LOAN DEFAULT DATA PIPELINE

DATA

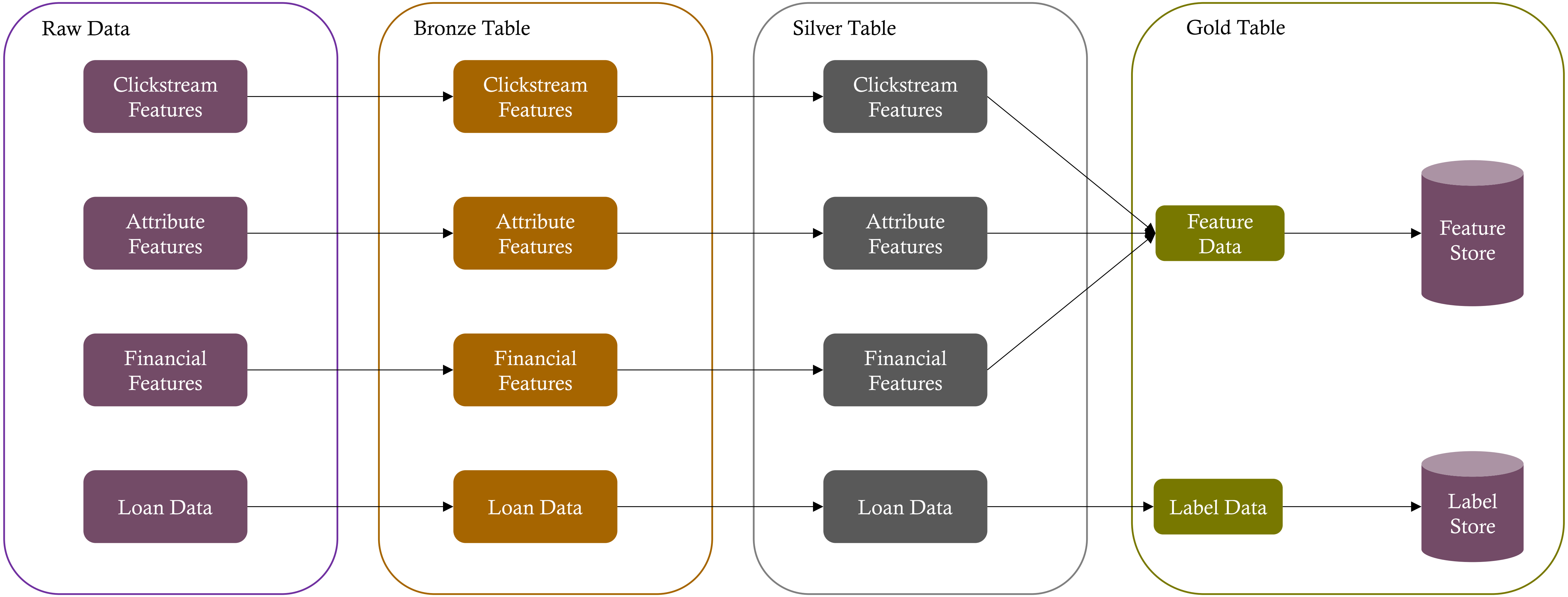
Inputs

- Clickstream Data
  - App user data
- Attribute Data
  - Job and Age Data
- Financial Data
  - Number and Types of Loans
  - Salary and Investment Data
  - Credit Score and Payment Behaviours
- Loan Data
  - Payments schedule and Amounts overdue
  - Instalments Number and amount

