

Week 4 - AI-Powered Data Analysis & Automation

Task 1: AI-Powered Data Cleaning and Preprocessing

Step 1: Upload the Dataset

I initiated the process by navigating to *Get Data* → *Text/Csv*, selecting my dataset, and then choosing *Load Data*. Rather than directly loading, I opted for *Transform Data* to access Power Query Editor for preprocessing.

Get Data → *Text/Csv* → *Load Data*
→ *Transform Data*

Next, I observed that the original data contained empty strings where data was missing. Power BI's automatic detection effectively handled this by replacing these empty strings with *null* values, a critical first step in standardizing missing data representation.

raw_dataset_week_4.csv

File Origin: 1252: Western European (Windows) | Delimiter: Comma | Data Type Detection: Based on first 200 rows

Date	Customer_ID	Age	Gender	Income	Spending_Score	Credit_Score	Loan_Amount	Previous_Defaults	Marketing_Sper
12-04-24	1	56	Female	142418	7	391	8083	1	
21-02-24	2	69	Male	63088	82	652	34328	2	
02-04-24	3	46	Male	136868	91	662	47891	2	
15-01-24	4	32	Female	null	34	644	25103	2	
16-04-24	5	60	Male	59811	91	469	44891	1	
12-03-24	6	25	Male	134825	17	655	15754	1	
01-03-24	7	38	Female	75479	43	490	39447	1	
21-01-24	8	56	Male	null	59	721	20901	2	
12-04-24	9	36	Male	107369	51	306	38004	1	
23-03-24	10	40	Male	137520	54	432	12446	0	
27-03-24	11	28	Female	39963	24	330	42211	0	
15-03-24	12	28	Female	75123	25	834	null	1	
15-03-24	13	41	Female	145911	71	725	31116	0	
28-03-24	14	53	Male	122468	52	693	46323	0	
26-04-24	15	57	Male	112705	70	null	23752	2	
09-04-24	16	41	Female	93477	88	736	37376	1	
13-04-24	17	20	Male	34388	33	381	21669	1	
24-01-24	18	39	Male	25569	49	655	37916	2	
03-01-24	19	19	Female	89836	29	557	null	0	
22-01-24	20	41	Female	124573	63	823	25421	0	

Extract Table Using Examples | Load | Transform Data | Cancel

A key observation was that the 'Date' column was incorrectly identified as a 'Text' data type. To rectify this, I selected the 'abc' icon next to the 'Date' column header. Using the 'Using Locale' option, I specified the correct locale to ensure proper interpretation of date formats and then changed the data type to 'Date'.

12 Date

12 Decimal Number
\$ Fixed decimal number
% Whole Number
% Percentage
Date/Time
Date
Time
Date/Time/Timezone
Duration
True/False
Binary
Using Locale...

26-04-24
09-04-24
13-04-24

4/12/2024
2/21/2024
4/2/2024
1/15/2024
4/16/2024
3/12/2024
3/1/2024
1/21/2024
4/12/2024

Query Settings

PROPERTIES

Name
Customer_Sales

All Properties

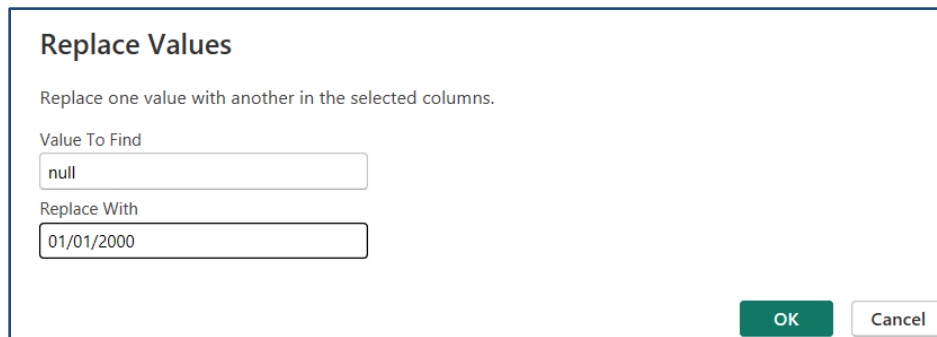
APPLIED STEPS

Source
Promoted Headers
Changed Type
X Changed Type with Locale

For better readability in PowerBI, I renamed the table to 'Customer_Sales' by typing it in the properties section in the panes. To save changes, I selected 'Close and Apply' in the top left.

