Thera Bank

Credit Card User Churn Prediction

PG-DSBA Project 6
Eric Green
May 2021

Background

The Thera bank recently saw a steep decline in the number of users of their credit card service. Credit cards are a good source of bank revenue via different kinds of fees charged e.g., annual fees, balance transfer fees, cash advance fees, late payment fees, foreign transaction fees and others. Some fees are charged on every user regardless of usage, while others are charged under specified usage circumstances.

Customers leaving credit card services lead to revenue loss for the bank, thus our objective is to analyze the data of customers and identify the those customers who are more likely to leave their credit card services and to understand which customer attributes are most common among them in order to intervene and prevent these customers from leaving.

Objectives

- ✓ Explore and visualize the dataset
- ✓ Build classification models to predict which customers are likely to churn
- ✓ Optimize classification models using appropriate techniques to improve performance
- ✓ Generate a set of insights and recommendations that will help the bank prevent lost revenue from credit card attrition

Raw Data Summary

BankChurners.csv shape: (10127, 21)

Raw Variables

0 CLIENTNUM 10127 non-null int64
1 Attrition_Flag 10127 non-null object
2 Customer_Age 10127 non-null int64
3 Gender 10127 non-null object
4 Dependent_count 10127 non-null int64
5 Education_Level 10127 non-null object
6 Marital_Status 10127 non-null object
7 Income_Category 10127 non-null object
8 Card_Category 10127 non-null object
9 Months_on_book 10127 non-null int64
10 Total_Relationship_Count 10127 non-null int64
11 Months_Inactive_12_mon 10127 non-null int64
12 Contacts_Count_12_mon 10127 non-null int64
13 Credit_Limit 10127 non-null float64
14 Total_Revolving_Bal 10127 non-null int64
15 Avg_Open_To_Buy 10127 non-null float64
16 Total_Amt_Chng_Q4_Q1 10127 non-null float64
17 Total_Trans_Amt 10127 non-null int64
18 Total_Trans_Ct 10127 non-null int64
19 Total_Ct_Chng_Q4_Q1 10127 non-null float64
20 Avg_Utilization_Ratio 10127 non-null float64

Null Values	Duplicates
CLIENTNUM 0 Attrition_Flag 0 Customer_Age 0 Gender 0 Dependent_count 0 Education_Level 0 Marital_Status 0 Income_Category 0 Card_Category 0 Months_on_book 0 Total_Relationship_Count 0 Months_Inactive_12_mon 0 Contacts_Count_12_mon 0 Credit_Limit 0 Total_Revolving_Bal 0 Avg_Open_To_Buy 0 Total_Amt_Chng_Q4_Q1 0 Total_Trans_Amt 0 Total_Trans_Ct 0 Total_Ct_Chng_Q4_Q1 0 Avg_Utilization_Ratio 0	CLIENTNUM 0 Attrition_Flag 0 Customer_Age 0 Gender 0 Dependent_count 0 Education_Level 0 Marital_Status 0 Income_Category 0 Card_Category 0 Months_on_book 0 Total_Relationship_Count 0 Months_Inactive_12_mon 0 Contacts_Count_12_mon 0 Credit_Limit 0 Total_Revolving_Bal 0 Avg_Open_To_Buy 0 Total_Amt_Chng_Q4_Q1 0 Total_Trans_Amt 0 Total_Trans_Ct 0 Total_Ct_Chng_Q4_Q1 0 Avg_Utilization_Ratio 0

Observations

- Raw data shape is 10127 rows x 21 columns
- Raw data is free of null values, NAs and duplicates
- Columns values will be inspected further to determine variants and subsequent cleansing requirements

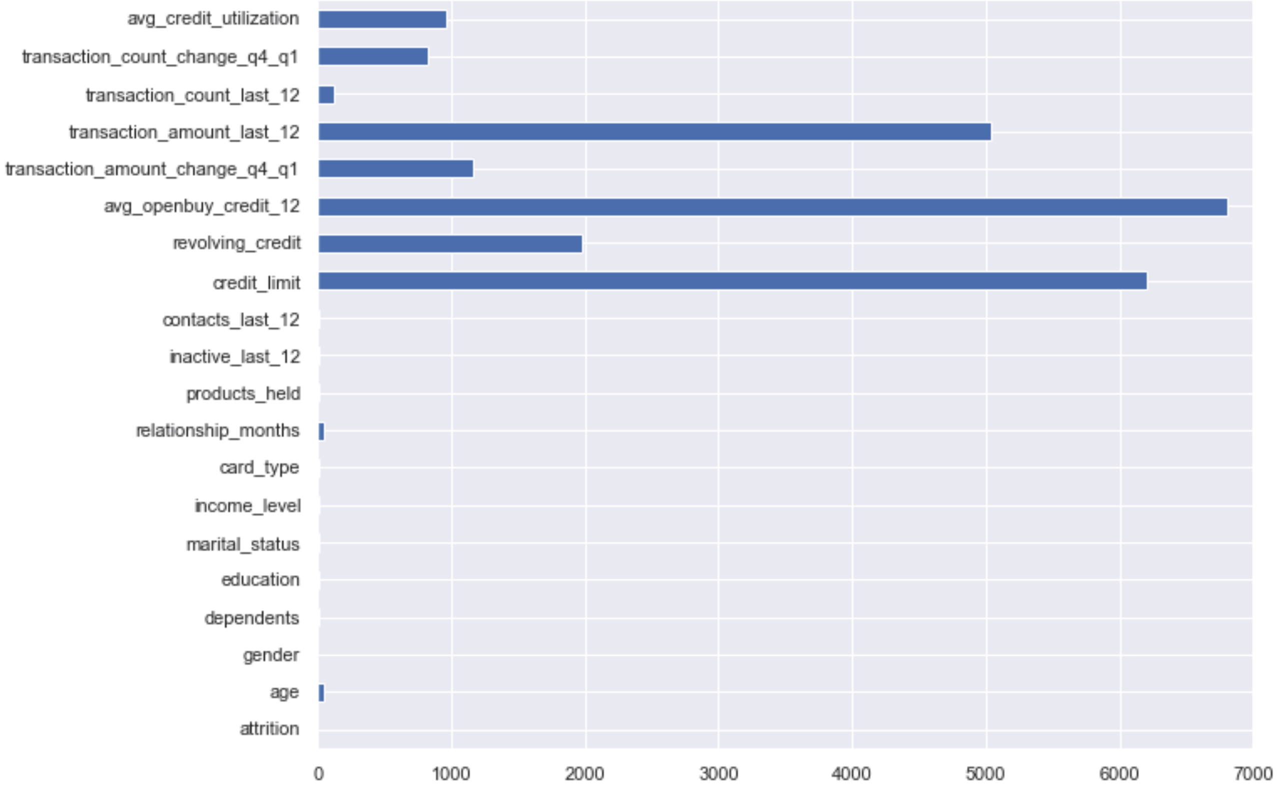
Raw Data Summary (continued)

Converted Variables

Variables given intuitive names to work with and converted to appropriate types

0 attrition 10127 non-null category
1 age 10127 non-null int64
2 gender 10127 non-null category
3 dependents 10127 non-null int64
4 education 10127 non-null category
5 marital_status 10127 non-null category
6 income_level 10127 non-null category
7 card_type 10127 non-null category
8 relationship_months 10127 non-null int64
9 products_held 10127 non-null int64
10 inactive_last_12 10127 non-null int64
11 contacts_last_12 10127 non-null int64
12 credit_limit 10127 non-null float64
13 revolving_credit 10127 non-null float64
14 avg_openbuy_credit_12 10127 non-null float64
15 transaction_amount_change_q4_q1 10127 non-null float64
16 transaction_amount_last_12 10127 non-null float64
17 transaction_count_last_12 10127 non-null float64
18 transaction_count_change_q4_q1 10127 non-null float64
19 avg_credit_utilization 10127 non-null float64

Variety of column values to understand categoricals & dense vectors



Observations - Raw Data

- BankChurners.csv shape: 10127 rows x 21 columns
- Nulls, NAs and duplicate observations are not an issue in the raw data set
- CLIENTNUM variable was removed as extraneous
- Columns are immediately converted to intuitive names that can be understood directly from the name
- We itemize the uniqies values across all columns to understand diversity of data present and how the variables might be visualized
- Columns are divided into logical type sets (cat, int, float) by how they will be visualized in the EDA and potentially modelled
- Our target variables is attrition
- Unknown values must be cleansed in education, martial_status and income_level
- Summary statistics show extreme variation (std) in credit_limit, revolving_credit, avg_openbuy_credit_12 and transaction_amount_last_12 variables

Data Preprocessing

Raw Data

1. product_conversion
2. customer_age
3. contact_type
4. city_scale
5. pitch_duration
6. occupation
7. gender
8. visitors
9. followups
10. pitched_product
11. property_rating
12. marital_status
13. avg_annual_trips
14. passport
15. pitch_sat_score
16. car_owner
17. job_role
18. monthly_income

Data Preprocessing

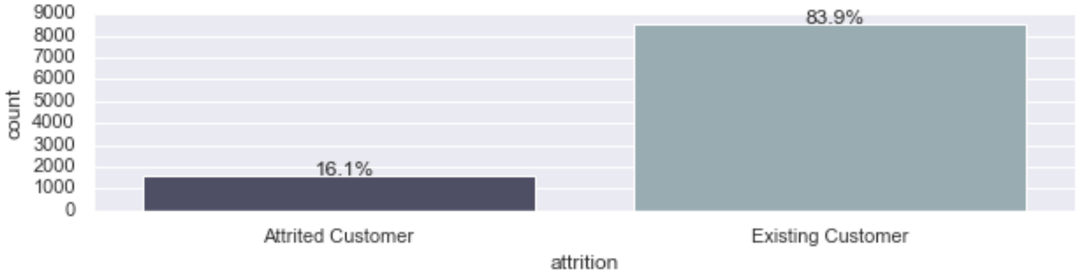
Modeling shape: (10127, 23)

Preprocessed Data
(cleansed & one-hot encoded)

