RISK PREDICTION

CASSANDRA

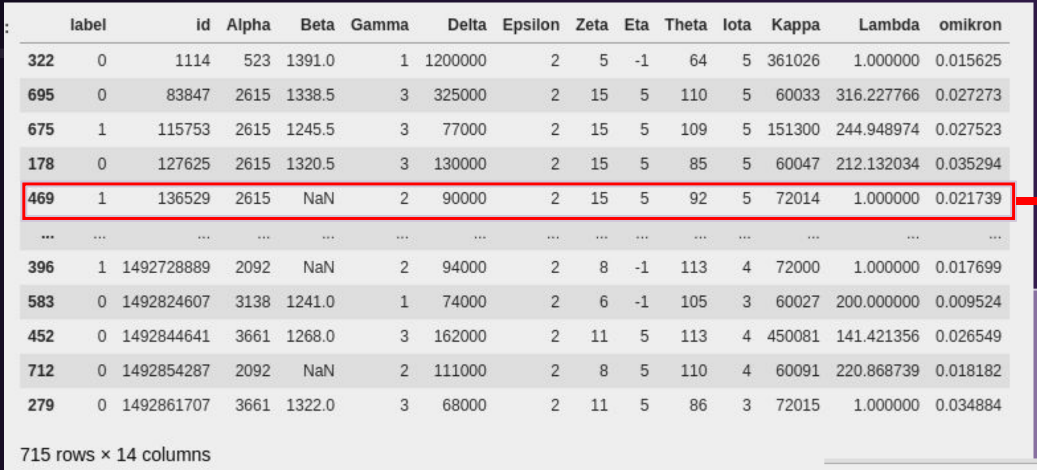
OUR APPROACH

The entire pipeline for prediction

PREDICTION PIPELINE

1. Data Cleaning
2. Data Analysis and Visualization
3. Merging Datasets
4. Feature Engineering
5. Handling Imbalanced dataset
6. Model Building
7. Hyperparameter Optimization
8. Prediction

A Minute to analyze the Data

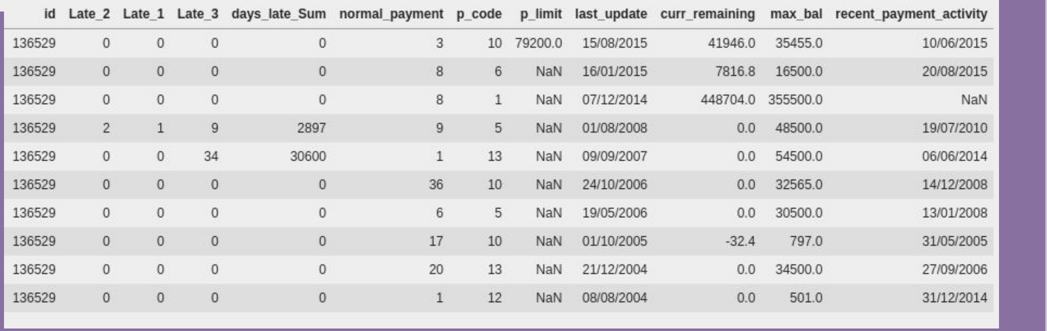


	label	id	Alpha	Beta	Gamma	Delta	Epsilon	Zeta	Eta	Theta	Iota	Kappa	Lambda	omikron
322	0	1114	523	1391.0	1	1200000	2	5	-1	64	5	361026	1.000000	0.015625
695	0	83847	2615	1338.5	3	325000	2	15	5	110	5	60033	316.227766	0.027273
675	1	115753	2615	1245.5	3	77000	2	15	5	109	5	151300	244.948974	0.027523
178	0	127625	2615	1320.5	3	130000	2	15	5	85	5	60047	212.132034	0.035294
469	1	136529	2615	NaN	2	90000	2	15	5	92	5	72014	1.000000	0.021739
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
396	1	1492728889	2092	NaN	2	94000	2	8	-1	113	4	72000	1.000000	0.017699
583	0	1492824607	3138	1241.0	1	74000	2	6	-1	105	3	60027	200.000000	0.009524
452	0	1492844641	3661	1268.0	3	162000	2	11	5	113	4	450081	141.421356	0.026549
712	0	1492854287	2092	NaN	2	111000	2	8	5	110	4	60091	220.868739	0.018182
279	0	1492861707	3661	1322.0	3	68000	2	11	5	86	3	72015	1.000000	0.034884

