BRANCH
CASH RETENTION LIMIT OPTIMIZATION

By

Manjis Majumdar

GUIDE:

Mr. Nitin Narang
(Senior Manager-Data Analyst,
Management Information System & Data Analytics Division, PNB)

Project Work undertaken at Punjab National Bank, HO, New Delhi

DECLARATION

I, **Manjis Majumdar**, hereby declare that the presented report of internship titled **"Branch Cash Retention Limit Optimization"** is uniquely prepared by me after the completion of one month work at Punjab National Bank, HO, New Delhi.

I also confirm that the report is only prepared for my academic requirement not for any other purpose. It might not be used with the interest of opposite party of the corporation.

Manjis Majumdar
MSc. In Big Data Analytics
Dept. of Computer Science
RKMVERI, Belur Math, Howrah

ACKNOWLEDGEMENT

I would like to take this opportunity to express my sincere gratitude to all those who have guided me and supported me throughout this project work. Foremost, I would like to express my gratitude to "Punjab National Bank" for giving me an opportunity to be a part of this esteemed organisation and enhance my knowledge by granting me permission to pursue my internship project with them.

I am immensely grateful to my guide for the project, Mr. Nitin Narang for his valuable guidance and continuous encouragement throughout the course of this project. He took out the time from his busy schedule and nudged me towards the right approach to carry out the project.

Last but not the least, I would like to thank my family and friends for their constant support throughout the project.

EXECUTIVE SUMMARY

Cash Retention Limit is the amount of money a certain branch of a bank can keep overnight in order to carry on day-to-day operations.

This limit is decided at the Head Office of the bank for the Zones. In case of circles, the Zones decide the limit, whereas in case of each branch, its corresponding circle office decides the limit. There are certain factors that allow the bank to decide the Cash Retention Limit for all their branches.

These are:
1. The size of a branch
2. Business they do on a day-to-day basis
3. Average daily pay and receipts
4. Location of the branch (market/residential area)
5. Category of the branch

Cash Retention Limits are mainly set to ensure smooth flow of all business operations at the branch but with reduced risk and a higher profitability as its by-product.

In this project, we are deriving the Cash Retention Limit for each branch by predicting the daily receipt and withdrawal. Also, instead of manually setting up the limit by each circle (as done presently), the limit will be set at one place by deriving mathematically through automation.

TABLE OF CONTENTS

CONTENTS	PAGE NO.
INTRODUCTION	
Company Profile	6
Organization Structure	7
Performance (Financial)	7-9
Awards & Accolades	9
PROJECT DETAILS	
Problem Summary	10
Objective	10
Data	10
Procedure	11
Model Building	12
Model Description	13-15
Calculation of CRL	16-17
Outcomes	18
Inference	19
Suggestions	19
Annexure-I	20-21