Step 2: Handle Missing Values

For the 'Date' column, I replaced all null values with a predetermined default date:



Replace Values

Replace one value with another in the selected columns.

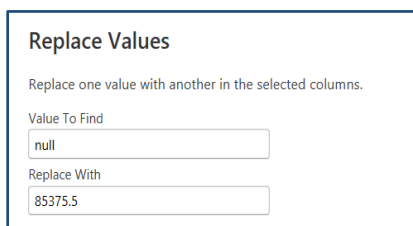
Value To Find
null

Replace With
01/01/2000

OK Cancel

To handle **null values** in the '*Income*' column, I chose a median imputation strategy, as the median is robust to outliers. My process was as follows:

- I first duplicated the 'Income' column and named the new column 'median_Income'.
- To isolate the median calculation, I then removed all other columns from this duplicated query, leaving only the 'Income' column.
- I navigated to *Transform* → *Statistics* → *Median*, which calculated the median value for the 'Income' column.
- I then copied this calculated median value.
- Finally, I returned to the original '*Customer_Sales*' query and used the 'Replace Values' function on the 'Income' column to replace all null entries with the copied median value.



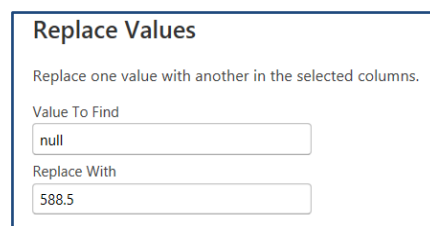
Replace Values

Replace one value with another in the selected columns.

Value To Find
null

Replace With
85375.5

Median for *Income*



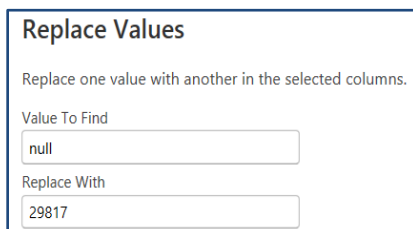
Replace Values

Replace one value with another in the selected columns.

Value To Find
null

Replace With
588.5

Median for *Credit Score*



Replace Values

Replace one value with another in the selected columns.

Value To Find
null

Replace With
29817

Median for *Loan Amount*

Step 3: Detect and Handle Outliers

```
# Remove outliers using IQR method

numeric_df = sales_data.select_dtypes(include=[np.number])
Q1 = numeric_df.quantile(0.25)
Q3 = numeric_df.quantile(0.75)
IQR = Q3 - Q1

non_outliers = ~((numeric_df < (Q1 - 1.5 * IQR)) | (numeric_df > (Q3 + 1.5 * IQR))).any(axis=1)

# Keep only non-outlier rows in the full dataset
df_cleaned = sales_data[non_outliers]
print(f'Before:', sales_data.shape, 'After:', df_cleaned.shape)
```

✓ 0.0s

Before: (500, 15) After: (405, 15)

As you can see, the output Before: (500, 15) After: (405, 15) clearly shows that the original **sales_data** had 500 rows and 15 columns. After applying the IQR method for outlier removal, the **df_cleaned** DataFrame now contains 405 rows and 15 columns. This means that 95 rows (500 - 405), which were identified as containing outliers in at least one of their numeric columns, were successfully removed from the dataset. This step is crucial for improving the robustness and accuracy of subsequent data analysis and model training by mitigating the influence of extreme values.

Step 4: Save the Cleaned Data

To save the transformed data, I used the EVALUATE Customer_Sales DAX function in POWERBI that allows me to view the table and copy it to excel.

