Mid Bootcamp Project

Case Study: Classification

Manuel Zimmermann, Polly Jenkinson, Sybille Kiziltan, Brett Hunt

Graphics: Freepik

Problem Definition

1. What are the demographics and characteristics of customers who accept or reject credit offers?

2. What insights did you discover within your analysis which would be meaningful to the bank?

Contextualizing the Data

1. Initial Exploration
   a. Examining the Data and the Data Imbalances
   b. Ensure to have a complete Data Set with a proper convention
   c. Get rid of data points that are not impactful for the prediction

Snake Casing

```
#Snake Casing - Lower case and replacing blanks with underscore
new_cols = [col.replace(" ", "_").lower() for col in data]
data.columns = new_cols
new_cols2 = [col.replace("#_", "") for col in data]
data.columns = new_cols2
```

Checking for NaN in dataset

```
#Checking for NaN in dataset
data.isna().sum()/data.shape[0]
```

```
data['offer_accepted'].value_counts()
```

```
No      16954
Yes      1020
Name: offer_accepted, dtype: int64
```

```
#Dropping Data at Index 38,101
data = data.drop(labels=[38,101], axis=0)
```

Contextualizing the Data

1. Cleaning Categoricals
    a. Specific Label Encoding
    b. Non-specific Label Encoding
    c. Ordinal Numerics

	offer_accepted	reward	mailer_type	income_level	overdraft_protection	credit_rating	own_your_home	bank_accounts_open	credit_cards_held	homes_own
0	1	3	2	1	2	1	1	1	2	
1	1	3	2	2	2	2	2	1	2	
2	1	3	1	1	2	2	2	2	2	
3	1	3	2	2	2	1	1	2	1	
4	1	3	2	2	2	2	2	1	2	

2. Cleaning Numericals
    a. Clean distributions
    b. MinMax Scaling

	average_balance	q1_balance	q2_balance	q3_balance	q4_balance
0	0.749189	0.483768	0.256358	0.286424	0.237722
1	0.262826	0.011304	0.030985	0.020403	0.086833
2	0.411248	0.106377	0.102894	0.037928	0.057414
3	0.760723	0.457391	0.514469	0.292702	0.099407
4	0.759172	0.620290	0.396668	0.256866	0.086595

Modelling Function

```
def sample_all (x, y):
```

1. Classification score of Logistic Regression Following Sample Adjustment
2. Logistic Regression Score on X_test, y_test
3. Confusion Matrix of y_test
4. Classification Report of the Logistic Regression on X_test, y_test
5. KNN Model Score
6. KNN Model Confusion Matrix
7. KNN Model Classification Report
8. KNN Model Accuracy Chart
9. ROC Curve - Logistic Regression
10. ROC Curve - KNN

The function runs various sampling techniques, and both Logistic and KNN regressions, and creates an output which allows you to decide which combination is optimal for your data

Results

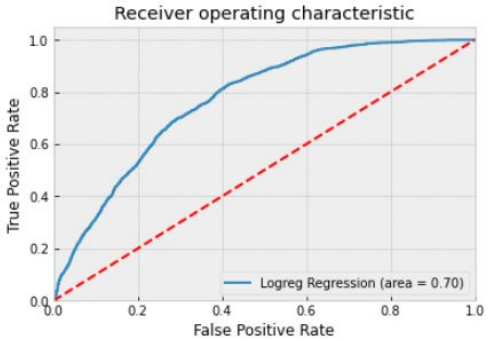
```
#Setting x and y prior to creating a training and test set
y = df['offer_accepted']
x = df.drop(['offer_accepted', 'household_size', 'overdraft_protection'], axis=1)

#converting y from object to int - Python didn't want to accept the object type
y = y.astype('float')
```

Logistic Regression:

- Random Oversampling Classification Score: 0.702
- Random Undersampling Classification Score: 0.696
- SMOTE Classification Score: 0.711

K-Nearest-Neighbor:

- Random Oversampling Classification Score: 0.926
- Random Undersampling Classification Score: 0.637
- SMOTE Classification Score: 0.909



