STUDENT PERFORMANCE ANALYSIS

Prepared and Presented By: Uyen Mai

MOTIVATION

High college dropout rate

Predict whether an student pass the course or not

Provide early insights into student progress to support academic achievement and course completion

Data Overview

Data Source: Student Performance Prediction (Kaggle)

```
df = pd.read_csv("student_performance_prediction.csv")
df.head(5)
```

	Student ID	Passed	Study Hours per Week	Attendance Rate	Previous Grades	Participation in Extracurricular Activities	Parent Education Level
0	S00001	Yes	12.5	NaN	75.0	Yes	Master
1	S00002	No	9.3	95.3	60.6	No	High School
2	S00003	No	13.2	NaN	64.0	No	Associate
3	S00004	No	17.6	76.8	62.4	Yes	Bachelor
4	S00005	No	8.8	89.3	72.7	No	Master

```
rows, variables = df.shape
print(f"Number of rows: {rows}")
print(f"Number of variables: {variables}")
```

```
Number of rows: 40000
Number of variables: 7
```

DATA PREPROCESSING

Step 1: Check missing values and unique values of all variables
Step 2: Handle outliers, missing and categorical data

- All columns: impute missing data for numerical features using mean and for categorical features using mode
- Study Hours per Week: replace negative value with absolute value
- Attendance Rate: remove values greater than 100 or less than 0
- Previous Grades: remove values greater than 100
- Passed, Participation in Extracurricular Activities: replace binary values with 0's and 1's.
- Educational Parent Level: one hot encoding for nominal categorical variable

Step 3: Check correlation among input features to remove one that have strong correlation to others

DATA PREPROCESSING

Step 3:

No correlation detected

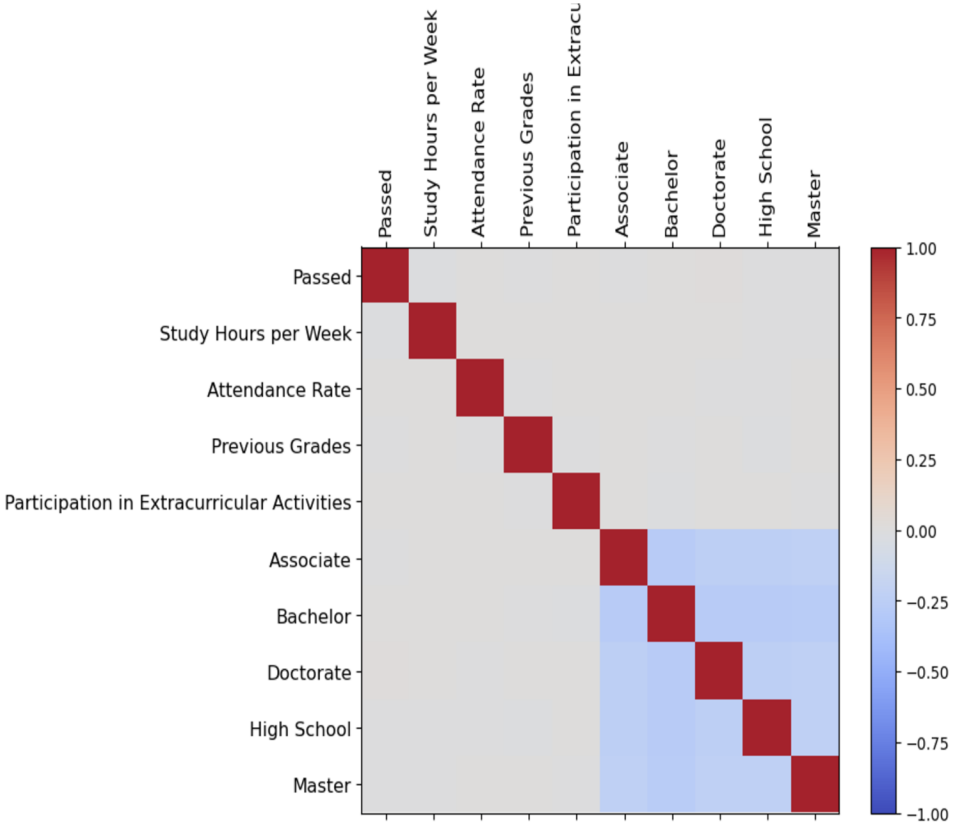
Step 4: Split dataset into: 80% train, 10% validation, 10% test

