CREDIT FRAUD DETECTION

SC1015 PRESENTATION

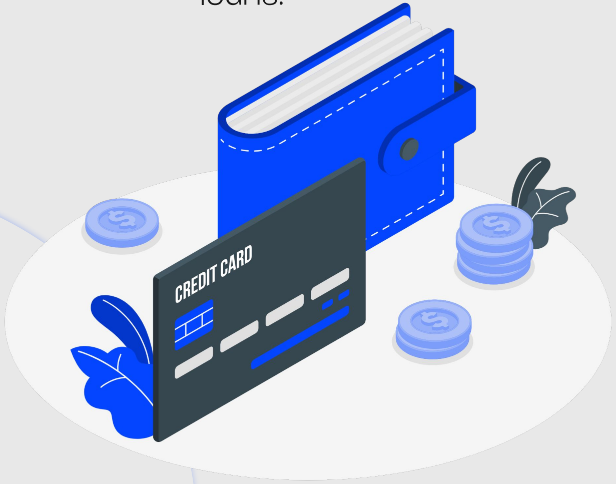
By: Jerald (U2221612D)
Bryan (U2222849J)
Jonathan (U2223191G)

TABLE OF CONTENTS

DATA SET

MOTIVATION

- Cashless payments are **increasingly common** with the push of such technology
- To **minimise** the **risk** of lending money to customers who may not pay back their loans.
- To understand which **factors** affect people from defaulting credit card loans.



DBS PayLah!

PAYNOW

73%

of Singaporeans own at least 1 credit card in 2023

DATA SET

application_data

columns_description

previous_application

FEATURE SELECTION

19/124 Columns in application_data

TARGET | NAME_CONTRACT_TYPE | FLAG_OWN_CAR | FLAG_OWN_REALTY | AMT_CREDIT | AMT_GOODS_PRICE, etc.

PROBLEM DEFINITION

"To find the **highest correlation** of credit card defaulters and to **predict** loan default risk of defaulters."

DATA CLEANING

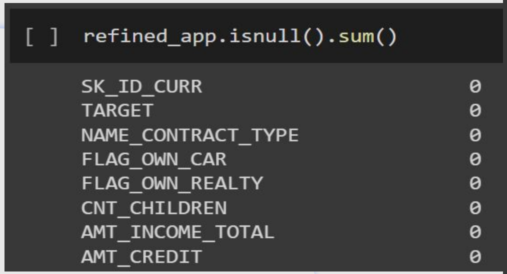
SPLITTING THE DATA

NUMERICAL

- Finding **NULL** values within dataset
- Fill in missing values with **MEDIAN**

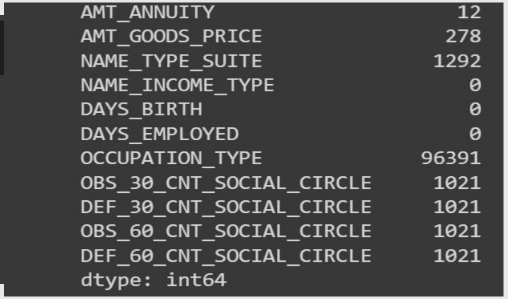
CATEGORICAL

- Finding **empty values** within dataset
- Fill in missing values with **NULL**

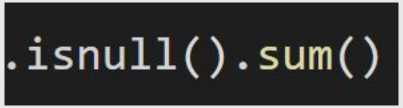


```
[ ]  refined_app.isnull().sum()

     SK_ID_CURR                  0
     TARGET                      0
     NAME_CONTRACT_TYPE          0
     FLAG_OWN_CAR                0
     FLAG_OWN_REALTY             0
     CNT_CHILDREN                0
     AMT_INCOME_TOTAL            0
     AMT_CREDIT                  0
```



```
AMT_ANNUITY                    12
AMT_GOODS_PRICE               278
NAME_TYPE_SUITE              1292
NAME_INCOME_TYPE                0
DAYS_BIRTH                      0
DAYS_EMPLOYED                   0
OCCUPATION_TYPE             96391
OBS_30_CNT_SOCIAL_CIRCLE     1021
DEF_30_CNT_SOCIAL_CIRCLE     1021
OBS_60_CNT_SOCIAL_CIRCLE     1021
DEF_60_CNT_SOCIAL_CIRCLE     1021
dtype: int64
```

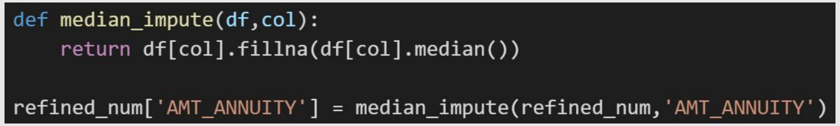


```
.isnull().sum()
```

FILLING MISSING VALUES

NUMERICAL

6/13 columns with missing data



```
def median_impute(df,col):
    return df[col].fillna(df[col].median())

refined_num['AMT_ANNUITY'] = median_impute(refined_num,'AMT_ANNUITY')
```

