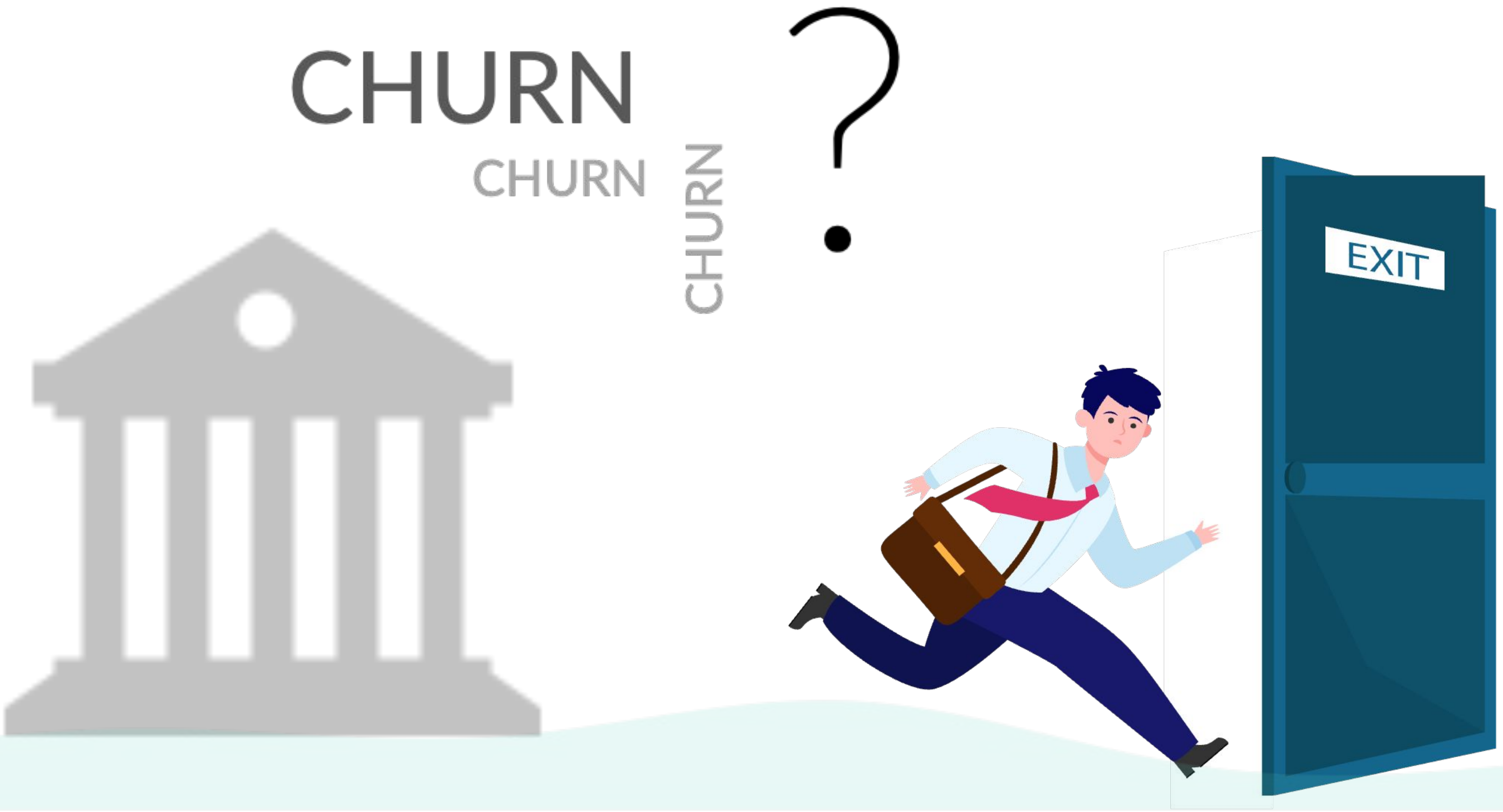
Dataset & Modelling Project - Customer Churn

Team Catalyst - DS10

Ayudha Hardian Pratama
Irvan Zidny
Muhammad Idris
Surahma Jaya

DigitalSkola

What's the problems ?



What's the problems ?