Task 2: AI-Powered Data Visualisation and Storytelling

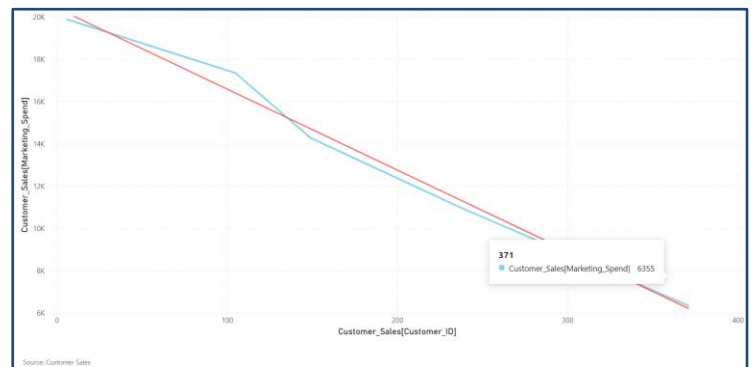
Step 2: Generate AI-Powered Visual Insights

I imported my transformed *Customer Sales* file into the PowerBI browser. Next, I used the 'Get Insights' options to Automatically build the visualizations.

- **Trends Over Time**

Marketing Spend Over Time Based on Customer ID

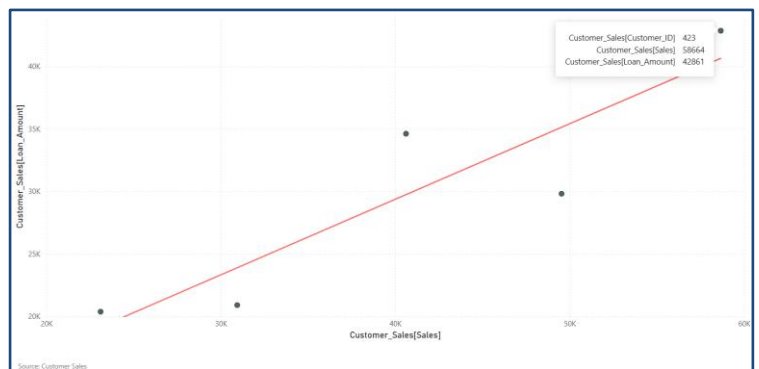
The line chart illustrates the relationship between *Marketing Spend* and *Customer ID* over time. The trend shows that there is a strong negative correlation as the customer id increases (associated with different customers) on the date 3/12/2024.



- **Correlation between Variables**

Correlation between Sales and Loan Amount Over Time

This scatter plot demonstrates a strong positive correlation between Sales and *Loan Amount*. The upward-sloping trend line (in red) clearly indicates that as sales increase, the associated loan amount also tends to increase. For instance, the highlighted data point shows that for a *Customer ID* of 423, Sales are 58664, and the corresponding *Loan Amount* is 42861, which aligns with the observed positive trend. This suggests that **higher sales figures are generally accompanied by larger loan amounts**.



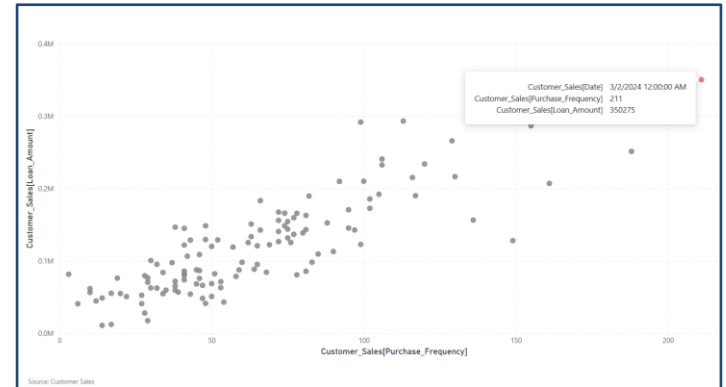
- **Anomalies/Outliers in Data**

Loan Behavior and Purchase Trends with Notable Anomaly on March 2, 2024

This scatter plot visualizes the relationship between *Loan Amount* and *Purchase Frequency*.

The general trend in the data suggests a **positive correlation**: as the purchase frequency increases, the loan amount also tends to increase. This indicates that customers who make more frequent purchases are often

associated with larger loan amounts. The data points are somewhat spread out, but the overall cluster of points moves upwards and to the right.