INTRODUCTION

COMPANY PROFILE

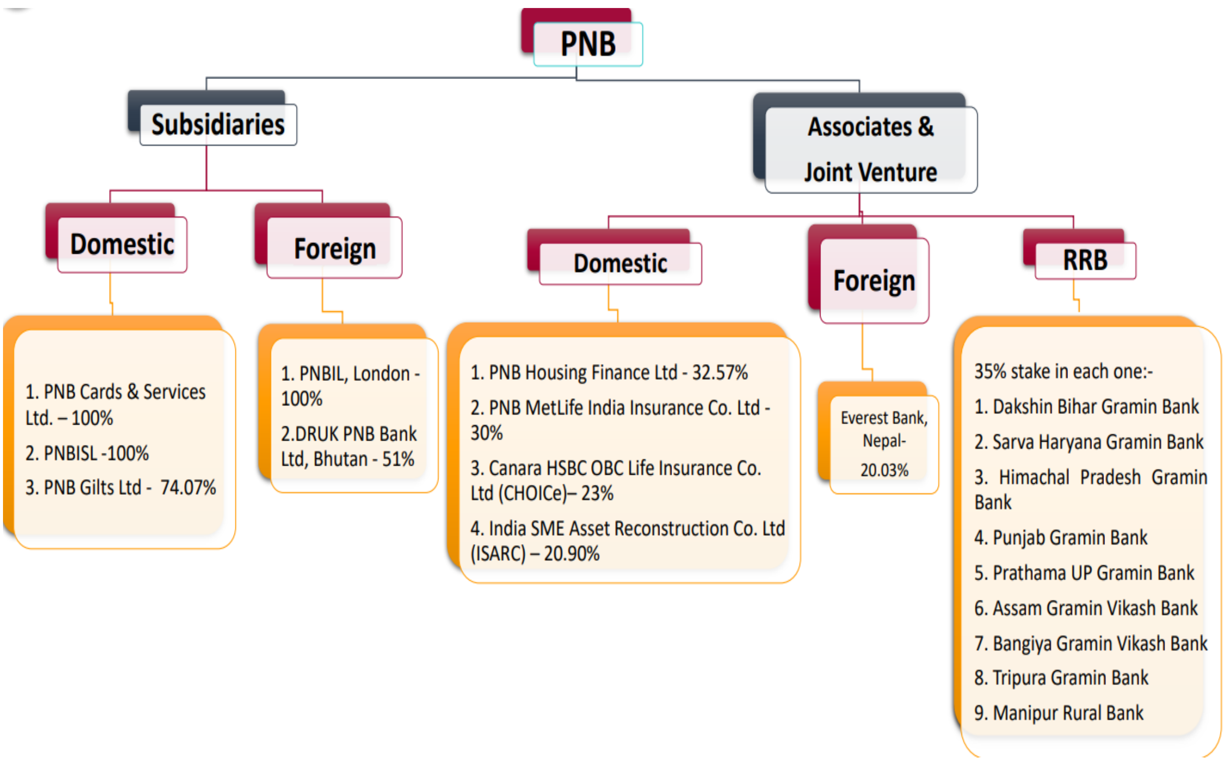
Punjab National Bank (PNB), India's first Swadeshi Bank, commenced its operations on April 12, 1895 from Lahore, with an authorized capital of Rs. 2 lac and working capital of Rs. 20,000. The Bank was established by the spirit of nationalism and was the first bank purely managed by Indians with Indian Capital. During the long history of the Bank, 9 banks have been merged/ amalgamated with PNB.

As at the end of March' 2022, Bank has total 39,167 delivery channels with a network of 10,098 domestic branches, 2 International branches, 13,350 ATMs & 15,719 Business Correspondents.

As on March'2022, Bank is having the 2 International branches in Gift City, Ahmedabad and Dubai. The Bank has two overseas subsidiaries viz. PNB International Ltd. London and Druk PNB Bank Ltd. Bhutan and one joint Venture Bank in Nepal under the name Everest Bank Ltd. Nepal. Bank has its representative offices in Myanmar and Bangladesh.

PNB is the second largest Public Sector Bank (PSB) in the country with Global Gross Business at ₹ 19,31,322 Crore. The Bank continues to maintain its forte in low-cost CASA deposits with a share of 47.43%. Bank's focus has been on qualitative business growth, recovery and arresting fresh slippages.

ORGANIZATION STRUCTURE



PERFORMANCE(FINANCIAL)

1. KEY FINANCIAL HIGHLIGHTS (JUNE 2022)

2. BUSINESS PERFORMANCE

Sl.	Parameters	Jun'21	Mar'22	Jun'22	Growth %	
					QoQ	YoY
1	Global Gross Business	1823685	1931322	1936923	0.29%	6.21%
	Overseas Gross Business	36666	47059	51019	8.42%	39.15%
	Domestic Gross Business	1787019	1884263	1885904	0.09%	5.53%
2	Global Deposits	1097649	1146218	1136747	-0.83%	3.56%
	Overseas Deposits	18712	21169	22040	4.12%	17.79%
	Domestic Deposits	1078937	1125049	1114706	-0.92%	3.32%
	Current Deposits	67611	81974	69332	-15.42%	2.55%
	Savings Deposits	419525	451680	447258	-0.98%	6.61%
	CASA Deposits	487136	533654	516590	-3.20%	6.05%
	CASA Share %	45.15%	47.43%	46.34%		
	Total Term Deposits	610513	612564	620157	1.24%	1.58%
3	Global Gross Advances	726036	785104	800177	1.92%	10.21%
	Overseas Gross Advances	17954	25890	28979	11.93%	61.41%
	Domestic Gross Advances	708082	759214	771198	1.58%	8.91%
4	CD Ratio %	66.14%	68.50%	70.39%	190 bps	425 bps

