

# UIUC Datathon 2023

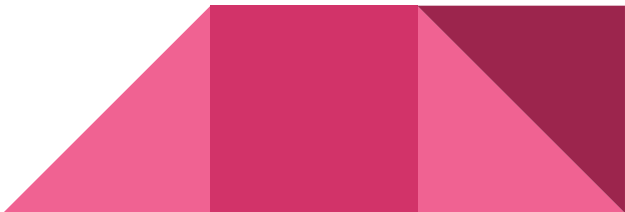
## Brendan's Beneficiaries

Chieh Hsu, Barret Li, Soroush Hoseini, Jason Cheng

# Problem Breakdown

For the training set, we were given data of customer accounts from Jan 2018 to Dec 2019. We are then asked to predict the number of monthly charge-offs over the period of Feb 2020 to Jan 2021, given data of customer accounts from Jan 2020.

The code for use in the following approach to the problem can be found in the drive, named “Brendan's Beneficiaries Code Used For Submission.” There is also an alternative approach related to machine learning that we were unfortunately unable to finish in time.



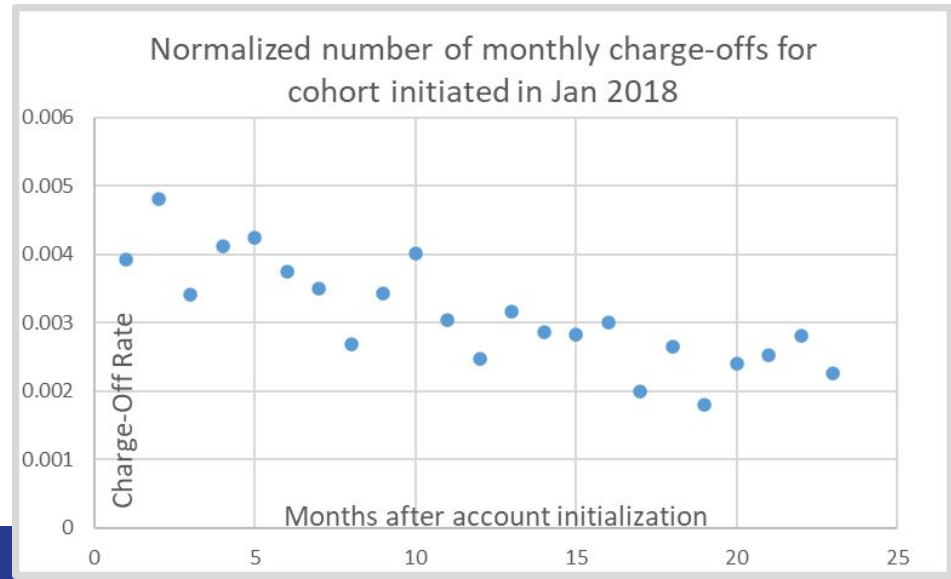
# Preliminary Approach

We first realized that the dataframe is very big (1.5GB) and that we could slice it into smaller data frames to approach it faster. Luckily, we can immediately group datapoints that have the same “snapshot” value, which tells us which month the account was initialized, into cohorts. Another column to note is “mth\_code,” which tells us when the datapoint was taken.



