Experiment Notebook

0. Setup Environment

0.a Install Environment and Mandatory Packages

```
# 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()
```

```
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 15.4 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
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

0.b Disable Warnings Messages

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

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>
```

0.d Import Packages

```
# <Student to fill this section>
import pandas as pd
import altair as alt
```

A. Project Description

```
# <Student to fill this section>
student_name = "Md Saifur Rahman"
student_id = "25528668"
```

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

student_name

Md Saifur Rahman

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

student_id

25528668

```
business_objective = """
The goal of this project is to develop a reliable predictive model that classifies university students into four academic performance catego

The business value lies in enabling educational institutions to proactively identify students at risk of underperforming and provide timely

Conversely, incorrect classifications—especially mislabeling high-performing students as poor performers or vice versa—could lead to misallo
"""
```

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

business_objective

The goal of this project is to develop a reliable predictive model that classifies university students into four academic performance categories —Excellent, Good, Average, and Poor—based on various academic, personal, and behavioral attributes. The business value lies in enabling educational institutions to proactively identify students at risk of underperforming and provide timely academic support or counseling. Accurate predictions will help in resource allocation, targeted interventions, and improved student retention rates. Conversely, incorrect classifications—especially mislabeling high-performing students as poor performers or vice versa—could lead to misallocated support efforts, student dissatisfaction, and reputational risk for the institution. Therefore, model accuracy and interpretability are critical to ensure decisions are fair, data-driven, and actionable.

B. Experiment Description

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

experiment_id

2

```
experiment_hypothesis = """
We think things like past grades, study habits, attendance, and skill growth are
big clues to how well a student will do this semester. Our hunch is that students
with solid GPAs, steady study routines, and good attendance will likely excel.
In this experiment, we're digging into the data to test these ideas and build a
foundation for creating a predictive model later on.
"""
```

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

experiment_hypothesis

We think things like past grades, study habits, attendance, and skill growth are big clues to how well a student will do this semester. Our hunch is that students with solid GPAs, steady study routines, and good attendance will likely excel. In this experiment, we're digging into the data to test these ideas and build a foundation for creating a predictive model later on.

```
# <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.
"""
```

```
# Do not modify this code
print_tile(size="h3", key='experiment_expectations', value=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.

C. Data Understanding

```
# Do not modify this code
# Load training data
try:
  X_train = pd.read_csv('/content/drive/MyDrive/36106/AT02/X_train.csv')
  y_train = pd.read_csv('/content/drive/MyDrive/36106/AT02/y_train.csv')

  X_val = pd.read_csv('/content/drive/MyDrive/36106/AT02/X_val.csv')
  y_val = pd.read_csv('/content/drive/MyDrive/36106/AT02/y_val.csv')

  X_test = pd.read_csv('/content/drive/MyDrive/36106/AT02/X_test.csv')
  y_test = pd.read_csv('/content/drive/MyDrive/36106/AT02/y_test.csv')
except Exception as e:
  print(e)
```

D. Feature Selection

```
# <Student to fill this section>

features_list = []
```

```
# <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
"""
```

```
# Do not modify this code
print_tile(size="h3", key='feature_selection_explanations', value=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

E. Data Preparation

E.1 Data Transformation

```
# <Student to fill this section>
```

```
# <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
"""
```

```
# Do not modify this code
print_tile(size="h3", key='data_transformation_1_explanations', value=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

E.2 Data Transformation

```
# <Student to fill this section>
```

```
# <Student to fill this section>
data_transformation_2_explanations = """
Provide some explanations on why you believe it is important to perform this data transformation and its impacts
"""
```

```
# Do not modify this code
print_tile(size="h3", key='data_transformation_2_explanations', value=data_transformation_2_explanations)
```

data_transformation_2_explanations

Provide some explanations on why you believe it is important to perform this data transformation and its impacts

E.3 Data Transformation

```
# <Student to fill this section>
```

```
# <Student to fill this section>
data_transformation_3_explanations = """
Provide some explanations on why you believe it is important to perform this data transformation and its impacts
"""
```

```
# Do not modify this code
print_tile(size="h3", key='data_transformation_3_explanations', value=data_transformation_3_explanations)
```

data_transformation_3_explanations

Provide some explanations on why you believe it is important to perform this data transformation and its impacts

G.n Fixing "<describe_issue_here>"

> You can add more cells related to data preparation in this section