1. attrition int64
2. age int64
3. gender int64
4. dependents int64
5. products_held int64
6. inactive_last_12 int64
7. contacts_last_12 int64
8. credit_limit float64
9. revolving_credit float64
10. transaction_amount_change_q4_q1 float64
11. transaction_amount_last_12 float64
12. transaction_count_last_12 float64
13. transaction_count_change_q4_q1 float64
14. avg_credit_utilization float64
15. education_tier int64
16. income_tier int64
17. marital_status_Divorced uint8
18. marital_status_Married uint8
19. marital_status_Single uint8
20. card_type_Blue uint8
21. card_type_Gold uint8
22. card_type_Platinum uint8
23. card_type_Silver uint8

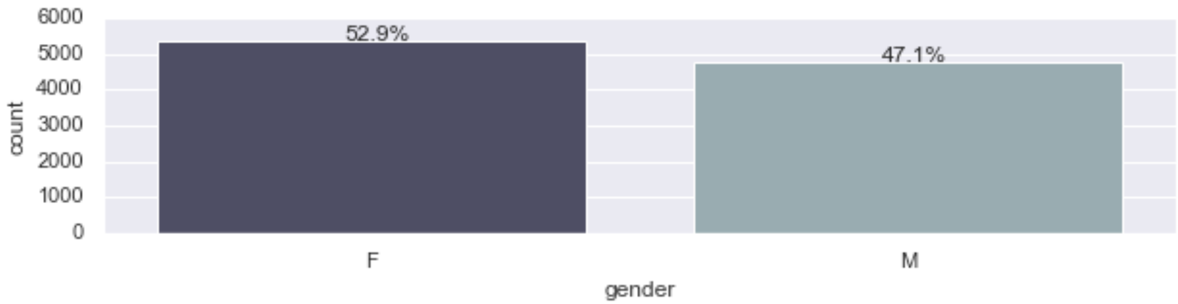
Classification Variable: attrition

```
********* Target Variable *********

Target var: attrition

Attrited Customer    1627
Existing Customer    8500
Name: attrition, dtype: int64
```

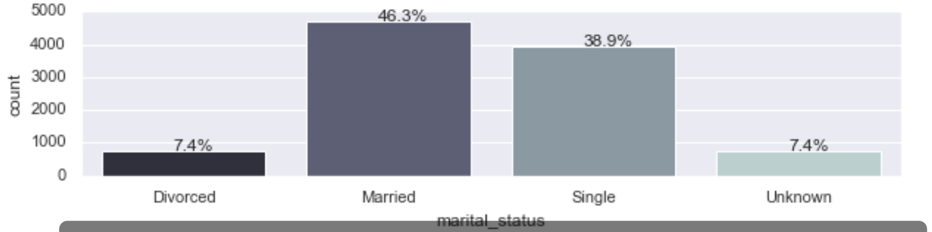


Univariate EDA – Customer Category Distributions

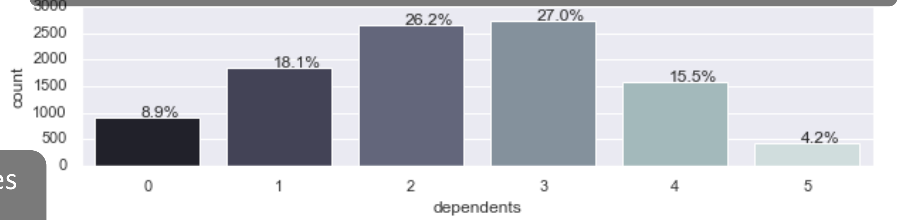
There is a reasonably balanced distribution between male and female customers



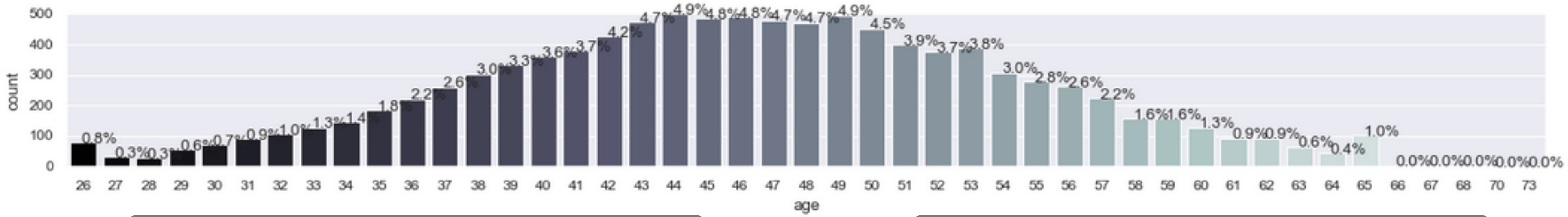
Majority of customer are married at 46.3%



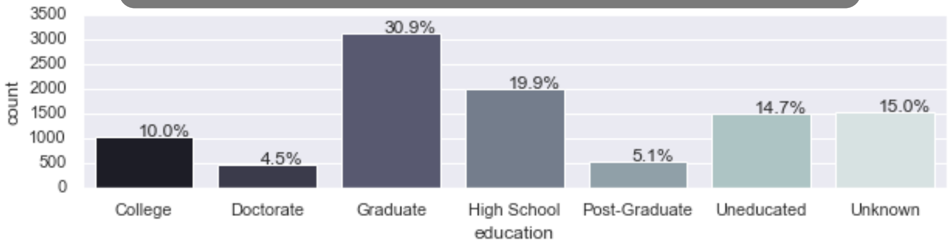
dependents are a normal distribution centered around 2 and 3



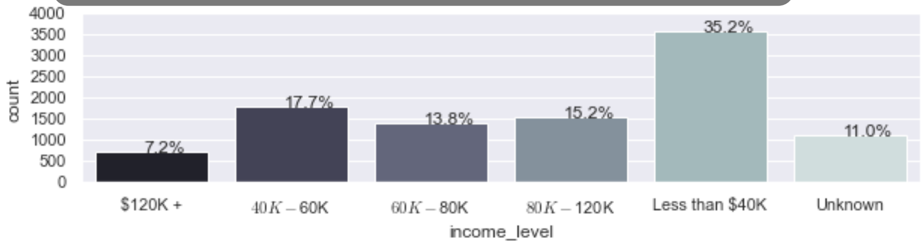
age exhibits a normal distribution with smalls modes on the ends



Graduate education is the most common among customers

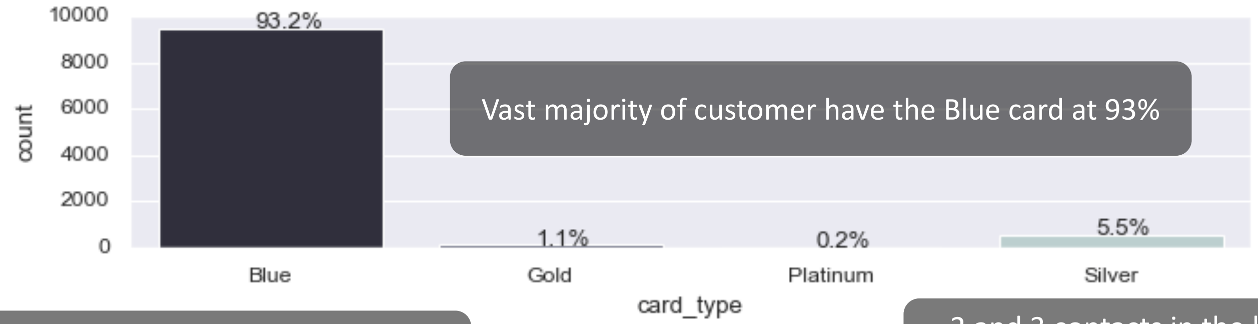


The mode is income is less than $40K, which is the most likely value among customers

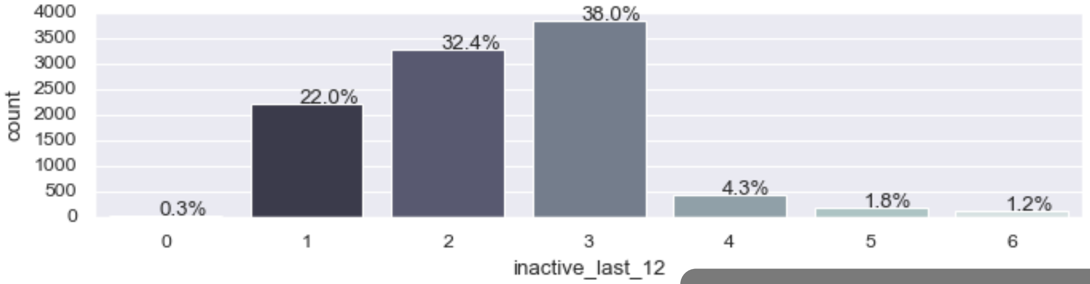


Univariate EDA – Card Category Distributions

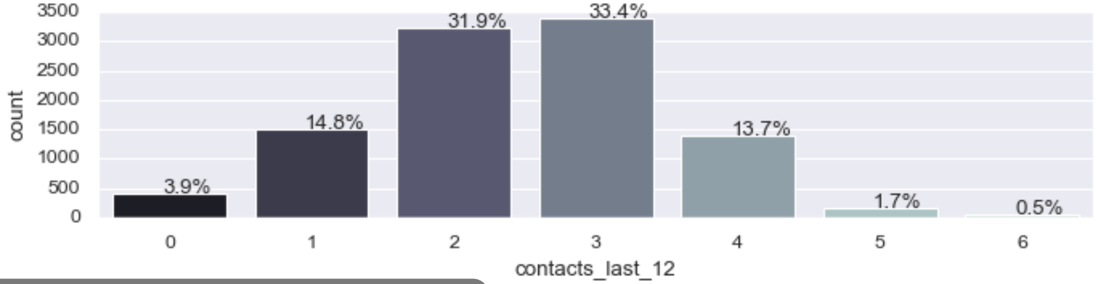
Vast majority of customer have the Blue card at 93%