Filling in with **MEDIAN**

CATEGORICAL

2/6 columns with missing data

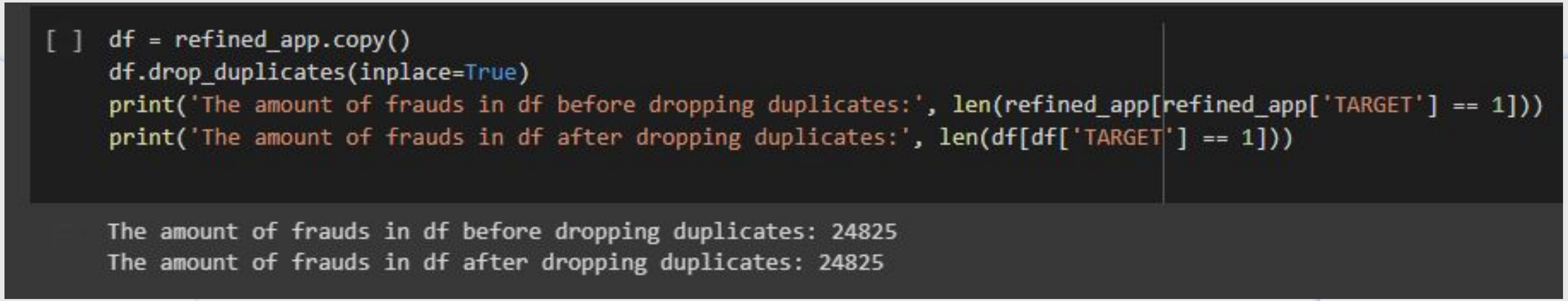
NAME_TYPE_SUITE: Fill with "***Unaccompanied***"

OCCUPATION_TYPE: Fill with "***NILL***"

```
.fillna
```

REMOVING DUPLICATES

- Ensure each record represents a unique observation
- Affects data analysis:
  - Skewness of results of statistical analysis
  - Increasing size of data set unnecessarily
  - Overfitting in machine learning



```
[ ] df = refined_app.copy()
    df.drop_duplicates(inplace=True)
    print('The amount of frauds in df before dropping duplicates:', len(refined_app[refined_app['TARGET'] == 1]))
    print('The amount of frauds in df after dropping duplicates:', len(df[df['TARGET'] == 1]))
```

```
The amount of frauds in df before dropping duplicates: 24825
The amount of frauds in df after dropping duplicates: 24825
```

DATA VISUALISATION

NUMERICAL DATA

Target (whether individuals defaulted)

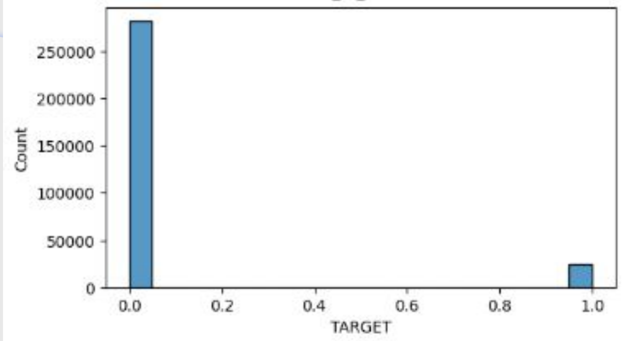
Heavily skewed - more 0s than 1s

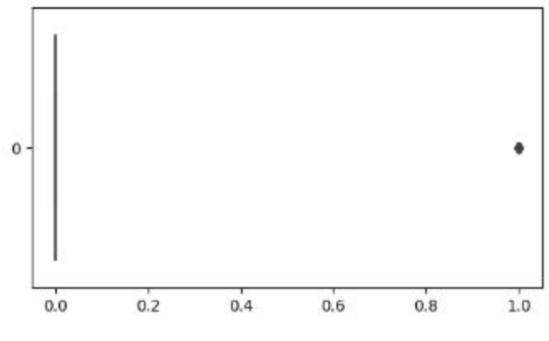
Repercussions:

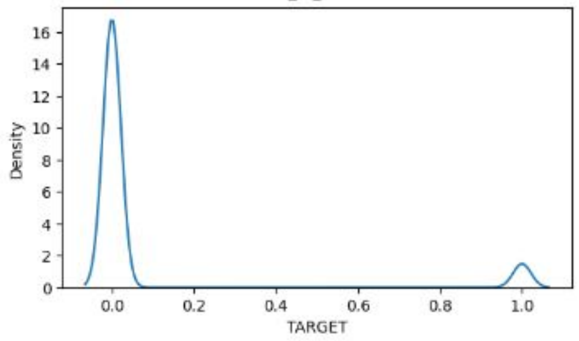
1. Biased model
2. Poor performance of minority class
3. Misleading evaluation metrics

Possible solutions:

1. Undersampling of majority class
2. Oversampling of minority class (SMOTE)
3. Cost-sensitive learning
4. Ensemble method