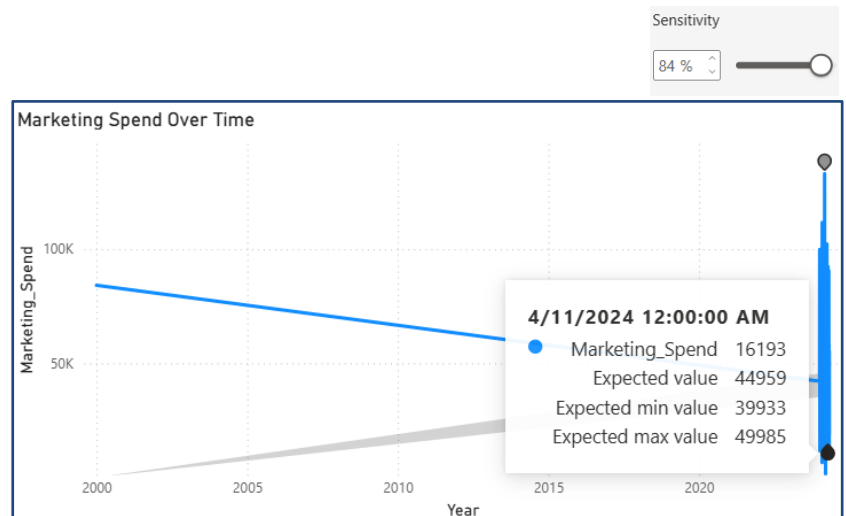


A notable **anomaly** is highlighted for March 2, 2024, where a single data point stands out from the main cluster. On this date, the *Purchase Frequency* was 211, which is among the highest frequencies observed in the dataset. Correspondingly, the *Loan Amount* for this entry was 350275, which is also an exceptionally high loan amount compared to most other data points.

Step 3: Use AI Features for Deeper Insights

➤ Anomaly Detection

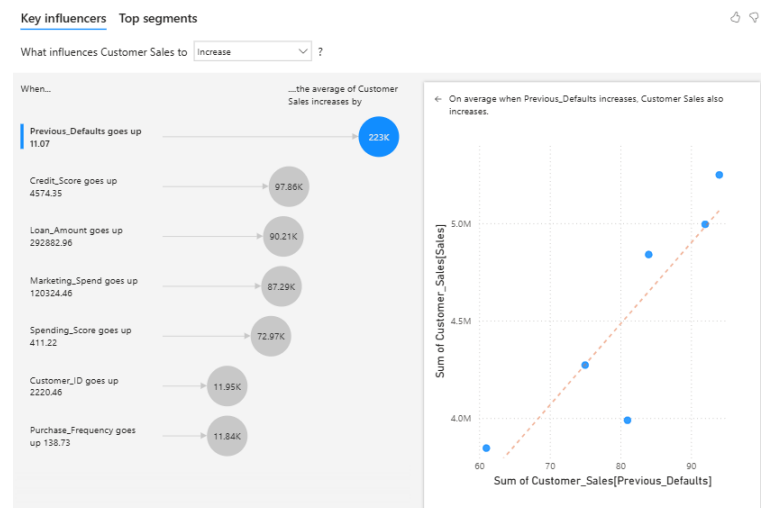
This line chart displays the trend of marketing expenditure from around 2000 to the present, with a projection. *Marketing Spend* was unexpectedly low on Thursday, April 11, 2024. It had a value of 16193, which is outside the expected range of 39933 – 49985.



➤ Key Influencers

The most significant factor influencing an increase in Customer Sales is *Previous Default* going up by 11.07, which leads to a substantial increase in the average of *Customer Sales* of 223K. This relationship is further illustrated in the scatter plot on the right, which shows a positive correlation between *Previous Defaults* and *Customer Sales*.

As the number of previous defaults increases, the customer sales also tend to increase. Whereas *Purchase Frequency* on the other hand has the least influence on *Sales*, increasing the average *Customer Sales* of 11.84K.



