samples_leaf': 2, 'n_estimators': 100}
8   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 20,
'min_samples_leaf': 5, 'n_estimators': 200}
9   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 20,
'min_samples_leaf': 5, 'n_estimators': 100}
10     {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 20,
'min_samples_leaf': 5, 'n_estimators': 200}
11     {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 20,
'min_samples_leaf': 5, 'n_estimators': 100}
12     {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 20,
'min_samples_leaf': 7, 'n_estimators': 200}
13     {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 20,
'min_samples_leaf': 7, 'n_estimators': 100}
14     {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 10,
'min_samples_leaf': 5, 'n_estimators': 100}
15  {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 20,
'min_samples_leaf': 7, 'n_estimators': 200}
16  {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 20,
'min_samples_leaf': 7, 'n_estimators': 100}
17     {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 10,
'min_samples_leaf': 5, 'n_estimators': 200}
18  {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 10,
'min_samples_leaf': 5, 'n_estimators': 200}
19  {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 10,
'min_samples_leaf': 5, 'n_estimators': 100}
20     {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 10,
'min_samples_leaf': 7, 'n_estimators': 200}
21     {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 10,
'min_samples_leaf': 7, 'n_estimators': 100}
22  {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 10,
'min_samples_leaf': 7, 'n_estimators': 200}
23  {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 10,
'min_samples_leaf': 7, 'n_estimators': 100}
24   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 5,
'min_samples_leaf': 2, 'n_estimators': 200}
25      {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 5,
'min_samples_leaf': 2, 'n_estimators': 200}
26      {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 5,
'min_samples_leaf': 2, 'n_estimators': 100}
27   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 5,
'min_samples_leaf': 2, 'n_estimators': 100}
28      {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 5,
'min_samples_leaf': 5, 'n_estimators': 200}
```

```
29   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 5,
'min_samples_leaf': 5, 'n_estimators': 200}
30      {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 5,
'min_samples_leaf': 5, 'n_estimators': 100}
31   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 5,
'min_samples_leaf': 5, 'n_estimators': 100}
32      {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 5,
'min_samples_leaf': 7, 'n_estimators': 200}
33      {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 5,
'min_samples_leaf': 7, 'n_estimators': 100}
34   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 5,
'min_samples_leaf': 7, 'n_estimators': 200}
35   {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 5,
'min_samples_leaf': 7, 'n_estimators': 100}
```

[ ]:
```
et_model = ExtraTreesClassifier(class_weight= 'balanced', criterion= 'entropy',␣
↪max_depth= 20, min_samples_leaf= 2, n_estimators= 200, random_state=42)
et_model.fit(X_train_resampled, y_train_resampled)

cv_scores = cross_val_score(et_model, X_train_resampled, y_train_resampled,␣
↪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): [0.88896401 0.91078651
0.87650452 0.94239112 0.96438039]
Mean CV Weighted F1 Score: 0.9166053095421278
```

### 1.7.4 G.4 Model Technical Performance

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

y_preds = et_model.predict(X_val)

print("ExtraTrees 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))
```

```
ExtraTrees Evaluation:


Recall_score: 0.7574257425742574

f1_score: 0.753208587449134

Classification Report:
```

```
              precision    recall  f1-score   support

           0       0.87      0.89      0.88       103
           1       0.78      0.57      0.66        51
           2       0.57      0.80      0.67        35
           3       0.40      0.31      0.35        13

    accuracy                           0.76       202
   macro avg       0.66      0.64      0.64       202
weighted avg       0.77      0.76      0.75       202


Confusion Matrix:
 [[92  6  5  0]
 [13 29  7  2]
 [ 1  2 28  4]
 [ 0  0  9  4]]
```

[21]:

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

The Extra Trees model achieved a weighted F1 score of 0.92 in cross-validation␣
 ↪and 0.753 on the validation set, with a recall of 0.76.
This indicates solid generalization across folds and acceptable performance on␣
 ↪validation data.

GridSearchCV was used to efficiently explore various parameter combinations for␣
 ↪the Extra Trees.
The best parameters identified by GridSearchCV were used, as they achieved␣
 ↪strong performance (F1 score of 92) without clear signs of overfitting.

The model shows strong recall for the 'Poor' class (label 0), correctly␣
 ↪identifying 92 out of 103 students.
It also performs moderately well for the 'Average' (label 1) class, identifying␣
 ↪29 out of 51 students.

