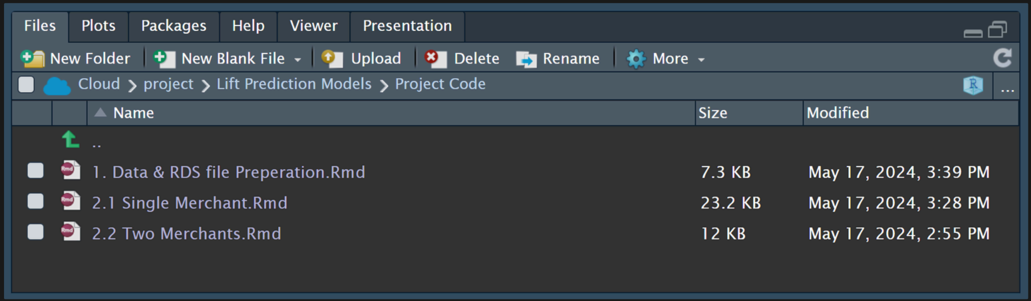
User Manual/ Instruction

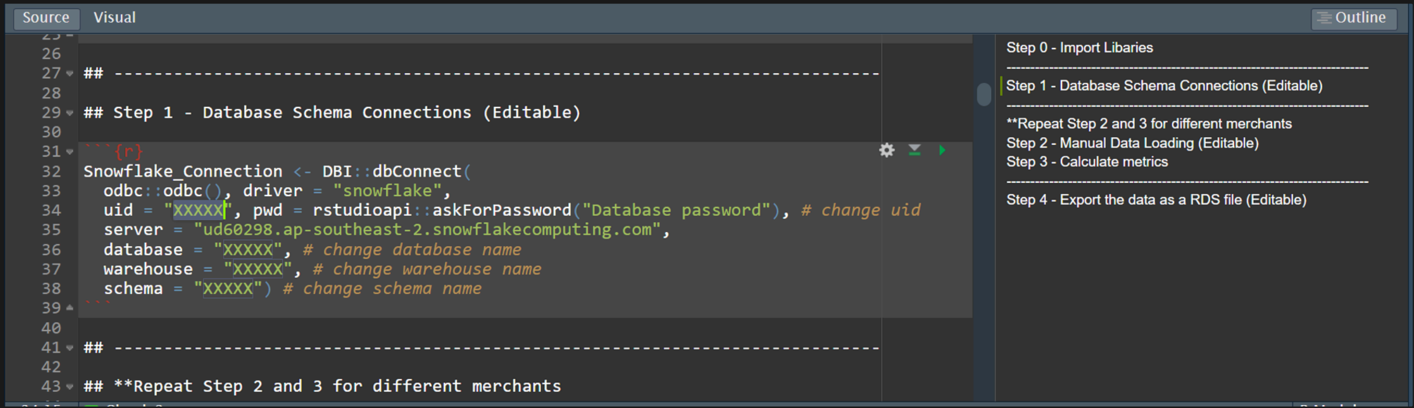
Instruction

The folder “Project Code” consists of three R Markdown files:

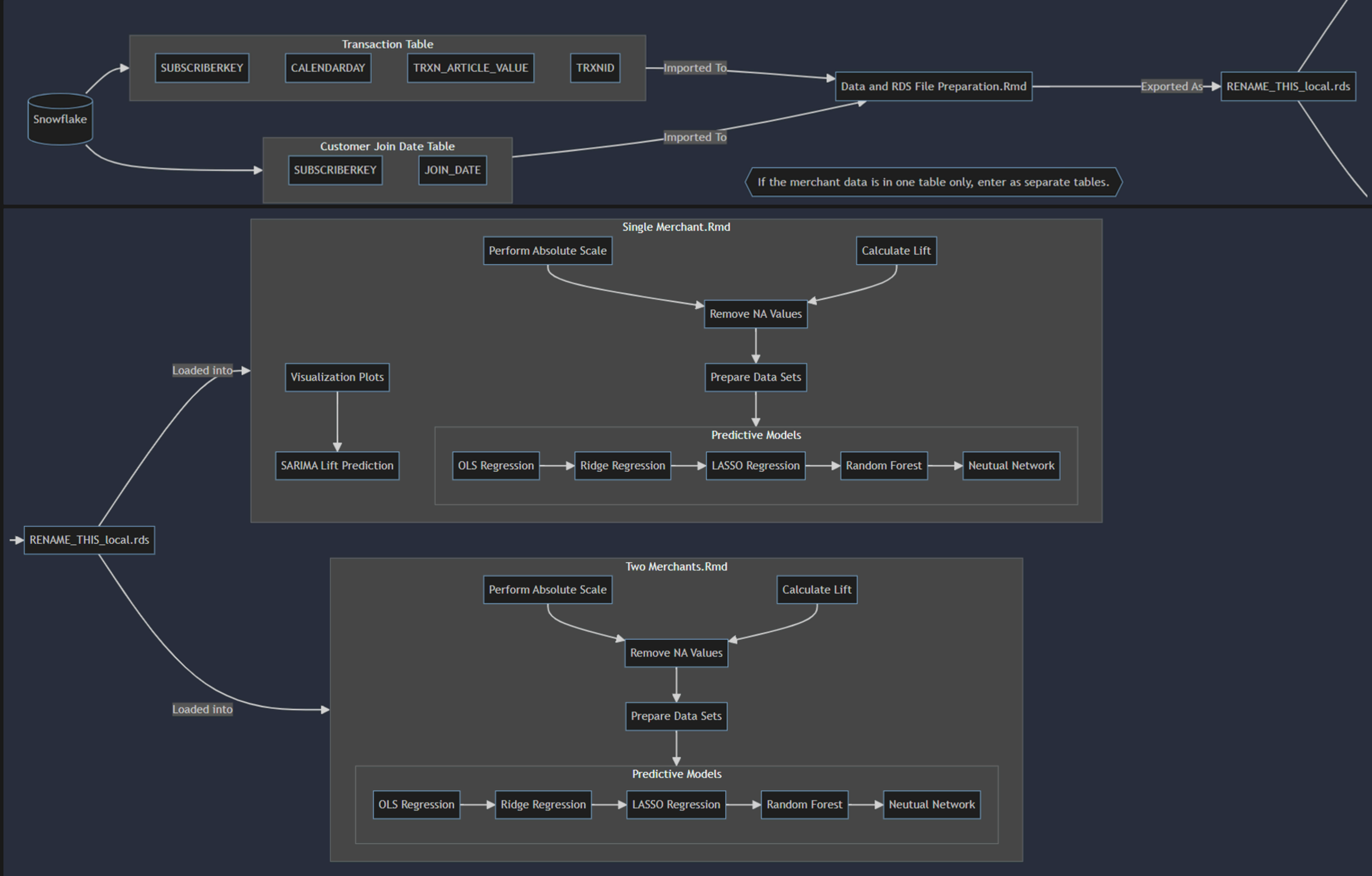
- `1. Data & RDS file Preperation.Rmd`
- `2.1 Single Merchant.Rmd`
- `2.2 Two Merchants.Rmd`



In each R Markdown file, sections marked with “(Editable)” require user input, typically prefilled with “XXXXX”. To navigate the files, open the Outline on the right side of the interface and click on the sections you wish to execute.



Flow diagram



Metric Specifications

Before delving into predictive modeling, it's crucial to define and accurately compute relevant metrics from the raw data. The table below outlines the metrics critical for evaluating merchant performance and customer behavior. Metrics like 'Total Sales', 'Total Customers', and 'Average Transaction Value' (ATV) provide direct insights into revenue and customer engagement. Customer dynamics are analyzed through metrics like 'Retention Rate', 'New Customer Rate', and 'Reactivated Rate', crucial for understanding customer engagement trends.

These preparations ensure our models are built on a solid foundation of accurately measured data, supporting strategic decision-making with robust, data-driven insights.

Table

Metric	Formula	Coded in R markdown?
Year_Period	CALENDARDAY (aggregated by year)	Done
Total_Sales	sum(TRXN_ARTICLE_VALUE)	Done
Total_Customer	n_distinct(SUBSCRIBERKEY)	Done
Total_Transactions	n_distinct(TRXNID)	Done
ATV (Average Transaction Value)	Total_Sales / Total_Transactions	Done
ATF (Average Transaction Freq.)	Total_Transactions / Total_Customer	Done
AS (Average Spend)	Total_Sales / Total_Customer	Done
Program_Penetration (by month)	Total_Customer / Total_Subscribers (Total_Subscribers is the total count of unique SUBSCRIBERKEYs in the database)	No avaliable data
Scan_Rate	Not directly mapped, requires additional context for calculation	No avaliable data
Redemption_Rate	Not directly mapped, requires additional context for calculation	No avaliable data
Retention_Rate (by month)	(Number of Customers whose Join Month equals Transaction Month) / (Total Unique Customers in Transaction Month)	Done
New_Customer_Rate (by month)	(Customers active in both the current and previous months) / (Total Unique Customers in the previous month)	Done
Reactivated rate (by month)	(Customers with a transaction after a gap of at least 1 month) / (Total Unique Customers in that month)	Done
Retained rate	1 - Reactivated Rate + New Customer Rate	Correlated
Churn	1 - Retention Rate	Correlated
One_time_Purchase_Rate	sum(ifelse(Total_Transactions == 1, 1, 0)) / Total_Customer	No avaliable data

+ New

Model Specifications

> 1. To proceed, select and execute one of the following modelling approaches:

1.1 Scaling absolute values and omitting NA for numeric columns

```
# Remove NA values and drop the YYYYMM column if it's not needed for modeling model_data_clean <- combined_data %>% na.omit() %>% dplyr::select(-YYYYMM) # Scale only numeric columns # Identify numeric columns numeric_columns <- sapply(model_data_clean, is.numeric) # Apply scaling to numeric columns only model_data_clean[numeric_columns] <- scale(model_data_clean[numeric_columns]) # Convert back to a data frame if it was converted to a matrix by scale() model_data_clean <- as.data.frame(model_data_clean)
```

1.2 Calculate Lift and omit NA for numeric columns

```
model_data_clean <- combined_data %>% group_by(Merchant) %>% # for dataset with two merchants mutate(across(where(is.numeric), ~ifelse(lag(.x) == 0 | is.na(lag(.x)), NA, .x / lag(.x) - 1))) %>% ungroup() model_data_clean <- na.omit(model_data_clean)
```

> 2. Run both for all the predictive models.

2.1 Splitting training and test data sets

75% of the data is randomly split into training set, the remaining 25% is used as the test set

```
## 75% of the sample size smp_size <- floor(0.75 * nrow(model_data_clean)) ## set the seed to make your partition reproducible
set.seed(1234) train_ind <- sample(seq_len(nrow(model_data_clean)), size = smp_size) train <- model_data_clean[train_ind, ] test <-
model_data_clean[-train_ind, ]
```

2.2 Prepare data for Ridge and LASSO Regression

We split the data again for Ridge and LASSO regression. This step ensures that our models have the correct input format for efficient computation and accurate predictions.

You may modify the model formula, for instance, by setting `Total_Customer` as the dependent variable.

```
# For the training set x_train <- model.matrix(Total_Sales ~ Total_Customer + Total_Transactions + ATV + AS + ATF + New_Customer_Rate +
Reactivated_Rate + Retention_Rate - 1, data = train) y_train <- train$Total_Sales # For the test set x_test <- model.matrix(Total_Sales ~
Total_Customer + Total_Transactions + ATV + AS + ATF + New_Customer_Rate + Reactivated_Rate + Retention_Rate - 1, data = test) y_test <-
test$Total_Sales # Case for Total_Customer as a dependent variable # For the training set x_train <- model.matrix(Total_Customer ~
Total_Sales + Total_Transactions + ATV + AS + ATF + New_Customer_Rate + Reactivated_Rate + Retention_Rate - 1, data = train) y_train <-
train$Total_Customer # For the test set x_test <- model.matrix(Total_Customer ~ Total_Sales + Total_Transactions + ATV + AS + ATF +
New_Customer_Rate + Reactivated_Rate + Retention_Rate - 1, data = test) y_test <- test$Total_Customer
```

> 3. Run all for all the predictive models.

OLS Regression

A linear regression model with `Total_Sales` as the dependent variable.

The model is trained on the train dataset.

You may modify the model formula, for instance, by setting `Total_Customer` as the dependent variable.

```
lm_model_train <- lm(Total_Sales ~ Total_Customer + Total_Transactions + ATV + AS + ATF + New_Customer_Rate + Reactivated_Rate +
Retention_Rate, data = train) # Case for Total_Customer as a dependent variable lm_model_train <- lm(Total_Customer ~ Total_Sales +
Total_Transactions + ATV + AS + ATF + New_Customer_Rate + Reactivated_Rate + Retention_Rate, data = train)
```

Ridge Regression

It is a variable selection method that could handle multicollinearity in data (correlation in variables).

This code finds the optimal lambda, shrinks coefficients close to 0, refits the model, and makes predictions on the test set.

```
# Perform cross-validation for Ridge Regression on the training data cv_fit_ridge <- cv.glmnet(x_train, y_train, alpha = 0) # Get the
optimal lambda for Ridge from the training data lambda_min_ridge <- cv_fit_ridge$lambda.min # Refit the Ridge Regression model using the
optimal lambda on the training data model_ridge <- glmnet(x_train, y_train, alpha = 0, lambda = lambda_min_ridge) # Use the fitted Ridge
model to make predictions on the test set predictions_ridge <- predict(model_ridge, s = lambda_min_ridge, newx = x_test)
```

LASSO Regression

It is a method that performs both variable selection and regularization to enhance the prediction accuracy and interpretability.

It does this by shrinking the coefficients of less important features to exactly 0, effectively eliminating them from the model.

```
# Perform cross-validation for Lasso Regression on the training data cv_fit_lasso <- cv.glmnet(x_train, y_train, alpha = 1) # Get the optimal lambda for Lasso from the training data lambda_min_lasso <- cv_fit_lasso$lambda.min # Refit the Lasso Regression model using the optimal lambda on the training data model_lasso <- glmnet(x_train, y_train, alpha = 1, lambda = lambda_min_lasso) # Use the fitted Lasso model to make predictions on the test set predictions_lasso <- predict(model_lasso, s = lambda_min_lasso, newx = x_test)
```

Random Forest

it is a machine learning method capable of performing both regression and classification tasks.

The code trains a Random Forest model on the training data, and then uses this model to make predictions on the test set.

```
rf_model <- randomForest(Total_Sales ~ Total_Customer + Total_Transactions + ATV + AS + ATF + New_Customer_Rate + Reactivated_Rate + Retention_Rate, data = train, importance = TRUE) # Use the fitted model to make predictions on the test set predictions_rf <- predict(rf_model, newdata = test)
```

Neutral Network

Designed to recognize patterns by algorithms modeled after the human brain, the code fits a neural network model to the training data, and then uses this model to make predictions on both the training and test sets.

```
# Fit a neural network model nn_model <- nnet(Total_Sales ~ Total_Customer + Total_Transactions + ATV + AS + ATF + New_Customer_Rate + Reactivated_Rate + Retention_Rate, data = train, size = 15, # Increased number of units in the hidden layer linout = TRUE, # Linear output neurons decay = 0.01, # Adding weight decay to prevent overfitting maxit = 500) # Increased max iterations for more training # Predict on the training set train_predictions_nn_complex <- predict(nn_model, newdata = train, type = "raw") # Predict on the test set predictions_nn <- predict(nn_model, newdata = test, type = "raw")
```

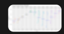
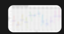
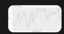
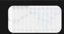
Visualization Plots

`Line Plots` track data changes over time, highlighting trends and anomalies.

`Seasonal Plots` illustrate patterns within each month, showing deviations from typical seasonal behaviors.

`Seasonal Subseries Plots` enable the underlying seasonal pattern to be seen clearly, and changes in seasonality over time to be visualized.

Visualization Plots

Name	Merchant	Plot type	Files & media
1	MJB	Line Plot	
2	MJB	Seasonal Plot	
3	MJB	Seasonal Subseries Plot	
4	Forever New	Line Plot	
5	Forever New	Seasonal Plot	
6	Forever New	Seasonal Subseries Plot	
7	Spotlight	Line Plot	
8	Spotlight	Seasonal Plot	
9	Spotlight	Seasonal Subseries Plot	
10	Synthetic	Line Plot	
11	Synthetic	Seasonal Plot	
12	Synthetic	Seasonal Subseries Plot	

+ New

Model Performance

Performance

↑ Name

Merchant | Model | Method

Name	Merchant	Model	Method	R-squared on Test set	MAE on Test set	RMSE on Test set
1	MJB	OLS Linear Regression	Absolute Scale	0.987196352994771	0.0732405546654706	0.088003361391959
2	MJB	OLS Linear Regression	Lift	0.97926817991676	0.0158882623005136	0.0204145620429069
3	MJB	Ridge Regression	Absolute Scale	0.970882344469285	0.0964750922284649	0.13271212913077
4	MJB	Ridge Regression	Lift	0.944982863160178	0.0250265513729297	0.0332560254956158
5	MJB	LASSO Regression	Absolute Scale	0.981748710261208	0.08876948597414	0.105070111841379
6	MJB	LASSO Regression	Lift	0.964307634263198	0.0192719588926407	0.026786071206212
7	MJB	Random Forest	Absolute Scale	0.90674515295867	0.196746380786792	0.237502465187024
8	MJB	Random Forest	Lift	0.802893039007624	0.0466361284311704	0.0629466080684115
9	MJB	Neural Network	Absolute Scale	0.992137789389171	0.0246228557483691	0.0689611906918178
10	MJB	Neural Network	Lift	0.951887462626349	0.020842498816342	0.0310992905956767
11	Forever New	OLS Linear Regression	Absolute Scale	0.972599748984071	0.103794918136639	0.223252215707665
12	Forever New	OLS Linear Regression	Lift	0.997634051813743	0.0113059228639219	0.0159850412891577
13	Forever New	Ridge Regression	Absolute Scale	0.99236948245244	0.0865133844229934	0.117813621155274
14	Forever New	Ridge Regression	Lift	0.998504210386187	0.0105212613356203	0.0127100249864247
15	Forever New	LASSO Regression	Absolute Scale	0.996661933000234	0.0585284045698689	0.0779230659263861
16	Forever New	LASSO Regression	Lift	0.997541642928562	0.0128901819783543	0.0162942220146016