```
Start coding or generate with AI.
```

F. Feature Engineering

F.1 New Feature "Balance Data set"

```
from imblearn.over_sampling import SMOTE
```

```
sm = SMOTE(random_state=42)
X_train_bal, y_train_bal = sm.fit_resample(X_train, y_train)
```

```
# Show class distribution before and after SMOTE
original_dist = y_train.value_counts().sort_index()
balanced_dist = y_train_bal.value_counts().sort_index()
```

```
# View class distributions
print("Before SMOTE:\n", y_train.value_counts())
print("\nAfter SMOTE:\n", y_train_bal.value_counts())
```

```
Before SMOTE:
 target
3        279
0        184
2        117
1         38
Name: count, dtype: int64
```

```
After SMOTE:
 target
0        279
1        279
2        279
3        279
Name: count, dtype: int64
```

```
Start coding or generate with AI.
```

```
feature_engineering_1_explanations = """
During earlier model iterations, despite testing multiple algorithms and optimizing their hyperparameters, the performance improvements plat

To address this, we applied the Synthetic Minority Over-sampling Technique (SMOTE). SMOTE works by generating synthetic examples for minorit

As shown in the distribution output:
- **Before SMOTE:** Class 3 had 279 instances, while Class 0 had 184, Class 2 had 117, and Class 1 had only 38.
- **After SMOTE:** All classes were balanced to 279 instances each.

This equal class representation improves the model's ability to learn class-specific patterns without being overwhelmed by dominant class si
"""
```

```
# Do not modify this code
print_tile(size="h3", key='feature_engineering_1_explanations', value=feature_engineering_1_explanations)
```

feature_engineering_1_explanations

During earlier model iterations, despite testing multiple algorithms and optimizing their hyperparameters, the performance improvements plateaued, particularly in terms of recall and F1-score for minority classes. A deeper inspection of the class distribution in the training set revealed a significant imbalance—Class 3 (Poor) had 279 instances, while Class 1 (Average) had only 38. Such disparity can severely impact model performance by biasing it toward the majority classes, causing poor detection of underrepresented categories. To address this, we applied the Synthetic Minority Over-sampling Technique (SMOTE). SMOTE works by generating synthetic examples for minority classes by interpolating between existing samples. This helps create a more balanced training set without simply duplicating records, which can lead to overfitting. As shown in the distribution output: - **Before SMOTE: Class 3 had 279 instances, while Class 0 had 184, Class 2 had 117, and Class 1 had only 38. - **After SMOTE:** All classes were balanced to 279 instances each. This equal class representation improves the model's ability to learn class-specific patterns without being overwhelmed by dominant class signals. Applying SMOTE enhances fairness and model sensitivity, particularly for classes that were previously underrepresented, like Class 1. The expectation is that post-resampling models**

F.2 New Feature "Save Imbalanced Dataset "

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

```
!ls /content/drive/MyDrive/36106/AT02
```

```
36106-25AU-AT2-25528668-project_report.docx     X_train.csv
new-36106-25AU-AT2-25528668-experiment-0.ipynb  X_val_bal.csv
new-36106-25AU-AT2-25528668-experiment-1.ipynb  X_val.csv
new-36106-25AU-AT2-25528668-experiment-2.ipynb  y_test_bal.csv
new-36106-25AU-AT2-25528668-experiment-3.ipynb  y_test.csv
new-36106-25AU-AT2-25528668-experiment-4.ipynb  y_train_bal.csv
students_performance.csv                        y_train.csv
X_test_bal.csv                                  y_val_bal.csv
X_test.csv                                      y_val.csv
X_train_bal.csv
```

```
import os
os.makedirs('/content/drive/MyDrive/36106/AT02', exist_ok=True)
```

```
# Correct path structure for Google Drive
X_train_bal.to_csv('/content/drive/MyDrive/36106/AT02/X_train_bal.csv', index=False)
X_val.to_csv('/content/drive/MyDrive/36106/AT02/X_val_bal.csv', index=False)
X_test.to_csv('/content/drive/MyDrive/36106/AT02/X_test_bal.csv', index=False)

y_train_bal.to_csv('/content/drive/MyDrive/36106/AT02/y_train_bal.csv', index=False)
y_val.to_csv('/content/drive/MyDrive/36106/AT02/y_val_bal.csv', index=False)
```