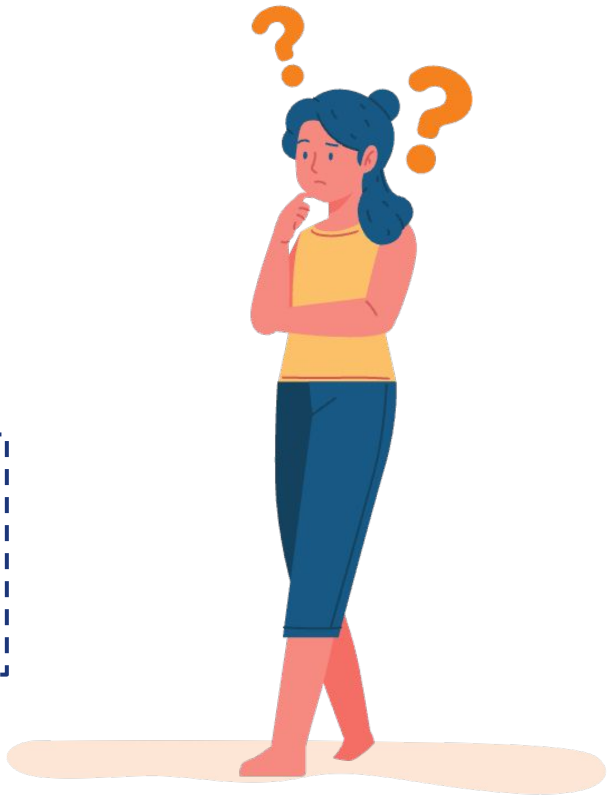
DigitalSkola

Problem Statement

- How to predict churning customers based on the data that we have?

Objective

- Develop and find best performing model to predict customer churn based on the data that we have



Methodology

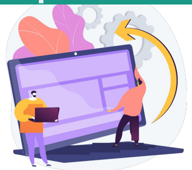
Available Data

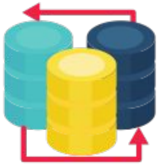
Credit Card Customers
10127 Baris
23 Features

kaggle

EDA

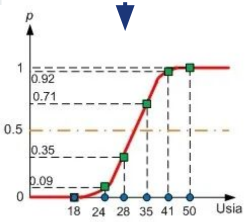
Data Preprocessing

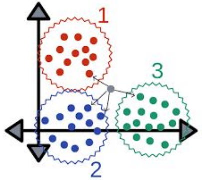


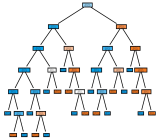


Modelling

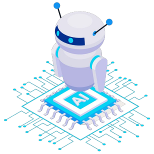
kNN







Final Model



What are we looking at ?

DigitalSkola

Dropped Variable

3

- Naive_Bayes_Classifier_Attrition_Flag_Card_Category ... Month 1
- Naive_Bayes_Classifier_Attrition_Flag_Card_Category ... Month 2
- CLIENTNUM

Numerical Variable

14

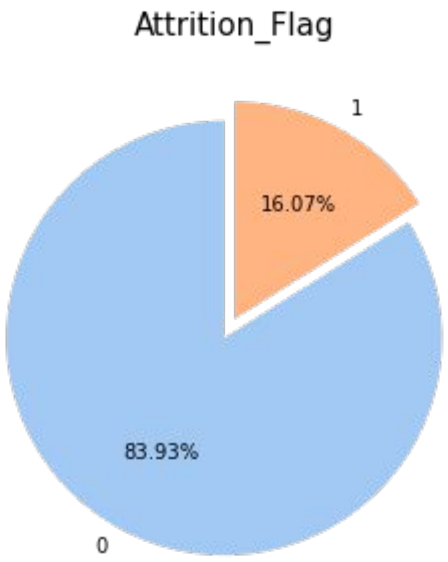
- Customer_Age
- Dependent_count
- Months_on_book
- Total_Relationship_Count
- Months_Inactive_12_mon
- Contacts_Count_12_mon
- Credit_Limit
- Total_Revolving_Bal
- Avg_Open_To_Buy
- Total_Amt_Chng_Q4_Q1
- Total_Trans_Amt
- Total_Trans_Ct
- Total_Ct_Chng_Q4_Q1
- Avg_Utilization_Ratio

