Data Analytics for a Credit Score

Key Findings & Visualisations

Start Slide

01

02

Janhvi Goje,

Overview of Dataset

- Both Categorical & Continuous Column
- ‘Dirty Data’
  - Incorrect Entries - Formating issues
  - Inconsistent Values
  - Multiple NULL Values

ID	Customer_ID	Month	Name	City	Street
	CUS_0xd40		Aaron Maashoh	Lonton	Oxford Street
1603	CUS_0xd40	February	Aaron Maashoh	Lonton	Oxford Street
1604	CUS_0xd40			Lonton	Oxford Street
1605	CUS_0xd40	April	Aaron Maashoh	Lonton	Oxford Street
	CUS_0xd40	May	Aaron Maashoh	Lonton	Oxford Street
1607	CUS_0xd40	June	Aaron Maashoh	Lonton	Oxford Street
1608		July	Aaron Maashoh	Lonton	Oxford Street
1609	CUS_0xd40	August		Standhampton	Oxford Street
160E	CUS_0x21b1	January	Rick Rothackerj	Standhampton	Old Street
160F		February	Rick Rothackerj	Lonton	Old Street
1610	CUS_0x21b1	March	Rick Rothackerj	Standhampton	Old Street

Therefore, we must deal with each of these points step-by-step

Data Cleaning - Categ. & Conti. Variables

Incorrect Entries – Formating issues	• Replaced invalid SSN entries (e.g., #F%$D@*&8) with None • Stripped whitespace and special characters from all string fields • Cleaned Type_of_Loan: split strings into structured tuples • Formatted all columns with suitable datatypes • Replaced placeholder values like _ and **NM** with None before imputation
Inconsistent Data	• Identified Inconsistent data by comparing it to the Mode value (per customer) • Substituted **Values != Mode** (for consistent column) with NULL • Outliers removal
Multiple NULL Values	• Used **forward-fill and backward-fill** based on month chronology • Applied **KNN imputation** per Customer_ID for fields that don't stay consistent

Logical Imputations

Annual_Income = Monthly_Inhand_Salary x 12

```
df["Annual_Income"] = df["Monthly_Inhand_Salary"].apply(
    lambda x: x * 12 if pd.notnull(x) else None
)
```

Delay_from_due_date
Num_of_Delayed_Payment

→ *Should Negative Values be interpreted as 0 (no delays) or + (delays misentered as -)?*

	Average Credit_Score
Delay_from due date < 0	**2.5**
Num_of_Delayed_Payment < 0	**2.3**

→ ***This means that negative values should be replaced with 0***

Data Imputation - Consistent Data

```
df['Month'].head(20)
```

```
0         None
1     February
2         None
3        April
4          May
5         June
6         July
7       August
8      January
9     February
10       March
11       April
12        None
13        June
14        July
15        None
16     January
17    February
18       March
19       April
Name: Month, dtype: objec
```

Since each customer had a record per month, we filled missing months using a **rolling calendar logic—January to August.**