Step 5: Compute class weight to handle class imbalance in training set

Step 6: Scale numerical input features (Study Hours per Week, Attendance Rate, Previous Grades)

Step 7: Check relationships among input features and target feature

Weak or negligible relationships with the target variable detected



Data Overview

Data Source: Student Performance Prediction (Kaggle)

Before Data Processing

Item	Stats
Total variables	7
Multi-categorical variables	2
Numerical variables	3
Binary variables	2
Total records (rows)	40,000
Class distribution	balanced
Missing values	yes

After Data Processing

Item	Stats
Total variables	10
Multi-categorical variables	0
Numerical variables	3
Binary variables	7
Total records (rows)	31,774
Class distribution	balanced
Missing values	no

Data Overview

Cleaned Dataset

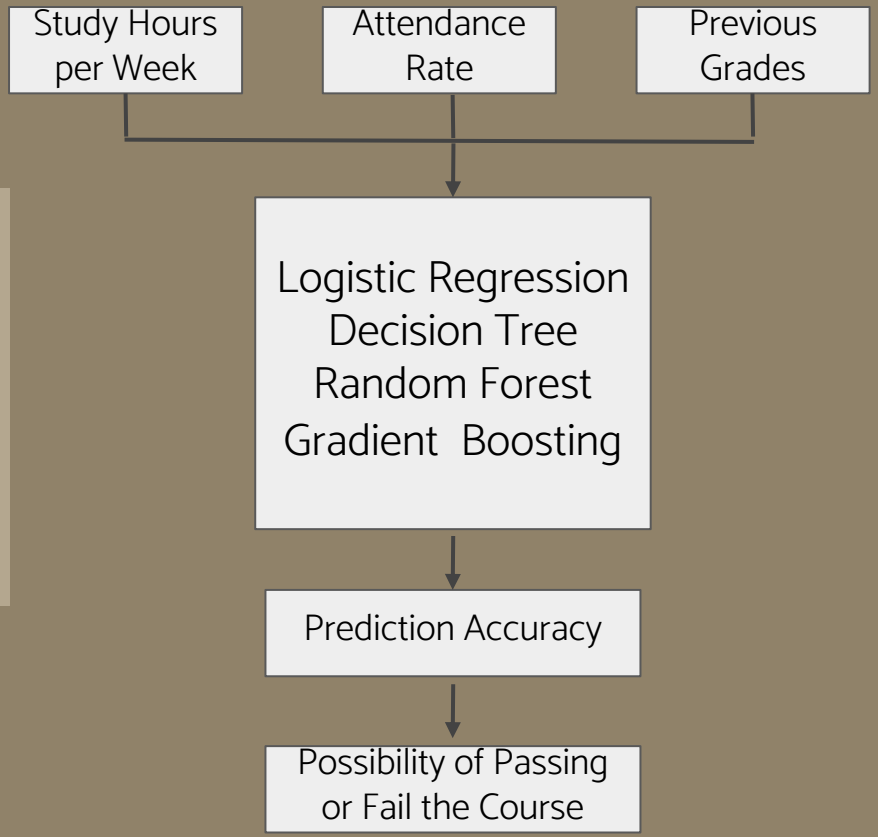
```
df_copy2.head(5)
```

	Passed	Study Hours per Week	Attendance Rate	Previous Grades	Participation in Extracurricular Activities	Associate	Bachelor	Doctorate	High School	Master
1	0	9.3	95.3	60.6	0	False	False	False	True	False
3	0	17.6	76.8	62.4	1	False	True	False	False	False
4	0	8.8	89.3	72.7	0	False	False	False	False	True
5	1	8.8	73.8	69.3	1	False	False	False	True	False
6	1	17.9	38.6	93.6	0	False	False	True	False	False

```
rows, variables = df_copy2.shape
print(f"Number of rows: {rows}")
print(f"Number of variables: {variables}")
```

```
Number of rows: 31774
Number of variables: 10
```

