

[Open in app](#)[Sign up](#)[Sign in](#)**Medium** Search Write

INST414: Data Science Tec...

Predicting Loan Default Risk Using Machine Learning: A Simple, Data-Driven Approach



Chelsy Gomes

[Follow](#)

14 min read · Nov 28, 2025



Making good decisions with data is important in many real-world situations, especially in finance. Banks and lending companies want to understand which customers are most likely to repay their loans and which customers might fall behind. Predicting this ahead of time can help them reduce financial risk and create safer lending practices.

In this work, I explore a large dataset of loan applicants and build a supervised machine learning model to see whether we can predict who might default on their loan. Using information like income, credit score, loan amount, employment type, and other personal factors, the goal is to see how well a computer model can learn patterns that separate reliable borrowers from high-risk ones.

This Medium post explains the question I started with, the data I used, how I trained a classification model, how I measured its performance, and what limitations I discovered along the way. My aim is to keep the process easy to understand while showing how data can support an important real-world decision.

Research Question, Stakeholder, and Decision

To guide my analysis, I focused on one main question:

Can we predict whether a loan applicant will default on their loan using their financial and personal information?

This question matters to a specific stakeholder: **the loan approval and risk management team at a financial institution**. These are the people who review each application and decide whether the bank should approve the

loan, deny it, or request more documents. Their goal is to protect the bank from financial loss while still offering loans to qualified customers.

A predictive model can help this stakeholder make safer and more informed decisions. For example:

- If the model predicts **low risk**, the loan may be approved with confidence.
- If the model predicts **high risk**, the bank might ask for a co-signer, adjust the interest rate, or choose not to approve the loan at all.

Understanding these predictions directly supports a real business decision, whether to lend money to a customer and under what conditions. My analysis explores how well a supervised model can assist with this type of financial decision-making.

Description of the Data

To explore this question, I used a publicly available [Loan Default dataset](#) from Kaggle. It contains information about more than 255,000 loan applicants. Each row represents one person who applied for a loan, and each column describes something about their financial situation, personal background, or the details of the loan they requested.

The dataset includes fields such as:

	Column_name	Column_type	Data_type	Description
0	LoanID	Identifier	string	A unique identifier for each loan.
1	Age	Feature	integer	The age of the borrower.
2	Income	Feature	integer	The annual income of the borrower.
3	LoanAmount	Feature	integer	The amount of money being borrowed.
4	CreditScore	Feature	integer	The credit score of the borrower, indicating their creditworthiness.
5	MonthsEmployed	Feature	integer	The number of months the borrower has been employed.
6	NumCreditLines	Feature	integer	The number of credit lines the borrower has open.
7	InterestRate	Feature	float	The interest rate for the loan.
8	LoanTerm	Feature	integer	The term length of the loan in months.
9	DTIRatio	Feature	float	The Debt-to-Income ratio, indicating the borrower's debt compared to their income.
10	Education	Feature	string	The highest level of education attained by the borrower (PhD, Master's, Bachelor's, High School).
11	EmploymentType	Feature	string	The type of employment status of the borrower (Full-time, Part-time, Self-employed, Unemployed).
12	MaritalStatus	Feature	string	The marital status of the borrower (Single, Married, Divorced).
13	HasMortgage	Feature	string	Whether the borrower has a mortgage (Yes or No).
14	HasDependents	Feature	string	Whether the borrower has dependents (Yes or No).
15	LoanPurpose	Feature	string	The purpose of the loan (Home, Auto, Education, Business, Other).
16	HasCoSigner	Feature	string	Whether the loan has a co-signer (Yes or No).
17	Default	Target	integer	The binary target variable indicating whether the loan defaulted (1) or not (0).

Together, these fields give a good picture of each applicant's financial health and personal situation. These features are commonly used by real banks when deciding whether to approve a loan, which makes the dataset very relevant to my question.

The most important column in the dataset is the **ground-truth label** called “Default.”

This label tells us what actually happened to each borrower:

- 0 means the person **did not default** on their loan.
- 1 means the person **did default**.

Because the true outcome is already recorded, this dataset works well for supervised learning. The model can use the other fields (income, credit

score, employment type, and so on) to learn patterns that separate people who repaid their loans from those who did not.