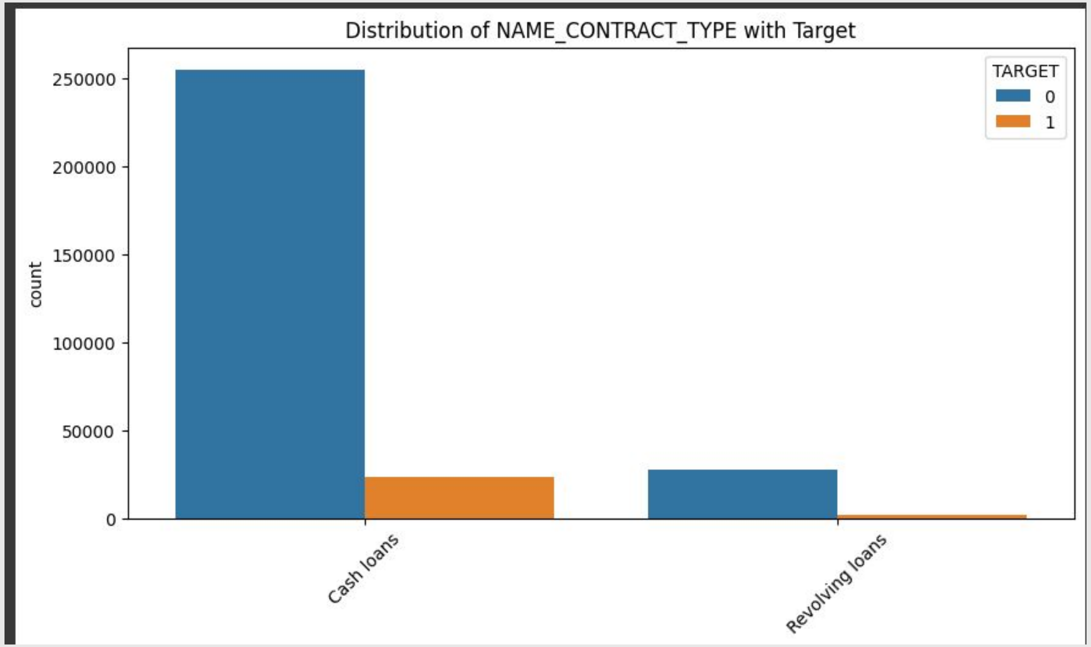






CATEGORICAL DATA

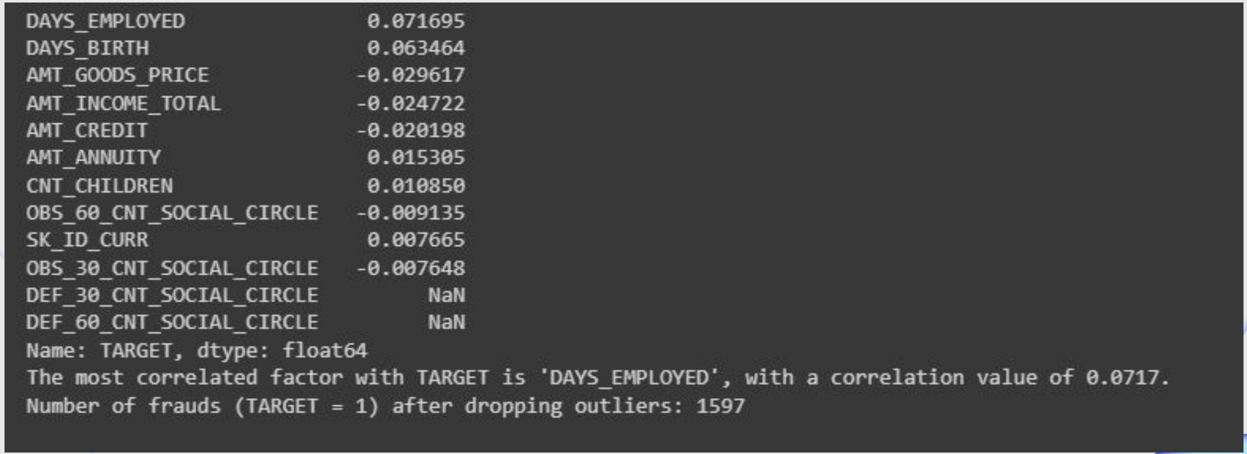
UNDERSAMPLING



CORRELATION (RAW DATA)

TARGET

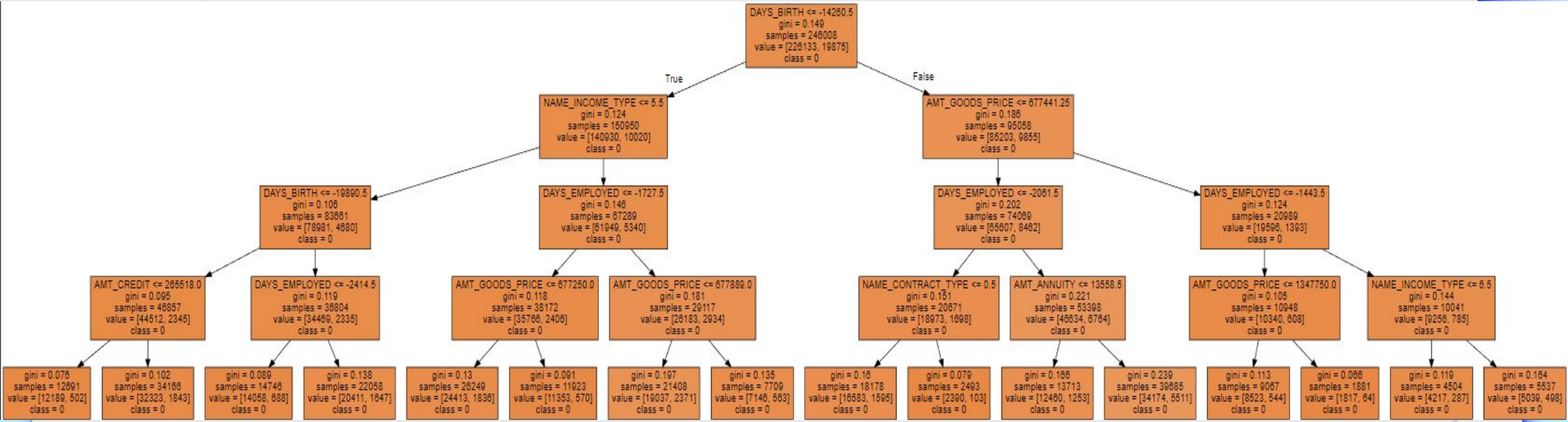
DAYS EMPLOYED
DAYS BIRTH



```
DAYS_EMPLOYED              0.071695
DAYS_BIRTH                 0.063464
AMT_GOODS_PRICE           -0.029617
AMT_INCOME_TOTAL          -0.024722
AMT_CREDIT                -0.020198
AMT_ANNUITY                0.015305
CNT_CHILDREN               0.010850
OBS_60_CNT_SOCIAL_CIRCLE  -0.009135
SK_ID_CURR                 0.007665
OBS_30_CNT_SOCIAL_CIRCLE  -0.007648
DEF_30_CNT_SOCIAL_CIRCLE        NaN
DEF_60_CNT_SOCIAL_CIRCLE        NaN
Name: TARGET, dtype: float64
The most correlated factor with TARGET is 'DAYS_EMPLOYED', with a correlation value of 0.0717.
Number of frauds (TARGET = 1) after dropping outliers: 1597
```