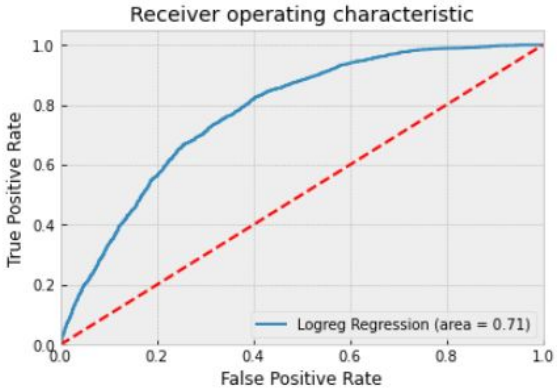
Outcome: SMOTE

- KNN provides unrealistic results
  - We will stick with the Logistic Regression for further analysis
- SMOTE is the optimal sampling technique for a logistic regression

Results - Model 2

Removing Columns with High P-values

```
#Create new X
x = df[['reward', 'income_level','mailer_type','credit_rating']]
print (y.shape, x.shape)
```



Logistic Regression:

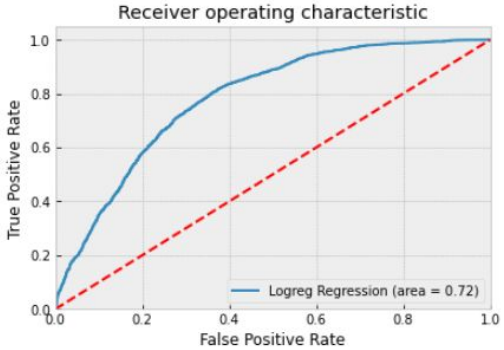
- SMOTE Classification Score: 0.71

Conclusion: Our model is, despite the number of removed columns, unchanged in terms of accuracy.

Results - Model 3

Removing Outliers from Average Balance

```
temp = df

#Removing IQR from average balance

iqr = np.percentile(temp['average_balance'],75) - np.percentile(temp['average_balance'],25)
upper_limit = np.percentile(temp['average_balance'],75) + 1.5*iqr
lower_limit = np.percentile(temp['average_balance'],25) - 1.5*iqr
temp = temp[(temp['average_balance']>lower_limit) & (temp['average_balance']<upper_limit)]

#Setting x and y prior to creating a training and test set
y = temp['offer_accepted']
x = temp.drop(['offer_accepted', 'household_size', 'overdraft_protection'], axis=1)

#converting y from object to int - Python didn't want to accept the object type
y = y.astype('int')
```

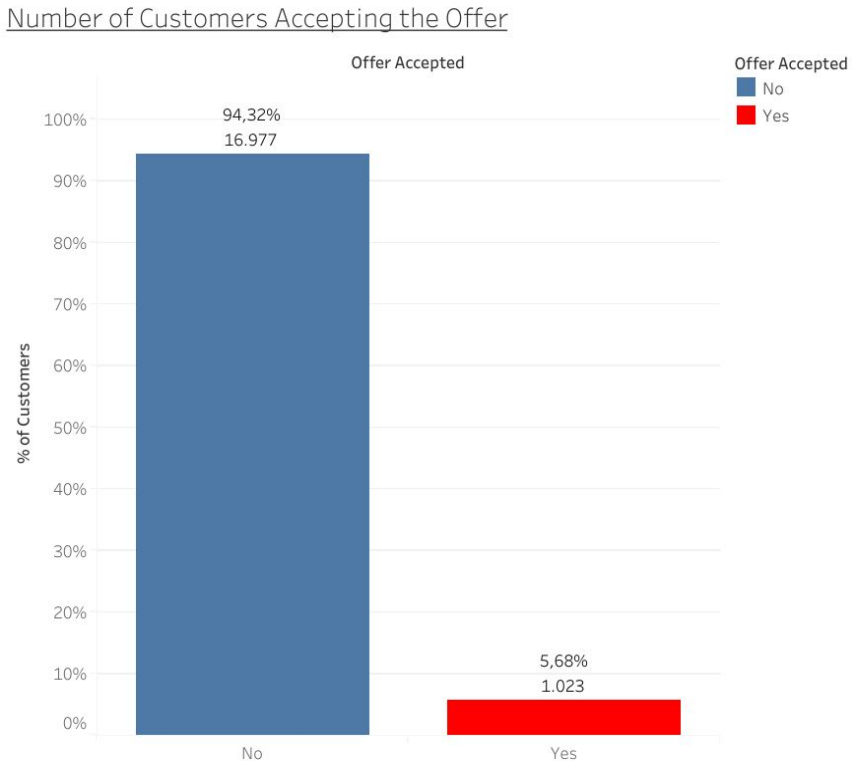


Logistic Regression:

- SMOTE Classification Score: 0.72

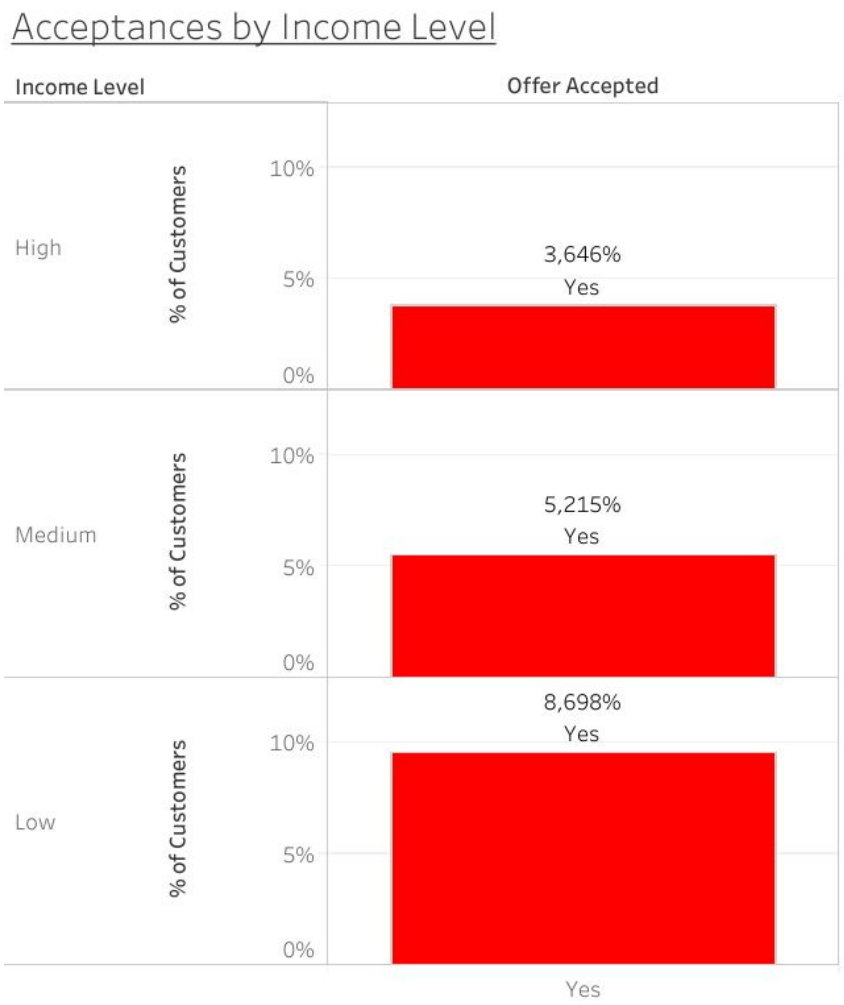
Conclusion: Our model has improved, slightly, with our accuracy score moving to 0.72.