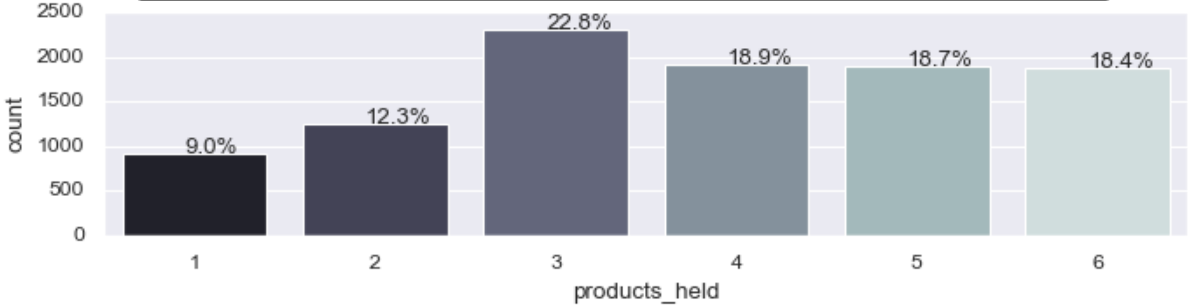
Most inactive_last_12 account at 92%



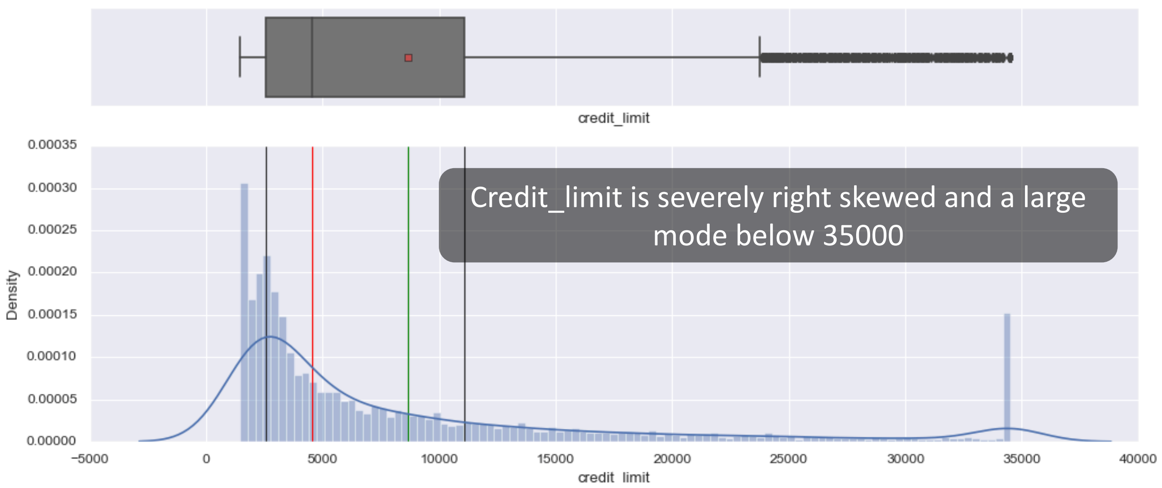
2 and 3 contacts in the last 12 months account for 65% of customers

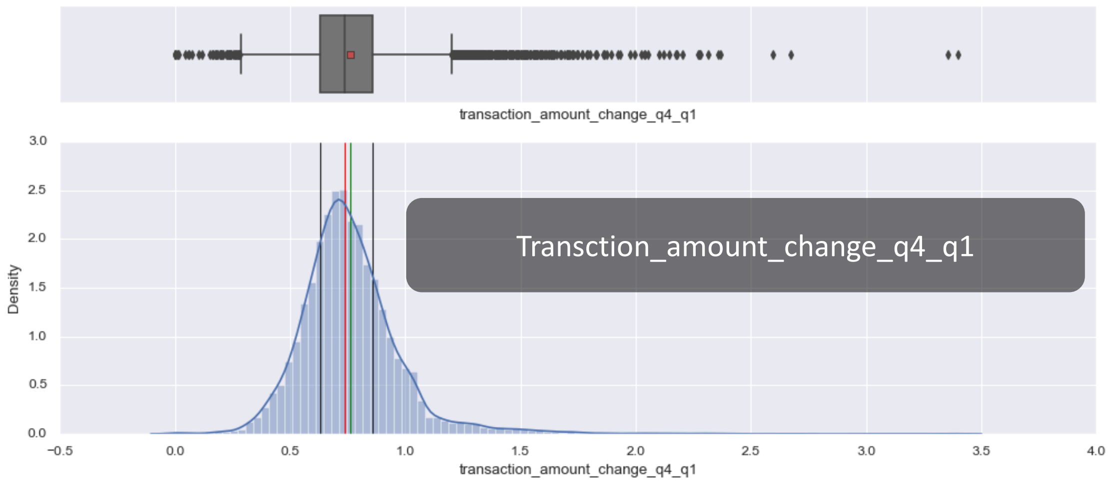


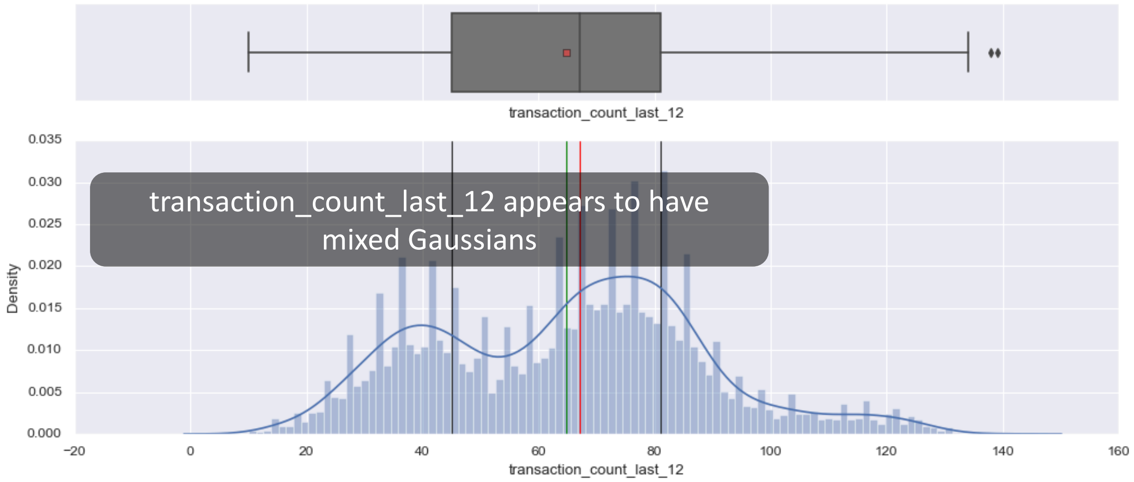
Product held are mostly distributed across 3-6 at of 78% of customers