MODELING PROCESS



1. Example Features

2. Prediction Models
(Using GridSearchCV to find the best parameters)

3. Model Evaluation

4. Impact Prediction

MODEL SELECTION

```
models = {
    "Logistic Regression": linear_model.LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Gradient Boosting": GradientBoostingClassifier()
}
param_grids = {
    "Logistic Regression": {'C': [0.01, 0.1, 1, 10], 'penalty': ['l2'], 'solver': ['liblinear', 'lbfgs'],
                            'max_iter': [500, 1000], 'class_weight':[class_weights_dict]},
    "Decision Tree": {'max_depth': [5, 10, 20],'min_samples_split': [2, 10, 20],'min_samples_leaf': [2, 5, 10],
                      'class_weight': [class_weights_dict]},
    "Random Forest": {'n_estimators': [50, 100, 200],'max_depth': [10, 20, None],'min_samples_split': [5, 10],
                      'min_samples_leaf': [2, 5],'class_weight': [class_weights_dict]},
    "Gradient Boosting": {'learning_rate': [0.01, 0.1, 0.2],'n_estimators': [50, 100, 200],'max_depth': [3, 5, 7],
                          'min_samples_split': [5, 10],'min_samples_leaf': [2, 5]}
}
for model_name, model in models.items():
    print(f"Running GridSearchCV for {model_name}")
    grid_search = GridSearchCV(estimator=model, param_grid=param_grids[model_name],
                               cv=5, n_jobs=-1, verbose=2, scoring='accuracy')
    grid_search.fit(X_train, y_train)

    print(f"Best parameters for {model_name}: {grid_search.best_params_}")
    best_model = grid_search.best_estimator_
    y_pred = best_model.predict(X_valid)
    print(f"Validation Accuracy for {model_name}: {accuracy_score(y_valid, y_pred)}")
    print("-" * 50)
```

MODEL SELECTION

Data Accuracy: 0.50 ~ 0.53

Running GridSearchCV for Logistic Regression
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best parameters for Logistic Regression: {'C': 0.01, 'class_weight': {0: 0.9478335446342009, 1: 1.0582431307243962}, 'max_iter': 500, 'penalty': 'l2', 'solver': 'lbfgs'}
Validation Accuracy for Logistic Regression: 0.5136921624173749

Running GridSearchCV for Decision Tree
Fitting 5 folds for each of 27 candidates, totalling 135 fits
Best parameters for Decision Tree: {'class_weight': {0: 0.9478335446342009, 1: 1.0582431307243962}, 'max_depth': 20, 'min_samples_leaf': 10, 'min_samples_split': 10}
Validation Accuracy for Decision Tree: 0.4976392823418319

Running GridSearchCV for Random Forest
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Best parameters for Random Forest: {'class_weight': {0: 0.9478335446342009, 1: 1.0582431307243962}, 'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 50}
Validation Accuracy for Random Forest: 0.506137865911237

Running GridSearchCV for Gradient Boosting
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Best parameters for Gradient Boosting: {'learning_rate': 0.01, 'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 50}
Validation Accuracy for Gradient Boosting: 0.5278564683663833

MODEL SELECTION

Rank	Models	Accuracy Score (Validation Set)
1	Logistics Regression	0.514
2	Decision Tree	0.498
3	Random Forest	0.506
4	Gradient Boosting	0.528

OUTCOME

- **Data Accuracy:** 0.50 ~ 0.53
- **Relationship between input features and target feature**: weak or negligible
- **Areas for Improvement:**

- Add more features with significant predictive power, such as participation in class through discussion, accumulative grade, teaching quality and etc.
- Deploy more advanced machine learning models to achieve better results

THANK YOU