Weighted F1 score was used to address class imbalance and ensure balanced␣
 ↪evaluation across categories.

Overall, the model outperforms SVC and offers a good balance of accuracy,␣
 ↪stability, and class-wise fairness.
While results are not as high as with the Decision Tree, they are more␣
 ↪realistic and likely to generalize better to real-world data.
"""
```

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

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

[22]:
```python
print("Model Performance Explanations:", model_performance_explanations)
```

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

The Extra Trees model achieved a weighted F1 score of 0.92 in cross-validation
and 0.753 on the validation set, with a recall of 0.76.
This indicates solid generalization across folds and acceptable performance on
validation data.

GridSearchCV was used to efficiently explore various parameter combinations for
the Extra Trees.
The best parameters identified by GridSearchCV were used, as they achieved
strong performance (F1 score of 92) without clear signs of overfitting.

The model shows strong recall for the 'Poor' class (label 0), correctly
identifying 92 out of 103 students.
It also performs moderately well for the 'Average' (label 1) class, identifying
29 out of 51 students.

Weighted F1 score was used to address class imbalance and ensure balanced
evaluation across categories.

Overall, the model outperforms SVC and offers a good balance of accuracy,
stability, and class-wise fairness.
While results are not as high as with the Decision Tree, they are more realistic
and likely to generalize better to real-world data.
```

### 1.7.5 G.5 Business Impact from Current Model Performance

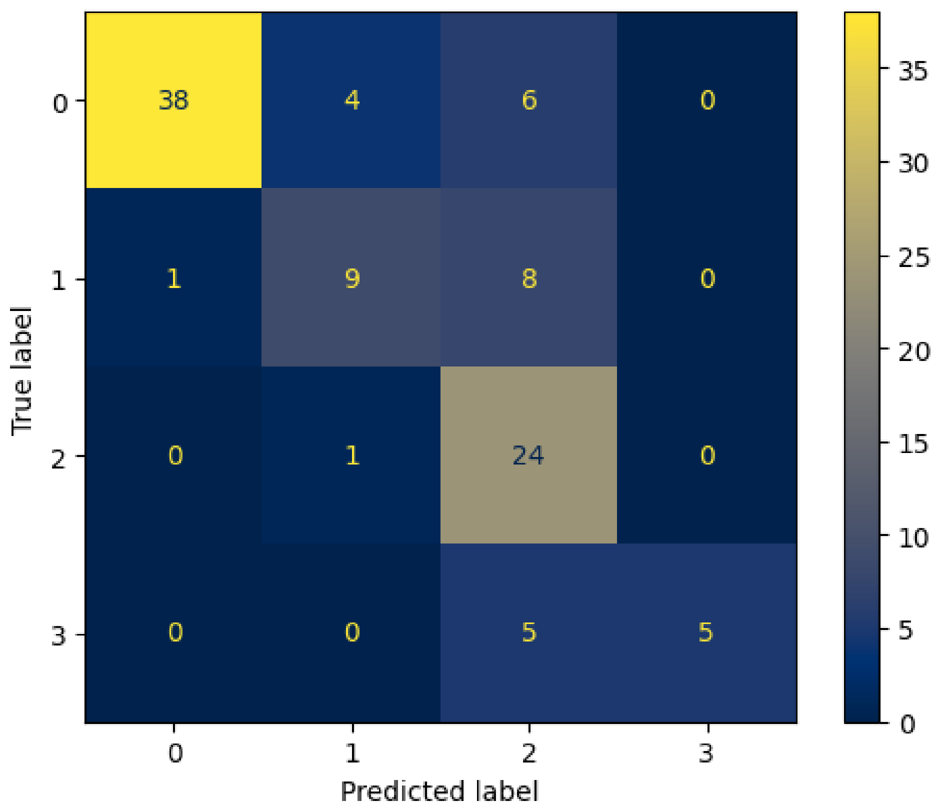
[ ]:
```python
# <Student to fill this section>

y_pred_final = et_model.predict(X_test)

print("ExtraTrees 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))
```

```
ExtraTrees Evaluation:
```

```
Recall_score: 0.7524752475247525

f1_score: 0.7561362373559871

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.79      0.87        48
           1       0.64      0.50      0.56        18
           2       0.56      0.96      0.71        25
           3       1.00      0.50      0.67        10

    accuracy                           0.75       101
   macro avg       0.79      0.69      0.70       101
weighted avg       0.81      0.75      0.76       101
```

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

[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x798cc7947490>
```



[23]:

```
# <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 Extra Trees model aligns well with identifying students at risk. It␣
↪achieved an F1-score of 0.76 on unseen data,
showing strong overall performance and reliability in real-world scenarios.