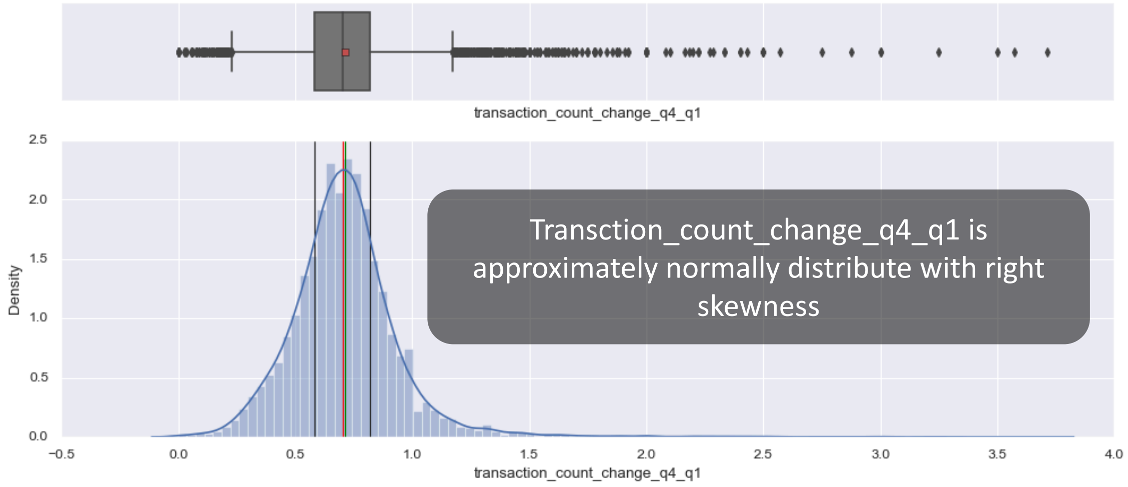


Univariate EDA – Continuous Distributions



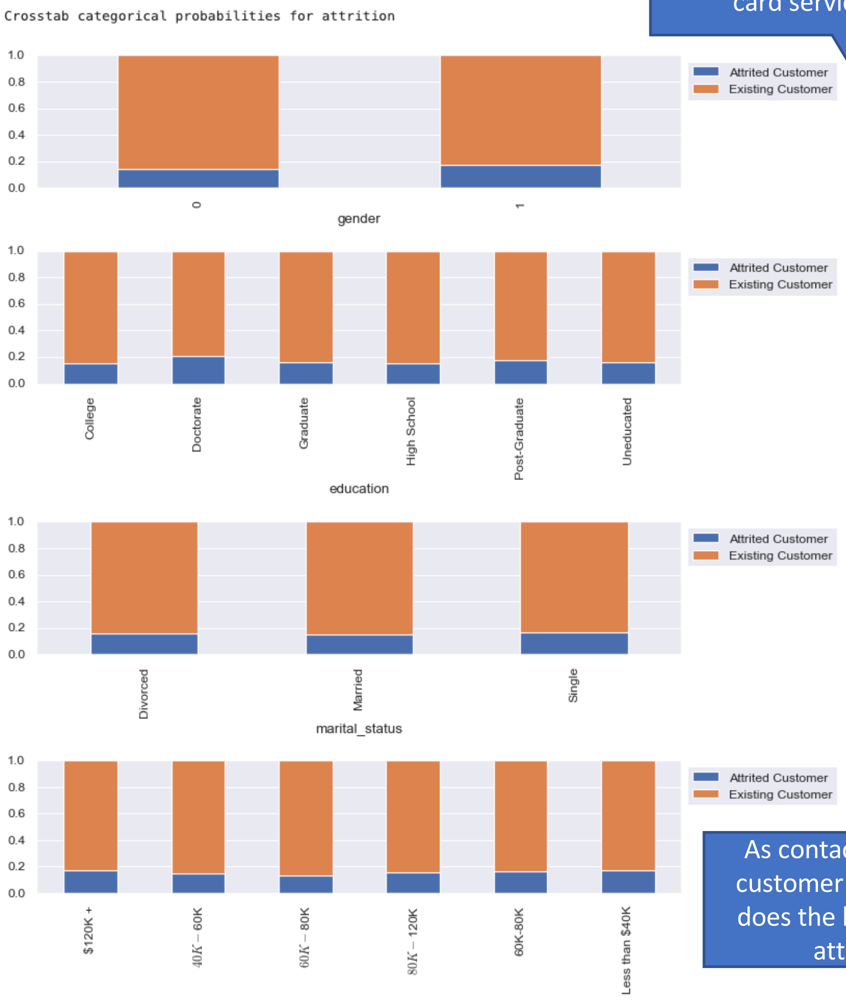


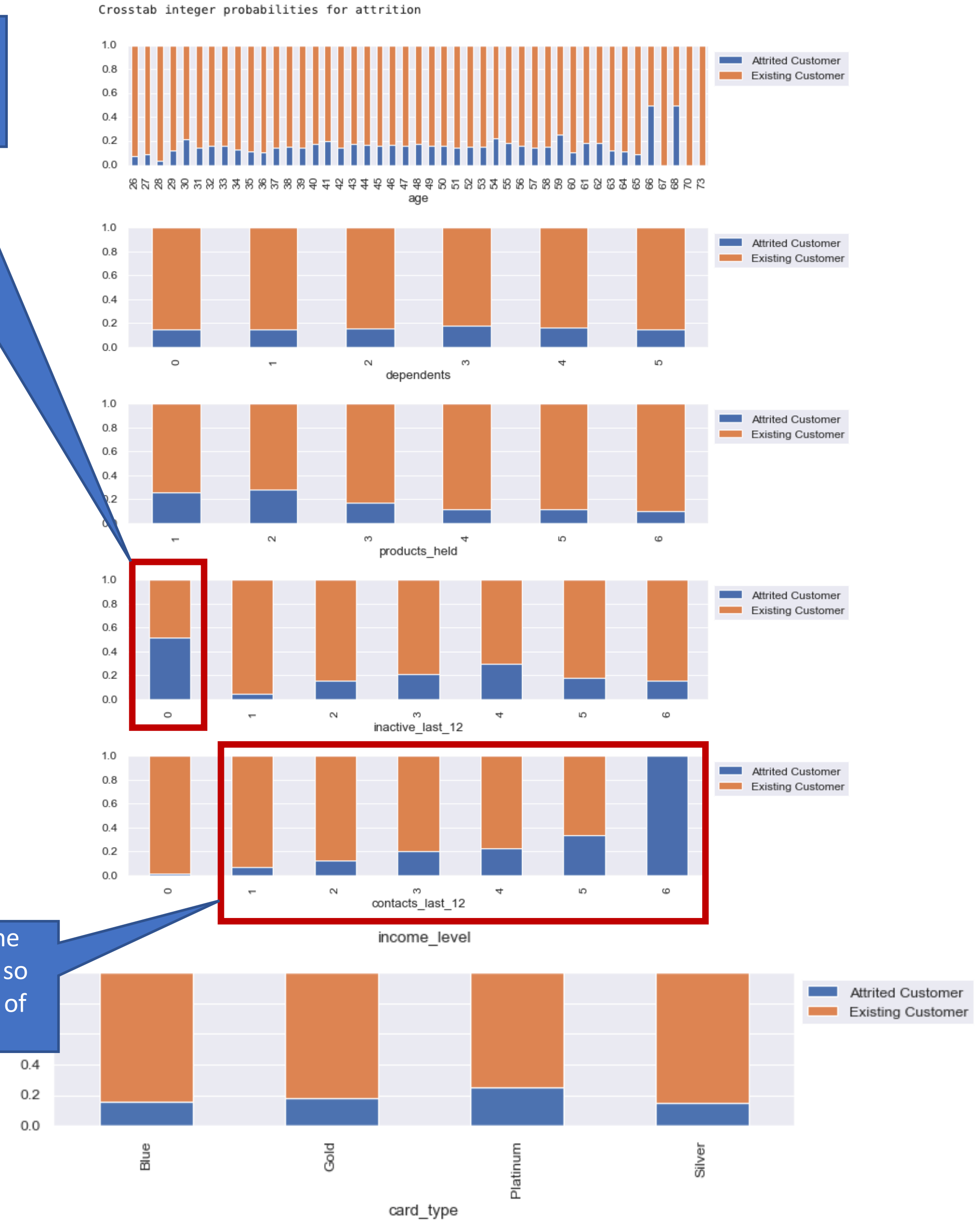




Univariate EDA – Crosstab Target Proportions across variables

People that are active commonly leave credit card services





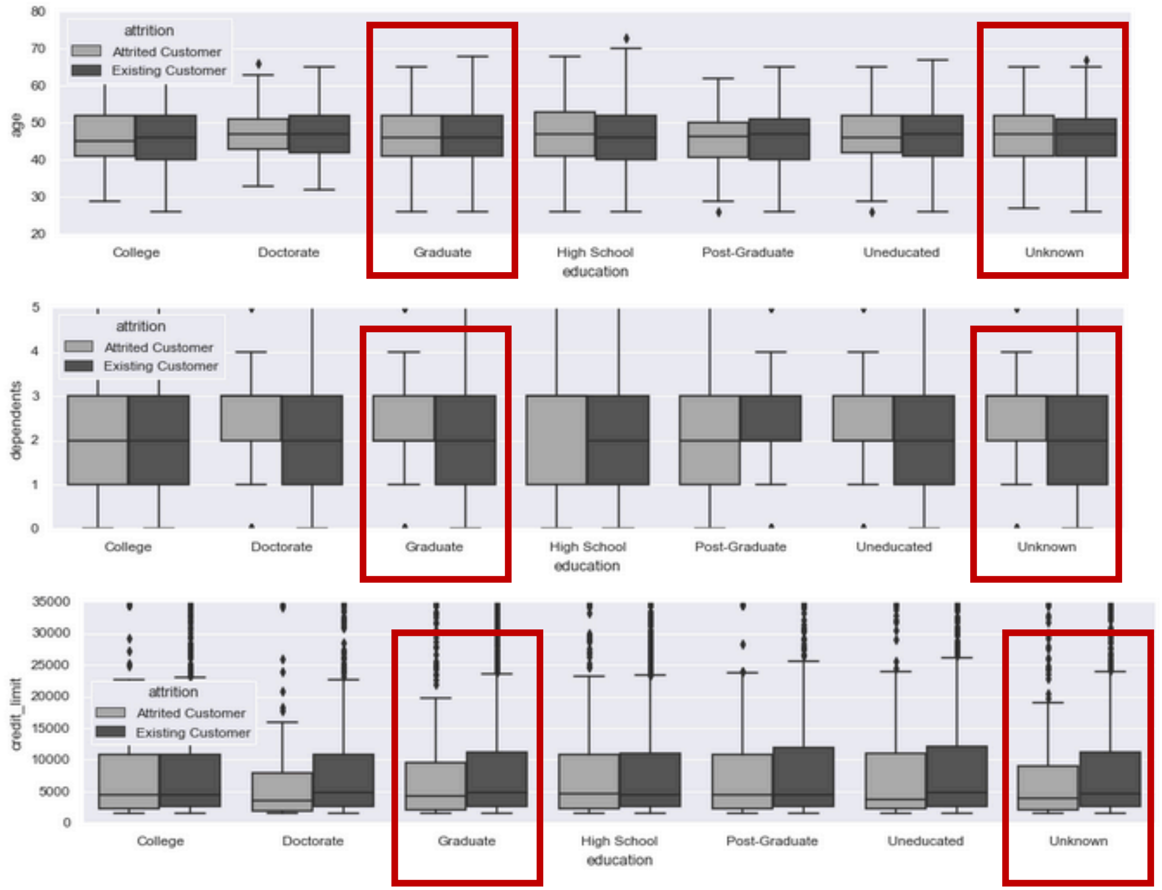
As contacts with the customer increase, so does the likelihood of attrition

Data Preprocessing – Cleansing Known values

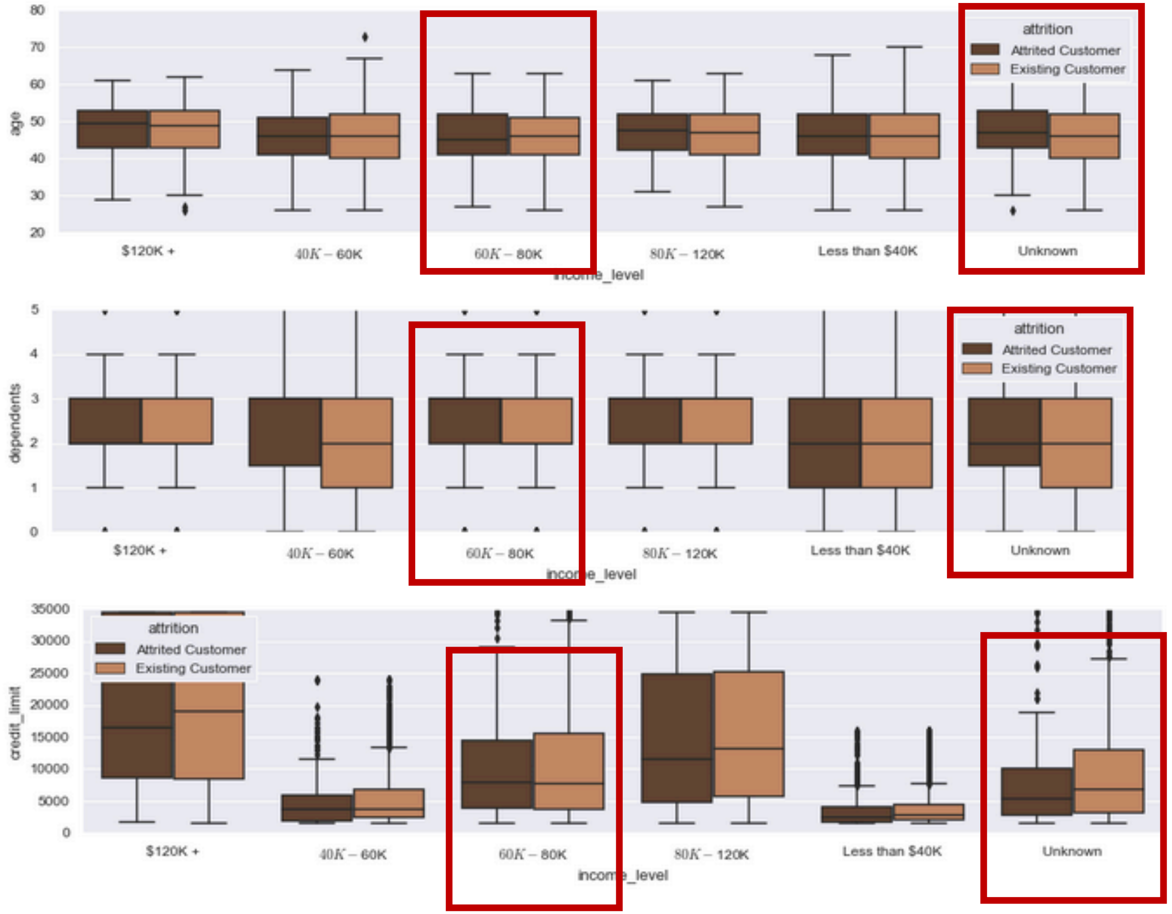
Mapping “Unknown” **education** to “Graduate” based on similarities across the other variables

Mapping “Unknown” **income** to “60K-80” based on similarities across other variables

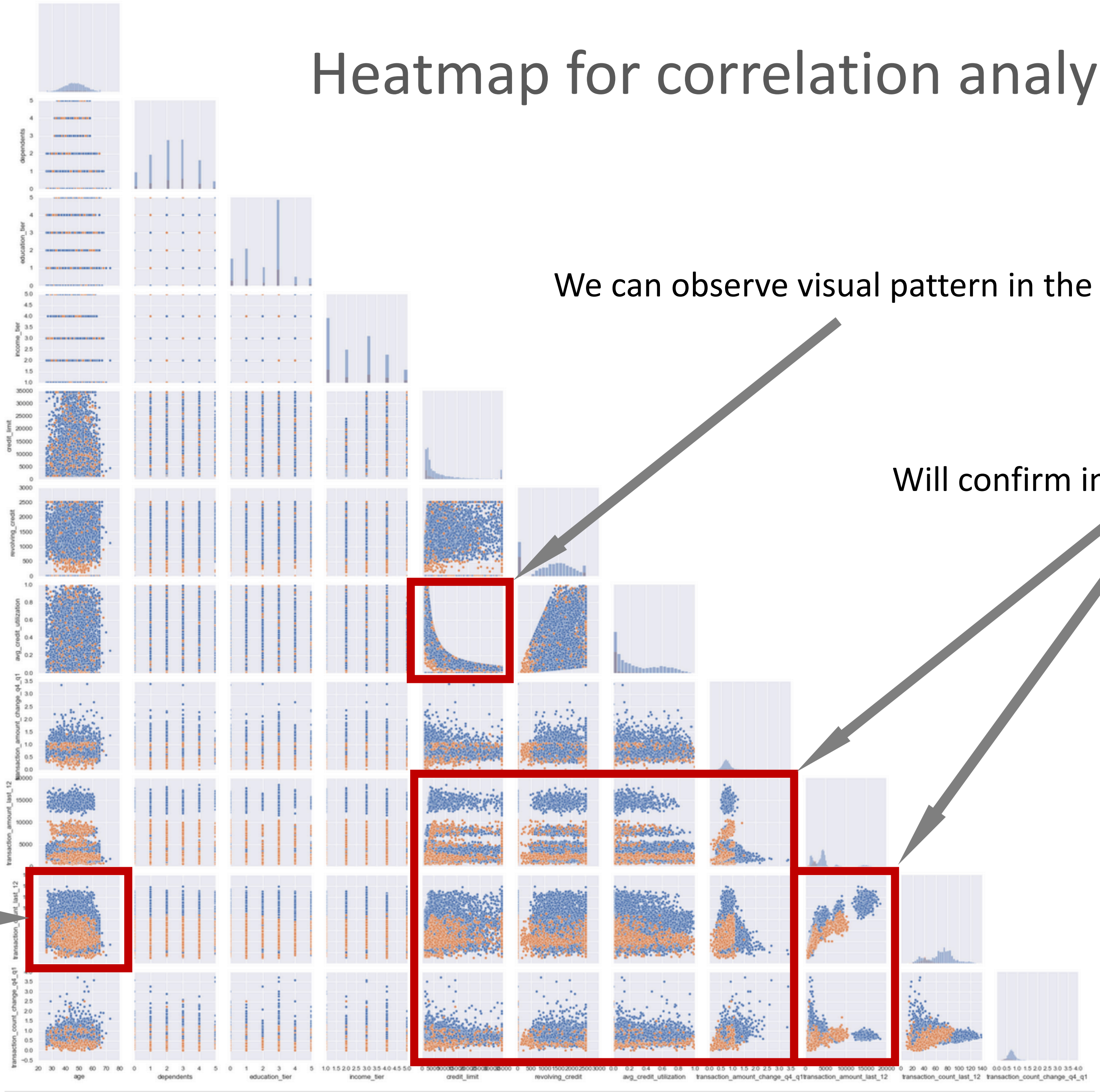
Which education level should Unknown values be mapped to?



Which income level should Unknown values be mapped to?



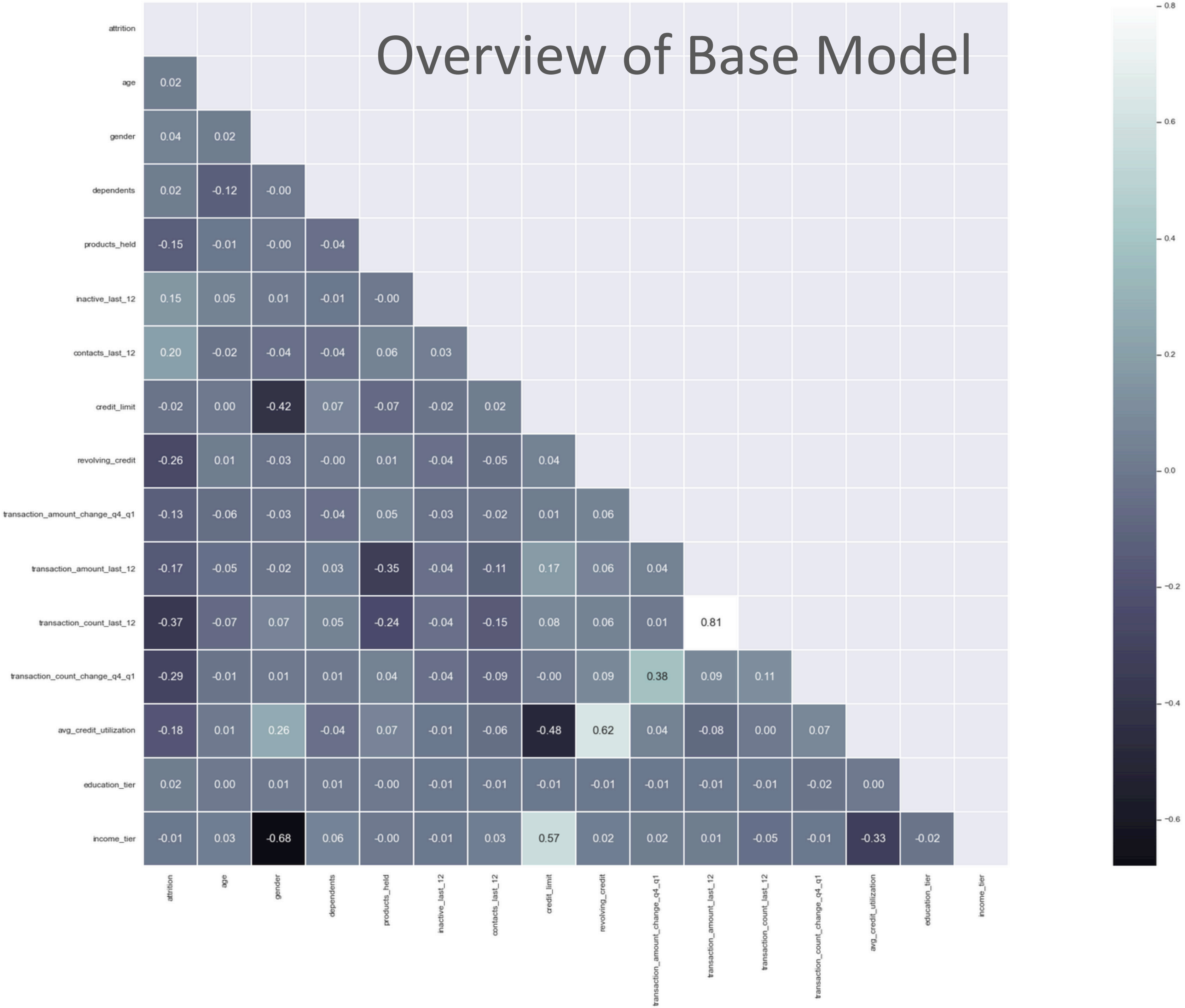
Heatmap for correlation analysis



We can observe visual pattern in the bivariate plots

Will confirm in the heatmap chart

Overview of Base Model



The strongest collinearities above +/- .9 have ben removed

Heatmap generate prior to one-hot encoding

Some collinearity is still present, We will performance model tuning and defer the decision to further reduce dimensions

Performance Measures

Accuracy = TP + TN / TP + TN + FP + FN
% of correct predictions overall

Recall = TP / TP + FN
% of correct pos predictions of all predictions made (use when FN is very expensive e.g., loan default)

Precision = TP / TP + FP
% of correct pos predictions of all pos data (use when FP is very expensive e.g., drone strike)

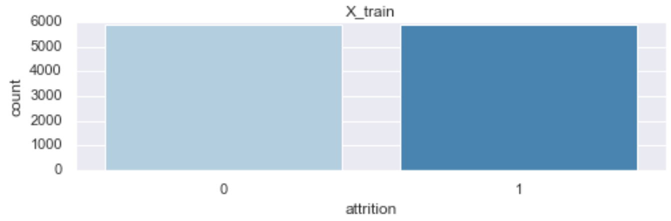
Given the business problem at hand (preempt customer attrition risk), we will need to find a reasonable balance point between Precision and Recall.

If we fail to identify and preempt customers that are likely to leave our credit card services (FN), the cost to the business is acute continued loss of revenue which in turn threatens loss of overall banking market share and profitability. This situation should be avoided as it has a greater immediate negative impact to the business performance.

Therefore, we should seek to avoid FN's by maximizing Recall while trading off (limited) Precision and/or Accuracy (which we expect to decrease as a result).

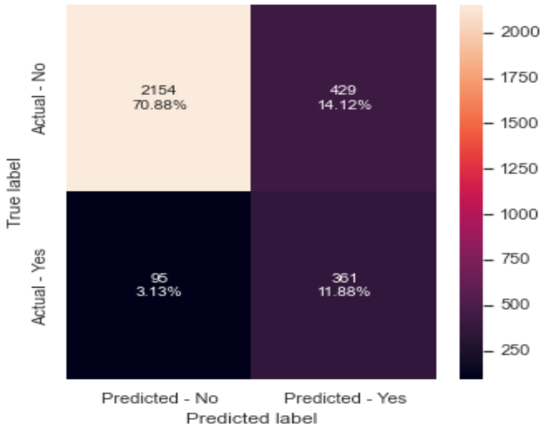
Logistics Regression with Over & Undersampling

```
Before UpSampling, counts of label '1': 1171
Before UpSampling, counts of label '0': 5917

After UpSampling, counts of label '1': 5917
After UpSampling, counts of label '0': 5917

After UpSampling, the shape of train_X: (11834, 22)
After UpSampling, the shape of train_y: (11834,)

Upsample target proportions
```

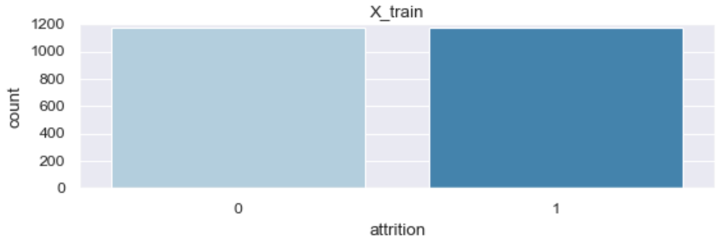


LR
performance

```
********* LR Upsampling *********

LogisticRegression(random_state=1) fit time: 0.12 secs
```

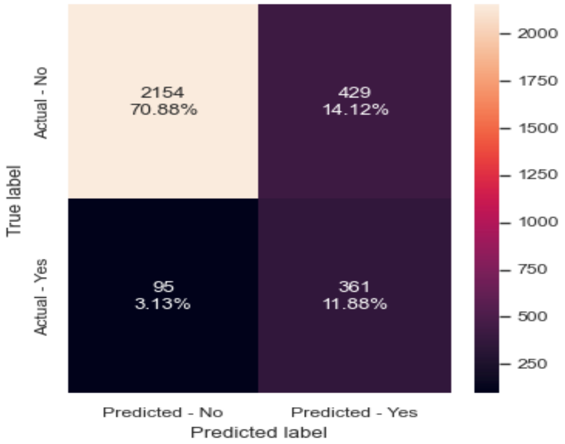



```
Before DownSampling, counts of label '1': 1171
Before DownSampling, counts of label '0': 5917

After DownSampling, counts of label '1': 1171
After DownSampling, counts of label '0': 1171

After DownSampling, the shape of train_X: (2342, 22)
After DownSampling, the shape of train_y: (2342,)

Downsample target proportions
```



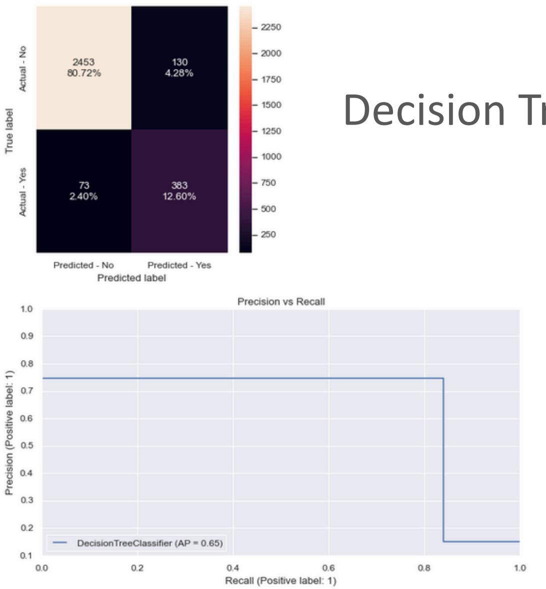
LR
performance

```
********* LR Downsampling *********

LogisticRegression(random_state=1)

fit time: 0.12 secs
```



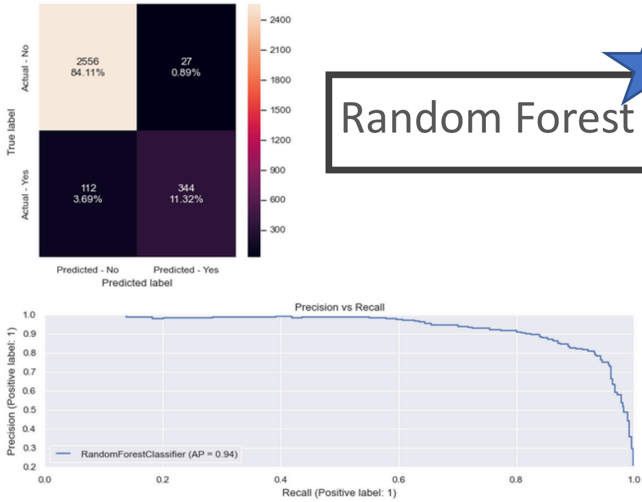
Overview of Base Model

```
DecisionTreeClassifier(class_weight={0: 0.25, 1: 0.85}, criterion='entropy',
                       random_state=1)

Fit time: 0.11 secs
```



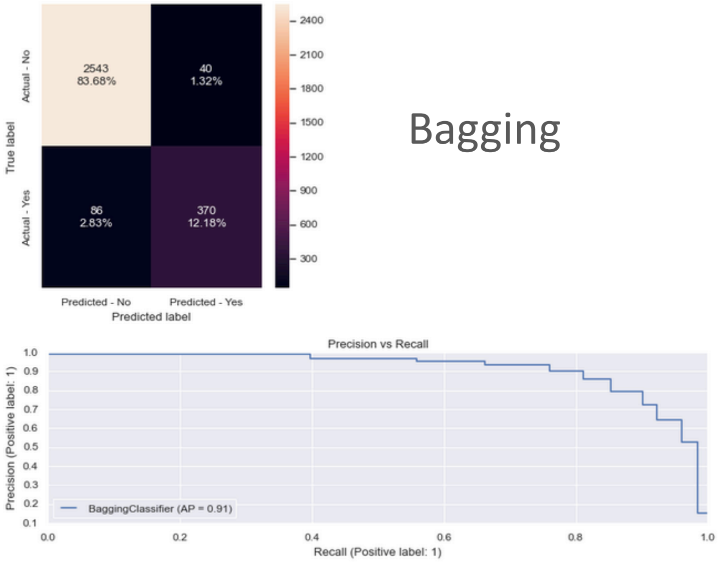
Decision Tree

```
RandomForestClassifier(random_state=1)

Fit time: 0.92 secs
```



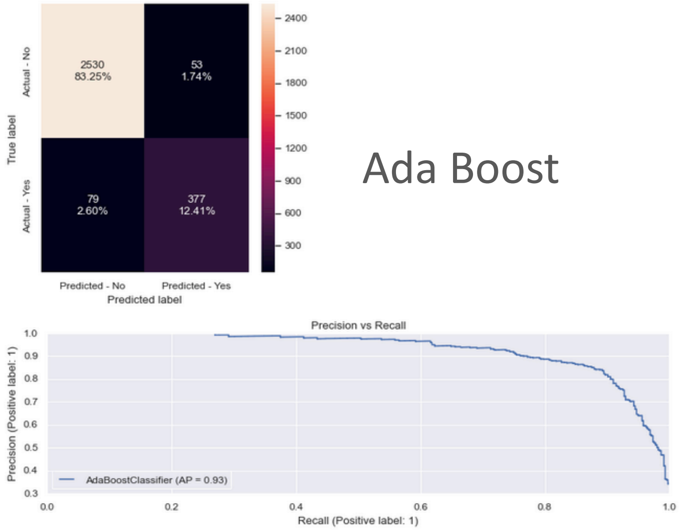
Random Forest

```
BaggingClassifier(random_state=1)

Fit time: 0.35 secs
```



