Microsoft Elevate

# CAPSTONE PROJECT

## AI-POWERED STUDENT PERFORMANCE WORKFLOW


**PRESENTED BY**

**STUDENT NAME: RISHIT GHOSH**

**COLLEGE NAME: GEETHANJALI COLLEGE OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT: CSE – AI&ML**

**EMAIL ID: rishitghosh06@gmail.com**

Microsoft Elevate

# OUTLINE:

- **Problem Statement** (Should not include solution)
- **Proposed System/Solution**
- **System Development Approach** (Technology Used)
- **Algorithm & Deployment**
- **Result (Output Image)**
- **Conclusion**
- **Future Scope**
- **References**

Microsoft Elevate

# PROBLEM STATEMENT:

Generally used, student performance evaluation is mostly rule-based (pass/fail decided by marks threshold).

This approach:

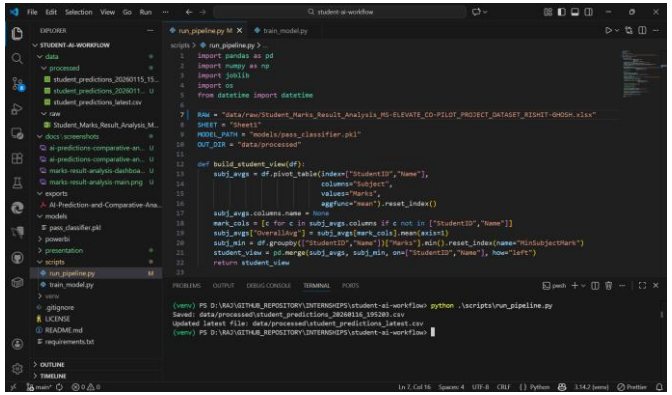
- Does not account for predictive insights
- Lacks automation in reporting
- Provides limited comparative analysis

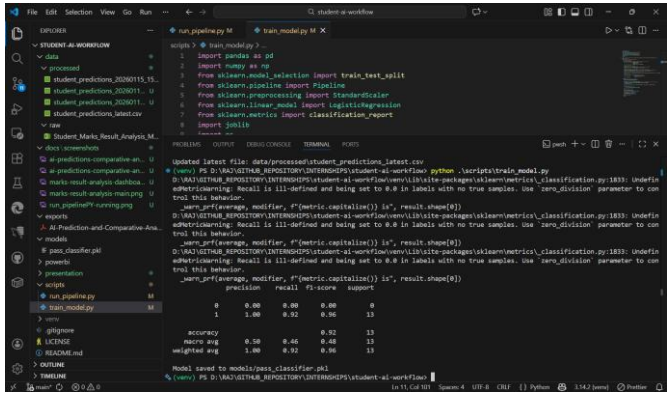
Challenge: Can we build a workflow that predicts student outcomes using AI/ML and integrates them into a dashboard for analysis?

Microsoft Elevate

# PROPOSED SOLUTION:

- **Python ML Pipeline**
  - Build predictive models to forecast pass/fail outcomes and grade distributions
  - Enable data-driven insights beyond static thresholds
- **Power BI Dashboard**
  - Compare rule-based vs AI-driven predictions
  - Provide interactive visualizations for deeper analysis
- **Copilot Integration**
  - Support in workflow design
  - Assist with repository organization and documentation
  - Enhance collaboration and productivity
- **Future Extension: Power Automate**
  - Automate scheduled data refreshes
  - Streamline report distribution to stakeholders





Microsoft Elevate

# SYSTEM APPROACH:

## ➢ System Requirements:

- **Python 3.11** → Used for data preprocessing, model training, and prediction pipeline.
- **Power BI Desktop** → Used for visualization of both rule-based and AI-based results.

## ➢ Libraries Used:

- **pandas** → Data cleaning, preprocessing, and tabular manipulation.
- **scikit-learn** → Training the ML classifier (Logistic Regression) and generating predictions.
- **joblib** → Saving and loading the trained model (*pass_classifier.pkl*) for reuse.
- **openpyxl** → Reading/writing Excel files for raw student marks dataset.