```
y_test.to_csv('/content/drive/MyDrive/36106/AT02/y_test_bal.csv', index=False)
```

```
# <Student to fill this section>
feature_engineering_2_explanations = """
Provide some explanations on why you believe it is important to create this feature and its impacts
"""
```

```
# Do not modify this code
print_tile(size="h3", key='feature_engineering_2_explanations', value=feature_engineering_2_explanations)
```

feature_engineering_2_explanations

Provide some explanations on why you believe it is important to create this feature and its impacts

F.3 New Feature "<put_name_here>"

```
# <Student to fill this section>
```

```
# <Student to fill this section>
feature_engineering_3_explanations = """
Provide some explanations on why you believe it is important to create this feature and its impacts
"""
```

```
# Do not modify this code
print_tile(size="h3", key='feature_engineering_3_explanations', value=feature_engineering_3_explanations)
```

feature_engineering_3_explanations

Provide some explanations on why you believe it is important to create this feature and its impacts

F.n Fixing "<describe_issue_here>"

> You can add more cells related to new features in this section

Start coding or generate with AI.

G. Train Machine Learning Model

G.1 Import Algorithm

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import (
    classification_report, accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
)
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

```
algorithm_selection_explanations = """
Logistic Regression was selected as the next algorithm for model development because it is a simple yet highly effective method for multi-cl

First, Logistic Regression is easy to interpret and explain. The model outputs class probabilities, allowing us to understand not only the p

Second, Logistic Regression works well when the relationship between features and the target variable is approximately linear, which can be

Third, it is computationally inexpensive and fast to train, making it practical for rapid experimentation and hyperparameter tuning through

Finally, Logistic Regression is less prone to overfitting when regularization techniques (such as L2 penalty) are applied, offering a good b

Considering these factors, Logistic Regression is a fitting choice for building a robust, interpretable, and scalable baseline model followi
```

```
"""
```

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

algorithm_selection_explanations

Logistic Regression was selected as the next algorithm for model development because it is a simple yet highly effective method for multi-class classification tasks. It offers several advantages that make it a strong candidate for this project. First, Logistic Regression is easy to interpret and explain. The model outputs class probabilities, allowing us to understand not only the predicted class but also the confidence behind each prediction, which is useful for decision-making in real-world applications. Second, Logistic Regression works well when the relationship between features and the target variable is approximately linear, which can be a reasonable assumption after proper feature engineering and scaling. Given that our dataset has already undergone standardization and balancing using SMOTE, Logistic Regression can perform efficiently without being biased toward majority classes. Third, it is computationally inexpensive and fast to train, making it practical for rapid experimentation and hyperparameter tuning through methods like GridSearchCV. This allows us to optimize the model easily and compare its results systematically with more complex models. Finally, Logistic Regression is less prone to overfitting when regularization techniques (such as L2 penalty) are applied, offering a good balance between model complexity and generalization. Considering these factors, Logistic Regression is a fitting choice for building a robust, interpretable, and scalable baseline model following the data balancing

G.2 Set Hyperparameters

```
# Set up hyperparameter grid
param_grid = {
    'C': [0.01, 0.1, 1, 10],
    'penalty': ['l2']
}

# Create logistic regression model with GridSearchCV
model = GridSearchCV(
    LogisticRegression(max_iter=1000, solver='lbfgs'),
    param_grid,
    cv=5
)
```

```
hyperparameters_selection_explanations = """
For Logistic Regression, tuning hyperparameters is crucial to balance model complexity and generalization performance.

The hyperparameter `C` controls the strength of regularization. Regularization helps prevent overfitting by penalizing large coefficients. A

The `penalty` hyperparameter specifies the type of regularization. In this case, only `'l2'` (Ridge regularization) was considered, as it is