Insights into the Data



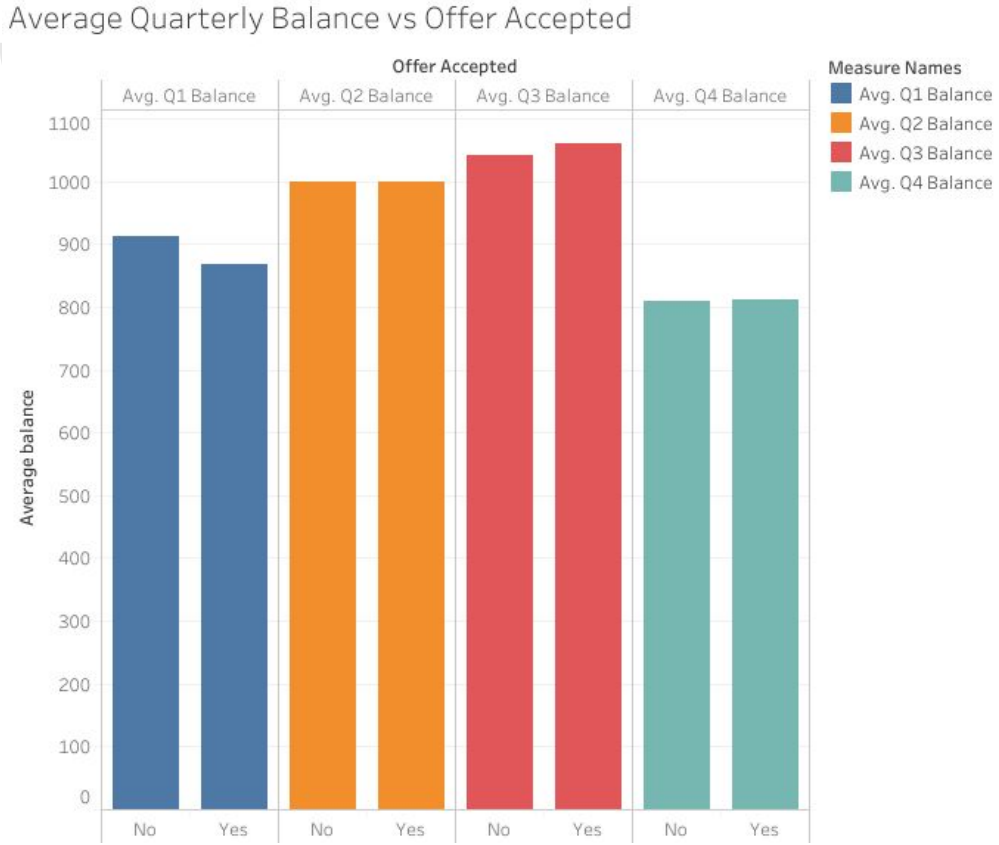
- Significant majority of clients have rejected offers

Insights Cont.



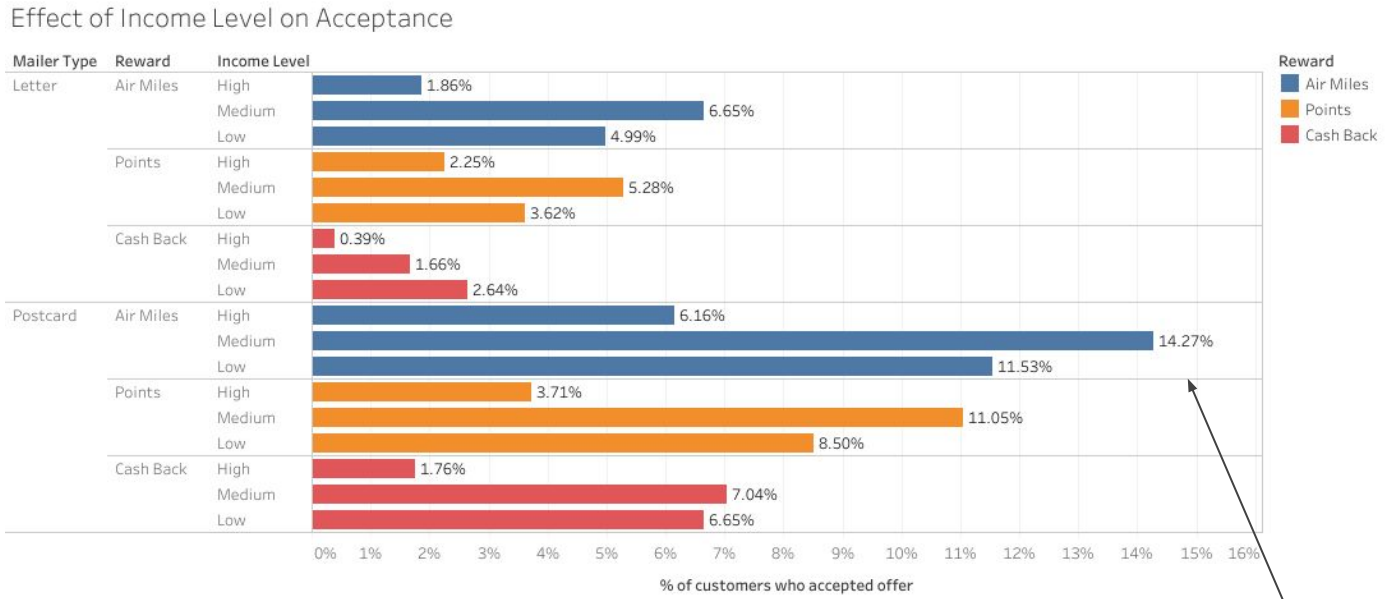
- More than twice as many people in the Low Income bracket accepted the offer compared to people with a high income

Insights Cont.