## ➢ Workflow Structure:

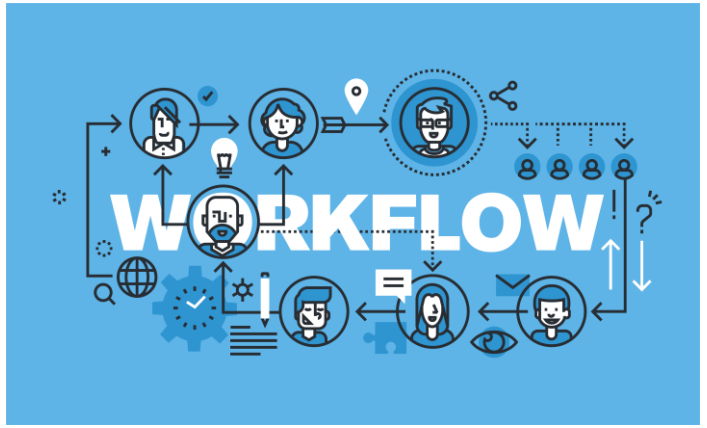
- Raw Data (Excel[*.xlsx*])
  - ❑ Student marks dataset collected in *.xlsx* format.
  - ❑ Stored in *data/raw/*.
- Processed Predictions (CSV)
  - ❑ Data cleaned and passed through ML pipeline.
  - ❑ Predictions (pass/fail, grade) exported to *data/processed/student_predictions_latest.csv*.
- Trained Model (PKL)
  - ❑ Logistic Regression model trained on historical marks.
  - ❑ Saved as *models/pass_classifier.pkl* for reproducibility.
- Dashboard Visualization (PBIX)
  - ❑ Power BI dashboard integrates both rule-based and AI predictions.

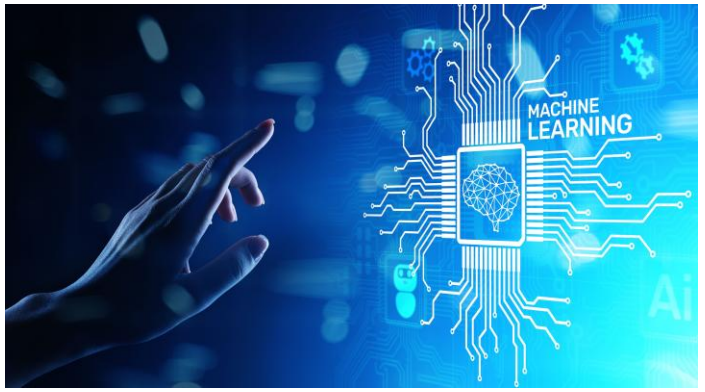
## ➢ Two pages:

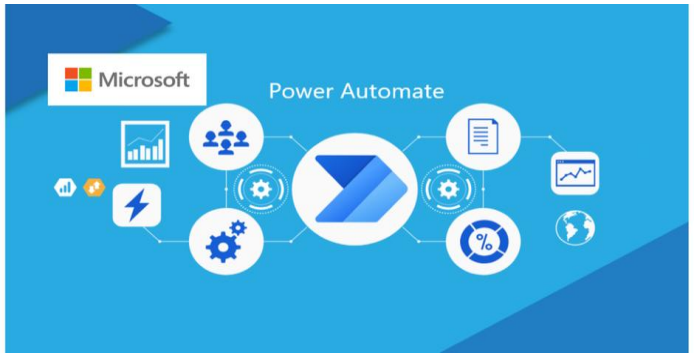
- Marks & Result Analysis (rule-based)
- AI Predictions & Comparative Analysis (ML-based)

## ➢ Final Flow:

- Raw Data → Preprocessing → ML Model Training → Predictions Export → Power BI Dashboard Visualization



# ALGORITHM & DEPLOYMENT:

## ➢ Algorithm Selection:

- The project uses a Logistic Regression classifier.
- This algorithm was chosen because it is simple, interpretable, and effective for binary classification tasks such as predicting whether a student will pass (1) or fail (0).
- Logistic Regression also provides probability scores, which can be useful for understanding confidence in predictions.

## ➢ Data Input:

- The input dataset consists of student marks stored in Excel format.
- Each record includes subject-wise marks and a rule-based pass/fail outcome.
- This dataset forms the basis for training and testing the machine learning model.

## ➢ Training Process:

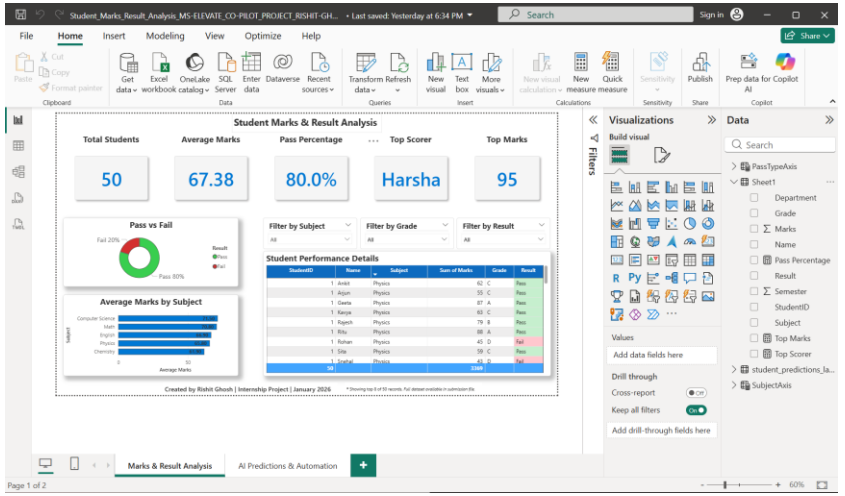
- Preprocessing → Clean the dataset, handle missing values, and prepare features.
- Model Training → Apply Logistic Regression using scikit-learn.
- Model Saving → Export the trained model as *pass_classifier.pkl* using joblib.
  - ❑ This ensures reproducibility and allows the model to be reused without retraining.