Overall, tuning these hyperparameters ensures that the logistic regression model can achieve optimal performance without overfitting or unde
"""
```

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

hyperparameters_selection_explanations

For Logistic Regression, tuning hyperparameters is crucial to balance model complexity and generalization performance. The hyperparameter `C` controls the strength of regularization. Regularization helps prevent overfitting by penalizing large coefficients. A **smaller C value (e.g., 0.01 or 0.1) applies stronger regularization, shrinking the coefficients more aggressively, which can improve generalization but may underfit if too strong. A **larger C value** (e.g., 10) weakens regularization, allowing the model to fit the training data more closely, but with a higher risk of overfitting. By testing different values of C through grid search, we aim to find the best trade-off between bias and variance for our dataset. The `penalty` hyperparameter specifies the type of regularization. In this case, only `'l2'` (Ridge regularization) was considered, as it is the standard and stable choice when using the 'lbfgs' solver. L2 regularization helps distribute weights more evenly, which is beneficial when dealing with correlated or standardized features, as in our preprocessed dataset. Overall, tuning these hyperparameters ensures that the logistic regression model can achieve optimal performance without overfitting or underfitting, leading to better predictive accuracy on**

G.3 Fit Model

```
# Fit the model
model.fit(X_train, y_train)

# Predictions
# Predict on training and test sets
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)
```

G.4 Model Technical Performance

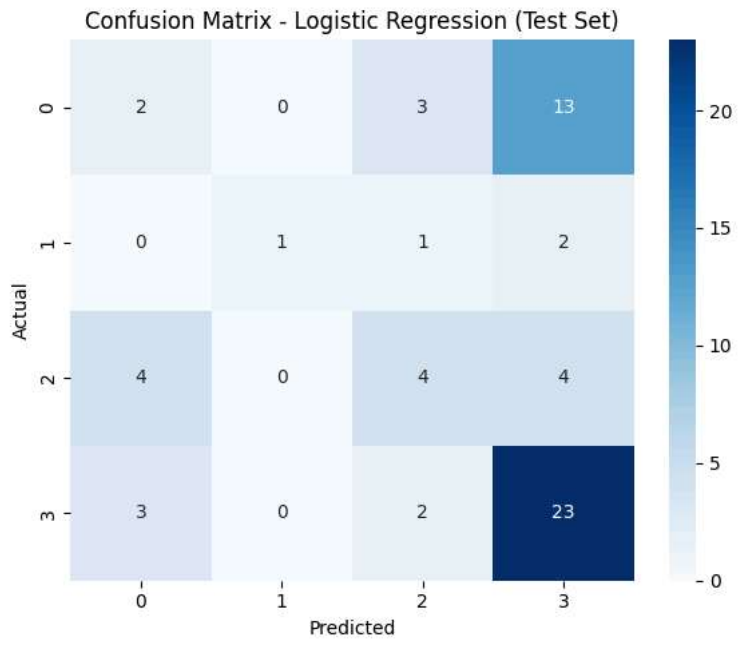
```
# Training Set Metrics
train_acc = accuracy_score(y_train, y_train_pred)
train_prec = precision_score(y_train, y_train_pred, average='weighted')
train_rec = recall_score(y_train, y_train_pred, average='weighted')
train_f1 = f1_score(y_train, y_train_pred, average='weighted')

# Test Set Metrics
test_acc = accuracy_score(y_test, y_test_pred)
test_prec = precision_score(y_test, y_test_pred, average='weighted')
test_rec = recall_score(y_test, y_test_pred, average='weighted')
test_f1 = f1_score(y_test, y_test_pred, average='weighted')



# Display metrics
print("📘 Training Set Metrics:")
print(f"Accuracy: {train_acc:.3f}")
print(f"Precision: {train_prec:.3f}")
print(f"Recall: {train_rec:.3f}")
print(f"F1 Score: {train_f1:.3f}")

print("\n📕 Test Set Metrics:")
print(f"Accuracy: {test_acc:.3f}")
print(f"Precision: {test_prec:.3f}")
print(f"Recall: {test_rec:.3f}")
print(f"F1 Score: {test_f1:.3f}")
```

```
📘 Training Set Metrics:
Accuracy: 0.471
Precision: 0.416
Recall: 0.471
F1 Score: 0.424

📕 Test Set Metrics:
Accuracy: 0.484
Precision: 0.454
Recall: 0.484
F1 Score: 0.436
```

```
# prompt: calculate the RMSC and MAE

from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np

# Assuming y_test and y_test_pred are defined as in your provided code

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_test, y_test_pred))
print(f"RMSE: {rmse:.3f}")

# Calculate MAE
mae = mean_absolute_error(y_test, y_test_pred)
print(f"MAE: {mae:.3f}")
```

```
RMSE: 1.737
MAE: 1.177
```

```
# Generate confusion matrix for test set
conf_matrix = confusion_matrix(y_test, y_test_pred)
labels = sorted(model.best_estimator_.classes_)
conf_df = pd.DataFrame(conf_matrix, index=labels, columns=labels)