Shape: (255347, 18)

	LoanID	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate	LoanTerm	DTIRatio	Education	EmploymentType	MaritalStatus	HasMortgage	HasDependents	LoanPurpose	HasCoSigner	Default
0	I38PQJQS96	56	85994	50587	520	80	4	15.23	36	0.44	Bachelor's	Full-time	Divorced	Yes	Yes	Other	Yes	0
1	HPSK72WA7R	69	50432	124440	458	15	1	4.81	60	0.68	Master's	Full-time	Married	No	No	Other	Yes	0
2	C1OZ6DFJBY	46	84208	129188	451	26	3	21.17	24	0.31	Master's	Unemployed	Divorced	Yes	Yes	Auto	No	1
3	V2KKSF3JUN	32	31713	44799	743	0	3	7.07	24	0.23	High School	Full-time	Married	No	No	Business	No	0
4	EY08JDHTZP	60	20437	9139	633	8	4	6.51	48	0.73	Bachelor's	Unemployed	Divorced	No	Yes	Auto	No	0

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 255347 entries, 0 to 255346
Data columns (total 18 columns):
 #  Column              Non-Null Count  Dtype
---  --
 0  LoanID              255347 non-null object
 1  Age                 255347 non-null int64
 2  Income              255347 non-null int64
 3  LoanAmount          255347 non-null int64
 4  CreditScore         255347 non-null int64
 5  MonthsEmployed      255347 non-null int64
 6  NumCreditLines       255347 non-null int64
 7  InterestRate        255347 non-null float64
 8  LoanTerm            255347 non-null int64
 9  DTIRatio            255347 non-null float64
10  Education            255347 non-null object
11  EmploymentType      255347 non-null object
12  MaritalStatus       255347 non-null object
13  HasMortgage         255347 non-null object
14  HasDependents       255347 non-null object
15  LoanPurpose         255347 non-null object
16  HasCoSigner         255347 non-null object
17  Default             255347 non-null int64
dtypes: float64(2), int64(8), object(8)
memory usage: 35.1+ MB