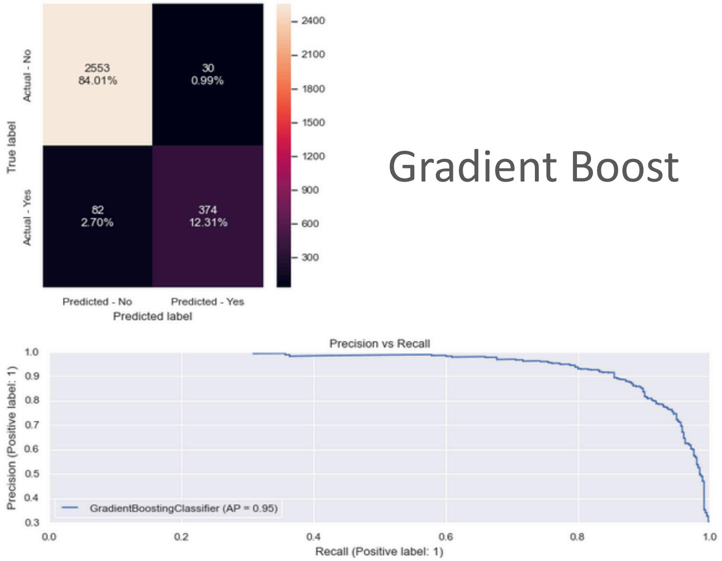
Bagging

```
AdaBoostClassifier(random_state=1)

Fit time: 0.41 secs
```



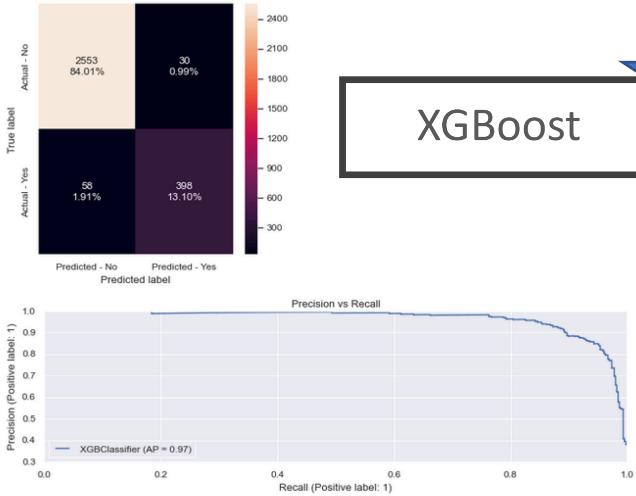
Ada Boost

```
GradientBoostingClassifier(random_state=1)

Fit time: 1.61 secs
```



Gradient Boost

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, eval_metric='logloss',
              gamma=0, gpu_id=-1, importance_type='gain',
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=8,
              num_parallel_tree=1, random_state=1, reg_alpha=0, reg_lambda=1,
              scale_pos_weight=1, subsample=1, tree_method='exact',
              validate_parameters=1, verbosity=None)

Fit time = 0.53 secs
```



XGBoost

Base Model Benchmarking

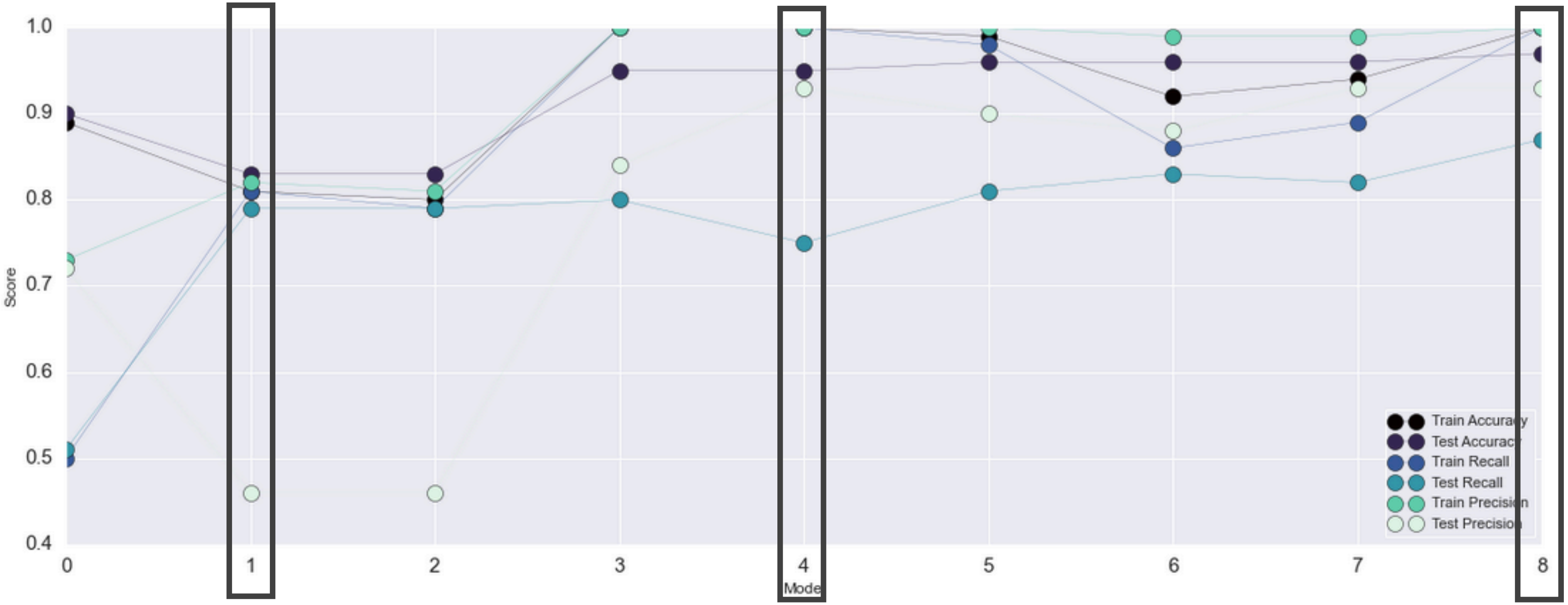
Prediction performance (all models)

	Classifier	Train Accuracy	Test Accuracy	Train Recall	Test Recall	Train Precision	Test Precision
0	lr_classifier	0.89	0.90	0.50	0.51	0.73	0.72
1	lr_classifier_upsample	0.81	0.83	0.81	0.79	0.82	0.46
2	lr_classifier_downsample	0.81	0.83	0.79	0.79	0.82	0.46
3	dtree_classifier	1.00	0.95	1.00	0.80	1.00	0.84
4	rforest_classifier	1.00	0.95	1.00	0.75	1.00	0.93
5	bagging_classifier	0.99	0.96	0.98	0.81	1.00	0.90
6	ab_classifier	0.92	0.96	0.86	0.83	0.97	0.88
7	gb_classifier	0.94	0.96	0.89	0.82	0.99	0.93
8	xgb_classifier	1.00	0.97	1.00	0.87	1.00	0.93

We observe base model performance with an eye to a reasonably high Recall with smallest delta between train and test data. The secondary concern is ensure precision does not tank.

From our observations we select the following base models for optimization:

1. **Linear Regression with Upsampling**
2. **Random Forest**
3. **XGBoost**



Tuned Model Performance vs Benchmarks

The top performance tuned model is the **xgboost classifier tuned with grid search CV**. This is the model recommend to be productionized.

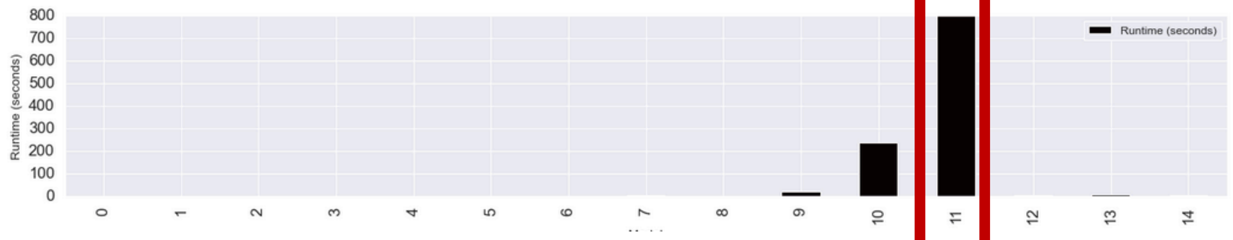
```
Pipeline(steps=[('xgbclassifier', XGBClassifier(base_score=0.5,
                 booster='gbtree', colsample_bylevel=1,
                 colsample_bynode=1, colsample_bytree=1,
                 eval_metric='mlogloss', gamma=3, gpu_id=-1,
                 importance_type='gain', interaction_constraints='',
                 learning_rate=0.1, max_delta_step=0, max_depth=9,
                 min_child_weight=1, missing=nan,
                 monotone_constraints='()', n_estimators=20,
                 n_jobs=8, num_parallel_tree=1, random_state=1,
                 reg_alpha=0, reg_lambda=0, scale_pos_weight=1,
                 subsample=0.9, tree_method='exact',
                 validate_parameters=1, verbosity=None))])
```

Runtime performance (all models)

	Runtime (seconds)
0	0.970
1	0.123
2	0.126
3	0.051
4	0.920
5	0.346
6	0.410
7	1.608
8	0.528
9	22.128
10	235.037
11	798.226
12	5.095
13	7.150
14	3.864

Runtime Performance

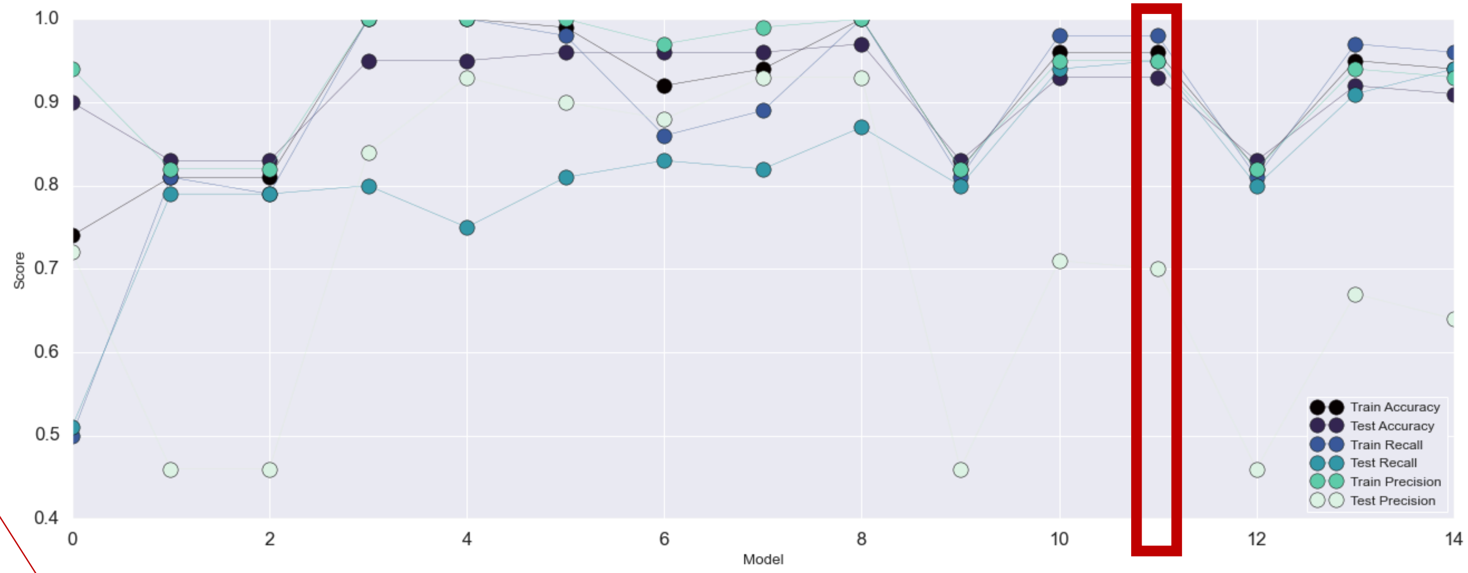
13 minute runtime



Prediction performance (all models)

	Classifier	Train Accuracy	Test Accuracy	Train Recall	Test Recall	Train Precision	Test Precision
0	lr_classifier	0.74	0.90	0.50	0.51	0.94	0.72
1	lr_classifier_upsample	0.81	0.83	0.81	0.79	0.82	0.46
2	lr_classifier_downsample	0.81	0.83	0.79	0.79	0.82	0.46
3	dtree_classifier	1.00	0.95	1.00	0.80	1.00	0.84
4	rforest_classifier	1.00	0.95	1.00	0.75	1.00	0.93
5	bagging_classifier	0.99	0.96	0.98	0.81	1.00	0.90
6	ab_classifier	0.92	0.96	0.86	0.83	0.97	0.88
7	gb_classifier	0.94	0.96	0.89	0.82	0.99	0.93
8	xgb_classifier	1.00	0.97	1.00	0.87	1.00	0.93
9	lr_classifier_upsample_grid	0.82	0.83	0.81	0.80	0.82	0.46
10	randomforest_classifier_grid	0.96	0.93	0.98	0.94	0.95	0.71
11	xgboost_classifier_grid	0.96	0.93	0.98	0.95	0.95	0.70
12	lr_classifier_upsample_random	0.82	0.83	0.81	0.80	0.82	0.46
13	randomforest_classifier_random	0.95	0.92	0.97	0.91	0.94	0.67
14	xgboost_classifier_random	0.94	0.91	0.96	0.94	0.93	0.64

model performance vs. benchmarks



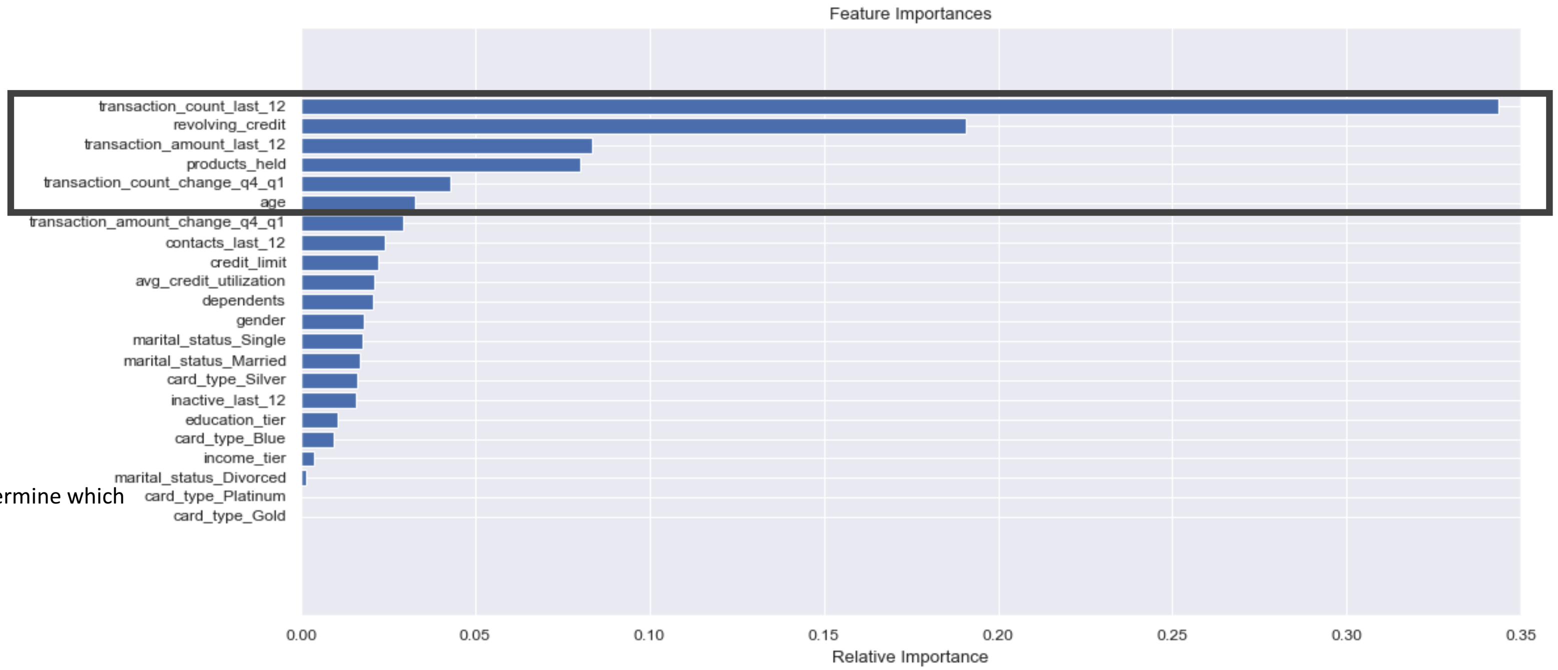
Production Model (pilot on unseen data)

(2nd to rand_search_CV only due to smaller precision delta between train & test)

Best Performing Model – XGBoost Grid Search CV Tuned

Best Performing Model is xgboost hyperparameter tuned via Grid Search CV