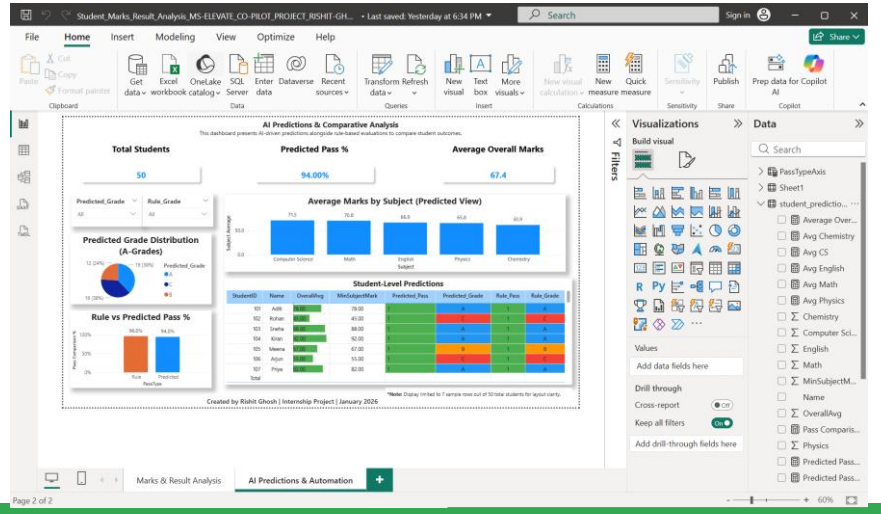
## ➢ Prediction Process:

- Pipeline Execution → Run run_pipeline.py to load the saved model.
- Generate Predictions → Predict pass/fail outcomes and assign grades (A/B/C).
- Export Results → Save predictions into *student_predictions_latest.csv*.
- Integration → Load the CSV into Power BI for visualization and comparative analysis.

## ➢ Deployment:

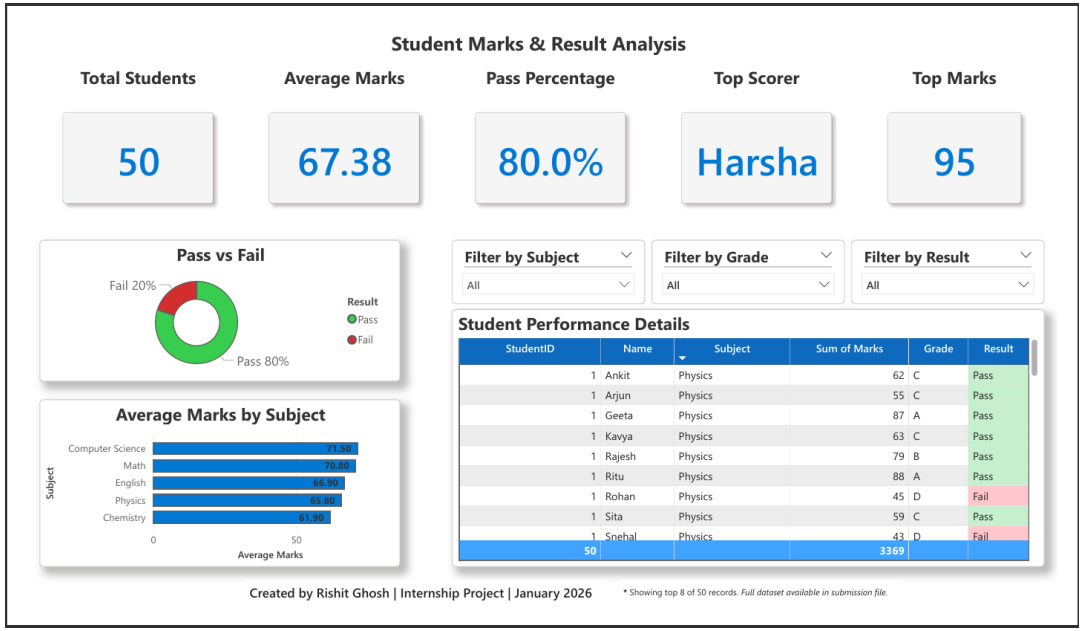
- The Power BI dashboard is refreshed with the latest predictions.
- Two pages are maintained:
  - ❑ Marks & Result Analysis → Rule-based evaluation.
  - ❑ AI Predictions & Comparative Analysis → ML-driven outcomes.
- This deployment ensures that both traditional and AI-based insights are available in a single, interactive interface.