proportion
Default
0      0.883872
1      0.116128

dtype: float64

```

Preview of the dataset showing sample rows, all columns, and the distribution of the default label.

Overall, the dataset gives all the information needed to train a model and test whether it can predict default risk in a meaningful way.

How I Collected the Data

To start working with my analysis, I first downloaded the [Loan Default dataset](#) from Kaggle, which is a public website where many datasets are shared. The dataset was available as a single CSV file, so I saved it directly to my computer.

After downloading the file, I opened **Google Colab** and uploaded the CSV into the Colab environment. This allowed me to read the file using pandas and begin exploring the data. I didn't need to collect any extra information or combine multiple sources, the dataset already contained everything needed for my analysis.

Type of Supervision and Why I Used Classification

Once the data was loaded, the next step was to decide what type of supervised learning model to use. In supervised learning, we try to predict a specific outcome using examples where the correct answer is already known. In my dataset, the outcome I want to predict is the “**Default**” column.

This column tells us whether each borrower ended up defaulting on their loan:

- 0 means the person did **not** default
- 1 means the person **did** default

Because the label has only two possible values (0 or 1), this makes the problem a **binary classification task**. The goal is to teach the model to separate applicants who are likely to repay their loans from those who may be at higher risk.

A regression model would not make sense here because regression is used for predicting continuous numbers, like prices or amounts. My outcome is not a number that increases or decreases, it is simply a yes-or-no situation. For this reason, a **classification model** is the correct choice for this analysis.

Using classification allows the model to look at each applicant’s financial and personal information and produce a prediction about which group they belong to: “will default” or “will not default.”

Features Used for Prediction

To build my model, I used the information provided in each row of the dataset as **features**. Features are the pieces of data that help the model

understand patterns and make predictions. In this case, the features describe each loan applicant's financial background, personal situation, and the details of the loan they applied for.

I used two types of features: **numeric features** and **categorical features**.

Numeric Features

These features are numbers that describe financial or loan-related values. The numeric features I used include:

- Age
- Income
- LoanAmount
- CreditScore
- MonthsEmployed
- NumCreditLines
- InterestRate
- LoanTerm
- DTIRatio (debt-to-income ratio)

These features help the model understand important financial factors, such as how much money a person earns, how long they have been employed, how much credit they already have, and how much they want to borrow. These details are strongly related to a person's ability to repay a loan.

Categorical Features

These features describe qualities or categories instead of numbers. The categorical features I used include:

- **Education**
- **EmploymentType**
- **MaritalStatus**
- **HasMortgage**
- **HasDependents**
- **LoanPurpose**
- **HasCoSigner**

These features give more context about each borrower's personal life and financial responsibilities. For example, a co-signer might lower the risk of default, while certain employment types or loan purposes might come with different levels of stability.

Why These Features Matter

Together, these numeric and categorical features give a well-rounded picture of each borrower. Real banks use the same kinds of information to make lending decisions, so these features are useful for predicting whether someone might default on a loan. By feeding these features into the model, it can learn patterns that help separate low-risk applicants from high-risk ones.

Train the Model and Check Its Performance

After preparing my features and label, I trained a **Logistic Regression** model. This type of model works well for binary classification, which fits my goal of

predicting whether someone will default (1) or not default (0). I trained the model using 80% of the data and saved the remaining 20% for testing so I could see how well the model performs on new, unseen examples.

Get Chelsy Gomes's stories in your inbox

Join Medium for free to get updates from this writer.

Enter your email

Subscribe

☐ Remember me for faster sign in

Once the model was trained, I used the test data to make predictions and then measured how accurate those predictions were. I looked at several important evaluation scores to understand the model's performance:

Accuracy

Accuracy tells me the percentage of predictions the model got right. My model reached an accuracy of around **0.885**, which means it correctly predicted the outcome for about 88.5% of the test cases. This seems high at first, but accuracy alone can be misleading because most people in the dataset did **not** default. So the model could predict "no default" most of the time and still appear accurate.

ROC-AUC Score

To get a better sense of how well the model separates the two classes, I looked at the **AUC score**. My model had an AUC of about **0.75**, which suggests it has a moderate ability to tell high-risk borrowers apart from low-risk ones. A score closer to 1 would mean the model is very strong, while a score around 0.5 would mean it is no better than random guessing.

Classification Report

The classification report breaks down precision, recall, and F1-score for both classes. This report showed that the model performs very well for people who **do not default**, but it struggles with people who **do default**. The reason is that there are far fewer default cases in the dataset, so the model has less information to learn from.

Confusion Matrix

The confusion matrix helped me see exactly where the model makes mistakes. It showed that while the model correctly identifies most non-defaulters, it misses many actual defaulters. This matches the imbalance in the dataset, where default cases are only a small portion of the total data.

