

# Week 3 Assignment Report

## Synthetic Data Pipeline Project

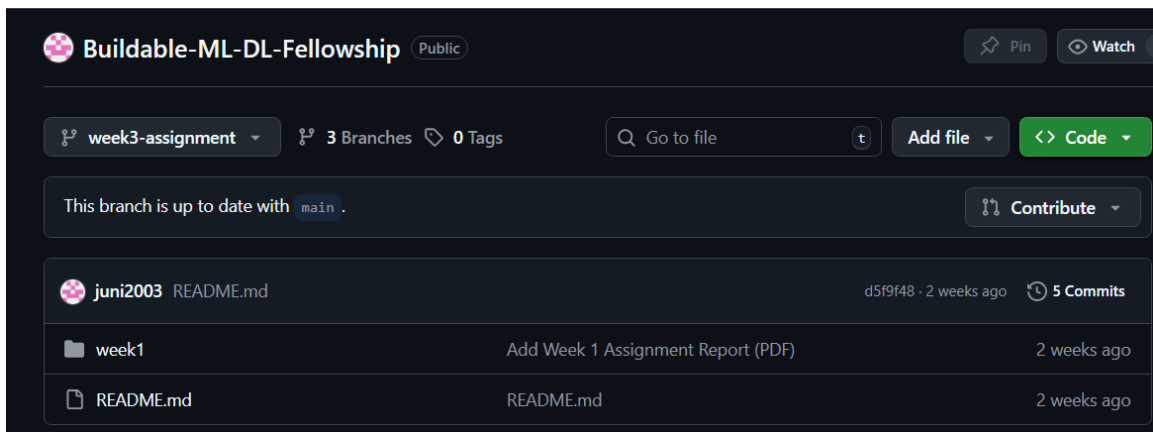
Name: Junaid Mohi Ud Din

Date: September 6, 2025

### Question 1 – Git & GitHub (Project Setup)

Include screenshots of branch creation, commits, and PR in your report.

**Screenshots:**



Buildable-ML-DL-Fellowship

Type [Z] to search

CodeIssuesPull requestsActionsProjectsWikiSecurityInsightsSettings

week3-assignment

Buildable-ML-DL-Fellowship / week4 / synthetic\_data\_pipeline\_project /

Go to file

Add file

...

juni2003

Add README for synthetic data pipeline project

f17b021 · now

History

This branch is 1 commit ahead of main.

Contribute

Name

Last commit message

Last commit date

README.md

Add README for synthetic data pipeline project

now

README.md

# Synthetic Data Pipeline Project

This project implements a complete ML pipeline for synthetic data generation, processing, and classification.

## Project Structure

- src/ - Source code modules

Buildable-ML-DL-Fellowship

Type [Z] to search

CodeIssuesPull requests1ActionsProjectsWikiSecurityInsightsSettings

## Week3 assignment #2

Merged

juni2003 merged 41 commits into main from week3-assignment 1 minute ago

Conversation 5Commits 41Checks 0Files changed 30+3,180 -0

juni2003 commented 9 minutes ago

No description provided.

juni2003 added 30 commits 5 hours ago

Add README for synthetic data pipeline project

Add .gitkeep to keep data directory in version control

Add .gitkeep file in src directory

Create .gitkeep in plots directory

Reviewers

Copilot

Still in progress? Convert to draft

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Merged

Week3 assignment #2

juni2003 merged 41 commits into main from week3-assignment 1 minute ago

juni2003 and others added 4 commits 6 minutes ago

Update week3/synthetic\_data\_pipeline\_project/src/ init\_.py

Update week3/synthetic\_data\_pipeline\_project/results/.gitkeep

Update week3/synthetic\_data\_pipeline\_project/src/data\_preparation.py

Update week3/synthetic\_data\_pipeline\_project/src/visuals.py

juni2003 merged commit 8c0ff27 into main 1 minute ago

Revert

juni2003 deleted the week3-assignment branch now

Restore branch

### Pull request successfully merged and closed

You're all set — the branch has been merged.