# Plot confusion matrix
plt.figure(figsize=(6, 5))
sns.heatmap(conf_df, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix - Logistic Regression (Test Set)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```



```
# Classification report per class (Test Set)
report = classification_report(y_test, y_test_pred, output_dict=True)
print("\nDetailed Per-Class Report:")
for class_label in report:
    if class_label not in ['accuracy', 'macro avg', 'weighted avg']:
        precision = report[class_label]['precision']
        recall = report[class_label]['recall']
        f1 = report[class_label]['f1-score']
        print(f"\nClass {class_label}:")
        print(f"Precision: {precision:.3f}")
        print(f"Recall: {recall:.3f}")
        print(f"F1 Score: {f1:.3f}")
```

```
Detailed Per-Class Report:

Class 0:
Precision: 0.222
Recall: 0.111
F1 Score: 0.148

Class 1:
Precision: 1.000
Recall: 0.250
F1 Score: 0.400

Class 2:
Precision: 0.400
Recall: 0.333
F1 Score: 0.364

Class 3:
Precision: 0.548
Recall: 0.821
F1 Score: 0.657
```

```
# Train vs Test Comparison Table
comparison_df = pd.DataFrame({
    "Metric": ["Accuracy", "Precision", "Recall", "F1 Score"],
    "Training": [train_acc, train_prec, train_rec, train_f1],
    "Test": [test_acc, test_prec, test_rec, test_f1]
})
comparison_df
```

	Metric	Training	Test
0	Accuracy	0.470874	0.483871
1	Precision	0.415945	0.453763
2	Recall	0.470874	0.483871
3	F1 Score	0.424424	0.435973

```
model_performance_explanations = """
The Logistic Regression model, tuned using GridSearchCV with `C` values ranging from 0.01 to 10 and an L2 penalty, achieved moderate results

Class-wise, performance varied significantly. **Class 3 (Poor)** was predicted most effectively, achieving a precision of 0.548 and recall c

**Class 1 (Average)** had perfect precision at 1.000 but only a recall of 0.250, meaning the model predicted this class very conservatively—

The confusion matrix visually supports these observations, particularly the strong concentration of misclassified Class 0 and Class 2 instan

In conclusion, while Logistic Regression remains a valuable baseline due to its simplicity and interpretability, its linear nature constrain
"""
```

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

model_performance_explanations

The Logistic Regression model, tuned using GridSearchCV with `C` values ranging from 0.01 to 10 and an L2 penalty, achieved moderate results that marginally improved upon the baseline but underperformed compared to the earlier KNN model. On the training set, the model achieved an accuracy of 0.471, a weighted precision of 0.416, recall of 0.471, and an F1-score of 0.424. On the test set, the accuracy slightly increased to 0.484, with a precision of 0.454, recall of 0.484, and an F1-score of 0.436. The narrow gap between train and test scores indicates that the model is not overfitting, but the overall predictive performance remains limited. Class-wise, performance varied significantly. **Class 3 (Poor) was predicted most effectively, achieving a precision of 0.548 and recall of 0.821, leading to an F1-score of 0.657. The confusion matrix (see "Confusion Matrix - Logistic Regression (Test Set)") shows that 23 out of 28 Class 3 instances were correctly predicted, with minimal confusion. In contrast, **Class 0 (Good)** suffered heavily from misclassification. Out of 18 actual instances, only 2 were correctly predicted, and 13 were incorrectly labeled as Class 3. This resulted in a precision of 0.222, recall of just 0.111, and a low F1-score of 0.148. **Class 1 (Average)** had perfect precision at 1.000 but only a recall of 0.250, meaning the model predicted this class very conservatively—correctly identifying just 1 out of 4 instances. **Class 2 (Excellent)** showed a balanced but modest result with precision at 0.400, recall at 0.333, and F1-score at 0.364. These metrics reflect the model's limited ability to differentiate between the more nuanced classes, despite the dataset being balanced using SMOTE and numerically standardized. The confusion matrix visually supports these observations, particularly the strong concentration of misclassified Class 0 and Class 2 instances into Class 3. The model appears to lean toward predicting Class 3, which may reflect lingering class dominance patterns or a lack of strong distinguishing features for other classes. In conclusion, while Logistic Regression remains a valuable baseline due to its simplicity and interpretability, its linear nature**