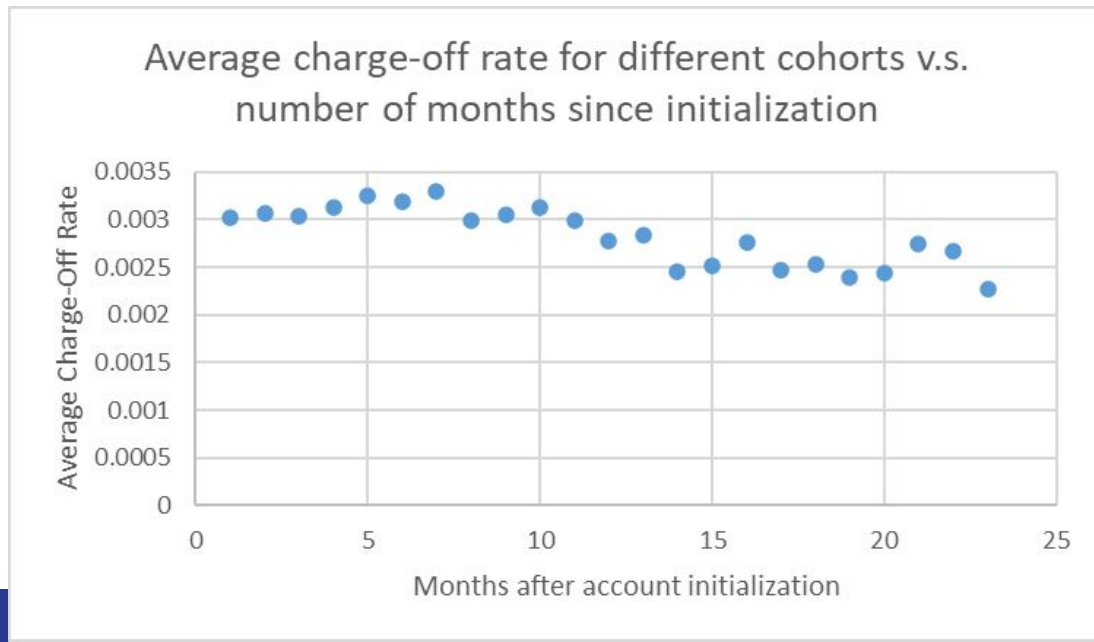
# Preliminary Approach

We discovered that the status of charge-off is encoded by 0 or 1, and therefore, if we were to sum up this value in a given cohort in a given month, we can easily obtain the number of charge-offs in a month. For the purpose of normalization, we divided the number of charge-offs by the number of active accounts in the same month to obtain the rate of charge-offs for that cohort at that point in time.



# Theory and calculation

We theorized that the biggest influencer for the monthly charge-off number is the number of months since the cohort was initialized. Thus, we calculated the average charge-off rate for the different cohorts for the number of months since it was initialized.



# Submission of Prediction

Our result from the previous step enabled us to calculate, iteratively, the number of charge-offs that should occur each month, according to our prediction.

	A	B
1	Month	accounts_charged_off
2	202002	60
3	202003	61
4	202004	60
5	202005	62
6	202006	66
7	202008	62
8	202009	64
9	202010	58
10	202011	59
11	202012	60
12	202101	58
13		

# Possible improvements on this model

Due to time constraint, we weren't able to explore the macrodata on this model. It could be worth looking into.

The calculations used to build this model were simple, as it emphasized on the average value for the monthly charge-offs for each cohort for a given number of months since it was initiated. It will be worth exploring more complex models in machine learning to see if there are intricacies that a rather simple model cannot explore.



# MLR Approach

- In file “mlr\_approach.ipynb”
- Steps:
  - Data cleaning and processing
    - Merge some of the categorical columns into one
    - Retrieve data from **macro\_data** to the trn\_data using the mth\_code column
  - Forward Selection





# MLR Approach

- Aggregate due\_account, due\_balance columns (as well as other categorical columns)

```
Code | Markdown | Run All | Clear All Outputs | Outline | Select Kernel
Python 3.7.6

trn['overdue_months'] = 0
trn['overdue_amount'] = 0

for n in range(2, 9):
    column_name = f'due_account_{n}'
    column_name_am = f'due_balance_{n}'
    trn['overdue_months'] += n*trn[column_name]
    trn['overdue_amount'] += trn[column_name_am]

# Check the result
print(trn[['due_account_2', 'due_account_3', 'due_account_4', 'due_account_5', 'due_account_6', 'due_account_7', 'due_account_8', 'overdue_months']])
```

[9]

	due_account_2	due_account_3	due_account_4	due_account_5	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	1.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	
...	...	...	...	...	
5758218	0.0	0.0	0.0	0.0	
5758219	1.0	0.0	0.0	0.0	
5758220	0.0	0.0	0.0	0.0	
5758221	0.0	0.0	0.0	0.0	
5758222	0.0	0.0	0.0	0.0	

	due_account_6	due_account_7	due_account_8	overdue_months
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	2.0

# MLR Approach

- Create new columns in trn\_data based on the data in the macro\_data, aggregate

```
macro_recent.tail()
```