## **Question 2 – OOP & Reusability (Synthetic Data Generator)**

Explain how OOP and reusability make it easier to expand or modify the generator.

### **Answer:**

OOP makes our code organized and easy to work with. Instead of writing separate functions everywhere, we put everything related to data generation inside one class called DataGenerator. This keeps all our code in one place.

#### Benefits of using OOP:

1. Easy to modify: If I want to change the age range from 18-80 to 20-70, I just change one line in the class.
2. Easy to add new features: To add a new feature like "city", I just add a few lines in the generate\_dataset() method. The rest of the code stays the same.
3. Reusable: I can create multiple generator objects with different settings. For example, one for customers and another for students, just by changing the parameters.
4. No code repetition: The class methods can be used multiple times without copying code. If I need to generate data 10 times, I just call the same method 10 times.

## Question 3 – Files & Exception Handling

Show at least one screenshot of your error handling in action.

### Screenshots:

```
C:\Users\LAPTOP CLINIC\Buildable-ML-DL-Fellowship\week3\synthetic_data_pipeline_project\src>python data_generator.py
Testing Exception Handling:

❑ Valid data generation successful
Error logged to ../logs/errors.txt
❑ Caught DataGenerationError: n_samples must be positive, got -5
Error logged to ../logs/errors.txt
❑ Caught InvalidFilePathError: File must have .csv extension, got bad<file>.txt

❑ Check logs/errors.txt for error logs!

C:\Users\LAPTOP CLINIC\Buildable-ML-DL-Fellowship\week3\synthetic_data_pipeline_project\src>python test_errors.py
❑ Exception Handling Demonstration

=====
TEST 1: Invalid n_samples (string instead of int)
=====
Error logged to ../logs/errors.txt
❑ Successfully caught error: n_samples must be integer, got str

=====
TEST 2: Invalid n_samples (negative number)
=====
Error logged to ../logs/errors.txt
❑ Successfully caught error: n_samples must be positive, got -100

=====
TEST 3: Invalid file path (wrong extension)
=====
Error logged to ../logs/errors.txt
❑ Successfully caught error: File must have .csv extension, got data.txt

=====
TEST 4: Invalid file path (special characters)
=====
Error logged to ../logs/errors.txt
❑ Successfully caught error: Filename contains invalid characters: bad<file>name.csv

❑ All exception handling tests completed!
❑ Check ../logs/errors.txt for logged errors!
```

## **Question 4 – Modular Programming (Creating Your Own Package)**

Why modular programming is important when projects grow larger?

### **Answer:**

**Code Organization:** Instead of having one huge file with 1000+ lines, I can split functionality into separate files. Each module handles one specific task - stats.py only does calculations, visuals.py only does plotting.

**Easy Maintenance:** When I need to fix a bug in the histogram function, I only look in specific file.

**Team Collaboration:** Different team members can work on different modules without having to handle code from others files.

**Reusability:** I can use my stats.py module in other projects without copying all the code. Just import it and use the functions.

**Testing:** I can test each module separately. If augment.py works but visuals.py has bugs, I know exactly where the problem is.

**Scalability:** Adding new features is easy. Need a new statistical function? Just add it to stats.py. Need a new plot type? Add it to visuals.py.