3. PROFIT & PROVISIONS

Sl.	Parameters	Q1 FY22	Q4 FY22	Q1 FY23	QoQ Variation		YoY Variation	
					Amt.	%	Amt.	%
1	Net Interest Income	7234	7304	7543	239	3.27%	309	4.27%
2	Other Income	3887	2450	2537	87	3.55%	-1350	-34.73%
3	Operating Income (1+2)	11122	9754	10080	326	3.34%	-1042	-9.37%
4	Operating Expenses	4722	4489	4701	212	4.71%	-21	-0.45%
5	Operating Profit	6400	5265	5379	114	2.17%	-1021	-15.95%
6	Provisions other than Tax	4980	4851	4790	-61	-1.26%	-190	-3.82%
	Of which							
a	NPAs	3248	4564	4814	250	5.48%	1566	48.21%
b	Standard Advances incl. Standard Restructured	1193	25	-278	-303	-	-1471	-
c	Depreciation on Investment	530	99	149	50	50.51%	-381	-71.89%
d	Other provisions	9	164	105	-59	-35.98%	96	-
7	Profit Before Tax	1420	413	589	176	42.62%	-831	-58.52%
8	Provision for Income Tax	397	212	281	69	32.55%	-116	-29.22%
9	Net Profit	1023	202	308	106	52.48%	-715	-69.89%

Sl.	Profitability Ratios	Q1 FY22	Q4 FY22	Q1 FY23
1	Return on Assets [%]	0.30%	0.06%	0.09%
2	Return on Equity [%]	7.13%	1.35%	2.01%
3	Earnings per share [₹] (Not annualized)	0.95	0.18	0.28
4	Book Value per Share [₹]	78.49	79.59	80.32
4a	Book Value per Share-Tangible [₹]	54.01	54.77	57.16
5	Cost to Income Ratio [%]	42.46%	46.02%	46.63%
5a	Staff Cost to Income Ratio [%]	26.58%	22.12%	25.27%
5b	Other Cost to Income Ratio [%]	15.88%	23.90%	21.37%
6	Operating Profit to AWF [%]	1.90%	1.59%	1.63%
7	Operating Expenses To AWF [%]	1.40%	1.36%	1.42%

AWARDS & ACCOLADES

1st

"Best Core Banking System Initiative"

13th Annual Retail Banker International Trailblazer Awards

"Most Innovative Branch Offering"

13th Annual Retail Banker International Trailblazer Awards

"National MSME Awards 2022" (3rd Prize)

Contribution towards the promotion and development of the MSME sector

Recognized by PFRDA for performance under National Pension System (NPS) in Quarterly Award Recognition Programme for Q4 FY'22

PROJECT DETAILS

PROBLEM SUMMARY

In this project, we are deriving the Cash Retention Limit for each branch by predicting the daily receipt and withdrawal. Also, instead of manually setting up the limit by each circle (as done presently), the limit will be set at one place by deriving mathematically through automation.

OBJECTIVE

To derive the Cash Retention Limit for each branch by prediction of daily cash receipt and pay.

DATA

- There are 10047 branches in the bank, as of May'22.
- These branches are divided into 27 zones and 137 circles.
- Out of those, there are 439 branches in ZO Delhi.
- In ZO Delhi, there are 7 circles:
    i. East Delhi
    ii. New Delhi
    iii. North Delhi
    iv. South Delhi
    v. West Delhi
    vi. Ghaziabad
    vii. Noida
- We have taken the South Delhi circle as a sample for our study, which is having 66 branches.