Categorical Variable

6

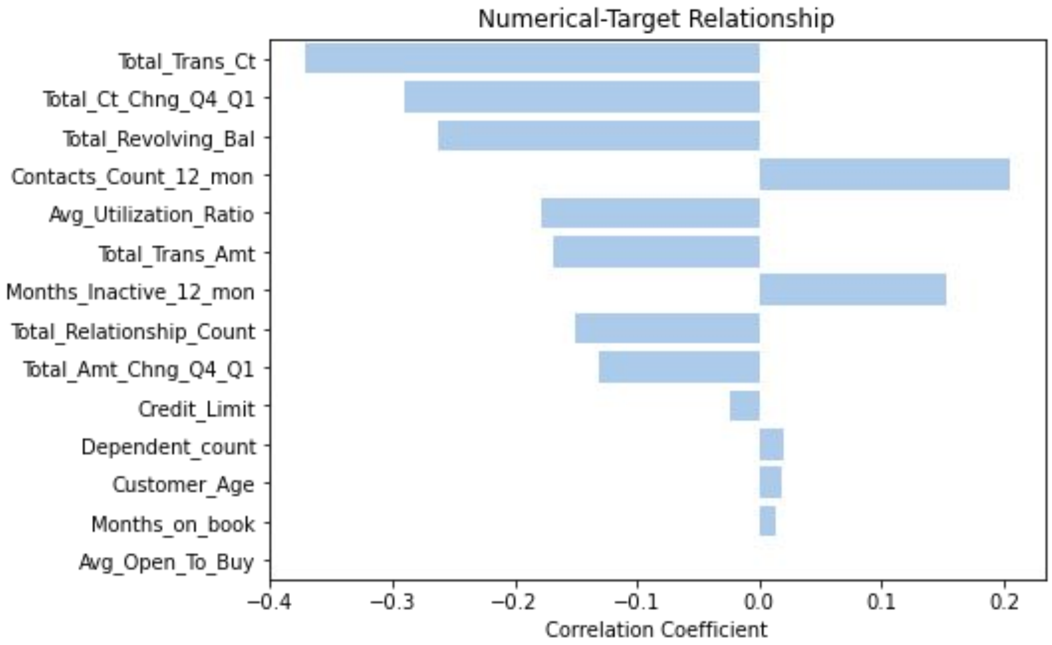
- Attrition_Flag
- Gender
- Education_Level
- Marital_Status
- Income_Category
- Card_Category

Exploratory Data Analysis



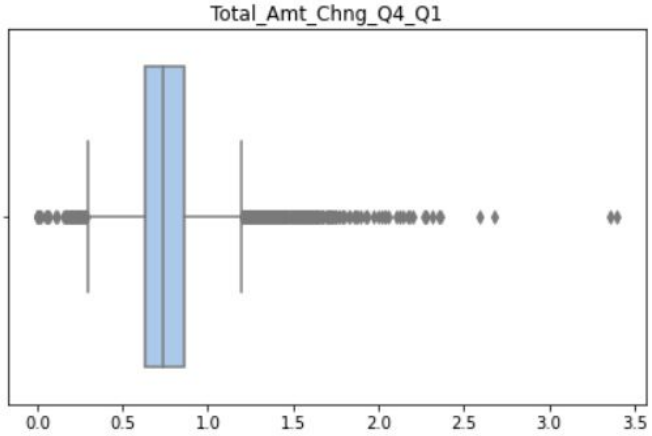
- There are 16.07% of customers that churned, or 1627 out of 10127
- Generally the distribution of churned customers of the categorical features are almost proportional with the distribution of non-churned customers
- The distribution of churned customers on numerical features like *Credit_Limit* and *Total_Trans_Amt* tend to the left