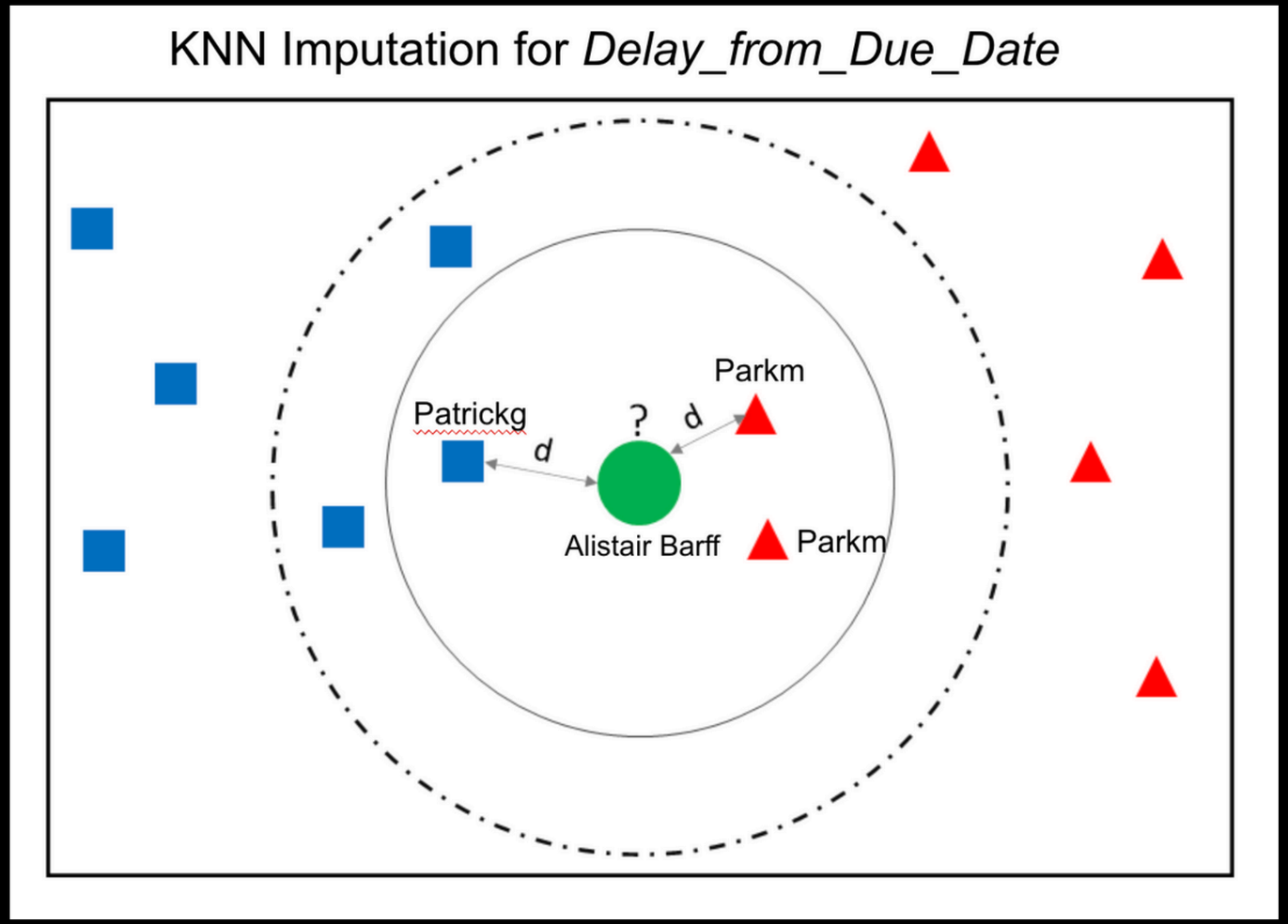
- This gave us confidence in forward/backward filling categorical values like occupation and credit mix in a time-aware and customer-specific way.
- We followed this logic for consistent columns:
  - **Month = January ? → Forward Filling**
  - **Month = August ? → Backward Filling**
  - **Else, both! (ensuring consistency)**

···	Name	Credit_Mix
28712	Charlieg	Good
28713	Charlieg	Good
28715	Charlieg	*Missing value*
28717	Charlieg	*Missing value*
28718	Charlieg	*Missing value*
28719	Charlieg	Good

	Name	Credit_Mix
28712	Charlieg	Good
28713	Charlieg	Good
28714	Charlieg	Good
28715	Charlieg	Good
28716	Charlieg	Good
28717	Charlieg	Good
28718	Charlieg	Good
28719	Charlieg	Good

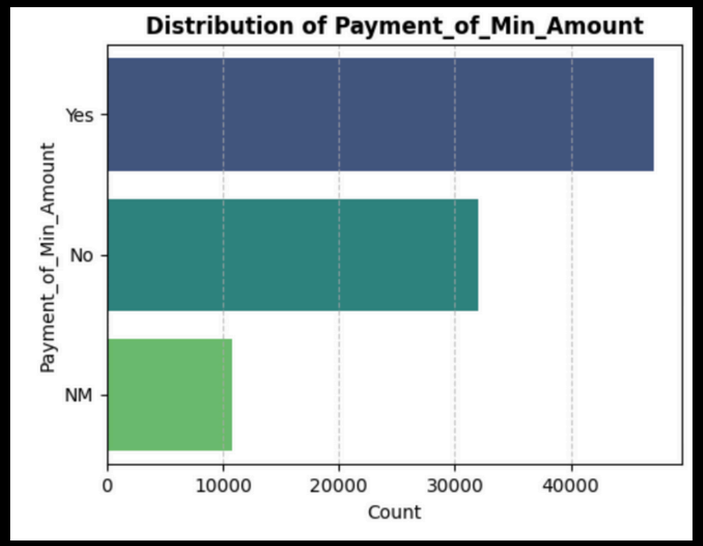
Data Imputation - Inconsistent Data

We use **KNN Imputer** to predict the missing values for inconsistent columns.