**Import Control:** With the init.py file, I control what gets imported, making the package clean and professional.

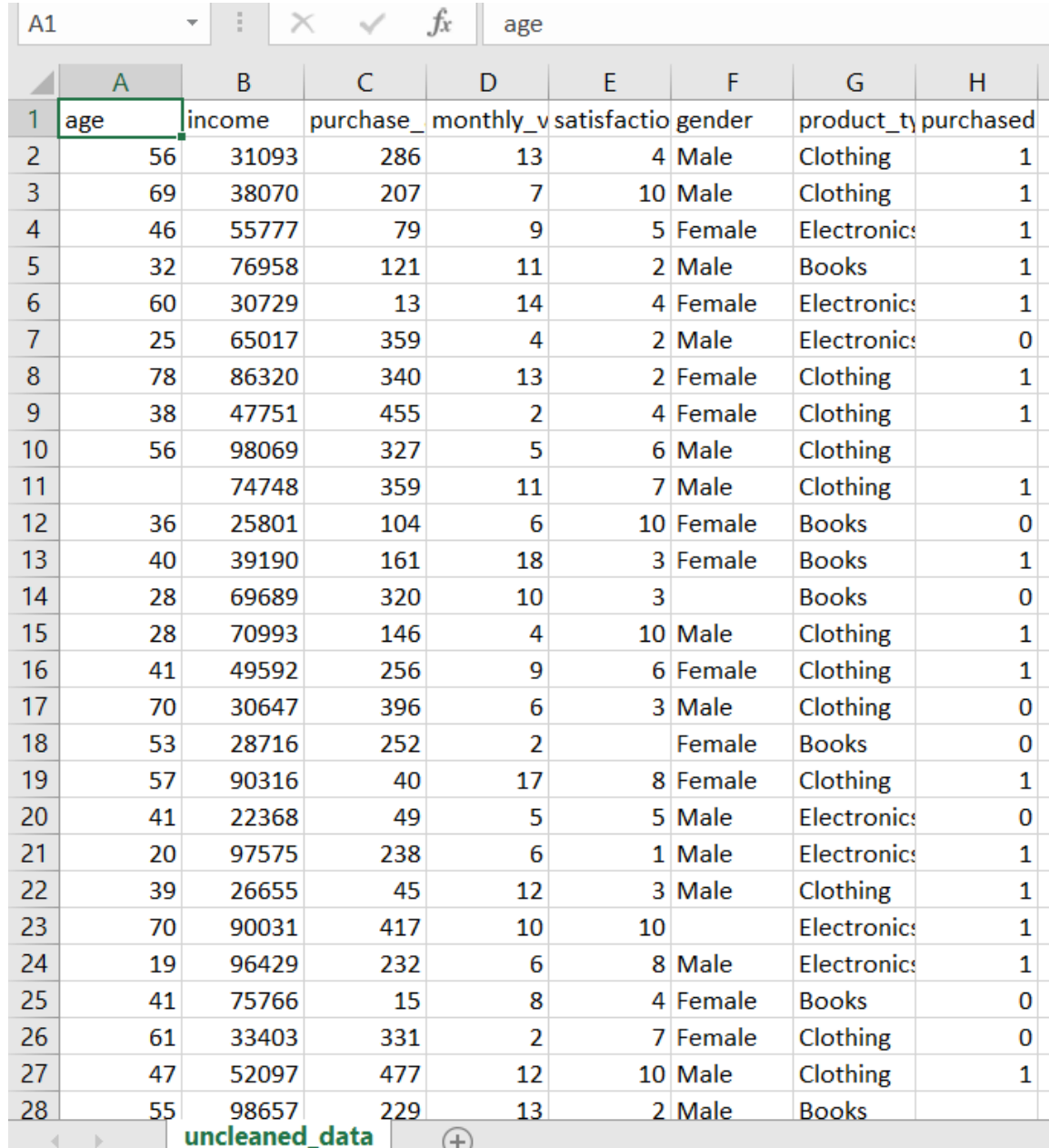
So in conclusion modular approach makes large projects manageable and professional.

## Question 5 – Data Preparation with Pandas

Include before-and-after screenshots of your dataset in your report.

### Screenshots:

Before:



	A	B	C	D	E	F	G	H
1	age	income	purchase_	monthly_v	satisfactio	gender	product_ty	purchased
2	56	31093	286	13	4	Male	Clothing	1
3	69	38070	207	7	10	Male	Clothing	1
4	46	55777	79	9	5	Female	Electronics	1
5	32	76958	121	11	2	Male	Books	1
6	60	30729	13	14	4	Female	Electronics	1
7	25	65017	359	4	2	Male	Electronics	0
8	78	86320	340	13	2	Female	Clothing	1
9	38	47751	455	2	4	Female	Clothing	1
10	56	98069	327	5	6	Male	Clothing	
11		74748	359	11	7	Male	Clothing	1
12	36	25801	104	6	10	Female	Books	0
13	40	39190	161	18	3	Female	Books	1
14	28	69689	320	10	3		Books	0
15	28	70993	146	4	10	Male	Clothing	1
16	41	49592	256	9	6	Female	Clothing	1
17	70	30647	396	6	3	Male	Clothing	0
18	53	28716	252	2		Female	Books	0
19	57	90316	40	17	8	Female	Clothing	1
20	41	22368	49	5	5	Male	Electronics	0
21	20	97575	238	6	1	Male	Electronics	1
22	39	26655	45	12	3	Male	Clothing	1
23	70	90031	417	10	10		Electronics	1
24	19	96429	232	6	8	Male	Electronics	1
25	41	75766	15	8	4	Female	Books	0
26	61	33403	331	2	7	Female	Clothing	0
27	47	52097	477	12	10	Male	Clothing	1
28	55	98657	229	13	2	Male	Books	

After:

A1				X		✓		fx		age	
	A	B	C	D	E	F	G	H			
1	age	income	purchase_	monthly_v	satisfactio	gender	product_t	purchased			
2	56	31093	286	13	4	1	1	1			
3	69	38070	207	7	10	1	1	1			
4	46	55777	79	9	5	0	2	1			
5	32	76958	121	11	2	1	0	1			
6	60	30729	13	14	4	0	2	1			
7	25	65017	359	4	2	1	2	0			
8	78	86320	340	13	2	0	1	1			
9	38	47751	455	2	4	0	1	1			
10	56	98069	327	5	6	1	1	0.618947			
11	49.75579	74748	359	11	7	1	1	1			
12	36	25801	104	6	10	0	0	0			
13	40	39190	161	18	3	0	0	1			
14	28	69689	320	10	3	0	0	0			
15	28	70993	146	4	10	1	1	1			
16	41	49592	256	9	6	0	1	1			
17	70	30647	396	6	3	1	1	0			
18	53	28716	252	2	5.425263	0	0	0			
19	57	90316	40	17	8	0	1	1			
20	41	22368	49	5	5	1	2	0			
21	20	97575	238	6	1	1	2	1			
22	39	26655	45	12	3	1	1	1			
23	70	90031	417	10	10	0	2	1			
24	19	96429	232	6	8	1	2	1			
25	41	75766	15	8	4	0	0	0			
26	61	33403	331	2	7	0	1	0			
27	47	52097	477	12	10	1	1	1			
28	55	98657	229	13	2	1	0	0.618947			
		cleaned_data									

## Question 6 – Visualization with Matplotlib

Explain what insights you can observe from these plots.

### Answer:

Age distribution: The ages are spread across a wide range (late teens up to around 80). There is no single sharp peak, but middle-aged and older groups (around 45–70) seem slightly more common than the very youngest ages. This tells me the dataset covers a mix of younger, middle, and older customers, so it is not focused on only one age group.

Gender distribution: The bar (or counts) show the two gender groups are fairly close in size, with a small difference. This is good because it means the model later will not be strongly biased toward one gender.

Correlation heatmap: All correlations are very small (close to 0). Income has the highest positive correlation with the target purchased, but even that is weak (only about 0.23). Other features like age, purchase\_amount, monthly\_visits, satisfaction\_score, and gender have almost no linear relationship with purchased. This suggests that simple linear relationships are weak.

Age vs income scatter (colored by purchase): Red (purchased) and blue (not purchased) points are mixed together across all ages and income levels. There is no clear age band where purchases dominate. However, at higher income levels there seem to be a few more red points, which matches the weak positive correlation between income and purchasing.

## Question 7 – Statistics & Augmentation with NumPy

Explain why augmentation might be useful in machine learning tasks.

### Answer:

Data augmentation helps when the dataset is small or not varied enough. By adding slightly modified copies (noise/jitter), the model sees more examples and learns to generalize better instead of memorizing. It can also reduce overfitting and sometimes helps balance distributions.



## Question 8 – Classification Model Training

Identify the best-performing model and explain why it worked better.

**Answer:**

The best model is the Random Forest.