Task 3: AI-Driven Predictive and Prescriptive Analytics

Step 1: Train a Predictive Model

Step 2: Evaluate Model Performance

```
# Prepare features and target
# One-hot encode 'Seasonality'
df_encoded = pd.get_dummies(df_cleaned, columns=['Seasonality'], drop_first=True)

# Select features and target
X = df_encoded[['Marketing_Spend']] + [col for col in df_encoded.columns if col.startswith('Seasonality')]
y = df_encoded['Sales']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Used StandardScaler to scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Linear Regression Model
lr = LinearRegression()
lr.fit(X_train_scaled, y_train)

# Predict
y_pred = lr.predict(X_test_scaled)

# Evaluate
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"RMSE: {rmse:.2f}")
print(f"R^2 Score: {r2_score(y_test, y_pred):.2f}")

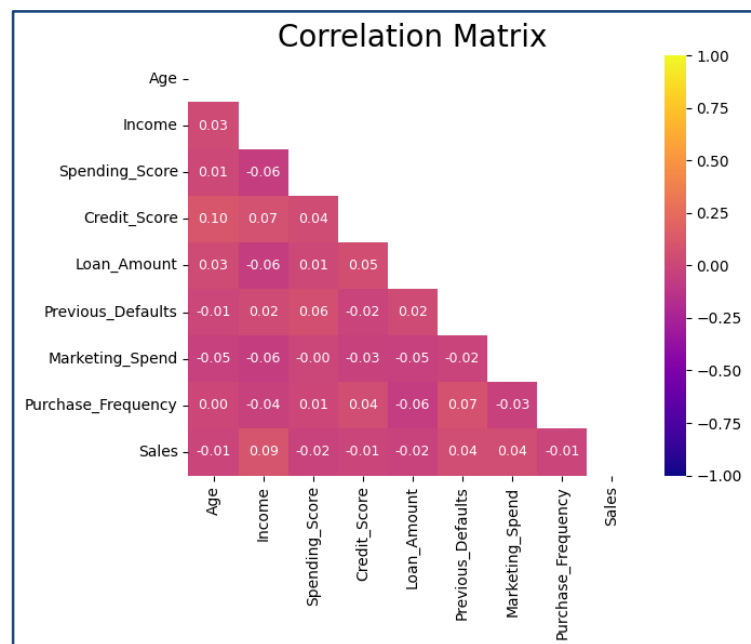
print(f"Regression Co-efficients: {lr.coef_}, lr.intercept_")
✓ 0.0s
RMSE: 167.82
R^2 Score: 0.00
Regression Co-efficients: [1345.79972656 -718.78510476 714.14399263] 53286.052959501554
```

I trained a **Linear Regression** model with '*Marketing_Spend*' and one-hot encoded '*Seasonality*' as features, aiming to predict '*Sales*'. After scaling the features using StandardScaler and splitting the data into training (80%) and testing (20%) sets, the model was fitted to the training data.

While the **RMSE** of 167.82 *looks* good in isolation when compared to the mean sales [53686.83], the **R² score of 0.00** is the **dominant indicator** here.

This shows that despite the small absolute average error, your Linear Regression model (with the current features) is not effectively learning the patterns in your data to explain 'Sales' variation. The low RMSE likely just reflects that the mean of 'Sales' is a good central tendency which is insignificant considering a 0 R² score

The correlation matrix reveals that most numerical variables in the dataset exhibit **very weak linear relationships** with each other, as indicated by correlation coefficients consistently close to zero. For instance, '*Sales*' show only negligible linear correlation with '*Marketing_Spend*' (-0.04) and 'Income' (0.09), and similarly weak or negligible relationships with all other features. This may explain the lack of variance explained by the linear regression model.



I also trained a **Random Forest Classifier** to predict '*Customer_Churn*' using '*Marketing_Spend*', '*Purchase_Frequency*', and one-hot encoded '*Seasonality*' as features. To address the class imbalance, I used *stratify=y* during the train-test split (25% test size default) and applied **SMOTE** to oversample the minority class in the training data.