```
... Accuracy: 0.885275112590562
ROC-AUC: 0.7531082963058535
```

```
Classification report:
```

	precision	recall	f1-score	support
0	0.89	1.00	0.94	45139
1	0.61	0.03	0.06	5931
accuracy			0.89	51070
macro avg	0.75	0.52	0.50	51070
weighted avg	0.85	0.89	0.84	51070

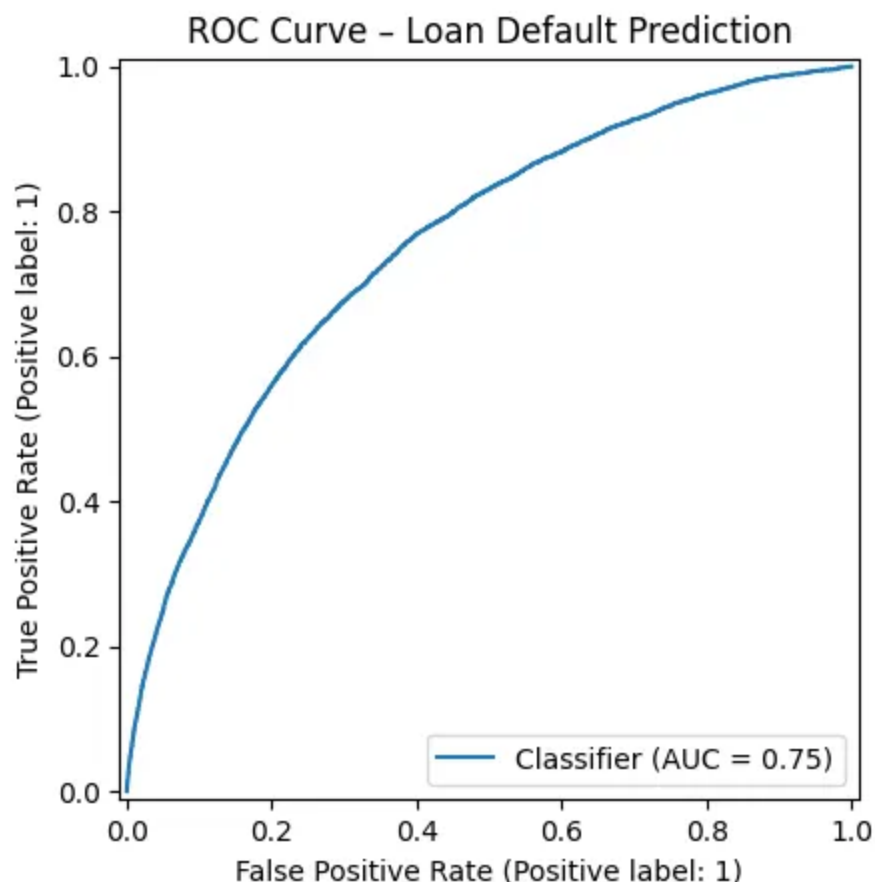
```
Confusion matrix:
```