MACHINE LEARNING

DECISION TREE


DAYS_BIRTH <= -14260.5 gini = 0.149 samples = 246008 value = [226133, 19875] class = 0
True
False
NAME_INCOME_TYPE <= 5.5 gini = 0.124 samples = 150950 value = [140930, 10020] class = 0
AMT_GOODS_PRICE <= 677441.25 gini = 0.186 samples = 95058 value = [85203, 9855] class = 0
DAYS_BIRTH <= -19890.5 gini = 0.106 samples = 83661 value = [78981, 4680] class = 0
DAYS_EMPLOYED <= -1727.5 gini = 0.146 samples = 67289 value = [61949, 5340] class = 0
DAYS_EMPLOYED <= -2061.5 gini = 0.202 samples = 74069 value = [65607, 8462] class = 0
DAYS_EMPLOYED <= -1443.5 gini = 0.124 samples = 20989 value = [19596, 1393] class = 0
AMT_CREDIT <= 265518.0 gini = 0.095 samples = 46857 value = [44512, 2345] class = 0
DAYS_EMPLOYED <= -2414.5 gini = 0.119 samples = 36804 value = [34469, 2335] class = 0
AMT_GOODS_PRICE <= 677250.0 gini = 0.118 samples = 38172 value = [35766, 2406] class = 0
AMT_GOODS_PRICE <= 677889.0 gini = 0.181 samples = 29117 value = [26183, 2934] class = 0
NAME_CONTRACT_TYPE <= 0.5 gini = 0.151 samples = 20671 value = [18973, 1698] class = 0
AMT_ANNUITY <= 13558.5 gini = 0.221 samples = 53398 value = [46634, 6764] class = 0
AMT_GOODS_PRICE <= 1347750.0 gini = 0.105 samples = 10948 value = [10340, 608] class = 0
NAME_INCOME_TYPE <= 6.5 gini = 0.144 samples = 10041 value = [9256, 785] class = 0
gini = 0.076 samples = 12691 value = [12189, 502] class = 0
gini = 0.102 samples = 34166 value = [32323, 1843] class = 0
gini = 0.089 samples = 14746 value = [14058, 688] class = 0
gini = 0.138 samples = 22058 value = [20411, 1647] class = 0
gini = 0.13 samples = 26249 value = [24413, 1836] class = 0
gini = 0.091 samples = 11923 value = [11353, 570] class = 0
gini = 0.197 samples = 21408 value = [19037, 2371] class = 0
gini = 0.135 samples = 7709 value = [7146, 563] class = 0
gini = 0.16 samples = 18178 value = [16583, 1595] class = 0
gini = 0.079 samples = 2493 value = [2390, 103] class = 0
gini = 0.166 samples = 13713 value = [12460, 1253] class = 0
gini = 0.239 samples = 39685 value = [34174, 5511] class = 0
gini = 0.113 samples = 9067 value = [8523, 544] class = 0
gini = 0.066 samples = 1881 value = [1817, 64] class = 0
gini = 0.119 samples = 4504 value = [4217, 287] class = 0
gini = 0.164 samples = 5537 value = [5039, 498] class = 0