PROCEDURE

1. Daily receipt and pay of 66 branches for the period Jan'21 to May'22 is considered.

2. These branches are having weekend holidays and festival on some dates. Their effect is incorporated in the model.

3. Daily receipt and pay value for June'22 is predicted using **Prophet Time Series model** and monthly average of predicted values(x) of receipt and pay are taken.

4. This average value(x) of each branch is adjusted (+/-) by Standard Deviation(y) of forecasted value on basis of distance between branch and Currency Chest.
   Calculation of CRL = Avg. value(x) ± (z * SD(y)) where, z is a constant and its value varies from 1 to 3 depending on distance between Branch-CC.

5. Predicted value of June'22 is validated through Monthly Average and Standard Deviation of actual vs predicted value.

6. Proposed CRL is derived and optimized on the predicted data of June'22.

7. Present and proposed CRL optimized by formula:
   Receipt + CRL – Pay >= 0

MODEL BUILDING

1. The data is read and pre-processed to make it in the desired form.

2. The model is defined in the following steps:

   a. Empty data frames are defined: for Pay (train & test) and for Receipt (train & test) which is later used to save the result of model.

   b. Each branch is taken at a time, and for each branch:

      (i) A holiday function is defined to form a holiday data frame with the list of dates. The dates are having holiday effects with the name of the holiday, i.e., weekends, national and local holidays. This data frame is incorporated while defining the model.

      (ii) Branch data is divided into two data frames, one for Pay and one for Receipt. These data frames are further divided into test and train sets. Train set is taken to be from Jan'21 to May'22 (17 months), and the test set is taken to be the month of Jun'22.

      (iii) A forecast function is defined for Prophet Time Series Model to predict values for the month of June'22.

      (iv) Two functions, RMSE, and MAPE, are also defined which are used for calculating the accuracy of the model.

      (v) Model outcomes are appended in the data frames that were defined in step 2a. Finally, we get 4 data frames: two for Pay (test & train) and two for Receipt (test & train). These 4 data frames are saved in the system directly.

Related Python code is attached in Annexure-I

MODEL DESCRIPTION

USAGE

```
Prophet (
   df = NULL,
   growth = “linear”,
   changepoints = NULL,
   n_changepoints = 25,
   changepoint_range = 0.8,
   yearly_seasonality = “auto”,
   weekly_seasonality = “auto”,
   daily_seasonality = “auto”,
   holidays = NULL,
   seasonality_mode = “additive”,
   seasonality_prior_scale = 10,
   holidays_prior_scale = 10,
   changepoint_prior_scale = 0.05,
   fit = True,
...
)
```

ARGUMENTS

df

(optional) Data frame containing the history. Must have columns ds (date type) and y, the time series. If growth is logistic, then df must also have a column cap that specifies the capacity at each ds. If not provided, then the model object will be instantiated but not fit; use fit.Prophet(m, df) to fit the model.

growth

String 'linear', 'logistic', or 'flat' to specify a linear, logistic or flat trend.

changepoints

Vector of dates at which to include potential changepoints. If not specified, potential changepoints are selected automatically.

n_changepoints

Number of potential changepoints to include. Not used if input `changepoints` is supplied. If `changepoints` is not supplied, then n_changepoints potential changepoints are selected uniformly from the first `changepoint_range` proportion of df["ds"].

changepoint_range

Proportion of history in which trend changepoints will be estimated. Defaults to 0.8 for the first 80 `changepoints` is specified.

yearly_seasonality

Fit yearly seasonality. Can be 'auto', TRUE, FALSE, or a number of Fourier terms to generate.

weekly_seasonality

Fit weekly seasonality. Can be 'auto', TRUE, FALSE, or a number of Fourier terms to generate.

daily_seasonality

Fit daily seasonality. Can be 'auto', TRUE, FALSE, or a number of Fourier terms to generate.

holidays

data frame with columns holiday (character) and ds (date type) and optionally columns lower_window and upper_window which specify a range of days around the date to be included as holidays. lower_window=-2 will include 2 days prior to the date as holidays. Also, optionally can have a column prior_scale specifying the prior scale for each holiday.