It performed better because it achieved higher accuracy (0.84 vs 0.59), higher F1-score (0.8865 vs 0.7267), and a much better ROC-AUC (0.8989 vs 0.5697). This means it both predicted the classes more correctly overall and ranked positive cases better. Random Forest works better here because it can capture non-linear patterns and interactions between features, while Logistic Regression is limited to a mainly linear relationship. The forest's many trees reduce overfitting by averaging, so it keeps high recall (it found most of the purchasers) while still keeping good precision. That balance is why its F1 and AUC are clearly higher.

## **Question 9 – Documentation & Report (Reflections)**

### **Challenges Faced:**

- Handling missing values without corrupting the target column; accidentally imputing the target created a continuous value (0.618...) that broke classification which took some extra time to fix.
- Keeping feature encoding consistent ( like gender originally turned into True/False then I have to fixed it to 0/1)
- Making augmentation add variety without distorting labels.
- Debugging why the classifier complained about continuous labels (traceback helped identify augmentation / imputation issue).

### **Design Choices Made:**

- Saved an “uncleaned” version before cleaning to preserve a reproducible baseline.
- Used mean for numeric and mode for categorical imputation (simple and fast).
- Excluded target column from missing-value injection, imputation, and augmentation.
- Applied light Gaussian noise (small fraction of std) to duplicate rows for augmentation (kept distribution realistic).
- Used LabelEncoder / manual mapping instead of one-hot where simplicity and fewer columns were preferred.
- Picked Logistic Regression (baseline linear) and Random Forest (non-linear ensemble) for clear comparison.
- Chose F1 and ROC-AUC (along with accuracy, precision, recall) to balance class performance.

### **How to Scale the Project Further:**

- Add a quick data check before training: make sure the target is only 0/1 and there are no weird missing values sneaking in. If something's off, just stop the script.
- Put settings (like file paths, test size, model names) into a small config file so I don't keep editing the code every time.

- Try handling class imbalance later (maybe SMOTE or just `class_weight='balanced'`) if one class starts getting tiny.
- Add a couple more models (like Gradient Boosting) and do a basic grid or random search to tune stuff instead of guessing.
- Replace all the print spam with simple logging, and maybe later hook in MLflow if I feel fancy.

## Output Screenshots of python programs:

```
Dataset saved to: data\raw\test_data.csv
Valid case worked
Caught data generation error: n_samples must be positive, got -5
Caught file path error: File must have .csv extension, got invalid<file>.txt

Check logs/errors.txt for error log!
```

```
Inserting missing values...
Missing values inserted: 180
Uncleaned data saved to: data/raw/uncleaned_data.csv

Cleaning missing values...
Missing values after cleaning: 0

Encoding categorical variables...
gender encoded: {'Female': np.int64(0), 'Male': np.int64(1)}
product_type encoded: {'Books': np.int64(0), 'Clothing': np.int64(1), 'Electronics': np.int64(2)}
Cleaned data saved to: data/processed/cleaned_data.csv
```

```
Data shape: (500, 8)
Columns: ['age', 'income', 'purchase_amount', 'monthly_visits', 'satisfaction_score', 'gender', 'product_type', 'purchased']
```

	age	income	purchase_amount	monthly_visits	satisfaction_score	gender	\
0	56.0	31093.0	286.0	13.0	4.0	1	
1	69.0	38070.0	207.0	7.0	10.0	1	
2	46.0	55777.0	79.0	9.0	5.0	0	
3	32.0	76958.0	121.0	11.0	2.0	1	
4	60.0	30729.0	13.0	14.0	4.0	0	

	product_type	purchased
0	1	1.0
1	1	1.0
2	2	1.0
3	0	1.0
4	2	1.0

```
Loading data from: data/processed/augmented_data.csv
logistic_regression: acc=0.5900 prec=0.6337 rec=0.8516 f1=0.7267 auc=0.5697
random_forest: acc=0.8400 prec=0.8117 rec=0.9766 f1=0.8865 auc=0.8989
Saved model: models/logistic_regression.joblib
Saved model: models/random_forest.joblib
Saved metrics: results/metrics.csv
Best model by F1: random_forest
```