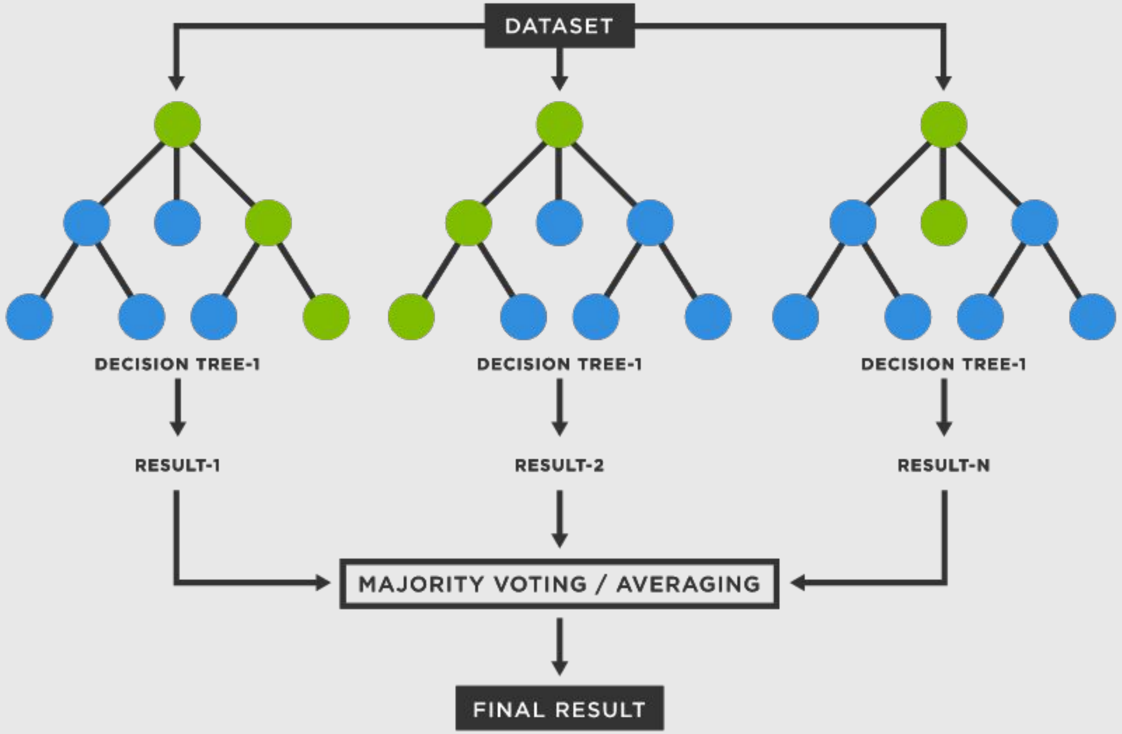
DECISION TREE

Train Accuracy	Test Accuracy	Precision	Recall	F1 Score
91.92%	91.95%	0	0	0

	Predict Safe (0)	Predict Fraud (1)
Actual Safe (0)	56553	**0**
Actual Fraud (1)	**4950**	0

True Positive Rate	0%
False Positive Rate	0%

RANDOM FOREST



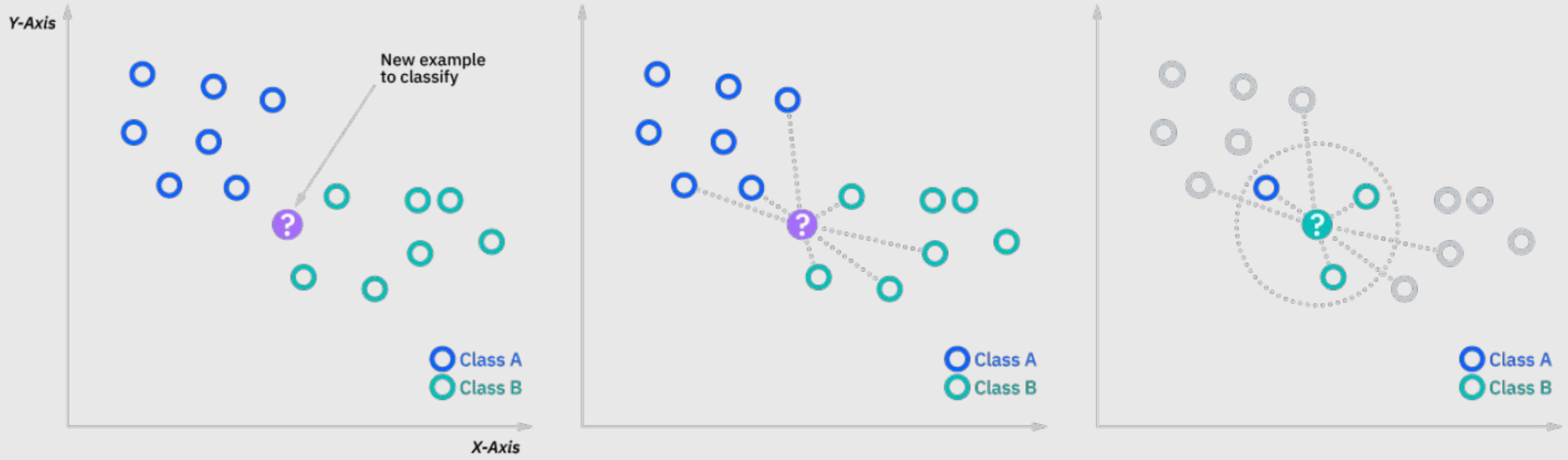
RANDOM FOREST

Train Accuracy	Test Accuracy	Precision	Recall	F1 Score
98.40%	91.87%	0.2298	0.0040	0.0079

	Predict Safe (0)	Predict Fraud (1)
Actual Safe (0)	56486	**67**
Actual Fraud (1)	**4930**	20