715 rows × 14 columns

Hierarchical connection between the datasets

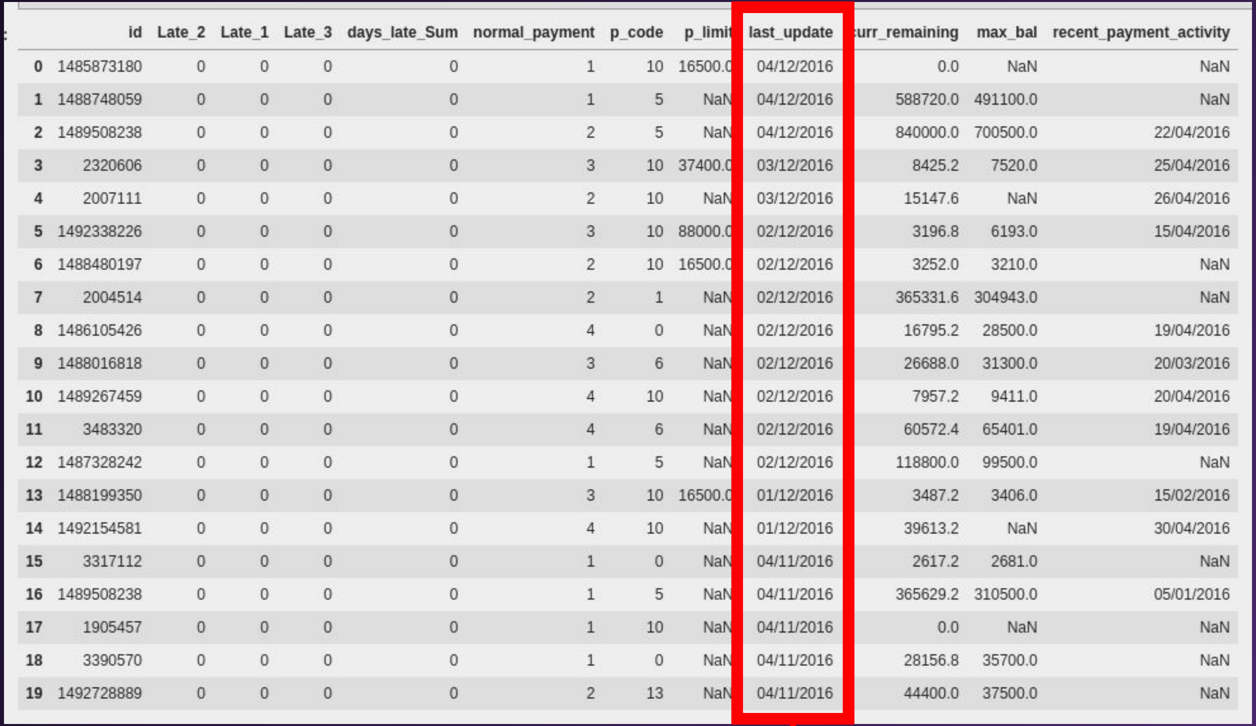
DataBase



id	Late_2	Late_1	Late_3	days_late_Sum	normal_payment	p_code	p_limit	last_update	curr_remaining	max_bal	recent_payment_activity
136529	0	0	0	0	3	10	79200.0	15/08/2015	41946.0	35455.0	10/06/2015
136529	0	0	0	0	8	6	NaN	16/01/2015	7816.8	16500.0	20/08/2015
136529	0	0	0	0	8	1	NaN	07/12/2014	448704.0	355500.0	NaN
136529	2	1	9	2897	9	5	NaN	01/08/2008	0.0	48500.0	19/07/2010
136529	0	0	34	30600	1	13	NaN	09/09/2007	0.0	54500.0	06/06/2014
136529	0	0	0	0	36	10	NaN	24/10/2006	0.0	32565.0	14/12/2008
136529	0	0	0	0	6	5	NaN	19/05/2006	0.0	30500.0	13/01/2008
136529	0	0	0	0	17	10	NaN	01/10/2005	-32.4	797.0	31/05/2005
136529	0	0	0	0	20	13	NaN	21/12/2004	0.0	34500.0	27/09/2006
136529	0	0	0	0	1	12	NaN	08/08/2004	0.0	501.0	31/12/2014

Payment History for each ID

Key Insight



	id	Late_2	Late_1	Late_3	days_late_Sum	normal_payment	p_code	p_limit	last_update	urr_remaining	max_bal	recent_payment_activity
0	1485873180	0	0	0	0	1	10	16500.0	04/12/2016	0.0	NaN	NaN
1	1488748059	0	0	0	0	1	5	NaN	04/12/2016	588720.0	491100.0	NaN
2	1489508238	0	0	0	0	2	5	NaN	04/12/2016	840000.0	700500.0	22/04/2016
3	2320606	0	0	0	0	3	10	37400.0	03/12/2016	8425.2	7520.0	25/04/2016
4	2007111	0	0	0	0	2	10	NaN	03/12/2016	15147.6	NaN	26/04/2016
5	1492338226	0	0	0	0	3	10	88000.0	02/12/2016	3196.8	6193.0	15/04/2016
6	1488480197	0	0	0	0	2	10	16500.0	02/12/2016	3252.0	3210.0	NaN
7	2004514	0	0	0	0	2	1	NaN	02/12/2016	365331.6	304943.0	NaN
8	1486105426	0	0	0	0	4	0	NaN	02/12/2016	16795.2	28500.0	19/04/2016
9	1488016818	0	0	0	0	3	6	NaN	02/12/2016	26688.0	31300.0	20/03/2016
10	1489267459	0	0	0	0	4	10	NaN	02/12/2016	7957.2	9411.0	20/04/2016
11	3483320	0	0	0	0	4	6	NaN	02/12/2016	60572.4	65401.0	19/04/2016
12	1487328242	0	0	0	0	1	5	NaN	02/12/2016	118800.0	99500.0	NaN
13	1488199350	0	0	0	0	3	10	16500.0	01/12/2016	3487.2	3406.0	15/02/2016
14	1492154581	0	0	0	0	4	10	NaN	01/12/2016	39613.2	NaN	30/04/2016
15	3317112	0	0	0	0	1	0	NaN	04/11/2016	2617.2	2681.0	NaN
16	1489508238	0	0	0	0	1	5	NaN	04/11/2016	365629.2	310500.0	05/01/2016
17	1905457	0	0	0	0	1	10	NaN	04/11/2016	0.0	NaN	NaN
18	3390570	0	0	0	0	1	0	NaN	04/11/2016	28156.8	35700.0	NaN
19	1492728889	0	0	0	0	2	13	NaN	04/11/2016	44400.0	37500.0	NaN

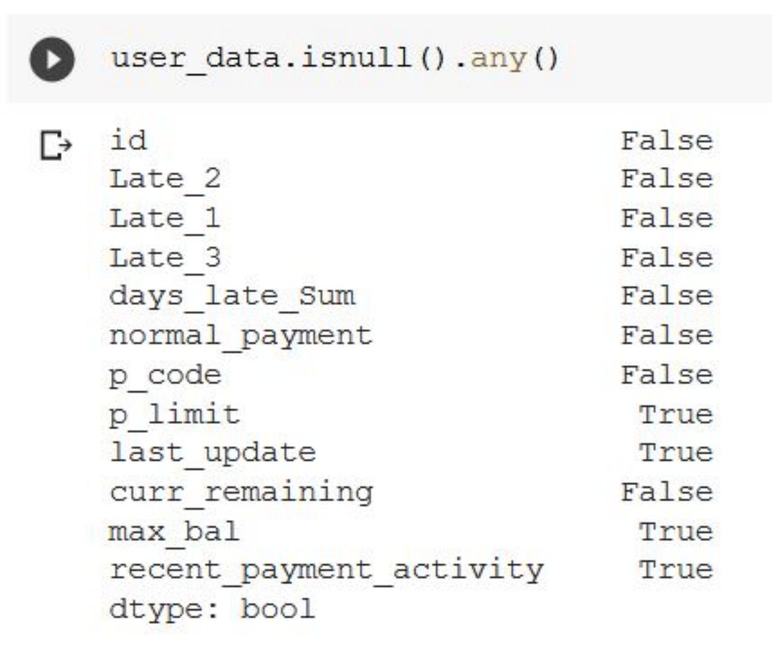
The table is sorted with respect to last_update

Data Cleaning/Analysis

User Payment History

4 columns has NaN values:

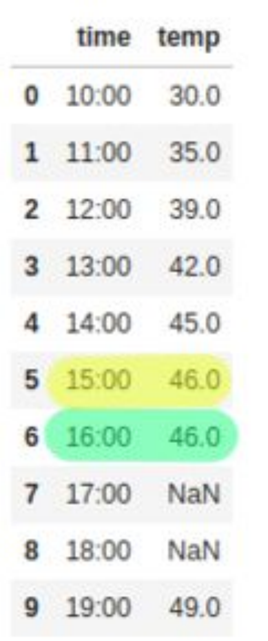
- last_update
- recent_payment_activity
- p_limit
- max_bal