```
[[45009  130]
 [ 5729  202]]
```

Model performance results, including accuracy, ROC-AUC, classification report, and confusion matrix.

ROC Curve

I also plotted a ROC curve, which visually shows how the model's true positive rate and false positive rate change at different thresholds. The curve helped confirm that the model is able to pick up useful patterns in the data, even if its performance on the minority class is limited.



ROC curve showing the model's ability to distinguish between defaulters and non-defaulters (AUC = 0.75).

Overall, the model performs fairly well at identifying low-risk borrowers but has more difficulty catching high-risk ones. This makes sense based on the nature of the dataset, and it is something I talk more about later in the limitations section.

Five Wrong Samples and Why the Model Misclassified Them

After checking the model's performance, I looked more closely at the cases where it made mistakes. These mistakes are important because they show the situations where the model struggles and help explain its weaknesses. I took five examples from the test set where the true outcome and the predicted outcome did not match.

Number of wrong predictions: 5859

	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate	LoanTerm	DTIRatio	Education	EmploymentType	MaritalStatus	HasMortgage	HasDependents	LoanPurpose	HasCoSigner	true_default	pred_default	proba_default
116085	32	19920	236895	558	9	2	22.66	36	0.41	High School	Part-time	Married	No	Yes	Other	No	0	1	0.574552
183752	43	82484	200209	387	49	2	19.92	12	0.77	High School	Self-employed	Divorced	Yes	No	Education	Yes	1	0	0.238188
174505	18	130606	187003	729	25	1	19.61	24	0.89	High School	Full-time	Divorced	Yes	Yes	Other	No	1	0	0.267787
137217	66	114370	21298	556	10	3	20.17	12	0.68	PhD	Part-time	Divorced	No	No	Home	Yes	1	0	0.046934
89447	29	26376	219146	518	22	1	5.56	24	0.56	PhD	Full-time	Married	Yes	Yes	Education	No	1	0	0.151332

Five misclassified examples from the test set, showing where the model's predictions did not match the true default outcomes.

When I reviewed these five samples, a few patterns stood out:

1. People with stable income but high debt

Some borrowers had a good or average income, but their **debt-to-income ratio (DTIRatio)** was very high. This means they owed a lot compared to what they earned. Even though they earned enough money, the model might have focused too heavily on income and not enough on the debt load, causing it to predict “no default” when the person actually defaulted.

2. Borrowers with good credit scores but risky loan amounts

A few people had strong credit scores but took out **large loan amounts** relative to their income. The model often sees a high credit score as a sign of low risk, so it might overlook the fact that the loan amount was unusually high for that person. This can lead to an incorrect “no default” prediction.

3. Short employment history

Some misclassified borrowers had only been employed for a short time. This can be a sign of instability or recent job changes, which may increase default risk. However, the model might not treat a short employment period as a major warning sign, especially if the person has other positive features like decent income or education level.

4. Categorical features that hide important details

Features like **LoanPurpose**, **EmploymentType**, or **MaritalStatus** are categorical. While the model uses one-hot encoding to convert these into

numbers, it still may not fully capture the deeper meaning behind these categories. For example, certain job types might be more unstable during economic shifts, but the model cannot easily understand this pattern, leading to mistakes.

5. Applicants with unusual combinations of features

Some people have a mix of traits that don't fit the typical pattern the model learned. For example, someone might have a low credit score but also a low DTIRatio, or a high loan amount but also a long employment history. These uncommon combinations can confuse the model because they don't match the patterns it saw most often during training.

Overall, these five wrong samples helped me understand that the model tends to focus heavily on the most common patterns in the dataset, especially for people who do not default. Because actual default cases are rare, the model sometimes struggles when it sees borrowers who fall somewhere in the middle, people who look safe in some ways but risky in others. These mistakes highlight the limits of the model and show why it can miss certain high-risk applicants.

Answer to the Original Question

The question I started with was:

Can we predict whether a loan applicant will default on their loan using their financial and personal information?

Based on my analysis, the answer is **yes, but with important limitations.**

The model was able to learn useful patterns from the dataset. It achieved a high overall accuracy and a solid AUC score, which shows that it can separate low-risk and higher-risk borrowers better than random guessing. This means the model does have predictive power, especially when it comes to identifying people who are likely *not* to default.

However, the model struggles much more with predicting actual default cases. Since default cases are rare in the dataset, the model doesn't see enough examples of them during training. As a result, it often predicts "no default" even when the person does end up defaulting. This makes the model less reliable for spotting high-risk borrowers, which is the part that matters most to a bank or lending company.

So overall, the model can help give a general idea of loan risk, especially for borrowers who clearly fit the pattern of being financially stable. But it should not be used on its own to make final loan decisions. A real lender would need more data, better handling of class imbalance, and possibly a more complex model to truly predict defaults with confidence.

This analysis shows that machine learning can support financial decision-making, but the predictions must be used carefully and should be combined with human judgment and additional checks.

Data Cleaning + Bugs

Before training the model, I took some time to make sure the dataset was clean and ready to use. The first thing I did was check for missing values in every column. Surprisingly, this dataset had **no missing entries at all**, so I didn't need to fill in any blanks or remove any incomplete rows. Even though everything looked clean, I still reviewed each column type to make sure

numbers were stored as numeric values and categories were stored as strings. This step helped avoid errors later when building the model.