Outputs

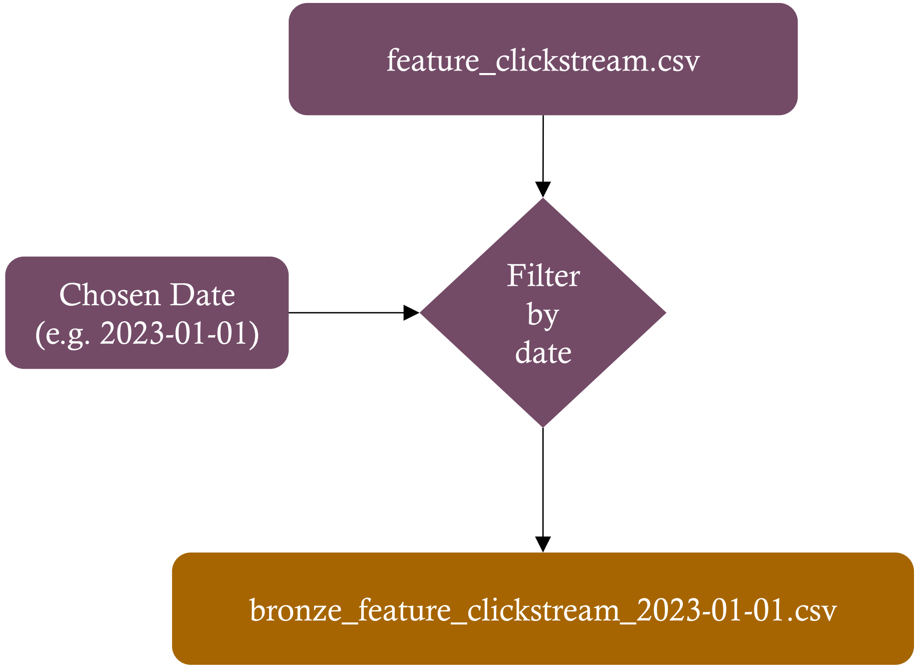
- Feature Store Containing all necessary features
- Label Store
  - Label classified as whether payments are 30 days past due as defaulters

DATA PIPELINE



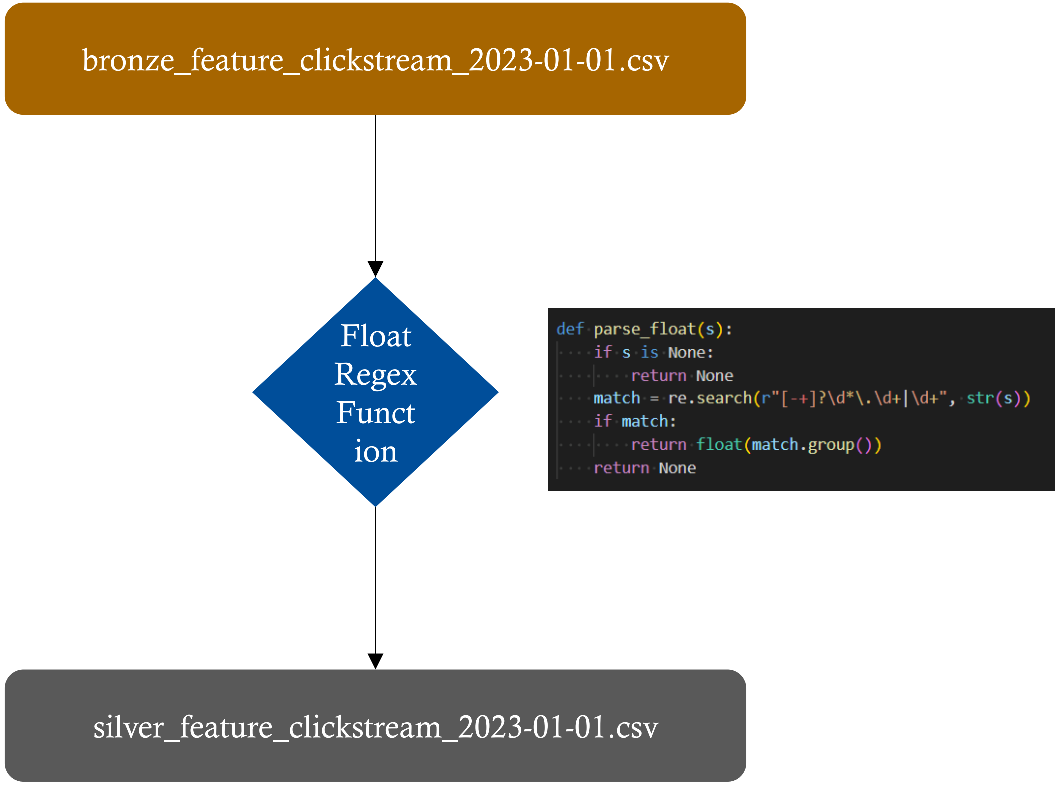
RAW DATA TO BRONZE TRANSFORMATION(ALL)