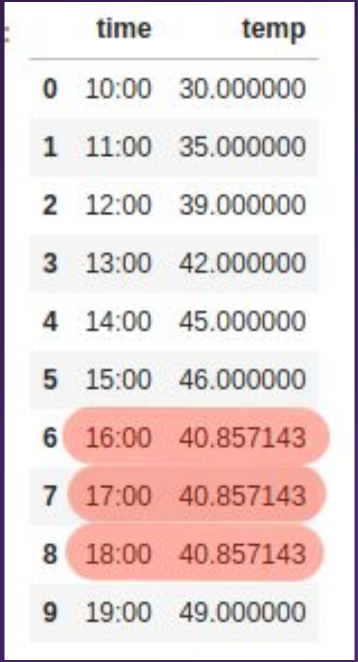
```
user_data.isnull().any()
```

```
id                         False
Late_2                     False
Late_1                     False
Late_3                     False
days_late_Sum              False
normal_payment             False
p_code                     False
p_limit                     True
last_update                 True
curr_remaining             False
max_bal                     True
recent_payment_activity     True
dtype: bool
```

BFill and FFill Vs Mean



	time	temp
0	10:00	30.0
1	11:00	35.0
2	12:00	39.0
3	13:00	42.0
4	14:00	45.0
5	15:00	46.0
6	16:00	46.0
7	17:00	NaN
8	18:00	NaN
9	19:00	49.0



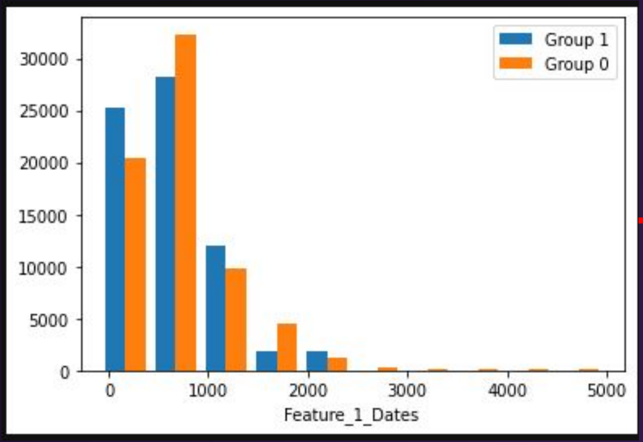
	time	temp
0	10:00	30.000000
1	11:00	35.000000
2	12:00	39.000000
3	13:00	42.000000
4	14:00	45.000000
5	15:00	46.000000
6	16:00	40.857143
7	17:00	40.857143
8	18:00	40.857143
9	19:00	49.000000

FFill improves the temporal consistency

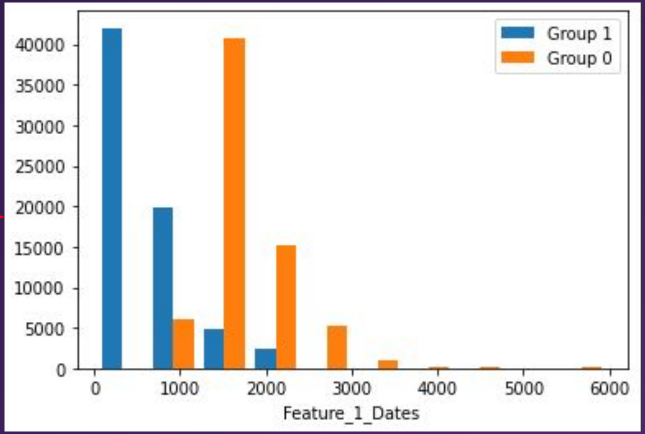
Mean fill giving discontinuous values.

Improvement with FFill and BFill

Our features which were dependent on “last_update” and “recent_activity” benefited with this method of fillna



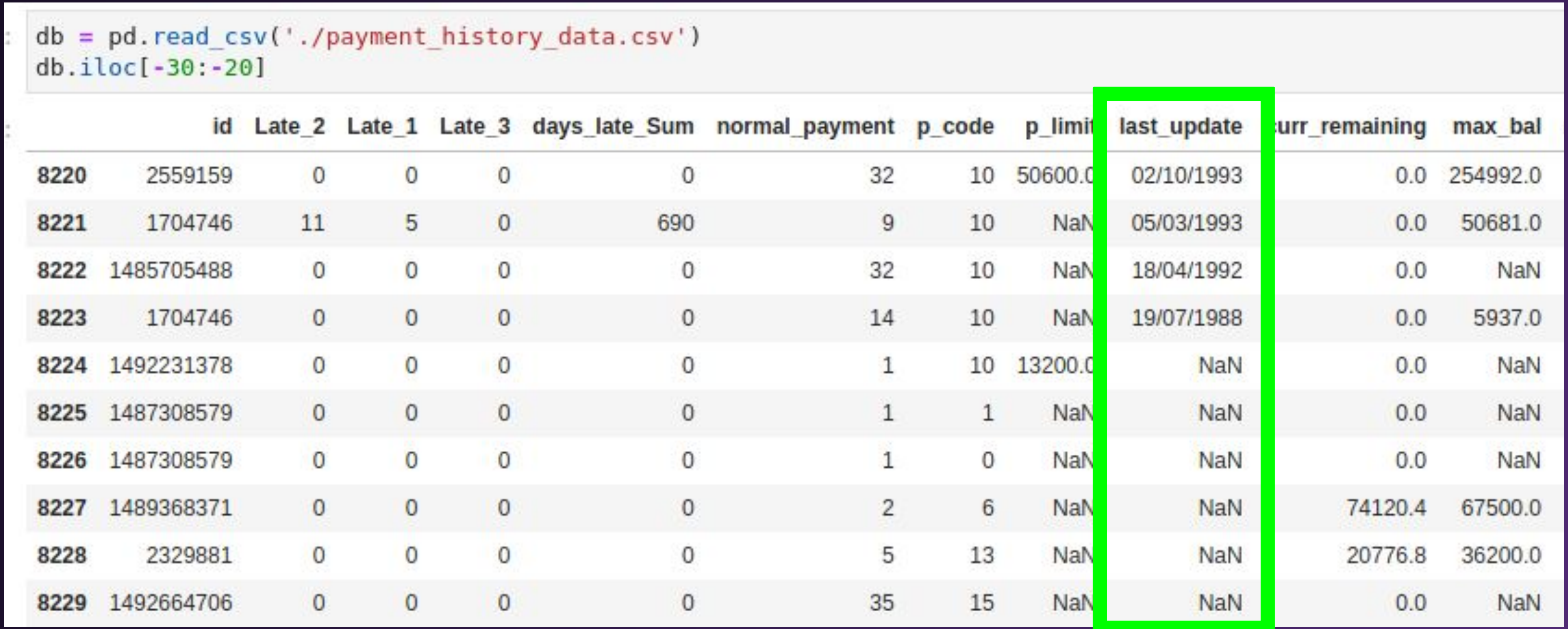
Filled with mean



Used FFill

Cleaning “last_update”