```
Missing values per column:
Age          0
Income       0
LoanAmount   0
CreditScore  0
MonthsEmployed 0
NumCreditLines 0
InterestRate 0
LoanTerm     0
DTIRatio     0
Education    0
EmploymentType 0
MaritalStatus 0
HasMortgage  0
HasDependents 0
LoanPurpose  0
HasCoSigner  0
Default      0
dtype: int64
New shape after dropping missing: (255347, 17)
Numeric columns found: ['Age', 'Income', 'LoanAmount', 'CreditScore', 'MonthsEmployed', 'NumCreditLines', 'InterestRate', 'LoanTerm', 'DTIRatio']
Categorical columns found: ['Education', 'EmploymentType', 'MaritalStatus', 'HasMortgage', 'HasDependents', 'LoanPurpose', 'HasCoSigner']
```

Missing-value check and identification of numeric vs categorical columns during data cleaning.

Another important step was removing the **LoanID** column. This column is just an identifier and doesn't help predict whether someone will default. Keeping it would only confuse the model, so I dropped it to keep the features meaningful.

While exploring the data, I also noticed a big **class imbalance**. Most borrowers did *not* default, and only a small portion actually did. This imbalance explains why the model performs better at predicting non-defaults. Even though this wasn't a bug, it was something I needed to keep in mind when looking at the results, especially the low recall for the default class.

During the coding process, I didn't run into major bugs, but there were a few moments where I had to double-check things like column names, shapes of arrays, and outputs from certain steps. Whenever something seemed off, I went back and printed the shapes or previewed the data to make sure everything matched correctly.

Limitations + Bias

Even though the model works, there are several limitations that are important to point out. The biggest issue in this dataset is the **class imbalance**. Most people in the data did *not* default, and only a small group actually did. Because of this, the model learns to predict “no default” most of the time. This is why the recall for the default class is low — the model struggles to catch the rare cases.

Another limitation is that this dataset only includes **basic financial and demographic information**. Real loan decisions depend on much more than that. For example, the dataset does not include things like recent financial behavior, credit history length, spending patterns, job stability, or late-payment history. These missing factors could make the model less accurate because it doesn't have the full picture.

There is also a chance of **bias** built into the model, even if it was not intentional. Some features, like income, employment type, education level, and marital status, are linked to real-world inequalities. If these patterns exist in the dataset, the model might learn to treat certain groups as “more risky,” even if that isn't fair. Since the dataset comes from real financial data, it could reflect existing societal biases, which then get reproduced by the model.

Another limitation is the choice of model. **Logistic Regression** is simple and easy to explain, but it may not capture deeper or more complicated patterns in the data. More advanced models (like random forests or gradient boosting) might perform better, especially with an imbalanced dataset.

Finally, because this dataset comes from a single source, it might not represent every type of borrower. People from different countries,

backgrounds, or banking systems might behave differently, so the model may not generalize well outside this dataset.

Overall, even though the model gives useful insights, these limitations show why results should be interpreted carefully and why real financial decisions require much more information.

Conclusion

Working with this dataset helped me understand how different borrower characteristics can affect the likelihood of a loan default. By training a logistic regression model, I was able to see which features matter, how the model behaves, and where it struggles. Even though the accuracy was high, the low recall for the default class showed me that predicting rare events is much harder than predicting common ones.

This experience also showed me how important it is to look beyond accuracy. The ROC curve, confusion matrix, and the five wrong examples helped me understand the model's strengths and weaknesses in a deeper way. I also learned that even a simple model can give meaningful insights when the data is clean and organized well.