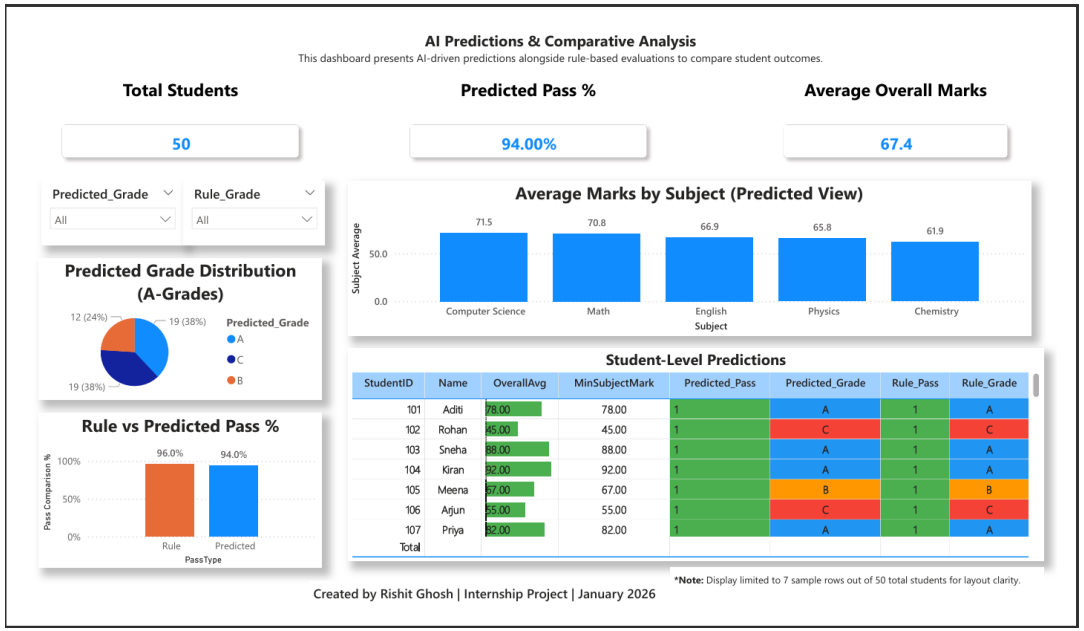



# RESULT:

The dashboards clearly show rule-based vs AI-predicted outcomes with KPIs, charts, and student-level tables. Comparative analysis highlights how the ML model aligns with traditional evaluation while adding predictive insights.




Marks & Result Analysis (Rule-based)




AI Predictions & Comparative Analysis

Microsoft Elevate

# CONCLUSION:

- Designed and implemented a modular workflow for student performance analysis using Python and Power BI.
- Built a Logistic Regression model to predict pass/fail outcomes and grades from student marks.
- Developed dashboards that provide comparative insights between rule-based evaluation and AI predictions.
- Ensured the workflow is reproducible, well-structured, and ready for extension with automation or advanced models.

Microsoft Elevate

# FUTURE SCOPE:

- Power Automate Integration → Automate report generation and scheduled dashboard refresh, reducing manual effort.
- Dataset Expansion → Include additional features such as attendance, assignments, and demographics to improve prediction accuracy.
- Advanced ML Models → Experiment with Random Forest, XGBoost, or ensemble methods for higher performance and deeper insights.
- Online Deployment → Publish dashboards via Power BI Service for accessibility across devices and real-time collaboration.
- Copilot Studio Integration → Build AI agents that can answer queries, provide insights, and interact with the dashboard dynamically.

Microsoft Elevate

# REFERENCES:

- **scikit-learn Documentation** -> https://scikit-learn.org/stable/index.html
- **Power BI Official Docs (Microsoft Learn)** → https://learn.microsoft.com/en-us/power-bi/
- **Microsoft Copilot Resources:**
  - ❑ Copilot Success Kit: https://adoption.microsoft.com/en-us/copilot/success-kit/
  - ❑ Copilot Studio Resources: https://microsoft.github.io/copilot-studio-resources/
  - ❑ Microsoft 365 Copilot Chat: https://copilot.cloud.microsoft/
- **GitHub Repository ->** https://github.com/rajghosh06-dev/student-ai-workflow/

[Click here for GitHub!].

Microsoft Elevate

# Thank You