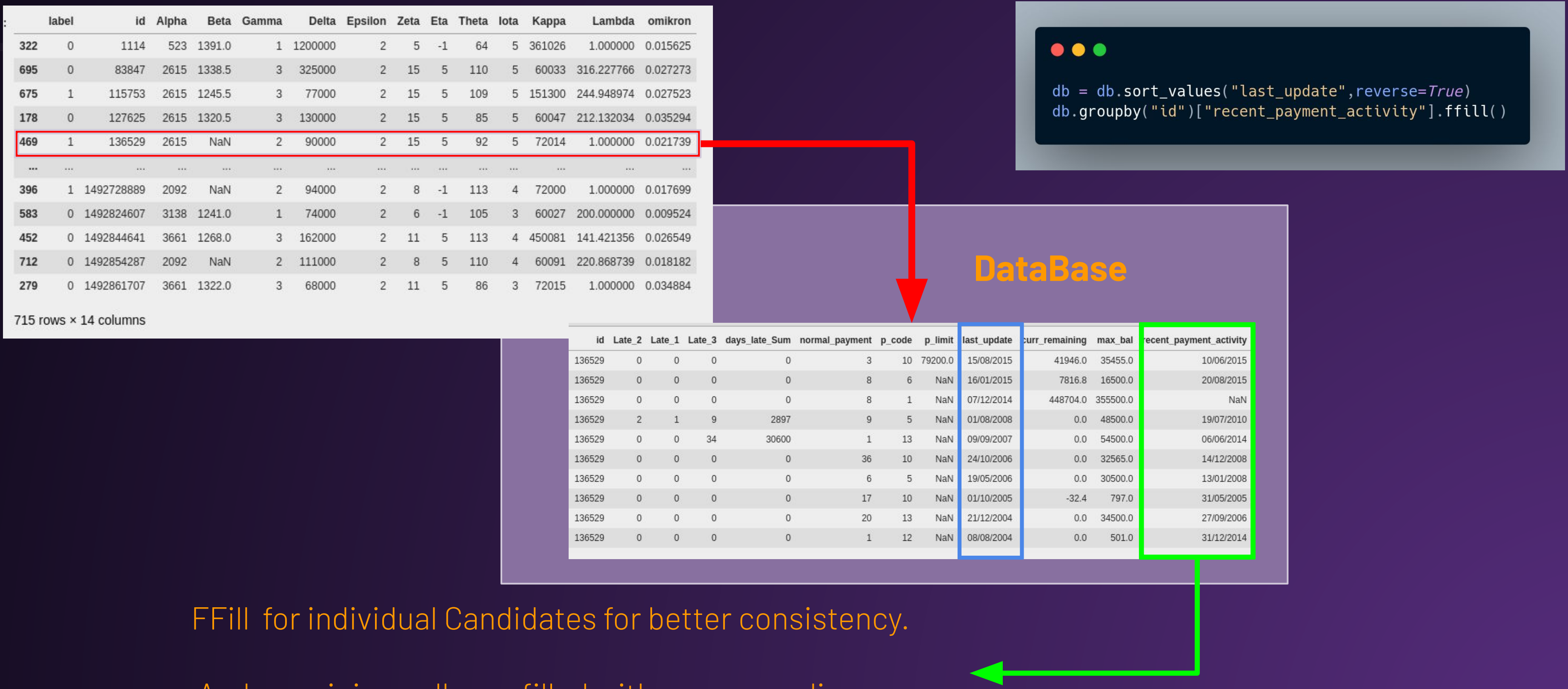
- Applied simple FFill as only the last 20 rows contained NaN. so the best estimate is the oldest known value



```
db = pd.read_csv('./payment_history_data.csv')
db.iloc[-30:-20]
```

	id	Late_2	Late_1	Late_3	days_late_Sum	normal_payment	p_code	p_limit	last_update	urr_remaining	max_bal
8220	2559159	0	0	0	0	32	10	50600.0	02/10/1993	0.0	254992.0
8221	1704746	11	5	0	690	9	10	NaN	05/03/1993	0.0	50681.0
8222	1485705488	0	0	0	0	32	10	NaN	18/04/1992	0.0	NaN
8223	1704746	0	0	0	0	14	10	NaN	19/07/1988	0.0	5937.0
8224	1492231378	0	0	0	0	1	10	13200.0	NaN	0.0	NaN
8225	1487308579	0	0	0	0	1	1	NaN	NaN	0.0	NaN
8226	1487308579	0	0	0	0	1	0	NaN	NaN	0.0	NaN
8227	1489368371	0	0	0	0	2	6	NaN	NaN	74120.4	67500.0
8228	2329881	0	0	0	0	5	13	NaN	NaN	20776.8	36200.0
8229	1492664706	0	0	0	0	35	15	NaN	NaN	0.0	NaN

Cleaning Recent_payment_activity



	label	id	Alpha	Beta	Gamma	Delta	Epsilon	Zeta	Eta	Theta	Iota	Kappa	Lambda	omikron
322	0	1114	523	1391.0	1	1200000	2	5	-1	64	5	361026	1.000000	0.015625
695	0	83847	2615	1338.5	3	325000	2	15	5	110	5	60033	316.227766	0.027273
675	1	115753	2615	1245.5	3	77000	2	15	5	109	5	151300	244.948974	0.027523
178	0	127625	2615	1320.5	3	130000	2	15	5	85	5	60047	212.132034	0.035294
469	1	136529	2615	NaN	2	90000	2	15	5	92	5	72014	1.000000	0.021739
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
396	1	1492728889	2092	NaN	2	94000	2	8	-1	113	4	72000	1.000000	0.017699
583	0	1492824607	3138	1241.0	1	74000	2	6	-1	105	3	60027	200.000000	0.009524
452	0	1492844641	3661	1268.0	3	162000	2	11	5	113	4	450081	141.421356	0.026549
712	0	1492854287	2092	NaN	2	111000	2	8	5	110	4	60091	220.868739	0.018182
279	0	1492861707	3661	1322.0	3	68000	2	11	5	86	3	72015	1.000000	0.034884