True Positive Rate	0.40%
False Positive Rate	0.11%

K-NEAREST NEIGHBORS



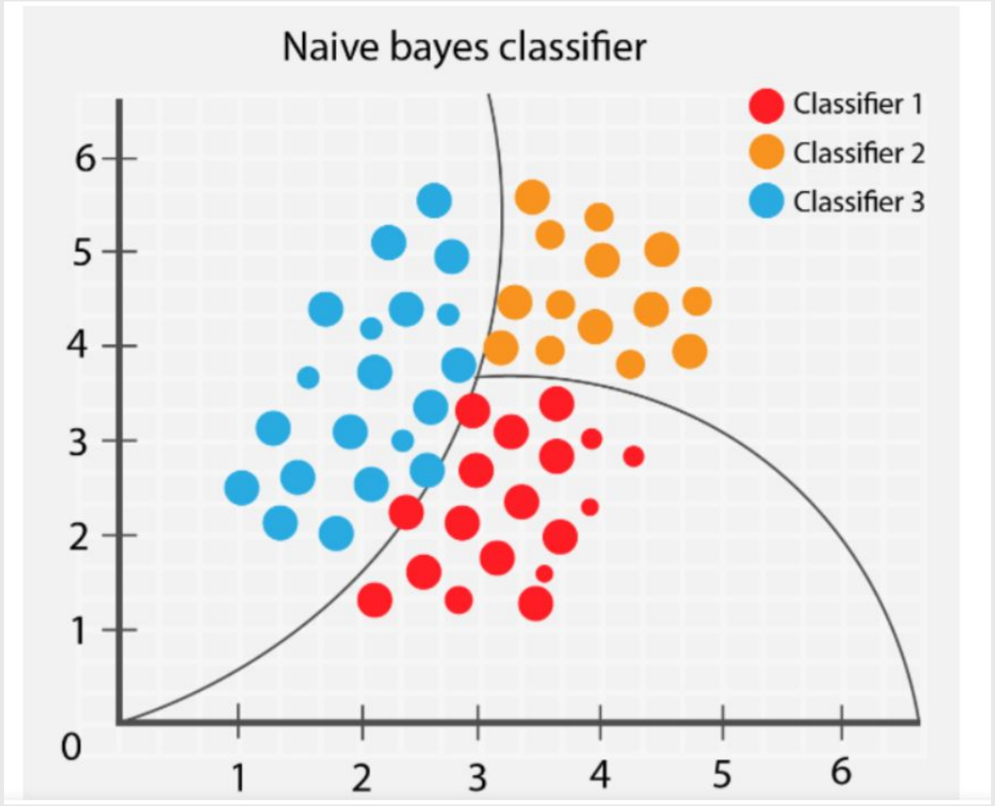
K-NEAREST NEIGHBORS

Train Accuracy	Test Accuracy	Precision	Recall	F1 Score
92.70%	90.29%	0.1308	0.0365	0.0571

	Predict Safe (0)	Predict Fraud (1)
Actual Safe (0)	55351	**1202**
Actual Fraud (1)	**4769**	181

True Positive Rate	3.65%
False Positive Rate	2.12%

NAIVE BAYES


Naive bayes classifier
Classifier 1
Classifier 2
Classifier 3
0
1
2
3
4
5
6
1
2
3
4
5
6

NAIVE BAYES

Train Accuracy	Test Accuracy	Precision	Recall	F1 Score
91.30%	91.40%	0.0687	0.0054	0.0101

	Predict Safe (0)	Predict Fraud (1)
Actual Safe (0)	56187	**366**
Actual Fraud (1)	**4923**	27

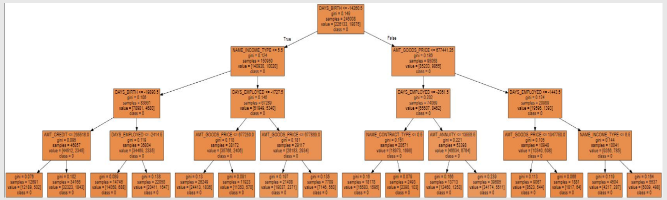
True Positive Rate	0.54%
False Positive Rate	0.64%

MODEL COMPARISON

	model	train_acc	test_acc	precision	recall	f1_score
1	Decision Tree	0.919210	0.919516	0.000000	0.000000	0.000000
2	Random Forest	0.984074	0.918752	0.229885	0.004040	0.007941
3	K-Nearest Neighbours	0.927096	0.902915	0.130875	0.036566	0.057161
4	Gaussian Naive Bayes	0.913023	0.914004	0.068702	0.005455	0.010107

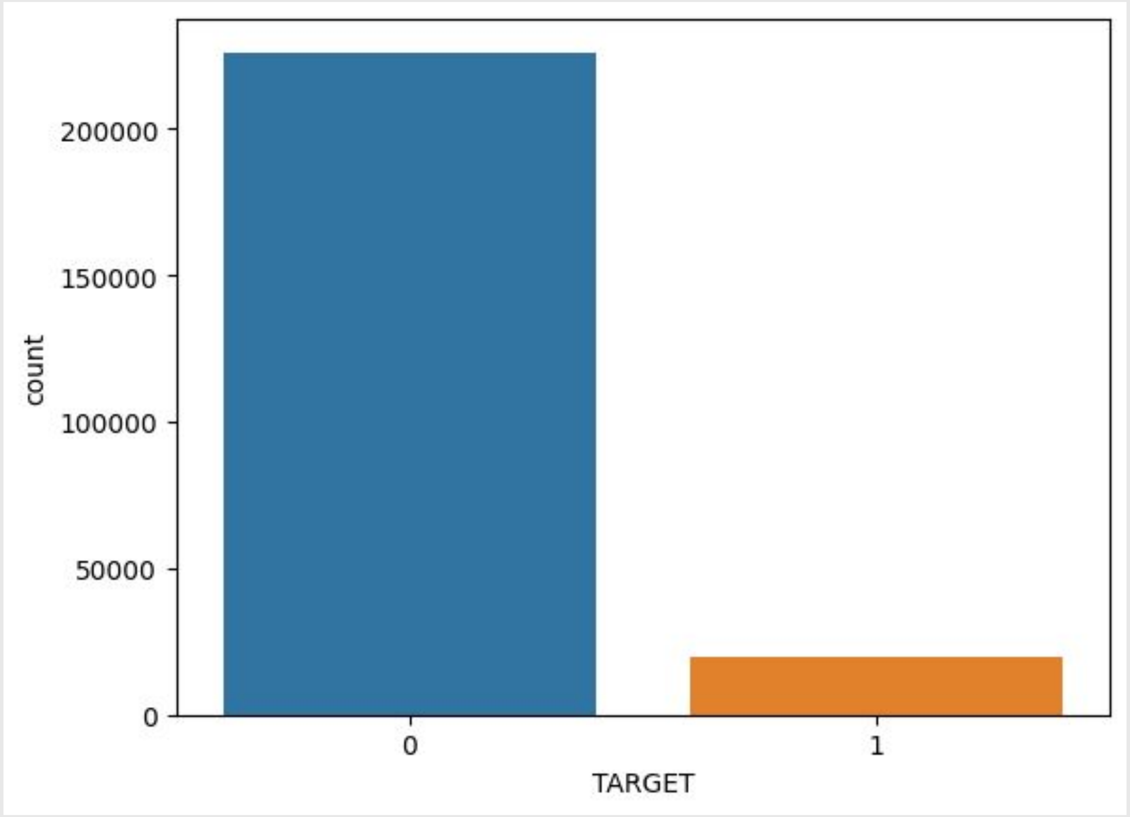
MODEL COMPARISON

- Highest test accuracy: **Decision Tree Classifier**
  - **91.95%**
  - Predicts all customers as safe (target=0)
  - Useless at detecting if customer will default