seasonality_mode

'additive' (default) or 'multiplicative'.

seasonality_prior_scale

Parameter modulating the strength of the seasonality model. Larger values allow the model to fit larger seasonal fluctuations, smaller values dampen the seasonality. Can be specified for individual seasonalities using add_seasonality.

holidays_prior_scale

Parameter modulating the strength of the holiday components model, unless overridden in the holidays input.

changepoint_prior_scale

Parameter modulating the flexibility of the automatic changepoint selection. Large values will allow many changepoints, small values will allow few changepoints.

fit

Boolean, if FALSE the model is initialized but not fit.

...

Additional arguments, passed to '*fit.Prophet*'

CALCULATION OF CRL

ASSUMPTIONS TO DERIVE CRL:

The branches are divided into two parts on basis of **Receipt > Pay** and **Receipt < Pay**.
Out of 66 branches, 38 branches belong to **Receipt > Pay** category, and 28 branches belong to **Receipt < Pay** category.

Rules for CRL according to distance between Branch and Currency Chest

Distance btw BR-CC (in kms)	Conditions	CRL Rule
0		5 lacs for all Branches – default
1-5	Receipt > Pay	5 lacs for all Branches – default
	Receipt < Pay	10 lacs for all Branches – default
5-10	Receipt > Pay	Min 5 lacs or as derived from proposed formula
	Receipt < Pay	Min 10 lacs or as derived from proposed formula
10 and above	Unconditional	Min 10 lacs or as derived from proposed formula

PROPOSED CRL MATRIX FOR RECEIPT > PAY (38 branches)

<table>
<tr><th></th><th colspan="5">Distance Between Branch and Currency Chest (in kms)</th></tr>
<tr><th>Conditions</th><th>5-10</th><th>10-20</th><th>20-30</th><th>30-40</th><th>40 & above</th></tr>
<tr><td colspan="6">No. of Rounds between Branch and Currency Chest < 10 in a month</td></tr>
<tr><td>Receipt > Pay + 3SD</td><td colspan="3">Pay – 3SD</td><td colspan="2">Pay – 2SD</td></tr>
<tr><td>Receipt > Pay + 2SD</td><td colspan="3">Pay – 2SD</td><td colspan="2">Pay – SD</td></tr>
<tr><td>Receipt > Pay + SD</td><td colspan="2">Pay – SD</td><td colspan="3">Pay</td></tr>
<tr><td>Receipt > Pay</td><td>Pay – SD</td><td colspan="4">Pay</td></tr>
<tr><td colspan="6">No. of Rounds between Branch and Currency Chest >= 10 in a month</td></tr>
<tr><td>Receipt > Pay + 3SD</td><td colspan="5">Pay – 3SD</td></tr>
<tr><td>Receipt > Pay + 2SD</td><td colspan="5">Pay – 2SD</td></tr>
<tr><td>Receipt > Pay + SD</td><td colspan="5">Pay – SD</td></tr>
<tr><td>Receipt > Pay</td><td>Pay – SD</td><td>Pay – 2SD</td><td>Pay – SD</td><td colspan="2">Pay</td></tr>
</table>

PROPOSED CRL MATRIX FOR RECEIPT < PAY (28 branches)

- For Receipt < Pay, we calculate Gap (or deficit): (Avg. Pay + SD Pay) – (Avg. Receipt + SD Receipt)
- Rules are defined on this gap for CRL calculation

Conditions	Gap	Pay	Proposed CRL
Condition – 1	< 10 lacs	< 10 lacs	10 or 15 lacs
Condition – 2	< 10-20 lacs	< 10-20 lacs	20 or 25 lacs
Condition – 3			
Rounds < 10	> 20 lacs	> 20 lacs	Gap + 20 or Gap + 15
Rounds > 10			Gap + 15 or Gap + 20

RECEIPT < PAY (Condition-3)