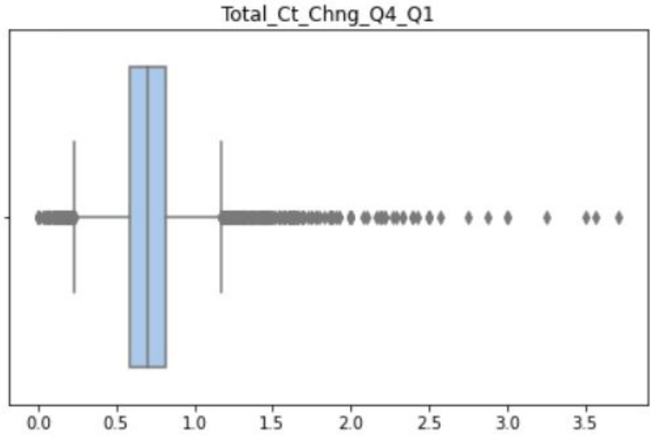
Exploratory Data Analysis

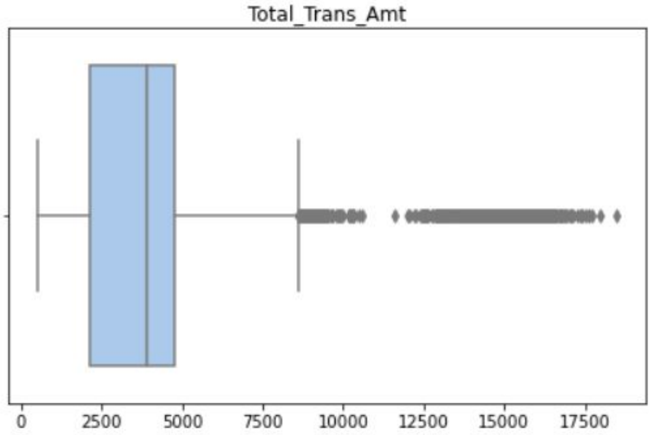


- There is no numerical feature that are highly correlated to the target variable
- Most of numerical features have low correlations with each other, with the exception of some like *Avg_Open_To_Buy* and *Credit_Limit* that have perfect positive correlation

Exploratory Data Analysis



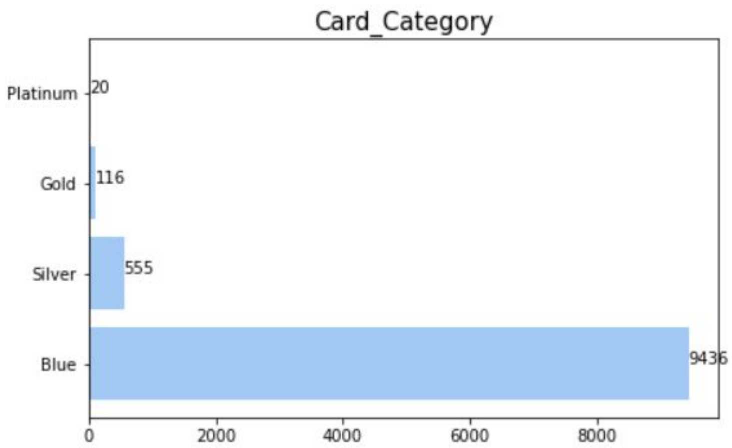




Based on Outer Fence criteria, we found outliers on:

- Total_Amt_Chng_Q4_Q1
- Total_Trans_Amt
- Total_Ct_Chng_Q4_Q1

Exploratory Data Analysis



- Card_Category is imbalanced, where the 'Platinum' and 'Gold' count is too few compared to the other category

Numerical

- Outlier Removal
- Scaling

Categorical

- Label Encoding
- One-Hot Encoding

Data Preprocessing - Numerical

- ## Outlier Removal

	feature	outlier_count_of	unique_outlier_count_of
0	Customer_Age	0	0
1	Dependent_count	0	0
2	Months_on_book	0	0
3	Total_Relationship_Count	0	0
4	Months_Inactive_12_mon	0	0
5	Contacts_Count_12_mon	0	0
6	Credit_Limit	0	0
7	Total_Revolving_Bal	0	0
8	Avg_Open_To_Buy	0	0
9	Total_Amt_Chng_Q4_Q1	90	83
10	Total_Trans_Amt	737	667
11	Total_Trans_Ct	0	0
12	Total_Ct_Chng_Q4_Q1	80	48
13	Avg_Utilization_Ratio	0	0

***Outer Fence**, percentage of removed rows: 8.77%*

Outliers can be calculated by using two criteria, Inner Fence and Outer Fence.