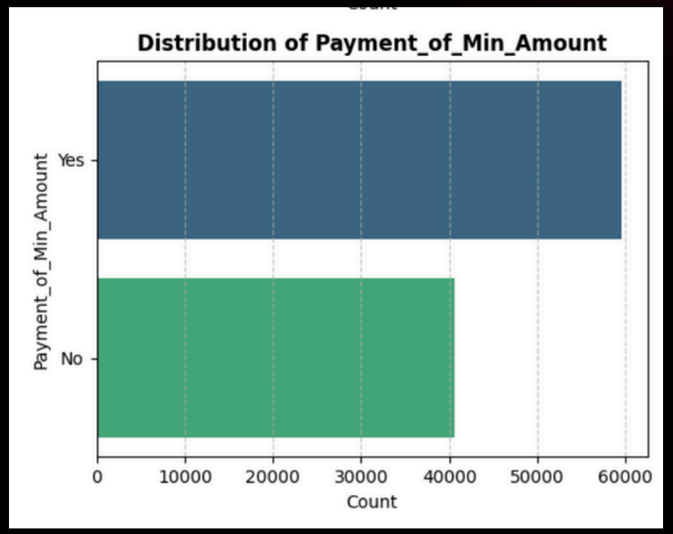


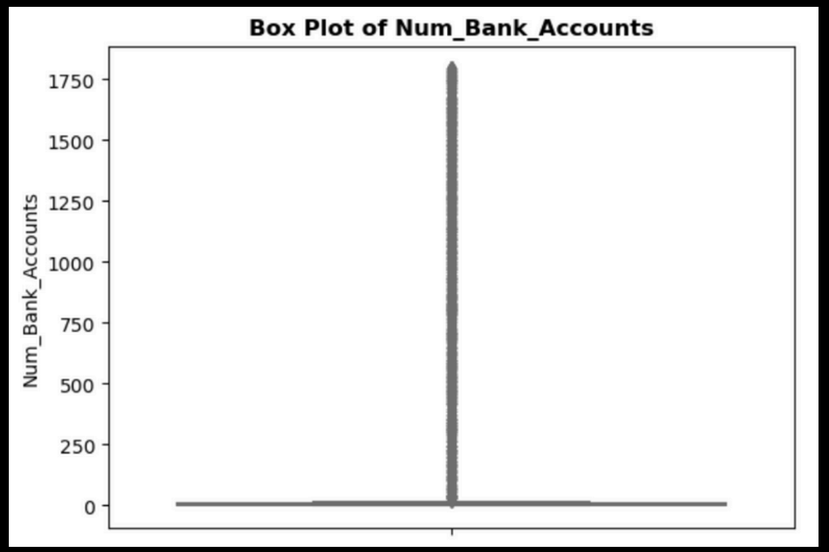
Pseudo-data for visualisation purposes, doesn't represent actual KNN Imputations.