Random Forest Accuracy: 0.536

Classification Report:				
	precision	recall	f1-score	support
0	0.73	0.60	0.66	93
1	0.23	0.34	0.28	32
accuracy			0.54	125
macro avg	0.48	0.47	0.47	125
weighted avg	0.60	0.54	0.56	125

```
# Random Forest Classifier to Customer Churn Prediction

# Encode target variable
le = LabelEncoder()
sales_data['Customer_Churn'] = le.fit_transform(sales_data['Customer_Churn']) # 'Yes'=1, 'No'=0

# One-hot encode 'Seasonality'
df_encoded = pd.get_dummies(sales_data, columns=['Seasonality'], drop_first=True)

# Select features and target
feature_cols = ['Marketing_Spend', 'Purchase_Frequency'] + \
               [col for col in df_encoded.columns if col.startswith('Seasonality_')]

X = df_encoded[feature_cols]
y = df_encoded['Customer_Churn']

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=42)

# Balance with SMOTE
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X_train, y_train)

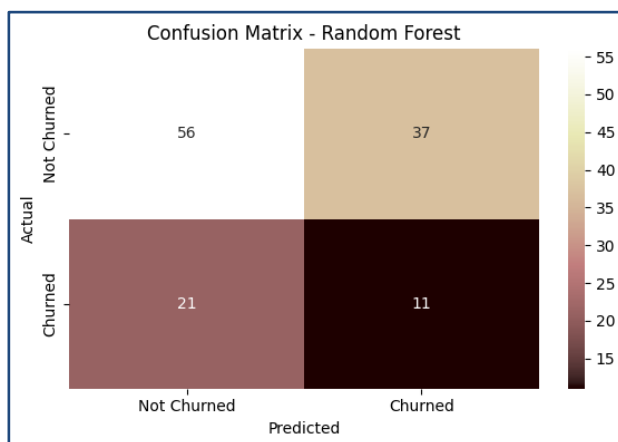
# Train model
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_res, y_res)

# Predict
y_pred_rf = rf.predict(X_test)
y_proba_rf = rf.predict_proba(X_test)[:, 1]

# Evaluate
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Classification Report:\n", classification_report(y_test, y_pred_rf))
```

The overall accuracy of 53.6% suggests that the model correctly predicts churned status for slightly more than half test instances. But

more important are the classification scores f1, precision, and recall. It is evident that the model is 135.7% (considering f1 scores) more effective at predicting customers who continue using the services [not churned] than those who stop using it [churned].



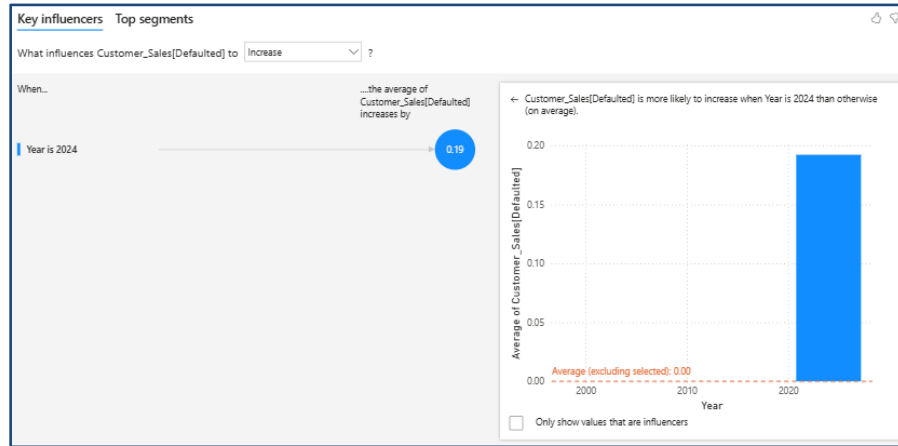
The **Confusion Matrix** confirms the output:

- It correctly identifies **56** customers as 'Not Churned' (True Negatives).
- It correctly identifies only **11** customers as 'Churned' (True Positives).

- Essentially, it **wrongly predicts 37 customers as 'Not Churned' when they did churn** (these are False Negative).
- It also **wrongly predicts 21 customers as 'Churned' when they did not churn** (these are False Positives).

Task 4: AI for Business Strategy and Risk Management

Step 1: Predict Loan Default Risk Using Power BI



This PowerBI visual strongly indicates that **the year 2024 is a significant factor contributing to an increase in customer defaults metric**). The average value of Defaulted jumps from effectively zero in other years to 0.19 specifically in 2024. This highlights 2024 as a period of elevated default risk, warranting further investigation into *why* defaults are higher in this particular year.