- We found that using Inner Fence, it would remove **32.84%** of the rows. Having too many rows removed is not preferred as it would make the data not representative of the original
- We preferred to use Outer Fence criteria, as it would only remove **8.77%** of the rows

Data shape after removing outliers

- 9240 rows
- 20 columns (including Target)

DigitalSkola

- **Scaling**

Normalization, will scale the data to have minimum value 0 and maximum value 1. It can be used on our remaining numerical features:

	count	mean	std	min	25%	50%	75%	max
Months_on_book	9240.0	0.534954	0.185191	0.0	0.441860	0.534884	0.627907	1.0
Total_Relationship_Count	9240.0	0.584286	0.304279	0.0	0.400000	0.600000	0.800000	1.0
Months_Inactive_12_mon	9240.0	0.392298	0.168777	0.0	0.333333	0.333333	0.500000	1.0
Credit_Limit	9240.0	0.204386	0.266254	0.0	0.031130	0.085154	0.263008	1.0
Total_Revolving_Bal	9240.0	0.453621	0.325515	0.0	0.000000	0.497815	0.701629	1.0
Total_Amt_Chng_Q4_Q1	9240.0	0.484147	0.126660	0.0	0.403372	0.473411	0.554475	1.0
Total_Trans_Amt	9240.0	0.259568	0.152118	0.0	0.133129	0.270682	0.338780	1.0
Total_Trans_Ct	9240.0	0.461655	0.179118	0.0	0.303571	0.491071	0.607143	1.0
Total_Ct_Chng_Q4_Q1	9240.0	0.450512	0.134683	0.0	0.367638	0.447896	0.526861	1.0
Avg_Utilization_Ratio	9240.0	0.281828	0.280618	0.0	0.000000	0.184184	0.520521	1.0

- **Scaling**

Numerical data should be scaled to improve our model's performance.
The technique used to scale numerical data depends on the type of the distribution

Standardization, will scale the data to have 0 mean, and 1 std. It can be used on data that have gaussian-like distribution, which are:

- Customer_Age
- Dependent_count
- Contacts_Count_12_mon

	count	mean	std	min	25%	50%	75%	max
Customer_Age	9240.0	-1.385213e-15	1.000054	-2.560983	-0.679604	-0.052478	0.700074	2.957729
Dependent_count	9240.0	-1.224670e-15	1.000054	-1.812915	-1.042349	-0.271783	0.498783	2.039915
Contacts_Count_12_mon	9240.0	1.202790e-15	1.000054	-2.221761	-0.426713	0.470812	0.470812	3.163385

Data Preprocessing - Categorical

- **Encoding**

Categorical data should be encoded to numerical, as models cannot understand categorical data. Based on the data type, we can use Label Encoding or One-Hot Encoding

Label Encoding encodes binary and ordinal categorical value into a range of integer starting from 0. It can be used on features that have binary and ordinal data types ,which are:

- Education_Level
- Income_Category
- Card_Category.

Note that on Card_Category, we are merging ‘Platinum’ and ‘Gold’ that have too few data into ‘Silver’, by encoding them into 1.

- **Encoding**

One-Hot Encoding creates a new feature for each categorical value. It can be used on other categorical data types that cannot be ordered, that is:

- Gender,
- Marital_Status

After encoding, Gender and Marital_Status are replaced by:

  - Gender_F
  - Gender_M
  - Marital_Status_Divorced
  - Marital_Status_Married
  - Marital_Status_Single
  - Marital_Status_Unknown

Data shape after One-Hot Encoding

- 9240 rows
- 24 columns (including Target)

Feature Selection

DigitalSkola

Why?

