# EDA No. 5 AAA Project Martin George mgeorgevienna@gmail.com

In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
df = pd.read_csv('member_sample.csv', index_col = 0)
```

# Application of classification model on AAA data

# Usage of SMOTE library to oversampling when imbalanced samples exists.

In [2]:
```
df.head()
df.info()
df.columns
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21344 entries, 0 to 99998
Columns: 112 entries, Individual Key to Was Towed To AAR Referral
dtypes: float64(35), object(77)
memory usage: 18.4+ MB
```

Out[2]:
```
Index(['Individual Key', 'Household Key', 'Member Flag', 'City',
       'State - Grouped', 'ZIP5', 'ZIP9', 'FSV CMSI Flag',
       'FSV Credit Card Flag', 'FSV Deposit Program Flag',
       ...
       'SC Vehicle Manufacturer Name', 'SC Vehicle Model Name',
       'SVC Facility Name', 'SVC Facility Type', 'Total Cost',
       'Tow Destination Latitude', 'Tow Destination Longitude',
       'Tow Destination Name', 'Was Duplicated', 'Was Towed To AAR Referral'],
      dtype='object', length=112)
```

In [3]:
```
df.head()
```

Out[3]:

| | Individual Key | Household Key | Member Flag | City | State - Grouped | ZIP5 | ZIP9 | FSV CMSI Flag | FSV Credit Card Flag | FSV Deposit Program Flag | ... | SC Vehicle Manufacturer Name | SC Vehicle Model Name | Fa I |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10000003.0 | 10462590.0 | Y | NEW HAVEN | CT | 6511.0 | 65111349.0 | N | N | N | ... | NaN | NaN | |
| **1** | 52211550.0 | 4500791.0 | Y | WEST WARWICK | RI | 2893.0 | 28933850.0 | N | Y | N | ... | TOYOTA | CAMRY | AS WREC SER |
| **2** | 52211550.0 | 4500791.0 | Y | WEST WARWICK | RI | 2893.0 | 28933850.0 | N | Y | N | ... | TOYOTA | CAMRY | Wr Sc |
| **3** | 52211550.0 | 4500791.0 | Y | WEST WARWICK | RI | 2893.0 | 28933850.0 | N | Y | N | ... | TOYOTA | CAMRY | AS WREC SER |
| **4** | 52211550.0 | 4500791.0 | Y | WEST WARWICK | RI | 2893.0 | 28933850.0 | N | Y | N | ... | TOYOTA | CAMRY | AS WREC SER |

5 rows × 112 columns

In [4]:
```
df.groupby('FSV CMSI Flag').mean()
```

Out[4]:

| FSV CMSI Flag | Individual Key | Household Key | ZIP5 | ZIP9 | Length Of Residence | Do Not Direct Mail Solicit | Email Available | ERS ENT Count Year 1 | ERS ENT Count Year 2 | ERS ENT Count Year 3 | ... | Meml Mat F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N | 3.403291e+07 | 1.600860e+07 | 2947.671848 | 2.948020e+07 | 11.552839 | 0.054041 | 0.52604 | 0.517824 | 0.921864 | 0.952447 | ... | |
| Y | 2.398762e+07 | 1.515128e+07 | 2885.457413 | 2.885794e+07 | 11.088766 | 0.027340 | 0.75184 | 0.531746 | 1.193878 | 1.090703 | ... | |

2 rows × 35 columns

# Python library SMOTE can be also used to populate samples

In [5]:
```
df['FSV CMSI Flag'].value_counts(normalize = True)
```

Out[5]:
```
N    0.955444
Y    0.044556
Name: FSV CMSI Flag, dtype: float64
```

In [6]:
```
yes = df.loc[df['FSV CMSI Flag'] == 'Y']
```

In [7]:
```
no = df.loc[df['FSV CMSI Flag'] == 'N']
```

In [8]:
```
no_sample = no.sample(951)
```

In [9]:
```
combined = pd.concat( [yes, no_sample]).dropna(subset = ['Total Cost'])
```

In [10]:
```
combined.shape
```

Out[10]:
```
(1359, 112)
```

In [11]:
```
combined['FSV CMSI Flag']. value_counts()
```

Out[11]:
```
Y    763
N    596
Name: FSV CMSI Flag, dtype: int64
```

In [12]:
```
X = combined[['Total Cost']]
y = combined[['FSV CMSI Flag']]
```

In [13]:
```
X.shape
```

Out[13]:
```
(1359, 1)
```

In [14]:
```
y.shape
```

Out[14]:
```
(1359, 1)
```

In [15]:
```
from sklearn.linear_model import LogisticRegression
```

In [16]:
```
lgr = LogisticRegression()
```

In [17]:
```
from sklearn.model_selection import train_test_split