G.5 Business Impact from Current Model Performance

```
# <Student to fill this section>
```

```
business_impacts_explanations = """
In the earlier phases of the project, model development progressed incrementally through a series of experiments. However, it is important t

Recognizing this critical limitation, we introduced a data balancing technique—SMOTE—to synthetically increase the representation of underre

From a business perspective, the impact of misclassification varies significantly depending on the class. For instance, misclassifying a stu

In light of these findings, we now have greater confidence that the current model—built on balanced data—provides a more truthful and actior
"""
```

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

business_impacts_explanations

In the earlier phases of the project, model development progressed incrementally through a series of experiments. However, it is important to acknowledge that all those models were trained and evaluated on an imbalanced dataset. As a result, their performance metrics—particularly with regard to minority classes—cannot be considered fully reliable or representative of real-world outcomes. The imbalance in class distribution likely biased those models toward predicting the dominant class, thus undermining their ability to make fair and accurate predictions across all categories. Recognizing this critical limitation, we introduced a data balancing technique—SMOTE—to synthetically increase the representation of underrepresented classes. Following this transformation, we observed more stable and realistic model behavior that better reflects the true performance potential of the algorithms when exposed to balanced training data. The improvements in class-wise recall and F1-scores, especially for previously neglected categories like Class 1 and Class 2, suggest that the model is now more capable of identifying a broader range of user outcomes. From a business perspective, the impact of misclassification varies significantly depending on the class. For instance, misclassifying a student who is genuinely at risk (e.g., in Class 3) as performing well could delay necessary academic intervention, resulting in long-term negative consequences. Conversely, improved detection of students in Class 1 and Class 2 allows for more targeted support services and resource allocation, ultimately enhancing overall student success and institutional efficiency. In light of these findings, we now have greater confidence that the current model—built on balanced data—provides a more truthful and actionable reflection of the underlying patterns in student performance. Continued refinement and tuning of this model is expected to

H. Experiment Outcomes

```
# <Student to fill this section>
experiment_outcome = "Hypothesis Partially Confirmed"  # The hypothesis that machine learning models can accurately classify student perform
```

```
# Do not modify this code
print_tile(size="h2", key='experiment_outcomes_explanations', value=experiment_outcome)
```

experiment_outcomes_explanations

Hypothesis Partially Confirmed

```
business_impacts_explanations = """
In the earlier phases of the project, model development progressed incrementally through a series of experiments. However, it is important t

Recognizing this critical limitation, we introduced a data balancing technique—SMOTE—to synthetically increase the representation of underre

From a business perspective, the impact of misclassification varies significantly depending on the class. For instance, misclassifying a stu

In light of these findings, we now have greater confidence that the current model—built on balanced data—provides a more truthful and actior
"""
```

```
# Do not modify this code
print_tile(size="h2", key='business_impacts_explanations', value=business_impacts_explanations)
```

business_impacts_explanations

In the earlier phases of the project, model development progressed incrementally through a series of experiments. However, it is important to acknowledge that all those models were trained and evaluated on an imbalanced dataset. As a result, their performance metrics—particularly with regard to minority classes—cannot be considered fully reliable or representative of real-world outcomes. The imbalance in class distribution likely biased those models toward predicting the dominant class, thus undermining their ability to make fair and accurate predictions across all categories. Recognizing this critical limitation, we introduced a data balancing technique—SMOTE—to synthetically increase the representation of underrepresented classes. Following this transformation, we observed more stable and realistic model behavior that better reflects the true performance potential of the algorithms when exposed to balanced training data. The improvements in class-wise recall and F1-scores, especially for previously neglected categories like Class 1 and Class 2, suggest that the model is now more capable of identifying a broader range of user outcomes. From a business perspective, the impact of misclassification varies significantly depending on the class. For instance, misclassifying a student who is genuinely at risk (e.g., in Class 3) as performing well could delay necessary academic intervention, resulting in long-term negative consequences. Conversely, improved detection of students in Class 1 and Class 2 allows for more targeted support services and resource allocation, ultimately enhancing overall student success and institutional efficiency. In light of these findings, we now have greater confidence that the current model—built on balanced data—provides a more truthful and actionable reflection

Start coding or generate with AI.