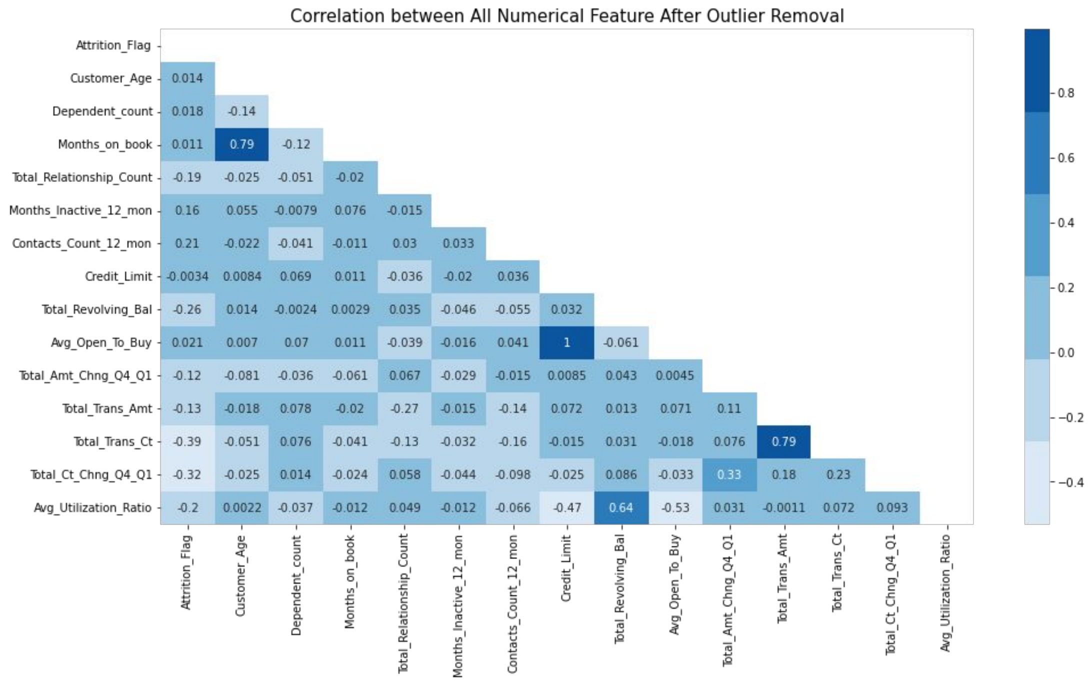
- Enables the model to train faster
- Reduce the complexity of model
- Improves performance if the right subset is selected
- Reduce overfitting

How?

- Correlation Statistics
  → Removing one of two highly correlated features
- Learning Algorithms
  → Using sklearn.feature_selection.SelectFromModel with sklearn.linear_model.Lasso

Reference: analyticsvidhya.com

Feature Selection - Correlation Statistics



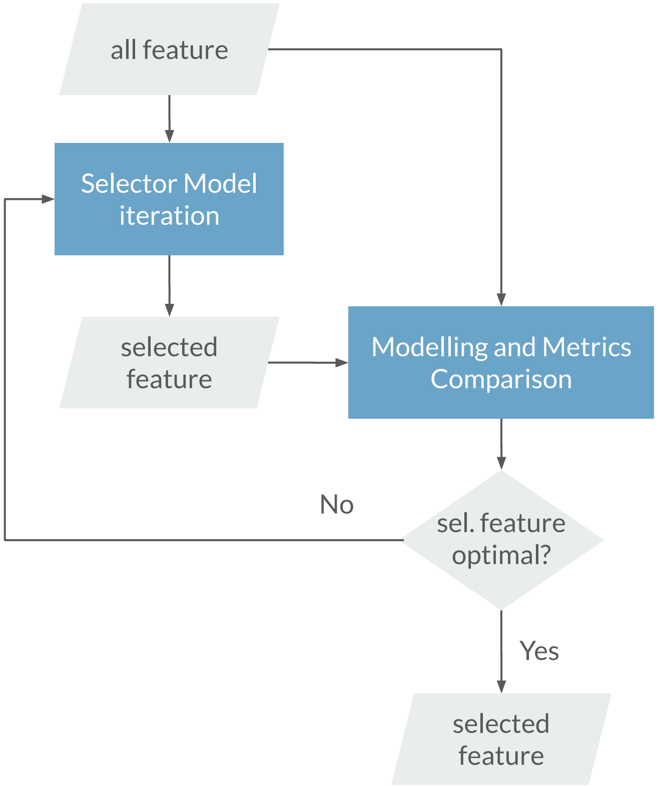
Avg_Open_To_Buy and *Credit_Limit* has perfect positive correlation, thus we only need one of them

- **Dropped feature**: *Avg_Open_To_Buy*

Data shape after Removing

- 9240 rows
- 23 columns (including Target)