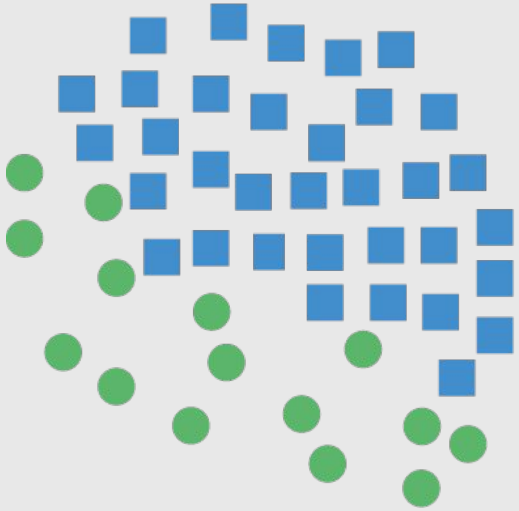

DAYS_BIRTH <= -14260.5 gini = 0.149 samples = 246008 value = [226133, 19875] class = 0
True
False
NAME_INCOME_TYPE <= 5.5 gini = 0.124 samples = 150950 value = [140930, 10020] class = 0
AMT_GOODS_PRICE <= 677441.25 gini = 0.186 samples = 95058 value = [85203, 9855] class = 0
DAYS_BIRTH <= -19890.5 gini = 0.106 samples = 83661 value = [78981, 4680] class = 0
DAYS_EMPLOYED <= -1727.5 gini = 0.146 samples = 67289 value = [61949, 5340] class = 0
DAYS_EMPLOYED <= -2061.5 gini = 0.202 samples = 74069 value = [65607, 8462] class = 0
DAYS_EMPLOYED <= -1443.5 gini = 0.124 samples = 20989 value = [19596, 1393] class = 0
AMT_CREDIT <= 265518.0 gini = 0.095 samples = 46857 value = [44512, 2345] class = 0
DAYS_EMPLOYED <= -2414.5 gini = 0.119 samples = 36804 value = [34469, 2335] class = 0
AMT_GOODS_PRICE <= 677250.0 gini = 0.118 samples = 38172 value = [35766, 2406] class = 0
AMT_GOODS_PRICE <= 677889.0 gini = 0.181 samples = 29117 value = [26183, 2934] class = 0
NAME_CONTRACT_TYPE <= 0.5 gini = 0.161 samples = 20671 value = [18973, 1698] class = 0
AMT_ANNUITY <= 13558.5 gini = 0.221 samples = 53398 value = [46634, 6764] class = 0
AMT_GOODS_PRICE <= 1347750.0 gini = 0.105 samples = 10948 value = [10340, 608] class = 0
NAME_INCOME_TYPE <= 6.5 gini = 0.144 samples = 10041 value = [9256, 785] class = 0
gini = 0.076 samples = 12691 value = [12189, 502] class = 0
gini = 0.102 samples = 34166 value = [32323, 1843] class = 0
gini = 0.089 samples = 14746 value = [14058, 688] class = 0
gini = 0.138 samples = 22058 value = [20411, 1647] class = 0
gini = 0.13 samples = 26249 value = [24413, 1836] class = 0
gini = 0.091 samples = 11923 value = [11353, 570] class = 0
gini = 0.197 samples = 21408 value = [19037, 2371] class = 0
gini = 0.135 samples = 7709 value = [7146, 563] class = 0
gini = 0.16 samples = 18178 value = [16583, 1595] class = 0
gini = 0.079 samples = 2493 value = [2390, 103] class = 0
gini = 0.166 samples = 13713 value = [12460, 1253] class = 0
gini = 0.239 samples = 39685 value = [34174, 5511] class = 0
gini = 0.113 samples = 9067 value = [8523, 544] class = 0
gini = 0.066 samples = 1881 value = [1817, 64] class = 0
gini = 0.119 samples = 4504 value = [4217, 287] class = 0
gini = 0.164 samples = 5537 value = [5039, 498] class = 0

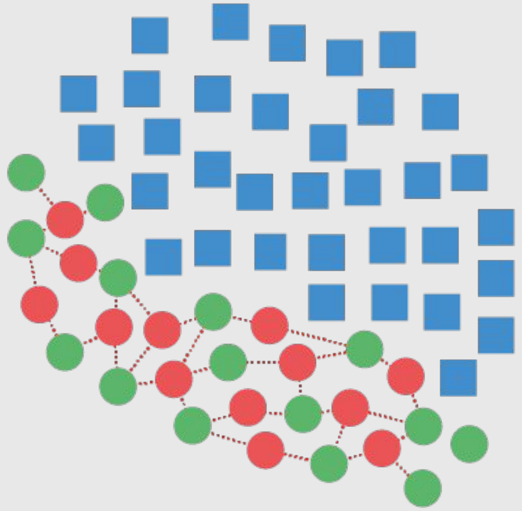
- Highest Recall (TPR): **K-Nearest Neighbors**
  - **3.65%**
  - Extremely low detection rate
  - Takes extremely long time to predict