	Mnemonic	M_FYPSAVQ.IUSA	FYPCPICA_FEDB.IUSA	M_FGDP.IUSA	M_FLBR.IUSA	M_FGDP\$.IUSA
242	201908	8.7	56918.9553	21548.02532	3.7	19150.28651
243	201909	8.9	57154.14298	21565.45763	3.5	19129.92383
244	201910	8.9	57397.55421	21597.53444	3.6	19112.85692
245	201911	8.9	57651.0369	21683.88241	3.6	19182.00586
246	201912	8.3	57916.66026	21838.1919	3.6	19352.21806

# MLR Approach

- In file “mlr\_approach.ipynb”
- Steps:
  - Data cleaning and processing
    - Merge some of the categorical columns into one
    - Retrieve data from **macro\_data** to the trn\_data using the *mth\_code* column
  - Forward Selection
  - Final Model
- Have the final model, which is able to predict the amount of charge off when providing one row of data
  - Fail to aggregate the prediction on a monthly basis



# MLR Approach

## Stepwise Forward Feature Selection

```
x=merged_trn[['mob']]
y=merged_trn['principal_amt_chrg_off']
lm=sm.OLS(y,x)
results=lm.fit()
print(results.summary())
```

```

=====
                        OLS Regression Results
=====
Dep. Variable:      principal_amt_chrg_off    R-squared (uncentered):      0.000
Model:              OLS                     Adj. R-squared (uncentered):  0.000
Method:             Least Squares           F-statistic:                 948.2
Date:               Sun, 26 Mar 2023        Prob (F-statistic):         3.46e-208
Time:              02:55:25                Log-Likelihood:             -3.6387e+07
No. Observations:   5758223                AIC:                       7.277e+07
Df Residuals:       5758222                BIC:                       7.277e+07
Df Model:           1
Covariance Type:    nonrobust

=====

```

	coef	std err	t	P> t	[0.025	0.975]
mob	0.0168	0.001	30.792	0.000	0.016	0.018

```

=====
Omnibus:            20726189.457    Durbin-Watson:           1.273
Prob(Omnibus):      0.000          Jarque-Bera (JB):    35361202096100.234
Skew:               73.401          Prob(JB):            0.00
Kurtosis:           12142.286       Cond. No.             1.00
=====
```

# MLR Approach

## Prediction using MLR Model

(Did not aggregate on monthly basis )

```
> result=results.predict(forecast_data)
> result[0:50]
[199]
... Output exceeds the size limit. Open the full output data in a text editor
0      0.000000
1      0.000000
2      0.000000
3      0.000000
4      0.000000
5      0.000000
6      0.000000
7      0.000000
8      0.000000
9      0.000000
10     0.000000
11     0.000000
12     0.000000
13     0.000000
14    64.239077
15     0.000000
16     0.000000
17     0.000000
18     0.000000
19     0.000000
20     0.000000
21     0.000000
22     0.000000
23     0.000000
24     0.000000
...
46     0.000000
47     0.000000
48     0.000000
49     0.000000
dtype: float64
```

# MLR Approach

- In file “mlr\_approach.ipynb”
- Steps:
  - Data cleaning and processing
    - Merge some of the categorical columns into one
    - Retrieve data from **macro\_data** to the trn\_data using the *mth\_code* column
  - Forward Selection
  - Final Model
- Have the final model, which is able to predict the amount of charge off when providing one row of data
  - Fail to aggregate the prediction on a monthly basis



# Challenges

- First time using python to build models
- Data-processing took up tremendous amount of time
- Adjust working time in one and a half day



# Future Studies

- Explore more using python when processing, analyzing, and modeling using dataset in real-world scenario
  - Reviewing statistical terms and model measurement
  - Try to implement complex model with provided data
- Use more of our knowledge to contribute to solve problem in reality





# Acknowledgment

Thank you Brendan Rhyno for being a wonderful TA for Phys 436 for Chieh, Soroush, and Jason. We were able to understand non-linear optics and wave guides thanks to you! We dedicated our team name to you in your spirit of spending hours and hours working on a complex problem, and we wanted to follow that spirit in this datathon :)