Feature Selection - Learning Algorithms



Selector Model

```
select_1 = SelectFromModel(Lasso(alpha=0.008, random_state=0))
select_1.fit(_X_train, _y_train)
feature_lasso_1 = _X_train.columns[(select_1.get_support())]
```

Metrics Comparison

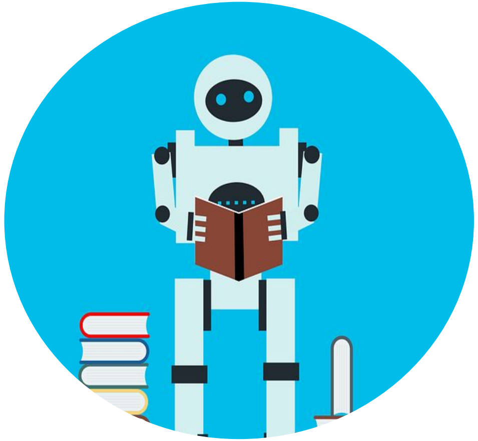
model	feature count	AUC score
all feature	22	0.9147
1st iteration	9	0.8691
2nd iteration	**10**	**0.9154**
3rd iteration	11	0.9154

Final Selected Input Features

- *Dependent_count*
- *Total_Relationship_Count*
- *Months_Inactive_12_mon*
- *Contacts_Count_12_mon*
- *Total_Revolving_Bal*
- *Total_Trans_Amt*
- *Total_Trans_Ct*
- *Total_Ct_Chng_Q4_Q1*
- *Gender_F*
- *Marital_Status_Married*

More on LASSO: spectdata.com

Modelling



Models

- Logistic Regression
- K Nearest Neighbors (k-NN)
- Random Forest

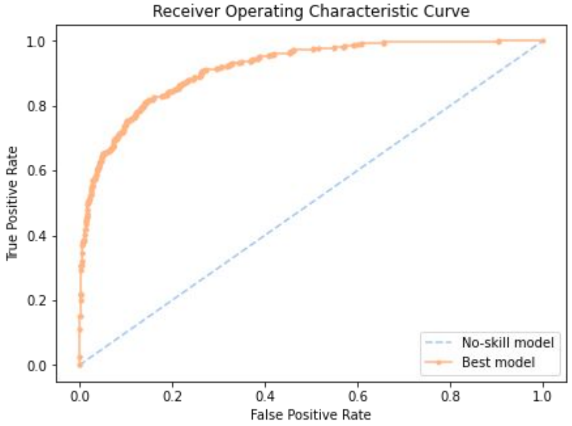
Evaluation Metrics

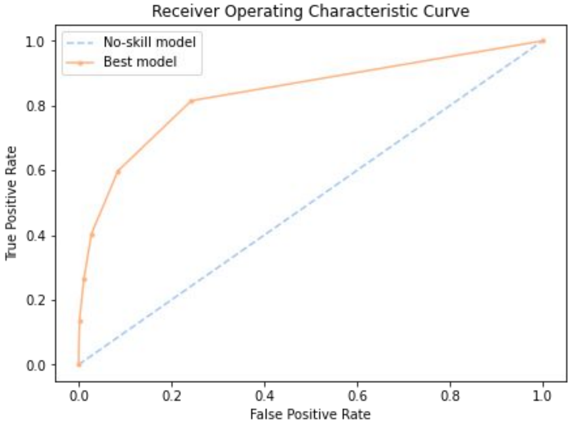
- We will use Area Under The Curve - Receiver Operating Characteristic (AUC-ROC) to measure the models performance, as we want to focus on predicting the **probability** of customer churns.

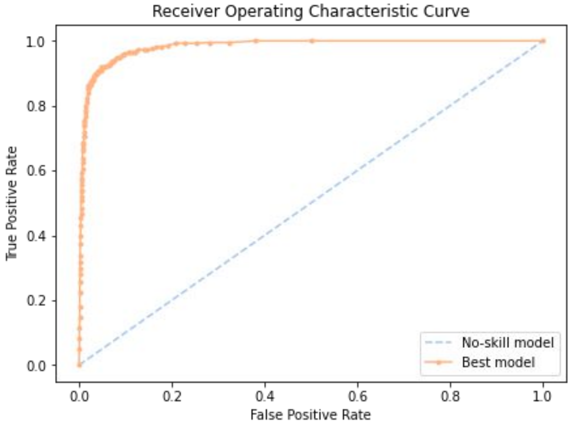
Train-Test Split

- For model evaluation purpose, we split the data into Train and Test set in ratio of 75:25