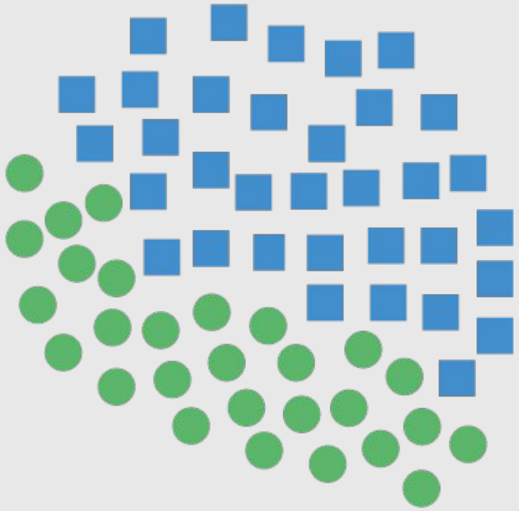
IMBALANCED TRAINING DATASET



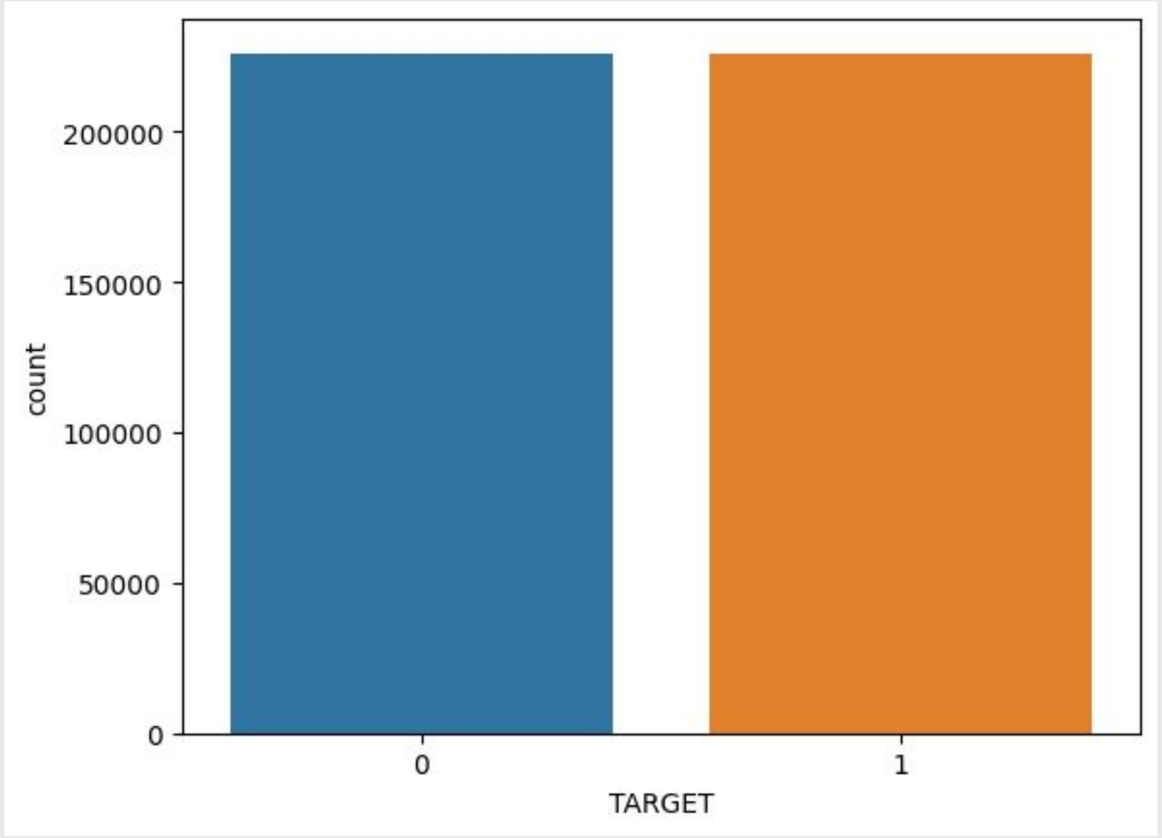
SYNTHETIC MINORITY OVER-SAMPLING TECHNIQUE (SMOTE)


Original Dataset


Generating Samples


Resampled Dataset

RESAMPLED TRAINING DATASET



Model Comparison - SMOTE

Rank	model	train_acc	test_acc	precision	recall	f1_score
4	Decision Tree	0.725909	0.692487	0.089975	0.309495	0.139418
1	Random Forest	0.993763	0.897013	0.128755	0.048485	0.070443
3	K-Nearest Neighbours	0.926815	0.724013	0.097429	0.293939	0.146349
2	Gaussian Naive Bayes	0.497342	0.893339	0.063922	0.023838	0.034726

Model Comparison - SMOTE

- Sacrifice accuracy to improve model robustness
    - Decreased Accuracy
    - Increased Recall / F1 Score

- Highest test accuracy: **Random Forest Classifier**
    - **89.70% Accuracy**
    - Low recall: **4.84%**

- Highest Recall (TPR): **Decision Tree Classifier**
    - **30.94% Recall**
    - Low accuracy: **69.24%**

Model Comparison - SMOTE

	model	train_acc	test_acc	precision	recall	f1_score
3	**K-Nearest Neighbours**	**0.926815**	**0.724013**	**0.097429**	**0.293939**	**0.146349**

BEST OVERALL MODEL: **K-Nearest Neighbours**

GOOD COMPROMISE **BETWEEN** ACCURACY & RECALL

CONCLUSION

INSIGHTS & TAKEAWAYS

1 EDA

Understand overview of data

2 FEATURE SELECTION

Reduce training time

AND

Model performance not affected

3 SMOTE

REDUCES ACCURACY

BUT

INCREASES PERFORMANCE

RECOMMENDATIONS

INSIGHTS & RECOMMENDATIONS

Contributing Factors

DAYS_EMPLOYED & DAYS_BIRTH

OUTCOME

SMOTE

✓ Address class **imbalance** in dataset

– under represented –

✗ **INTRODUCE** NOISE

OVERFITTING

THANK YOU

slidesgo