715 rows × 14 columns

```
db = db.sort_values("last_update",reverse=True)
db.groupby("id")["recent_payment_activity"].ffill()
```

DataBase

id	Late_2	Late_1	Late_3	days_late_Sum	normal_payment	p_code	p_limit	last_update	curr_remaining	max_bal	recent_payment_activity
136529	0	0	0	0	3	10	79200.0	15/08/2015	41946.0	35455.0	10/06/2015
136529	0	0	0	0	8	6	NaN	16/01/2015	7816.8	16500.0	20/08/2015
136529	0	0	0	0	8	1	NaN	07/12/2014	448704.0	355500.0	NaN
136529	2	1	9	2897	9	5	NaN	01/08/2008	0.0	48500.0	19/07/2010
136529	0	0	34	30600	1	13	NaN	09/09/2007	0.0	54500.0	06/06/2014
136529	0	0	0	0	36	10	NaN	24/10/2006	0.0	32565.0	14/12/2008
136529	0	0	0	0	6	5	NaN	19/05/2006	0.0	30500.0	13/01/2008
136529	0	0	0	0	17	10	NaN	01/10/2005	-32.4	797.0	31/05/2005
136529	0	0	0	0	20	13	NaN	21/12/2004	0.0	34500.0	27/09/2006
136529	0	0	0	0	1	12	NaN	08/08/2004	0.0	501.0	31/12/2014

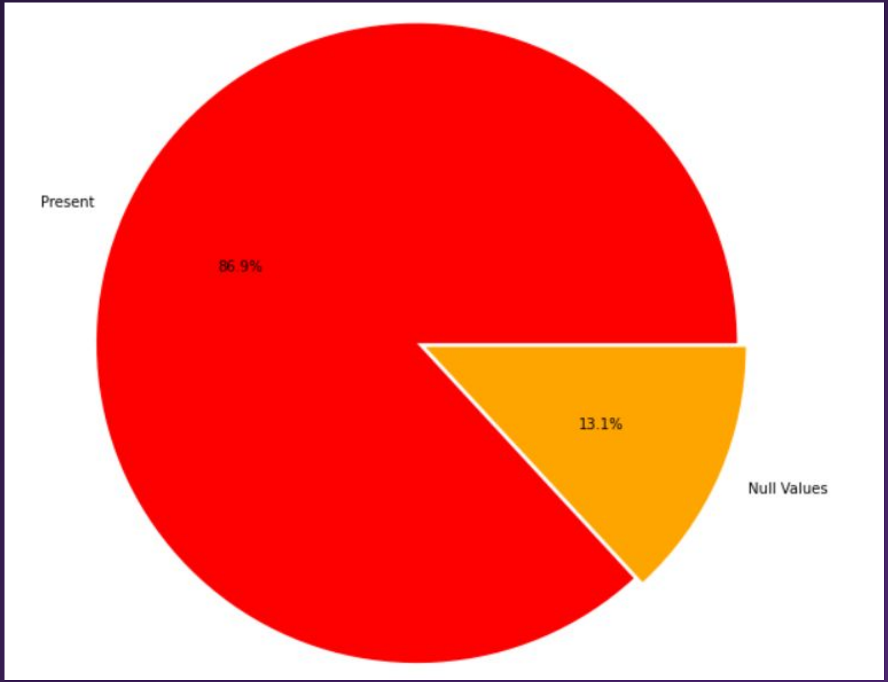
FFill for individual Candidates for better consistency.

And remaining nulls are filled with corresponding "last_update".

Data Cleaning/Visualization

User Training Data

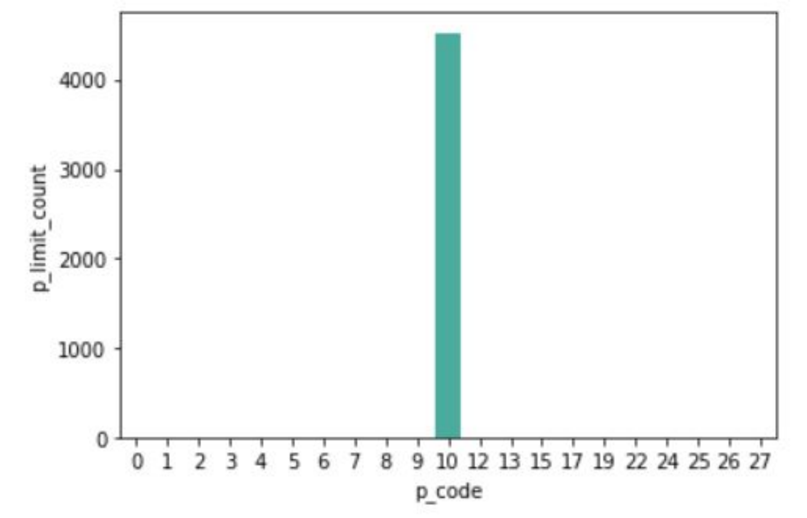
- Large proportion of values in the column “Beta” were missing
- Filled with mean, since the meaning of the column was not given
- Rest of the dataset was complete

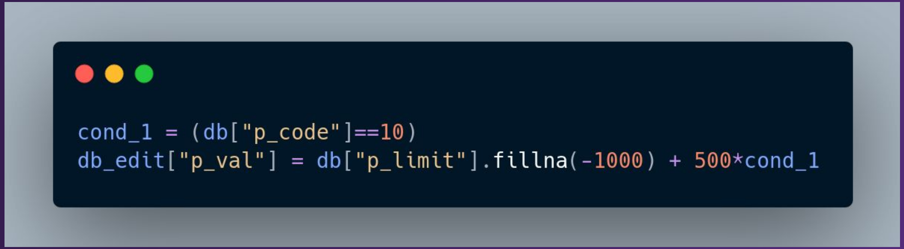


Data Cleaning

P_limit Data Field

- Refers to the credit limit of the product
- Wherever this field was not null, the corresponding p_code (code of the product) was 10
- If p_code was 10, we set p_limit to 1 else -1