Overall, this work gave me a clearer understanding of how data science can support decision-making in real situations, like helping a lender identify which loans may carry more risk. At the same time, it reminded me that models have limits, and they should always be used carefully and responsibly. Even with these challenges, this exploration made me more confident in analyzing data, training models, and understanding what the results really mean.

GitHub Repository Link

All the code, visualizations, and model-building steps from this analysis are available in my GitHub repository. This includes the Python code used for loading the dataset, checking for missing values, preparing numeric and categorical features, building the preprocessing pipeline, training the logistic regression model, and generating evaluation outputs such as the confusion matrix and ROC curve.

You can explore the complete project here: [Loan Default Prediction](#)

References

The dataset used in this analysis comes from the [Loan Default Prediction Dataset](#) on Kaggle, which contains financial and demographic information about loan applicants. The dataset was originally created for educational and modeling purposes.

All data exploration, preprocessing, and model training were carried out in Google Colab, which provided a cloud-based environment for running Python code and generating visualizations.

Python libraries used in this analysis include Pandas and NumPy for data cleaning and manipulation, Scikit-learn for preprocessing, splitting the data, and training the logistic regression model, and Matplotlib for plotting the ROC curve. These tools made it possible to prepare the features, build the classification model, evaluate its performance, and visualize the results clearly.



Published in INST414: Data Science Techniques

247 followers · Last published 4 days ago

Follow

This publication includes student-authored articles from INST414, focusing on data science techniques in a wide range of areas



Written by Chelsy Gomes

26 followers · 2 following

Follow

Information Science major and Creative Writing minor at UMD. I code, design, and write stories that bring ideas to life ✨°

No responses yet




Write a response

What are your thoughts?

More from Chelsy Gomes and INST414: Data Science Techniques




 In INST414: Data Science Techni... by Chelsy Go...

Matters in the Gowalla Social Network? A Simple Network...

What network data reveals about influence and engagement on Gowalla.

Oct 9, 2025  6




 In INST414: Data Science Techni... by Brandon Fu...

Finding the Best and Worst Times to Travel

In the bustling heart of New York City, where the metropolitan area is home to over 20...

Feb 9, 2024




 In INST414: Data Science Techni... by Alexander K...

An Analysis of Fiction Vs. Nonfiction Book Popularity

Introduction

Mar 30, 2025



 In INST414: Data Science Techni... by Chelsy Go...

Is a Higher Degree Really Worth It?

Does more schooling really pay off? A data-driven look

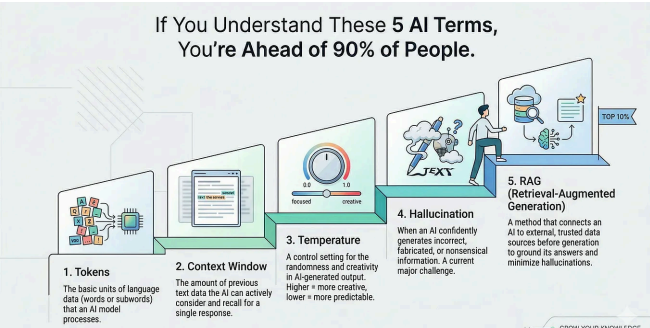
Oct 2, 2025  7



See all from Chelsy Gomes

See all from INST414: Data Science Techniques

Recommended from Medium

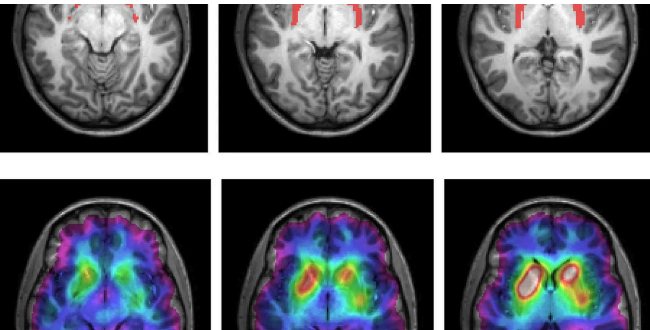



 In Towards AI by Shreyas Naphad

If You Understand These 5 AI Terms, You're Ahead of 90% of...

Master the core ideas behind AI without getting lost

★ Mar 29 🖱 9K 💬 180



 In Write A Catalyst by Dr. Patricia Schmidt

As a Neuroscientist, I Quit These 5 Morning Habits That Destroy You...

Most people do #1 within 10 minutes of waking (and it sabotages your entire day)

★ Jan 14 🖱 46K 💬 941



 In ILLUMINATION by Sufyan Maan, M.Eng

I Woke Up at 4:30 AM Every Day for 30 Days — Here Is What Nobody...

Here is what actually happened, from someone who did it & tracked everything.

★ Apr 3 🖱 7.6K 💬 329

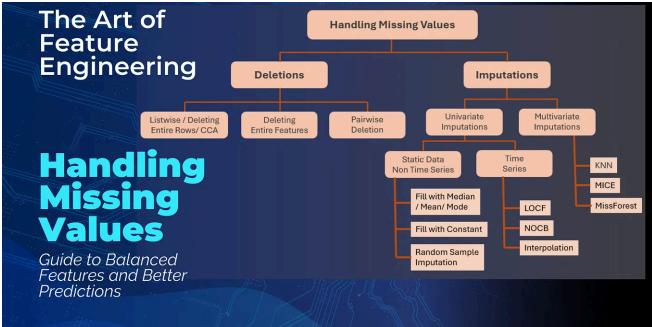


 Mihai Timoficiuc

Predicting Jet Engine Failures with NASA's C-MAPSS Dataset and...

Predictive maintenance is a maintenance strategy where you use data and machine...

Oct 11, 2025 🖱 6 💬 1



 Suparna Chowdhury

Handling Missing Data in Python: Practical Deletion and Imputation...

From simple deletions to advanced imputation, learn how to tackle MCAR, MAR,...

Nov 20, 2025

 22

 1





 Oleh Dubetcky

Difference-in-Differences: A People Analytics Case Study

Econometrics often sounds intimidating, but many of its most powerful ideas are...

Jan 21

 3



See more recommendations