Visualise, Clean, Visuaise

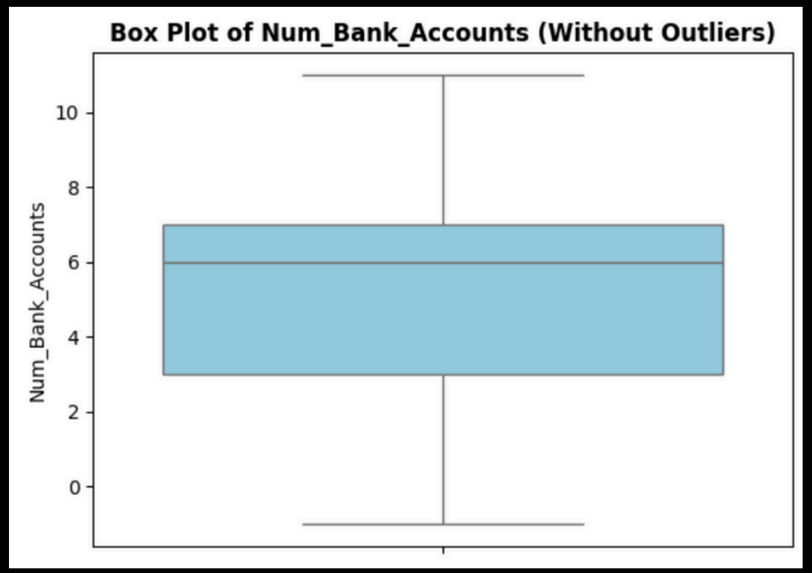


Replace NM → NULL



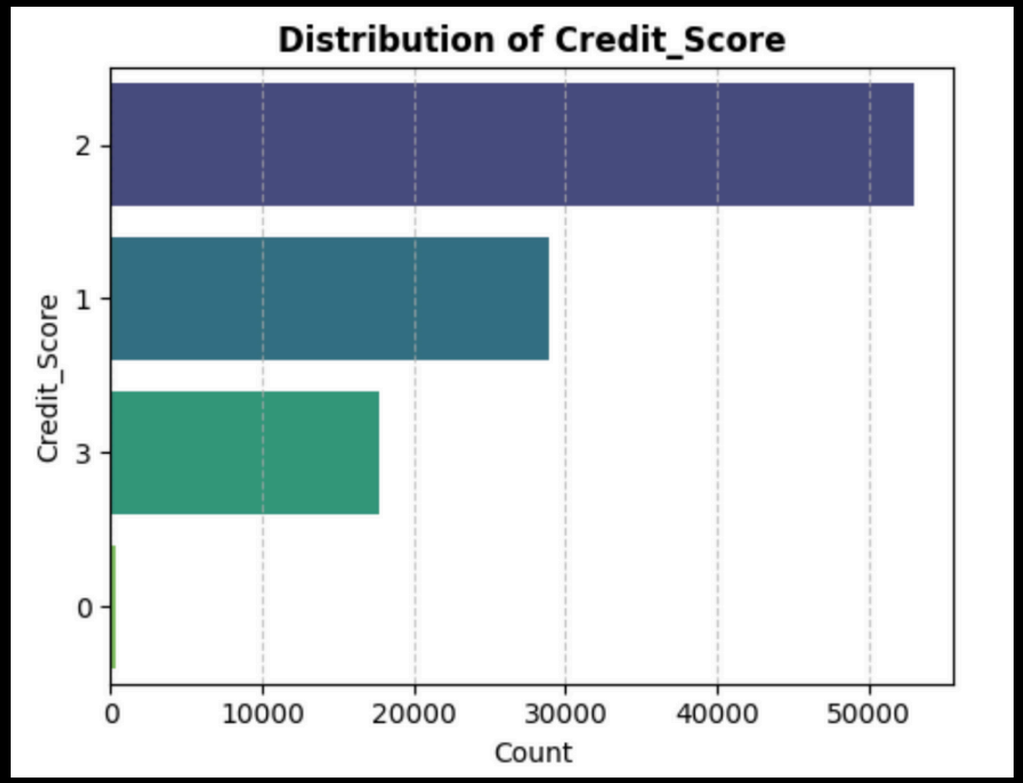


Outlier Removal

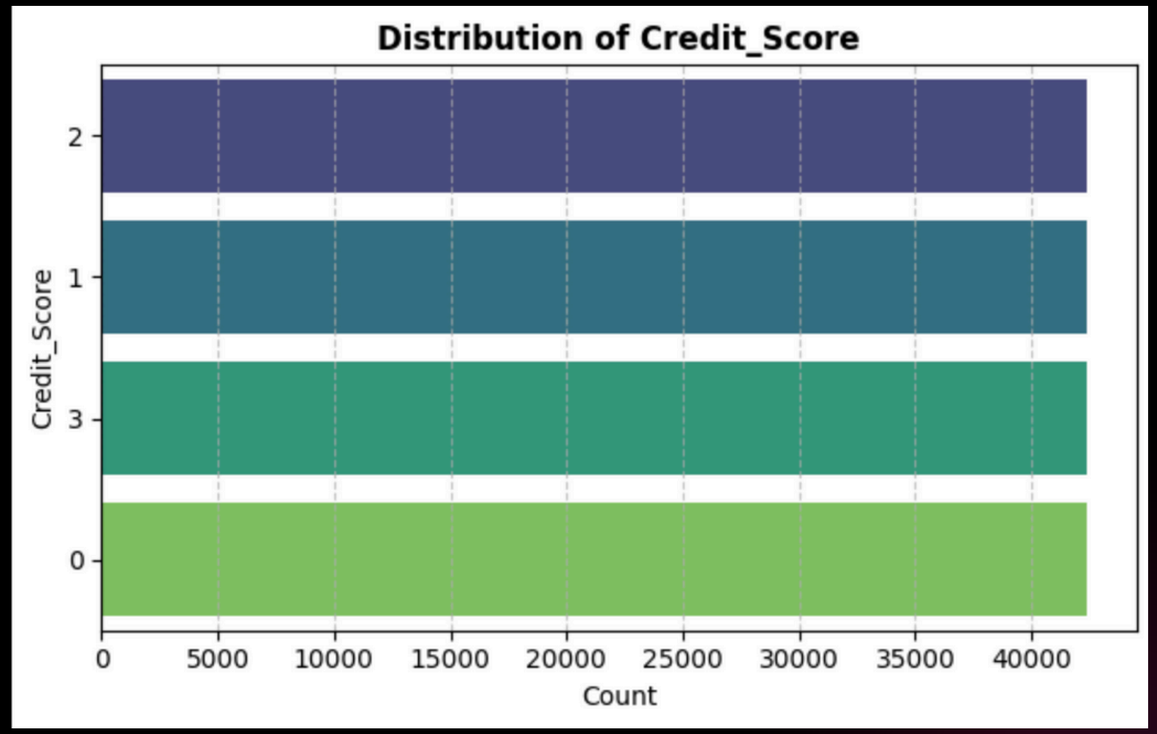


Addressing Class Imbalance

Before SMOTE



After SMOTE



Imbalanced data: "2" class dominates for Target Variable

Model Performance - Consistent & High!

	Accuracy
Logistic Regression	~ 0.98
Random Forest	~ 0.98
XGBClassifier	~ 0.98
GradientBoosting	~ 0.98

High Performance = Good Data Cleaning

High Performance != Good Model (presence of biases!)

Biases

Results from Feature Importance

	Original_Feature	Score	P_Value
4	City	70014.470896	0.000000e+00
6	Credit_Mix	13226.365952	0.000000e+00
21	Payment_of_Min_Amount	10663.080750	0.000000e+00
19	Occupation	4319.168485	0.000000e+00
10	Interest_Rate	3907.032415	0.000000e+00
23	Street	2929.628048	0.000000e+00
16	Num_Credit_Inquiries	2666.629419	0.000000e+00
20	Outstanding_Debt	2624.296968	0.000000e+00
9	Delay_from_due_date	2440.546990	0.000000e+00
18	Num_of_Loan	1920.563273	0.000000e+00
14	Num_Bank_Accounts	1732.567606	0.000000e+00

An evident Bias!

Credit_Score	0	1	2	3
City				
BadShire	4	28248	670	67
Lonton	1	53	434	17203
Standhampton	8	652	51865	450
ZeroVille	339	0	5	1

```
df[df['Credit_Score']==0][["Age", "Credit_Score"]]
```

	# Age	# Credit_Score
726	56.0	0
727	56.0	0
3545	56.0	0
3546	56.0	0

An evident Bias!

Thus, we drop ‘City’ and ‘Age’

Performance after Bias Removal

	Accuracy
Logistic Regression	~ 0.76
Random Forest	~ 0.81
XGBClassifier	~ 0.80
GradientBoosting	~ 0.78

The previous high accuracy of 98% doesn't accurately represent the models' performance on a newer, unbiased dataset and our re-trained models are likely to perform better.

Thank You

Any Questions?

Janhvi Goje,