Step 2: Predict Risk with Python (Optional)

I also trained a Random Forest Classifier to predict 'Defaulted' status, building upon the sales_data dataset. The features used for this model were *'Income'*, *'Loan_Amount'*, and *'Credit_Score'*. The target variable 'Defaulted' was assumed to be numerically encoded (e.g., 0 for 'No Default', 1 for 'Default').

Again, the scores for precision, recall, and f1-score are higher for not defaulted [0] than the other, suggesting that the model accurately predicts not defaulted more effectively.

```
# Random Forest Classifier to predict default risk

# Encode target variable
le = LabelEncoder()
sales_data['Defaulted'] = le.fit_transform(sales_data['Defaulted']) # 'Yes'=1, 'No'=0

X = sales_data[['Income', 'Loan_Amount', 'Credit_Score']]
y = sales_data['Defaulted']

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=42)

# Apply SMOTE on training data
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X_train, y_train)

# Train Random Forest on balanced data
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_res, y_res)

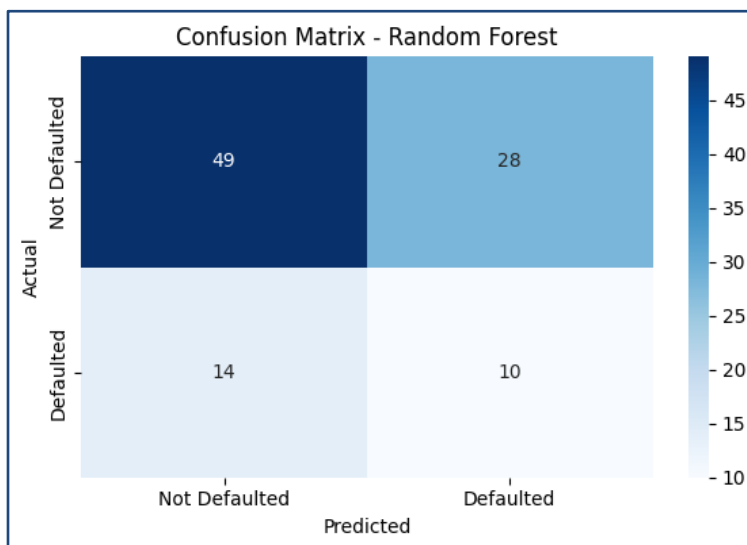
# Predict & Evaluate
y_pred_rf = rf.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Classification Report:\n", classification_report(y_test, y_pred_rf))
```

✓ 0.2s

Random Forest Accuracy: 0.648

Classification Report:

	precision	recall	f1-score	support
0	0.81	0.73	0.77	101
1	0.21	0.29	0.24	24
accuracy			0.65	125
macro avg	0.51	0.51	0.51	125
weighted avg	0.70	0.65	0.67	125



Conversely, the very low precision (0.21) and recall (0.29) for 'Defaulted' (Class 1) are clearly reflected by:

- The **high number of False Positives (28)**, indicating that many customers were wrongly flagged as defaulters.
- The **significant number of False Negatives (14)**, which represents actual defaulters that the model failed to detect.
- The **low True Positive count (10)**, meaning the model correctly identified only a small fraction of actual defaulters.

The **Confusion Matrix** further confirms the claims:

➤ The **high precision (0.81) and recall (0.73) for 'Not Defaulted' (Class 0)** are evident in the confusion matrix's **high True Negative count (49)** and relatively lower False Negative count (14 for the other class). The **model excels at identifying those customers who won't default.**

Business Recommendations

- While the model predicted slightly more than half of the test instances, the prediction was still weak, possibly due to lack of linear correlation as confirmed by the correlation heatmap. To overcome this, **more impactful features need to be assessed.**
- Urgently **investigate the surge in default risk identified for 2024.**

Re-evaluate and potentially tighten loan approval criteria for specific demographics or loan types that show higher default rates in this period. Look into intervention strategies such as early reminders or financial counselling for customers to not run into default.

For customers who do not default, offer more favorable loan terms.

Simultaneously, refine the default prediction model to significantly increase its ability to correctly identify actual defaulters while managing false alarms, crucial for minimizing financial losses.