Modelling - Models and Evaluation







Logistic Regression	k-NN Classifier	Random Forest Classifier
AUC: 91.54%	**AUC:** 83.48%	**AUC:** 98.27%

Modelling - Hyperparameter Tuning

How?

```python
# specify parameters
knn_param_grid_1 = {'n_neighbors': list(range(1,22,2)),
                    'metric': ['euclidean', 'manhattan', 'minkowski'],
                    'weights': ['uniform', 'distance'],
                    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                    }

# search algorithm
knn_search_1 = GridSearchCV(knn_model,
                            knn_param_grid_1,
                            scoring = 'roc_auc',
                            cv = 10,
                            n_jobs = -1,
                            verbose = 1)

# train
knn_tune_result_1 = knn_search_1.fit(X_train, y_train)
```

```
Fitting 10 folds for each of 264 candidates, totalling 2640 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done 128 tasks      | elapsed:    2.1s
[Parallel(n_jobs=-1)]: Done 728 tasks      | elapsed:   13.5s
[Parallel(n_jobs=-1)]: Done 1728 tasks      | elapsed:   37.7s
[Parallel(n_jobs=-1)]: Done 2640 out of 2640 | elapsed:  1.2min finished
```

```python
# best hyperparameter
print('Best hyperparameter:', knn_tune_result_1.best_params_)
```

```
Best hyperparameter: {'algorithm': 'auto', 'metric': 'manhattan', 'n_neighbors': 21, 'weights': 'distance'}
```

Example for k-NN Classifier::

- Specify possible hyperparameter values of KNeighborsClassifier, and search the best value

- Use best hyperparameter output of the 1st iteration on 2nd iteration, and re-tune hyperparameters that still have possible values.
  In this case, re-tune the n_neighbors by trying integer larger than 21

Modelling - Hyperparameter Tuning

Metrics Comparison

model	AUC before tuning	AUC after tuning
Logistic Regression	91.54%	92.15%
k-NN Classifier	83.48%	90.73%
Random Forest Classifier	98.27%	98.29%

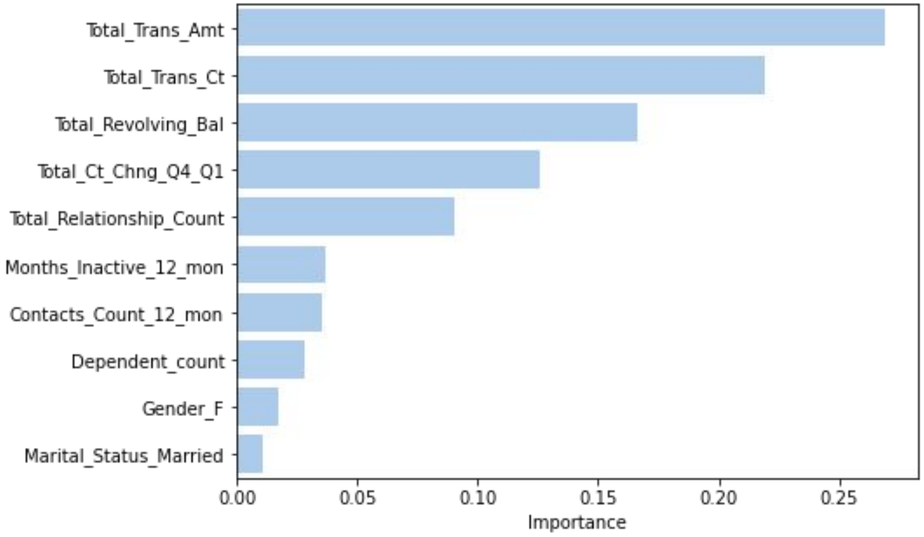
Conclusion

Best Models

Random Forest after tuning,
with hyperparameters as follows:

- **criterion**: ‘entropy’
- **max_depth**: 15
- **max_features**: ‘auto’
- **min_samples_leaf**: 1
- **min_samples_split**: 2
- **n_estimators**: 900

Feature Importance



Thank You