For bronze table, all data is filtered by snapshot date, where the first day of each month is chosen and the raw data is split by date



BRONZE TO SILVER TRANSFORMATION (CLICKSTREAM)

- For clickstream data, all values are seemingly floats
- To ensure that all values in the data are floats, a regex function was run on all the feature columns to ensure the output was a float

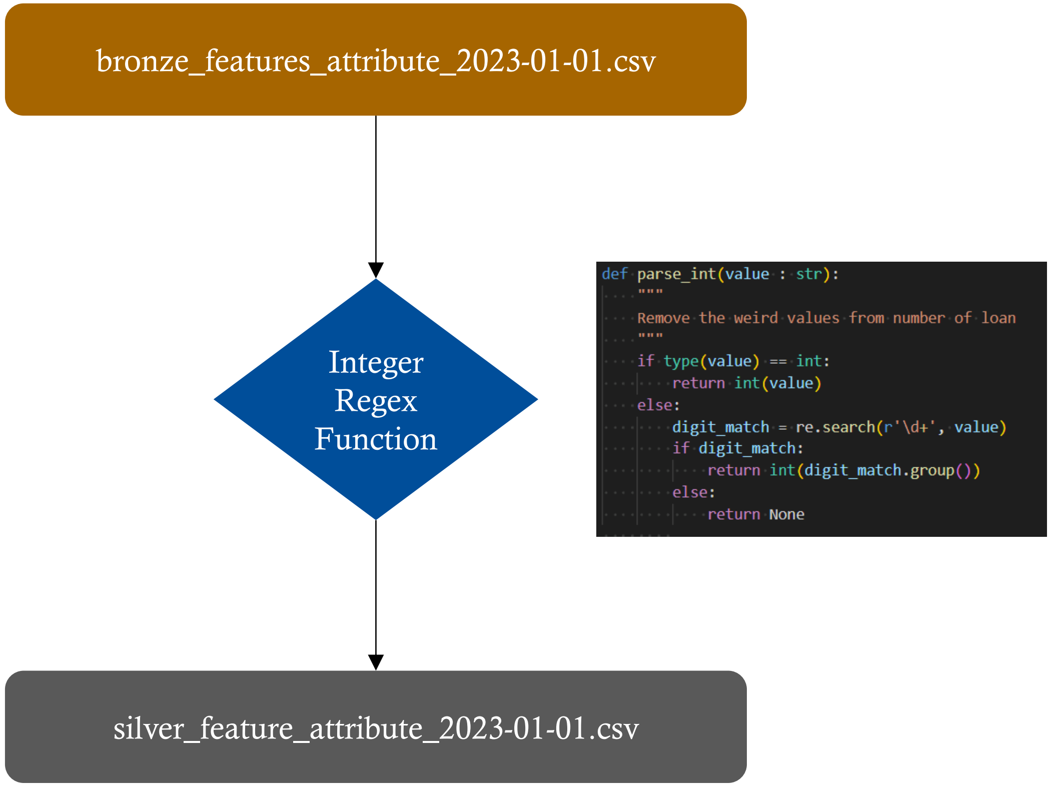


bronze_feature_clickstream_2023-01-01.csv
Float Regex Funct ion
silver_feature_clickstream_2023-01-01.csv

```python
def parse_float(s):
    if s is None:
        return None
    match = re.search(r"[-+]?\d*\.\d+|\d+", str(s))
    if match:
        return float(match.group())
    return None
```

BRONZE TO SILVER TRANSFORMATION (ATTRIBUTES)

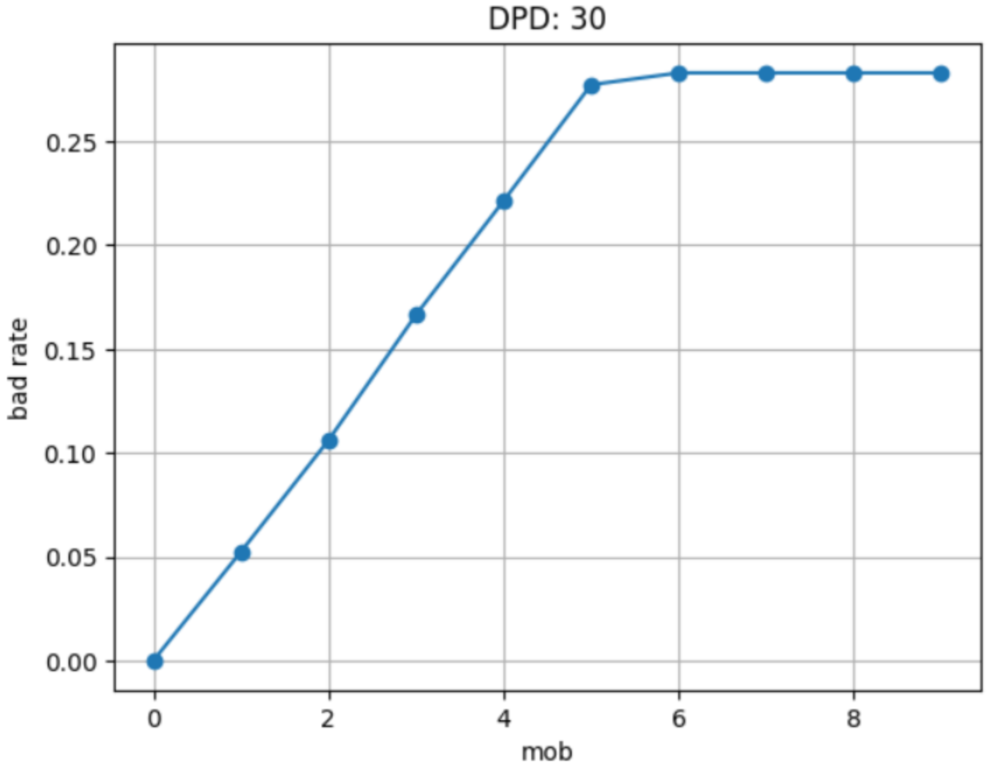
- For attribute data, some people do not have jobs listed, so value was changed to “Unknown”
- The age data is also unclean, so we needed to parse the integer data using regex as well


bronze_features_attribute_2023-01-01.csv
Integer Regex Function
silver_feature_attribute_2023-01-01.csv

```python
def parse_int(value : str):
    """
    Remove the weird values from number of loan
    """
    if type(value) == int:
        return int(value)
    else:
        digit_match = re.search(r'\d+', value)
        if digit_match:
            return int(digit_match.group())
        else:
            return None
```

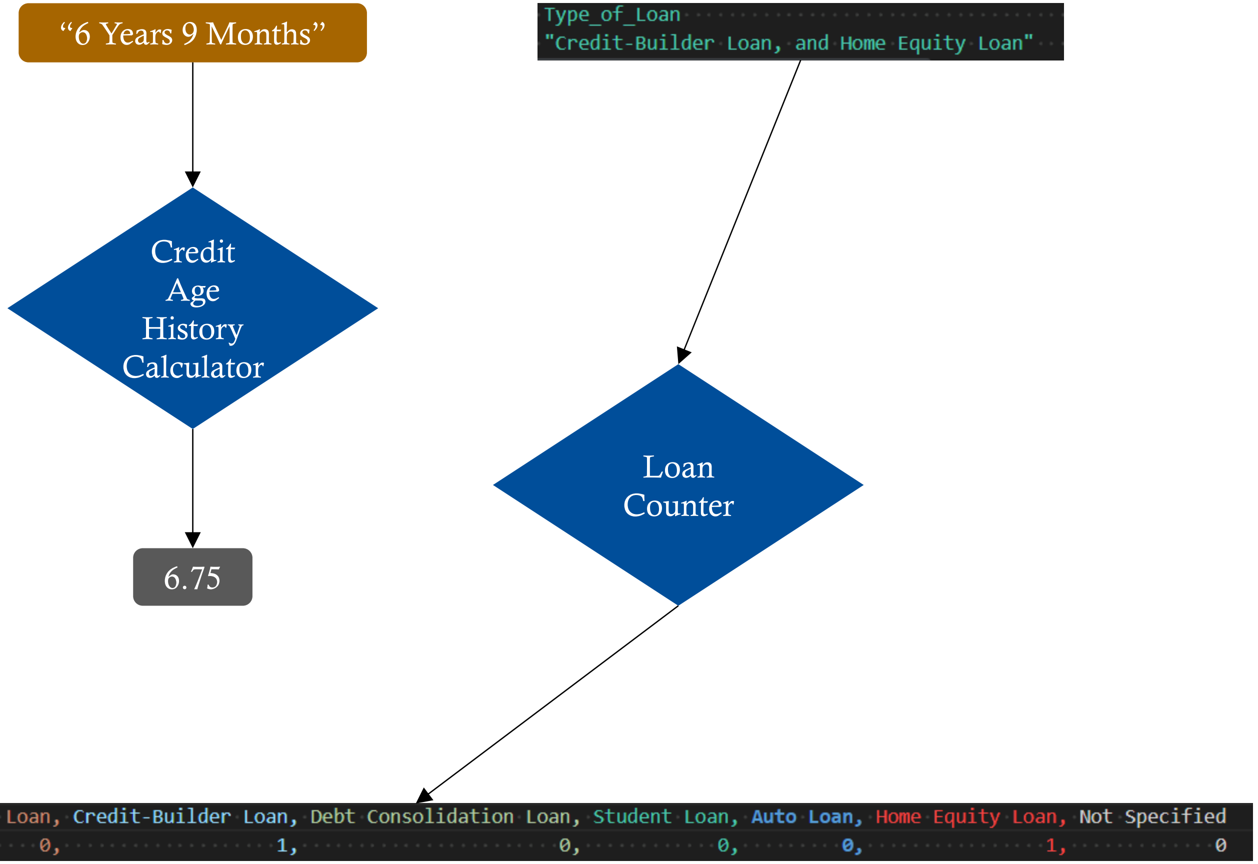
BRONZE TO SILVER TRANSFORMATION (LOAN DATA)

- Following Lab 3, I decided to re-use the previous days-past-due flag as a metric for defaulting a loan
  - Just in case, I ran the recalculation for dpd again to see if the same values corresponded and they did
- Upon inspection of the dataset, there does not seem to be any anomalous data, so cleaning was not done



BRONZE TO SILVER TRANSFORMATION (FINANCIALS)

- Financials had very dirty integer and float values, which were fixed with the previous int and float parsers
- Numerical variables that were not represented numbers we parsed into floats
- Recounted the number of loans based on “Type_of_loan”
- Split type of loan into columns which counted the number of loans an individual had



```
Type_of_Loan
"Credit-Builder Loan, and Home Equity Loan"
```

```
Personal Loan, Mortgage Loan, Payday Loan, Credit-Builder Loan, Debt Consolidation Loan, Student Loan, Auto Loan, Home Equity Loan, Not Specified
            0,            0,          0,                  1,                        0,            0,         0,                1,             0
```

SILVER TO GOLD TRANSFORMATION

Label Table

- Used the 6th installment and checked whether any installment was passed 30 days, was deemed as a default
- Made a new label called 30dpd_6mob to represent this

Feature Table

- Joined all data using Customer_ID as the key
- Join order was Clickstream -> attributes -> financials in order from cleanest to dirtiest, using inner join

THINGS TO NOTE

- Due to the time sensitive nature of data, a default is not known for new customers who have not made a loan. So data must be back-dated for each specific customer in the dataset which may exist in a different date. For matching feature to label, use Customer_ID as the key AFTER combining all dates together