Distance between Branch and Currency Chest 5-10 kms		
Conditions	**Receipt < Pay – 3SD or -2SD or -SD**	**Receipt < Pay**
Rounds < 10	Gap + 25	Gap + 15
Rounds > 10	Gap + 20	Gap + 10
Distance between Branch and Currency Chest 10-20 kms		
Conditions	**Receipt < Pay – SD or Receipt < Pay**	**Receipt < Pay – 3SD or Receipt < Pay-2SD**
Rounds < 10	Gap + 25	Gap + 15
Rounds > 10	Gap + 20	Gap + 10
Distance between Branch and Currency Chest 20-30 kms		
Conditions	**Receipt < Pay – 3SD or -2SD or -SD or Receipt < Pay**	
Rounds < 10	Gap + 25	
Rounds > 10	Gap + 20	
Distance between Branch and Currency Chest 30 kms and above		
Conditions	**Receipt < Pay – 3SD or -2SD or -SD or Receipt < Pay**	
Rounds < 10	Gap + 20	
Rounds > 10	Gap + 15	

OUTCOMES

No. of Branches in Bank	10047
Total CRL (in Rs. Cr.)	2080
No. of branches in ZO Delhi	439
Total CRL ZO Delhi (in Rs. Cr.)	129.85
No. of branches under South Delhi CO	66
Present CRL (in Rs. Cr.)	18.74
Proposed CRL (in Rs. Cr.)	11.02

COMPARISON IN PRESENT AND PROPOSED METHOD

Distance (in km)	Number of branches	Total CRL (in Rs. lacs)		Change in CRL
		Present	Proposed	
1-5	11	274	136	-138.0
5-10	25	660	375.8	-284.2
10-20	27	870	546.3	-323.7
20-30	3	70	43.9	-26.1
Total:	**66**	**1874**	**1102**	**-772**

INFERENCE

- There is a **41.19% decline** in the total CRL of the 66 branches of the South Delhi zone.

- Out of the 66 branches:
  - CRL has **decreased in 53 branches** by a total of Rs. 8.37 crores
  - CRL has **increased in 8 branches** by a total of Rs. 0.65 crore
  - CRL has remained the **same for 5 branches**

- The maximum decline (37.2%) has occurred for the branches which are located within 10-20 kms from the Currency Chest.

- The per-branch decline is maximum for the branches within 5-10 kms from the Currency Chest.

SUGGESTIONS

- Instead of manually setting up the limit by each circle (as done presently), the limit can now be set through prediction model so that human biasness is reduced.

- Model building can be done for each individual circle, because for each circle is having different patterns of receipt and pay. Also, the pattern of holidays varies from circle to circle.

Annexure-I

MODEL PYTHON CODE

1. Defining the data frames

```
global df_pay_train
df_pay_train = pd.DataFrame(columns = ['rmse', 'mape', 'avg_act', 'avg_pred', 'stdev', 'min', '
global df_pay_test
df_pay_test = pd.DataFrame(columns = ['rmse', 'mape', 'avg_act', 'avg_pred', 'stdev', 'min', 'm
global df_receipt_train
df_receipt_train = pd.DataFrame(columns = ['rmse', 'mape', 'avg_act', 'avg_pred', 'stdev', 'min
global df_receipt_test
df_receipt_test = pd.DataFrame(columns = ['rmse', 'mape', 'avg_act', 'avg_pred', 'stdev', 'min'
```