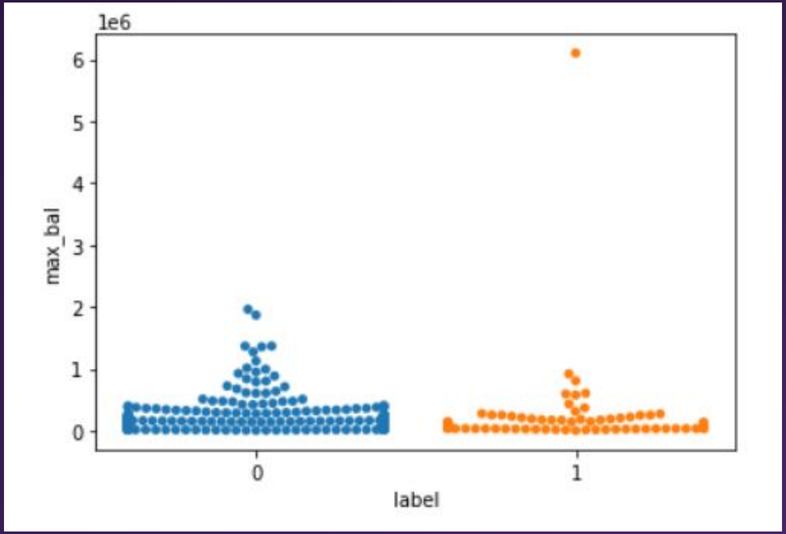


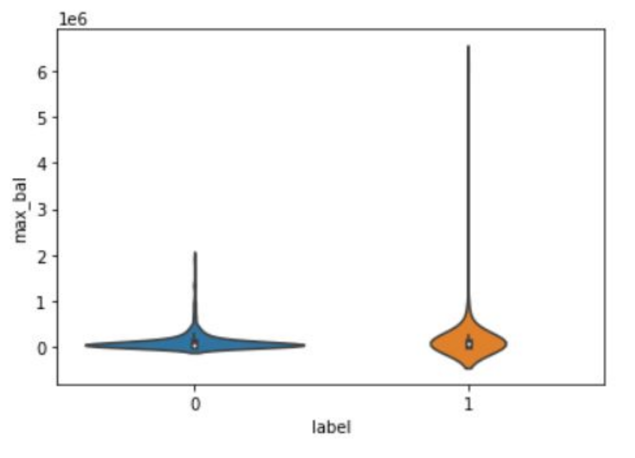
```
cond_1 = (db["p_code"]==10)
db_edit["p_val"] = db["p_limit"].fillna(-1000) + 500*cond_1
```

Data Cleaning

Max Bal Data Field

- Refers to the maximum balance of the user
- Can't simply fill the mean of the entire column
- Mean of the maximum balances of that particular user





Feature Extraction

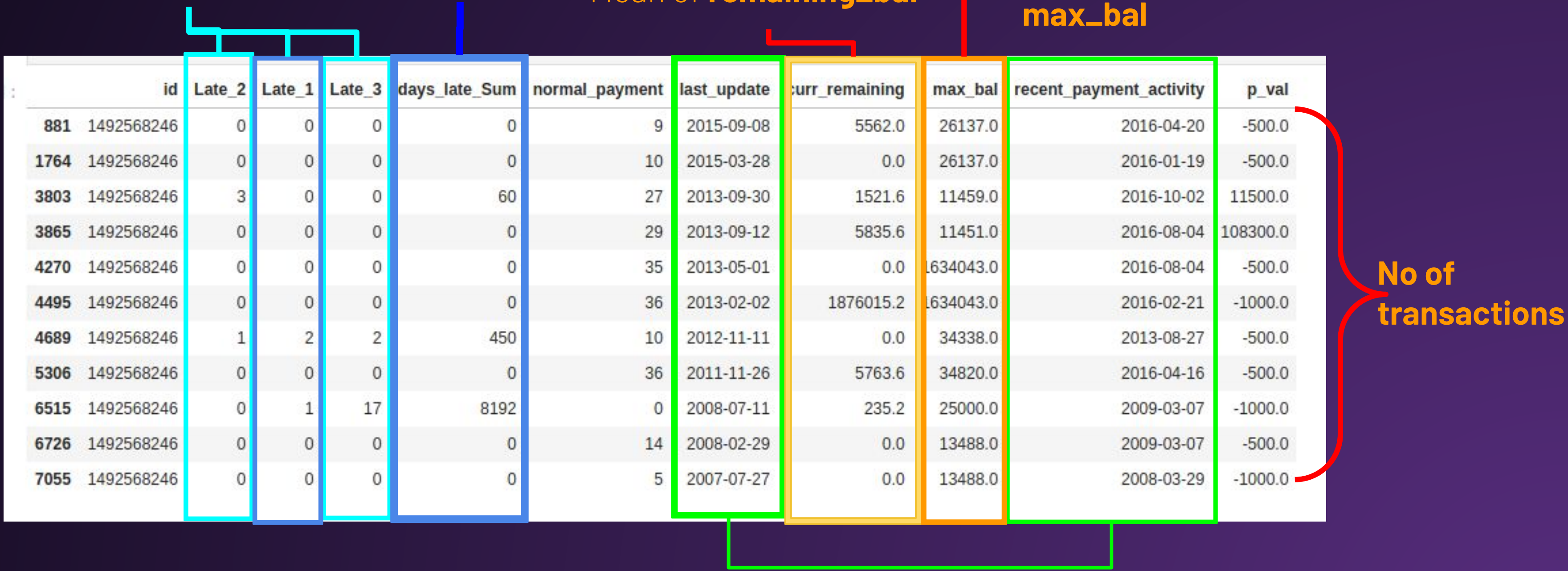
For each ID in User_data we get the features from payment_history as follows

Max of each type of **last_count**

* time the **no of days**

Mean of **remaining_bal**

Mean and Std deviation of **max_bal**



	id	Late_2	Late_1	Late_3	days_late_Sum	normal_payment	last_update	curr_remaining	max_bal	recent_payment_activity	p_val
881	1492568246	0	0	0	0	9	2015-09-08	5562.0	26137.0	2016-04-20	-500.0
1764	1492568246	0	0	0	0	10	2015-03-28	0.0	26137.0	2016-01-19	-500.0
3803	1492568246	3	0	0	60	27	2013-09-30	1521.6	11459.0	2016-10-02	11500.0
3865	1492568246	0	0	0	0	29	2013-09-12	5835.6	11451.0	2016-08-04	108300.0
4270	1492568246	0	0	0	0	35	2013-05-01	0.0	1634043.0	2016-08-04	-500.0
4495	1492568246	0	0	0	0	36	2013-02-02	1876015.2	1634043.0	2016-02-21	-1000.0
4689	1492568246	1	2	2	450	10	2012-11-11	0.0	34338.0	2013-08-27	-500.0
5306	1492568246	0	0	0	0	36	2011-11-26	5763.6	34820.0	2016-04-16	-500.0
6515	1492568246	0	1	17	8192	0	2008-07-11	235.2	25000.0	2009-03-07	-1000.0
6726	1492568246	0	0	0	0	14	2008-02-29	0.0	13488.0	2009-03-07	-500.0
7055	1492568246	0	0	0	0	5	2007-07-27	0.0	13488.0	2008-03-29	-1000.0

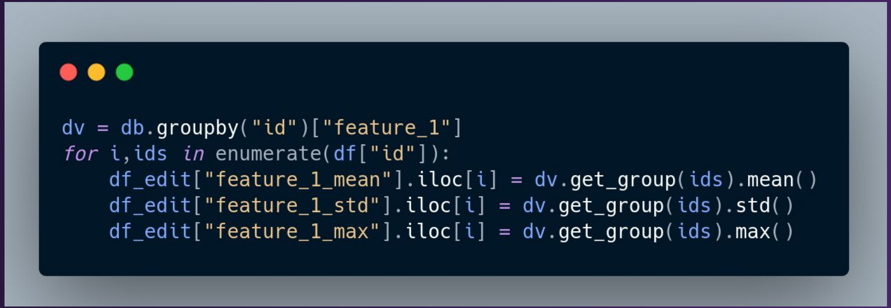
No of transactions

Difference of **Days** and a condition that if any last_update is less than recent_payment_activity then flag **ambiguity**

Merging Datasets

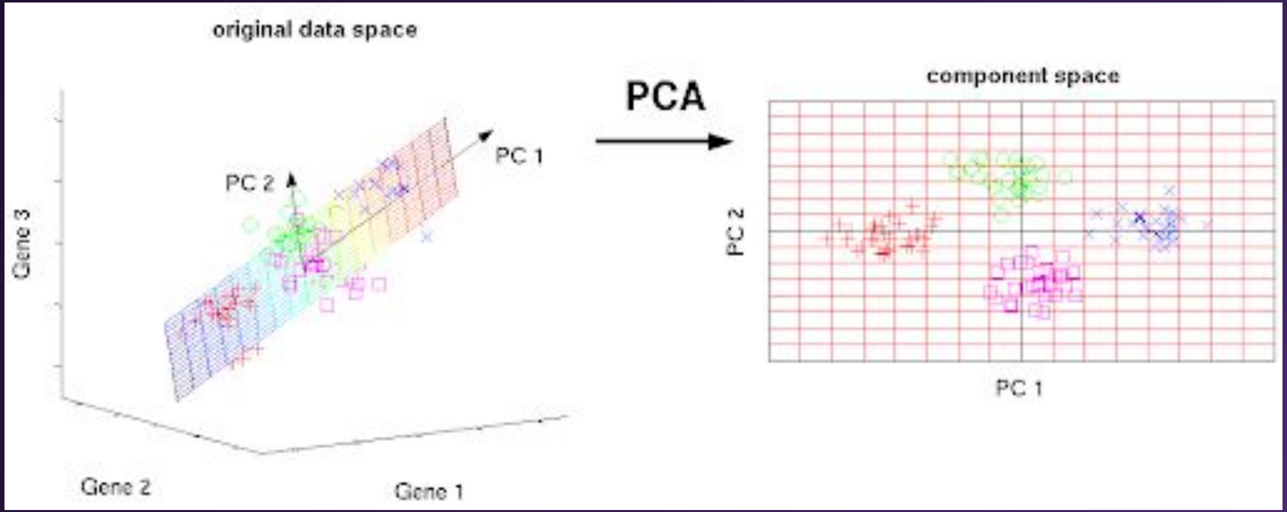
- Need of merging User Payment History
- Simply grouping by user id, and attaching user payment history to user ids common in both

Like so.,



```
dv = db.groupby("id")["feature_1"]
for i,ids in enumerate(df["id"]):
    df_edit["feature_1_mean"].iloc[i] = dv.get_group(ids).mean()
    df_edit["feature_1_std"].iloc[i] = dv.get_group(ids).std()
    df_edit["feature_1_max"].iloc[i] = dv.get_group(ids).max()
```

Principal Component Analysis



We reduced the no. of features using PCA to train more efficiently

Dataset Imbalance

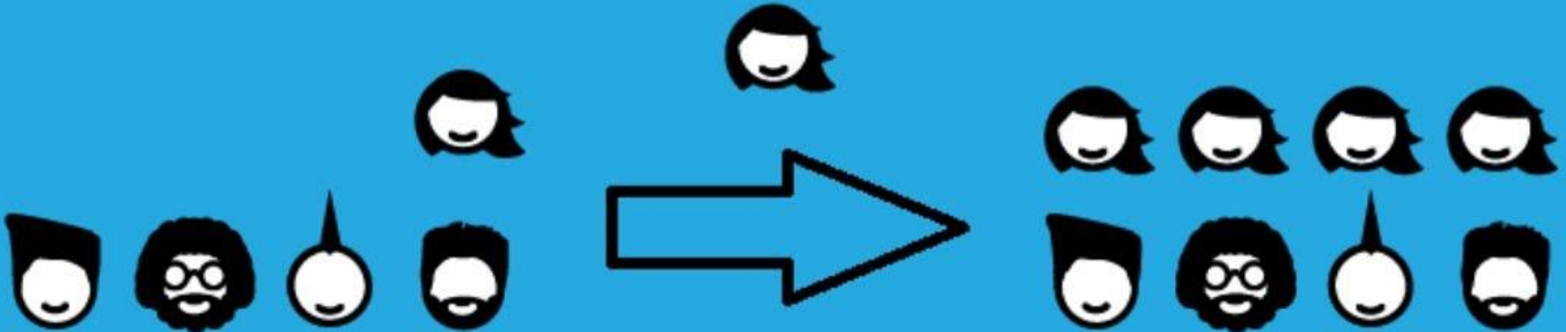
The biggest hurdle in risk prediction problems

Class Imbalance in dataset

- Might overlook minority class
- Accuracy doesn’t give the correct picture
- ONE SOLUTION:
- Simple oversampling of minority class
- Doesn’t add any extra information to the dataset
- Need something else to check this imbalance

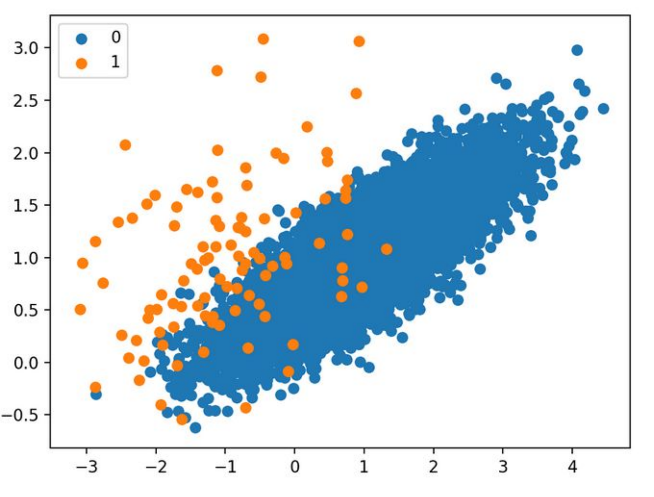
SMOTE

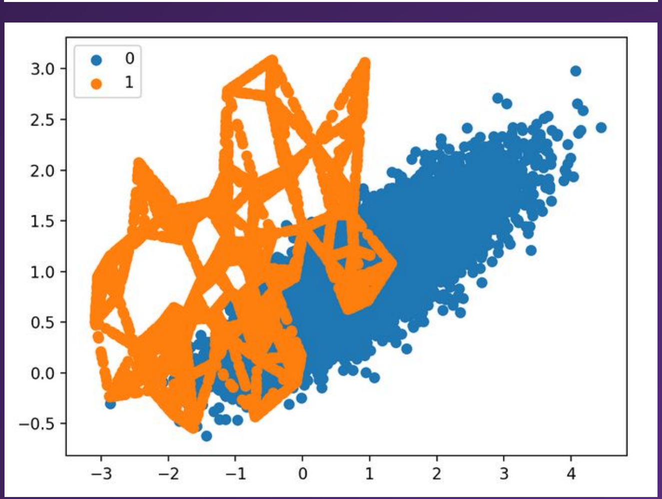
SYNTHETIC MINORITY OVER-SAMPLING TECHNIQUE



WHAT IS SMOTE?

- Data Augmentation
- Helps in increasing number of examples of minority class
- Diminishes class imbalance in dataset

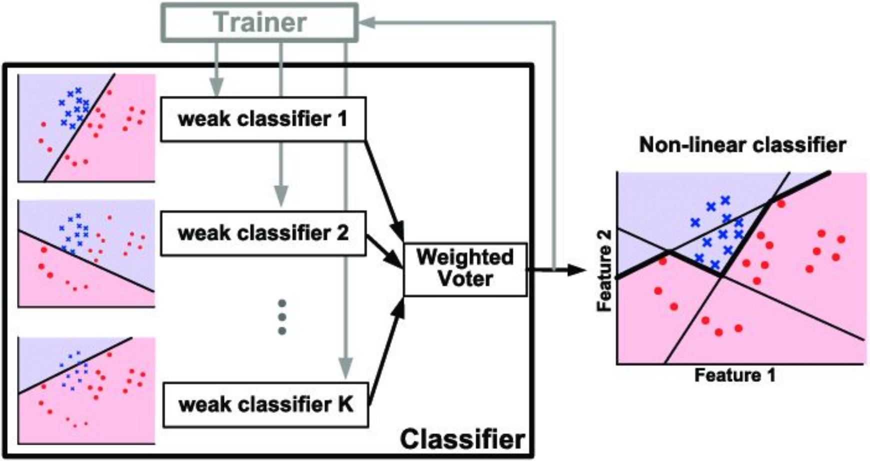




Model Building

Model Building

- Utilized the concept of Boosting
- Used an AdaBoost Classifier
- Base estimator in Adaboost was a Decision Tree Classifier



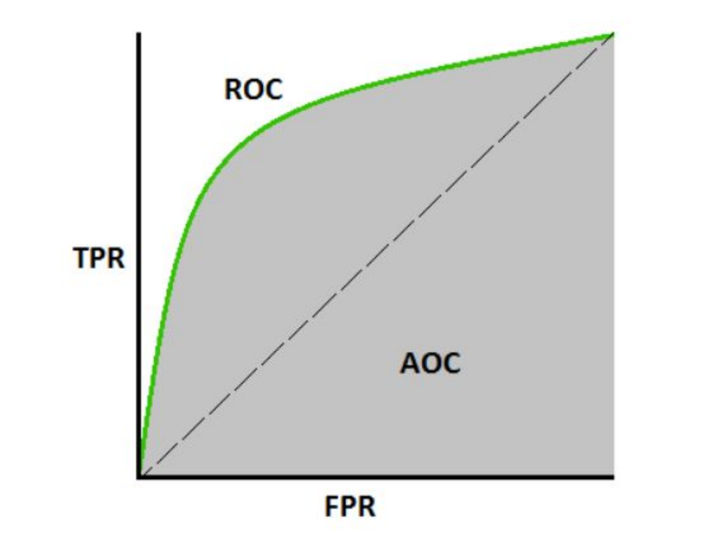
Hyperparameter Optimization

Hyperparameter Optimization

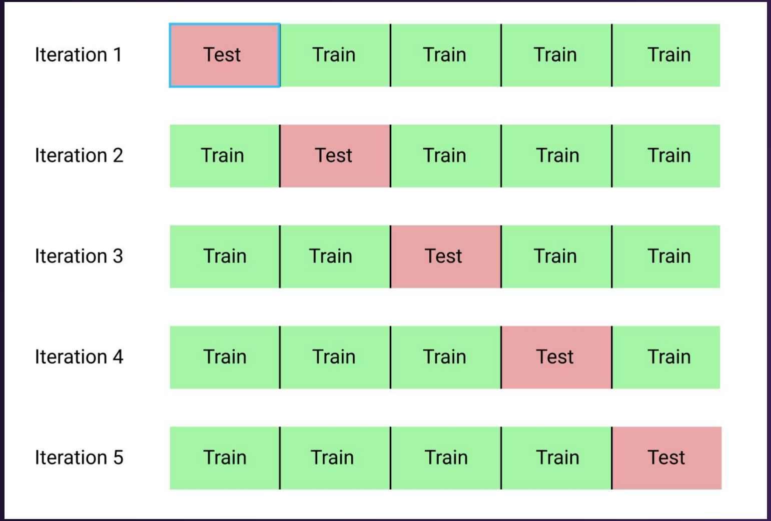
- Adaboost and Decision Tree Classifiers along with SMOTE have a a large number of hyperparameters
- Optuna Framework
- Define a range for all hyperparameters
- Run trials

What metric to use?

- K Fold Cross Validation
- Using ROC AUC Score



ROC AUC Score



Iteration 1	Test	Train	Train	Train	Train
Iteration 2	Train	Test	Train	Train	Train
Iteration 3	Train	Train	Test	Train	Train
Iteration 4	Train	Train	Train	Test	Train
Iteration 5	Train	Train	Train	Train	Test

K Fold Cross Validation

The Final Step

Training and Prediction

- Training with optimal hyperparameters
- Predicting on test dataset merged with the user payment history
- Submitting in hope of a better score :)