# split into 70:30 ration
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)

# describes info about train and test set
print("Number transactions X_train dataset: ", X_train.shape)
print("Number transactions y_train dataset: ", y_train.shape)
print("Number transactions X_test dataset: ", X_test.shape)
print("Number transactions y_test dataset: ", y_test.shape)
```

```
Number transactions X_train dataset:  (951, 1)
Number transactions y_train dataset:  (951, 1)
Number transactions X_test dataset:  (408, 1)
Number transactions y_test dataset:  (408, 1)
```

In [18]:
```
lgr.fit(X,y)
```

```
C:\Users\unodc\anaconda3\lib\site-packages\sklearn\utils\validation.py:760: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)
```

Out[18]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [19]:
```
lgr.predict_proba(X)
```

Out[19]:
```
array([[0.44327799, 0.55672201],
       [0.436015  , 0.563985  ],
       [0.43962616, 0.56037384],
       ...,
       [0.43734705, 0.56265295],
       [0.44145129, 0.55854871],
       [0.44304957, 0.55695043]])
```

In [20]:
```
lgr.score(X,y)
```

Out[20]:
```
0.5614422369389257
```

In [21]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, classification_report
```

In [22]:
```
predictions = lgr.predict(X_test)

# print classification report
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           N       0.00      0.00      0.00       179
           Y       0.56      1.00      0.72       229

    accuracy                           0.56       408
   macro avg       0.28      0.50      0.36       408
weighted avg       0.32      0.56      0.40       408


C:\Users\unodc\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: P
recision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_divi
sion` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

# Now we can consider the Logistics Regression with original data set without making the classifying sample equal.

In [23]:
```
df_t= X = df.dropna(subset = ['Total Cost'])
```

In [24]:
```
df_t.shape
```

Out[24]:
```
(13944, 112)
```

In [25]:
```
X = df_t[['Total Cost']].dropna(subset = ['Total Cost'])
y = df_t[['FSV CMSI Flag']]
```

In [26]:
```
X.shape
```

Out[26]:
```
(13944, 1)
```

In [27]:
```
y.shape
```

Out[27]:
```
(13944, 1)
```

In [28]:
```
df_t['FSV CMSI Flag']. value_counts()
```

Out[28]:
```
N    13181
Y      763
Name: FSV CMSI Flag, dtype: int64
```

In [29]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

In [30]:
```
# describes info about train and test set
print("Number transactions X_train dataset: ", X_train.shape)
print("Number transactions y_train dataset: ", y_train.shape)
print("Number transactions X_test dataset: ", X_test.shape)
print("Number transactions y_test dataset: ", y_test.shape)
```

```
Number transactions X_train dataset:  (9760, 1)
Number transactions y_train dataset:  (9760, 1)
Number transactions X_test dataset:  (4184, 1)
Number transactions y_test dataset:  (4184, 1)
```

In [31]:
```
lgr.fit(X,y)
```

```
C:\Users\unodc\anaconda3\lib\site-packages\sklearn\utils\validation.py:760: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)
```

Out[31]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [32]:
```
lgr.score(X,y)
```

Out[32]:
```
0.945281124497992
```

In [33]:
```
predictions = lgr.predict(X_test)

# print classification report
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           N       0.95      1.00      0.97      3968
           Y       0.00      0.00      0.00       216

    accuracy                           0.95      4184
   macro avg       0.47      0.50      0.49      4184
weighted avg       0.90      0.95      0.92      4184


C:\Users\unodc\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: P
recision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_divi
sion` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

# The recall of the minority class in very less. It proves that the model is more biased towards majority class. So, it proves that this is not the best model.