```
Pipeline(steps=[('xgbclassifier', XGBClassifier(base_score=0.5,
                booster='gbtree', colsample_bylevel=1,
                colsample_bynode=1, colsample_bytree=1,
                eval_metric='mlogloss', gamma=3, gpu_id=-1,
                importance_type='gain', interaction_constraints='',
                learning_rate=0.1, max_delta_step=0, max_depth=9,
                min_child_weight=1, missing=nan,
                monotone_constraints='()', n_estimators=20,
                n_jobs=8, num_parallel_tree=1, random_state=1,
                reg_alpha=0, reg_lambda=0, scale_pos_weight=1,
                subsample=0.9, tree_method='exact',
                validate_parameters=1, verbosity=None))])
```



Model Performance & Benchmarking

1.Benchmarking was performed on base machine learning classification models to determine which subset should be selected for subsequent tuning and prediction

2.The base classification models tested include:

1. Linear Regression
2. Linear Regression (with upsampling)
3. Linear Regression (with downsampling)
4. Decision Tree
5. Random Forest
6. Bagging
7. AdaBoost
8. Gradient Boost
9. XGBoost

3.The top 3 best performing models (based on Recall score) selected for tuning and optimization were:

1. Linear Regression (with upsampling)
2. Random Forest
3. XGBoost

4.The top 3 base classification models were hyper-parameter-tuned via both Grid Search cross validation and Random Search cross validation

5.sklearn pipelines were employed to automate model optimization and tuning and aggregate results were collected and compared to determine an overall winner (tuned model)

6.The tuning models were subsequently evaluated for both Recall score and runtime performance to understand efficacy of classification as well as cost (time and resources needed to make the prediction)

This model performs well in terms of both Recall score (high train & test w/small delta) and efficient runtime performance of

Key Finding & Insights

Using grid search cross validation to tune the best base models produced an XGBoost classifier that achieves Recall of .97 (train) and .85 (test) with a reasonable runtime cost of 13 minutes - this model is a candidate to deploy in production

Based on the best XGB estimator determined by grid search CV, the most important feature to predict attrition are:

- transction_count_last_12
- revolving_credit
- transaction_amount_last_12
- product_help
- transaction_count_change_q4_q1
- tranction_amount_change_q4_q1
- age
- gender
- credit limit
- contacts_last_12
- inactive_last_12
- education_tier
- income_tier
- card_type_Blue

Strategy to preempt credit card attrition should target the above mentioned features

All data anomalies (e.g., Unknown and income_level same-meaning variants) should be corrected in the source systems

Recommendations to the Business

1.Recommend initial steps of having analysis generate weekly report and review with stakeholders customer transaction_count, revolving_credit, tranaction_amount_change_q4_q1 high-light attrition risk

2.Target age, gender, education and income categories with credit card offers that differentiate from competitor incentives

3.Automated reporting and review of contacts_last_12, inactive_last_12and card_type to build marketing campaigns around

4.Instantiate a continuous effort internally to clean data anomalies at the various data sources as this will make future model prediction more accurate

5.For the customer how have already left credit card services, since they may have other services with the bank and development new offers which reduce credit card fees based on other bank service customers use

6.Rebuild and optimize classification models iteratively on the monthly basis and view in risk and strategy meeting with senior leaders