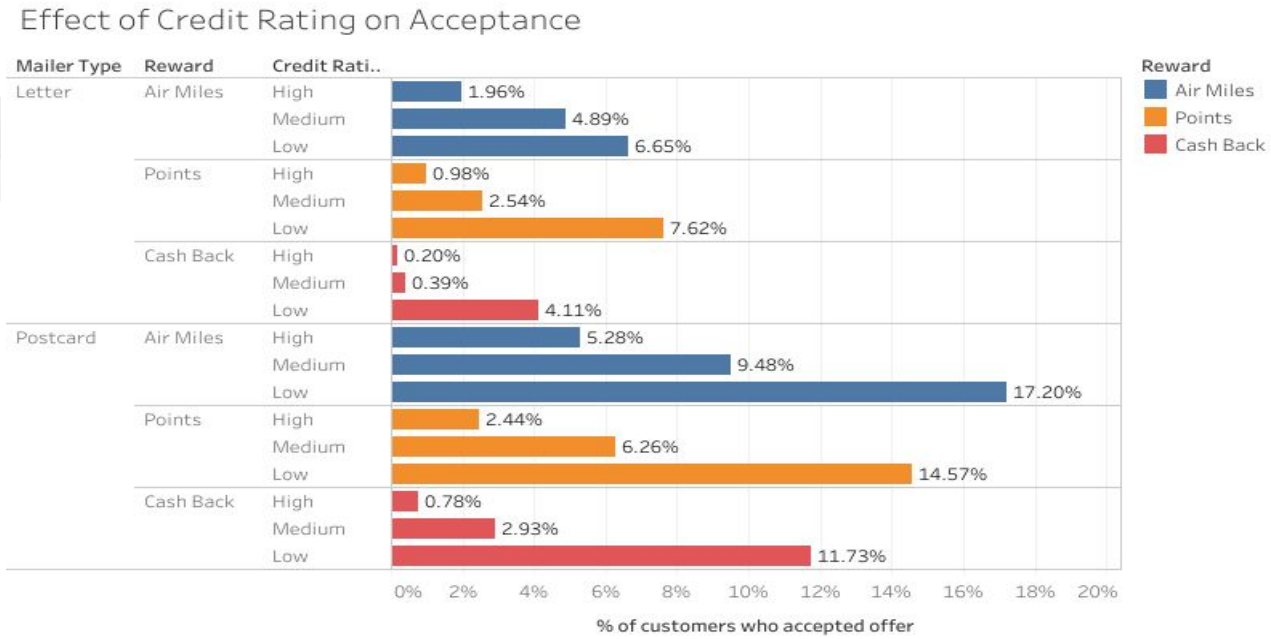


- Customers with a lower average balance in the first quarter we slightly more likely to accept the offer

Insights Cont.



- Mailer type and Reward have the biggest effect on acceptance
- Postcards are more effective than Letters
- E.g. Postcard + Medium Income + Air Miles = Most likely to accept (14.27%)



- People with a low credit rating are the most likely to accept the offer regardless of the reward
- People with high credit ratings are more than twice as likely to accept the offer if the reward is Air Miles
- Postcards are more likely to lead to accepted offers

Recommendations

Expand what is already working -

- Use postcards for credit card offer campaign.
- Profile of customer most likely to accept offer: low income, low credit rating, receiving Air Miles offer by postcard.
- Increase use of Air Miles as reward to attract higher income customers with higher credit ratings.

Improve what is not currently as effective -

- Rewards make a difference - improve terms of cashback reward and test uptake
- Gather more data on e.g. age, gender, location, employment status to improve prediction
- Consider timing of campaign to optimise uptake of customers with lower balance in first quarter

Pitfalls 

- Encoding Decisions
- Multicollinearity
- Optimizing the finished model
- Understanding why high accuracy scores occur, post-sampling adjustments