2. Holiday function

```
def holiday(data1):
    global hol
    upper_window = []
    lower_window = [0]
    da1 = data1['Date']

    for i in range(len(da1) - 1):
        deltap = da1[i+1] - da1[i]
        upper_window.append(deltap.days - 1)
    for i in range(len(da1)):
        if i != 0:
            deltan = da1[i-1] - da1[i]
            lower_window.append(deltan.days + 1)
    upper_window.append(0)

    h = pd.concat([da1, pd.DataFrame(lower_window), pd.DataFrame(upper_window)], axis = 1)
    h.columns = ["ds", "lower_window", "upper_window"]

    h2 = h.drop(h[(h['lower_window'] == 0) & (h['upper_window'] == 0)].index)
    h2 = h2.reset_index(drop = True)

    ld = list(h2['ds'])
    lw = list(h2['lower_window'])
    lu = list(h2['upper_window'])

    holiday = []
    for i in range(len(h2)):
        if((ld[i] == pd.to_datetime('2021-01-25')) | (ld[i] == pd.to_datetime('2021-01-27')) | (ld[i] == pd.to_datetime('2021-04-01')) |
        (ld[i] == pd.to_datetime('2021-04-03')) | (ld[i] == pd.to_datetime('2021-08-14'))| (ld[i] == pd.to_datetime('2021-08-16')) |
        (ld[i] == pd.to_datetime('2021-10-01')) | (ld[i] == pd.to_datetime('2021-10-03')) | (ld[i] == pd.to_datetime('2021-12-24')) |
        (ld[i] == pd.to_datetime('2021-12-26'))):
            holiday.append('national')

        elif (lw[i] == 0) & (lu[i] == 1):
            holiday.append('hol_wk_sat1')

        elif (lw[i] == -1) & (lu[i] == 0):
            holiday.append('hol_wk_mon1')

        elif (lw[i] == 0) & (lu[i] == 2):
            holiday.append('hol_wk_fri2')

        elif (lw[i] == -2) & (lu[i] == 0):
            holiday.append('hol_wk_mon2')

        else:
            holiday.append('local')

    hol = pd.concat([pd.DataFrame(holiday), h2], axis = 1)
    hol.columns = ['holiday', 'ds', 'lower_window', 'upper_window']
    return hol
```

3. Train-test Split and Pay and Receipt Split

```
data1_p = data1.loc[:, ['Date','TOT_PAY']]
data1_r = data1.loc[:, ['Date', 'TOT_RCPT']]
data1_p = data1_p[data1_p['TOT_PAY'] > 0.10]
data1_r = data1_r[data1_r['TOT_RCPT'] > 0.10]
data1_p.columns = ['ds', 'y']
data1_r.columns = ['ds', 'y']

p_train = data1_p.loc[(data1_p['ds'] <= '2022-05-31')]
p_test = data1_p.loc[(data1_p['ds'] > '2022-05-31')]
r_train = data1_r.loc[(data1_r['ds'] <= '2022-05-31')]
r_test = data1_r.loc[(data1_r['ds'] > '2022-05-31')]
```

4. Forecast function

```
def forecast(data2, future):
    m = Prophet(yearly_seasonality = False, weekly_seasonality = True, daily_seasonality = False,
                holidays = holi)
    m.fit(data2)
    forecast1 = m.predict(future)
    return forecast1

pred_p_train = forecast(p_train, pd.DataFrame({'ds' : p_train['ds']}))
pred_p_test = forecast(p_train, pd.DataFrame({'ds' : p_test['ds']}))
pred_r_train = forecast(r_train, pd.DataFrame({'ds' : r_train['ds']}))
pred_r_test = forecast(r_train, pd.DataFrame({'ds' : r_test['ds']}))
```

5. RMSE, MAPE, and details function

```
def rmse(actual, pred):
    result = sqrt(mean_squared_error(actual, pred))
    return result

def mape(actual1, pred1):
    actual1 = np.array(actual1)
    pred1 = np.array(pred1)
    result1 = np.mean(np.abs((actual1 - pred1) / actual1)) * 100
    return result1

def details(data, fore_data):
    rms_err = rmse(data['y'], fore_data['yhat'])
    map_err = mape(data['y'], fore_data['yhat'])
    avg_act = data['y'].mean()
    avg_pred = fore_data['yhat'].mean()
    stdev = fore_data['yhat'].std()
    minimum = fore_data['yhat'].min()
    maximum = fore_data['yhat'].max()
    cnt_neg_pred = (fore_data['yhat'] <= 0).sum()

    df1 = pd.DataFrame([rms_err, map_err, avg_act, avg_pred, stdev, minimum, maximum, cnt_neg_pre
    df1 = df1.T
    df1.columns = ['rmse', 'mape', 'avg_act', 'avg_pred', 'stdev', 'min', 'max', 'cnt_neg_pred']
    return df1
```