Moderate recall for the 'Good' class (label 2) and strong recall for the 'Poor'␣
↪class (label 0) suggest the model is effective at detecting students who␣
↪need support,
while minimizing false negative in the most critical category.

However, moderate misclassification in the 'Average' class (label 1) and low␣
↪recall for the 'Excellent' class (label 3)
could lead to some students being overlooked for enrichment opportunities or␣
↪being misprioritized.

Overall, the impact of incorrect predictions is more serious for the 'Poor'␣
↪category. Since the model handles this group well,
it meets the key business goal of enabling early intervention for struggling␣
↪students.
"""
```

[ ]:

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

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

[24]:

```
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 Extra Trees model aligns well with identifying students at risk. It achieved
an F1-score of 0.76 on unseen data,
showing strong overall performance and reliability in real-world scenarios.

Moderate recall for the 'Good' class (label 2) and strong recall for the 'Poor'
class (label 0) suggest the model is effective at detecting students who need
support,
```

```
while minimizing false negative in the most critical category.

However, moderate misclassification in the 'Average' class (label 1) and low
recall for the 'Excellent' class (label 3)
could lead to some students being overlooked for enrichment opportunities or
being misprioritized.

Overall, the impact of incorrect predictions is more serious for the 'Poor'
category. Since the model handles this group well,
it meets the key business goal of enabling early intervention for struggling
students.
```

## 1.8 H. Experiment Outcomes

[25]:
```
# <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>
```

[26]:
```
print("Experiment Outcome:", experiment_outcome)
```

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

[27]:
```
# <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=20, min_samples_leaf=2, and n_estimators=200.
The Extra Trees model partially confirmed the initial hypothesis by providing␣
↪stable and realistic performance on unseen data, outperforming SVC and␣
↪generalizing better than the Decision Tree.
```

```
It achieved a cross-validated weighted F1 score of 0.92 and a validation F1␣
↪score of 0.76, with strong recall for the 'Poor' class and balanced␣
↪class-wise performance.
While it did not surpass the Decision Tree in raw performance, the results were␣
↪more realistic and more likely to generalize to real-world applications.

Next steps include:

1. **Engineer additional features (e.g., academic_status_level,␣
↪attendance_flag)**
   - Expected gain: Moderate to high (potential improvement in interpretability␣
↪and recall)
   - Priority: High

2. **Experiment with different encoding methods (e.g., label encoding for␣
↪dimensionality reduction)**
   - Expected gain: Moderate to high (may enhance interpretability and reduce␣
↪noise in high-dimensional data)
   - Priority: Medium

The model shows strong potential for real-world deployment, particularly for␣
↪early academic intervention.
If further refinements confirm consistent performance, the Extra Trees␣
↪classifier can be considered a suitable choice for meeting stakeholder␣
↪requirements.
"""
```

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

<IPython.core.display.HTML object>

[28]:
```
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=20, min_samples_leaf=2, and n_estimators=200.

The Extra Trees model partially confirmed the initial hypothesis by providing stable and realistic performance on unseen data, outperforming SVC and generalizing better than the Decision Tree.
It achieved a cross-validated weighted F1 score of 0.92 and a validation F1 score of 0.76, with strong recall for the 'Poor' class and balanced class-wise performance.
While it did not surpass the Decision Tree in raw performance, the results were more realistic and more likely to generalize to real-world applications.

Next steps include:

1. **Engineer additional features (e.g., academic_status_level, attendance_flag)**
   - Expected gain: Moderate to high (potential improvement in interpretability and recall)
   - Priority: High

2. **Experiment with different encoding methods (e.g., label encoding for dimensionality reduction)**
   - Expected gain: Moderate to high (may enhance interpretability and reduce noise in high-dimensional data)
   - Priority: Medium

The model shows strong potential for real-world deployment, particularly for early academic intervention.
If further refinements confirm consistent performance, the Extra Trees classifier can be considered a suitable choice for meeting stakeholder requirements.