# Usage of SMOTE

In [34]:
```
!pip install imblearn
```

```
Collecting imblearn
  Downloading imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Collecting imbalanced-learn
  Downloading imbalanced_learn-0.6.2-py3-none-any.whl (163 kB)
Requirement already satisfied: scikit-learn>=0.22 in c:\users\unodc\anaconda3\lib\site-packages (from imbalan
ced-learn->imblearn) (0.22.1)
Requirement already satisfied: numpy>=1.11 in c:\users\unodc\anaconda3\lib\site-packages (from imbalanced-lea
rn->imblearn) (1.18.1)
Requirement already satisfied: joblib>=0.11 in c:\users\unodc\anaconda3\lib\site-packages (from imbalanced-le
arn->imblearn) (0.14.1)
Requirement already satisfied: scipy>=0.17 in c:\users\unodc\anaconda3\lib\site-packages (from imbalanced-lea
rn->imblearn) (1.4.1)
Installing collected packages: imbalanced-learn, imblearn
Successfully installed imbalanced-learn-0.6.2 imblearn-0.0
```

In [35]:
```
from imblearn.over_sampling import SMOTE
```

In [38]:
```
sm = SMOTE(random_state = 2)
X_train_res, y_train_res = sm.fit_sample(X_train, y_train)

print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))

#print("After OverSampling, counts of label '1': {}".format(sum(y_train_res == 1)))
#print("After OverSampling, counts of label '0': {}".format(sum(y_train_res == 0)))
```

```
After OverSampling, the shape of train_X: (18426, 1)
After OverSampling, the shape of train_y: (18426, 1)
```

In [41]:
```
print("Number transactions X_train dataset: ", X_train.shape)
print("Number transactions y_train dataset: ", y_train.shape)
print("Number transactions X_test dataset: ", X_test.shape)
print("Number transactions y_test dataset: ", y_test.shape)
```

```
Number transactions X_train dataset:  (9760, 1)
Number transactions y_train dataset:  (9760, 1)
Number transactions X_test dataset:  (4184, 1)
Number transactions y_test dataset:  (4184, 1)
```

In [40]:
```
lr1 = LogisticRegression()
lr1.fit(X_train_res, y_train_res)
predictions = lr1.predict(X_test)

# print classification report
print(classification_report(y_test, predictions))
```

```
C:\Users\unodc\anaconda3\lib\site-packages\sklearn\utils\validation.py:760: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)

              precision    recall  f1-score   support

           N       0.94      0.37      0.54      3968
           Y       0.05      0.56      0.09       216

    accuracy                           0.38      4184
   macro avg       0.49      0.47      0.31      4184
weighted avg       0.89      0.38      0.51      4184
```

In [42]:
```
lr1.score(X,y)
```

Out[42]:
```
0.39945496270797476
```

## precision recall f1-score support ( Equal samples)

## N 0.00 0.00 0.00 179

## Y 0.56 1.00 0.72 229

## precision recall f1-score support ( Original samples)

## N 0.95 1.00 0.97 3968

## Y 0.00 0.00 0.00 216

## precision recall f1-score support ( Sampling with SMOTE)

## N 0.94 0.37 0.54 3968

## Y 0.05 0.56 0.09 216

# We can see a clear advantage of using SMOTE in recall value for both minority class and majority class.

In [ ]: