Project 4: Machine Learning Model

By: Shauntel Phillips, Fabi Estrada, Uzor Francis, and Ally Blitch

March 26, 2024

Table of Contents

Introduction

We chose the “Credit Card Classification” dataset from Kaggle because we were interested in investigating which demographics have the biggest effects on predicting whether someone is high or low risk for a credit card loan. We picked this dataset because the data is clean and it has a high usability rating. Our target is loan risk, with 1 being high risk and 0 being low risk.

Predictions

We think age and income will predict high risk and that marital status and gender will be the least effective in predicting risk.

Data Engineering

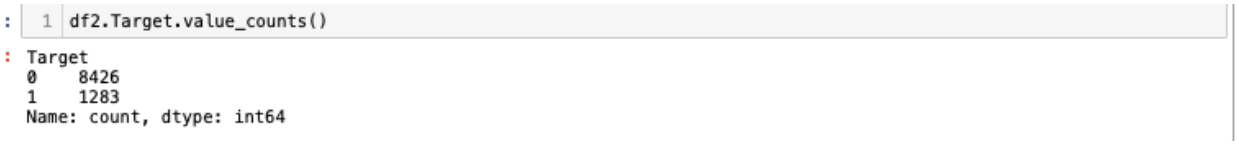
To begin, we converted categorical values into numerical values to make the data easier to manipulate.

ən	Num_family	Account_length	Total_income	Age	Years_employed	Income_type	Education_type	Family_status	Housing_type	Occupation_type	Target
0	2	15	427500.0	32.868574	12.435574	4	1	0	4	12	1
0	2	29	112500.0	58.793815	3.104787	4	4	1	1	17	0
0	1	4	270000.0	52.321403	8.353354	0	4	3	1	15	0
0	1	20	283500.0	61.504343	0.000000	1	1	2	1	12	0
0	2	5	270000.0	46.193967	2.105450	4	1	1	1	0	0

Next, we dropped the “phone”, “email”, and “id” columns because they did not have any effect on the outcomes. We also updated “Gender”, “Own_car”, “Own_property”, “Unemployed”, and “Target” columns to either 0 or 1, or “yes” or “no” values.

Machine Learning/Interpretations

Before running any ML models, the group chose to predict based on the “Target” variable, which dictates whether an individual is of high or low risk. Immediately, noticed the following imbalance of the ML predictor, as this imbalance has the tendency to favor the majority class and perform poorly on the minority class. To correct this, the SMOTE technique was initialized to address the imbalance. The SMOTE split the data into a stratified train/test split utilizing 30% of the data, thus allowing class distribution to be equal for the train test and original data.



```
: 1 df2.Target.value_counts()

: Target
  0    8426
  1    1283
  Name: count, dtype: int64
```

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 2)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train.ravel())
```

```
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: (12638, 43)
After OverSampling, the shape of train_y: (12638,)

After OverSampling, counts of label '1': 6319
After OverSampling, counts of label '0': 6319
```

Logistic Regression

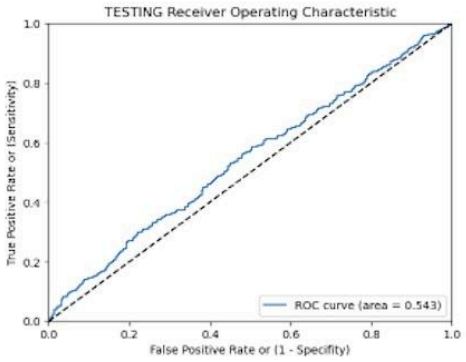
```
TESTING METRICS

    Test Confusion Matrix:
     [[2107    0]
 [ 321    0]]

    Test Report:
                  precision    recall  f1-score   support

           0       0.87      1.00      0.93      2107
           1       0.00      0.00      0.00       321

    accuracy                           0.87      2428
   macro avg       0.43      0.50      0.46      2428
weighted avg       0.75      0.87      0.81      2428
```



The model performs well in predicting class 0, with high precision, recall, and F1-score, thus showing no false positives. However, the model completely fails to predict class 1, suggesting that the model is unable to identify any true positives for class 1. The accuracy is still relatively high at 87%, but this is heavily influenced by the accurate predictions for class 0. This scenario suggests that the model has a significant issue with class imbalance or may require further tuning to better predict instances of class 1.

Logistic Regression-Smote

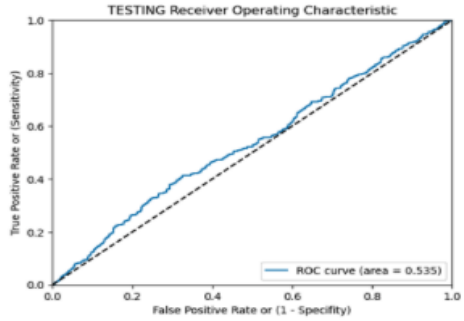
```
TESTING METRICS

   Test Confusion Matrix:
   [[1536  571]
 [ 210  111]]

   Test Report:
              precision    recall  f1-score   support

           0       0.88      0.73      0.80      2107
           1       0.16      0.35      0.22       321

    accuracy                           0.68      2428
   macro avg       0.52      0.54      0.51      2428
weighted avg       0.78      0.68      0.72      2428
```



The model shows decent performance in predicting class 0 but the model's performance in predicting class 1 is poor, with low precision and recall. The F1-score for class 0 is considerably higher than for class 1, indicating better overall performance in predicting class 0. The overall accuracy of the model is 68%, suggesting moderate performance.

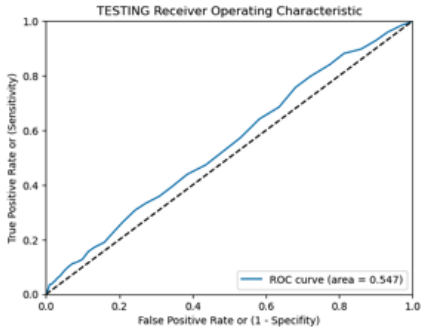
Random Forest

```
   Test Confusion Matrix:
   [[2105    2]
 [ 318    3]]

   Test Report:
              precision    recall  f1-score   support

           0       0.87      1.00      0.93      2107
           1       0.60      0.01      0.02       321

    accuracy                           0.87      2428
   macro avg       0.73      0.50      0.47      2428
weighted avg       0.83      0.87      0.81      2428
```



The model performs well in predicting class 0, as indicated by high precision, recall, and F1-score. However, the model's performance in predicting class 1 is poor, with low precision, recall, and F1-score. It struggles particularly with recall, meaning it fails to identify a significant portion of class 1 instances. The weighted average of precision, recall, and F1-score is also

showing a balanced measure across classes, but this is largely influenced by the dominant class). Overall accuracy is relatively high at 87%, but this might be misleading due to class imbalance

Random Forest-Smote

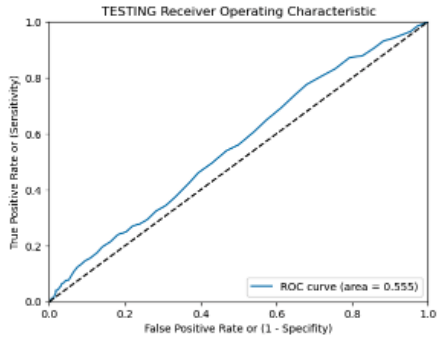
```
TESTING METRICS

    Test Confusion Matrix:
    [[2074   33]
 [ 312    9]]

    Test Report:
                precision    recall  f1-score   support

           0       0.87      0.98      0.92      2107
           1       0.21      0.03      0.05       321

    accuracy                           0.86      2428
   macro avg       0.54      0.51      0.49      2428
weighted avg       0.78      0.86      0.81      2428
```



Performs well in predicting class 0, with high precision, recall, and F1-score.,yet the performance in predicting class 1 remains poor, with low precision, recall, and F1-score. The accuracy is also high but is dominated by the majority class, so it might be misleading about the model's actual performance, especially for imbalanced datasets. Overall, the model's ability to predict class 1 is notably weaker than its ability to predict class 0.

Svc

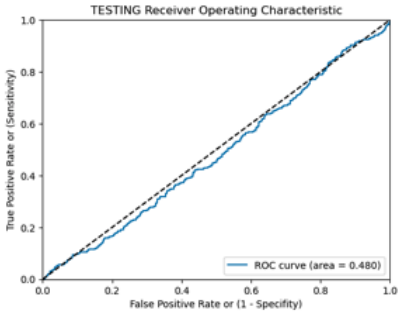
```
TESTING METRICS

    Test Confusion Matrix:
     [[2107    0]
 [ 321    0]]

    Test Report:
                  precision    recall  f1-score   support

           0       0.87      1.00      0.93      2107
           1       0.00      0.00      0.00       321

    accuracy                           0.87      2428
   macro avg       0.43      0.50      0.46      2428
weighted avg       0.75      0.87      0.81      2428
```



The model performs exceptionally well in predicting class 0, with high precision, recall, and F1-score, as indicated by a confusion matrix showing no false positives. The model fails to predict class 1, as indicated by precision, recall, and F1-score of 0 for class 1. This suggests that the model is unable to identify any true positives for class 1. The accuracy is still relatively high at 87%, but this is heavily influenced by the accurate predictions for class 0.

Svc-Smote

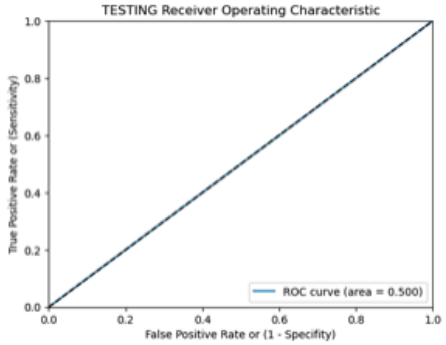
```
TESTING METRICS

    Test Confusion Matrix:
     [[ 802 1305]
 [ 106  215]]

    Test Report:
                  precision    recall  f1-score   support

           0       0.88      0.38      0.53      2107
           1       0.14      0.67      0.23       321

    accuracy                           0.42      2428
   macro avg       0.51      0.53      0.38      2428
weighted avg       0.79      0.42      0.49      2428
```



The model performs relatively well in predicting class 1, as indicated by a higher recall for class 1 than class 0. Yet, the precision for class 1 is quite low, suggesting that many instances predicted as class 1 are actually not. The overall accuracy of the model is low at 42%, indicating that it's not performing well overall. The F1-score for class 1 is also low, indicating a lack of balance between precision and recall for class 1. The model seems to struggle particularly with classifying instances of class 0, as shown by the low recall and F1-score for class 0.

K-Nearest Neighbor

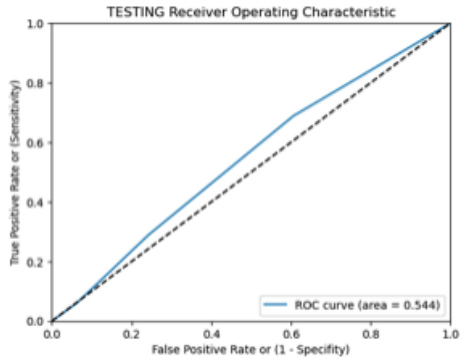
```
TESTING METRICS

    Test Confusion Matrix:
    [[2086   21]
 [ 318    3]]

    Test Report:
                  precision    recall  f1-score   support

           0       0.87      0.99      0.92      2107
           1       0.12      0.01      0.02       321

    accuracy                           0.86      2428
   macro avg       0.50      0.50      0.47      2428
weighted avg       0.77      0.86      0.80      2428
```



The model performs relatively well in predicting class 0, however, the performance in predicting class 1 is very poor, with extremely low precision, recall, and F1-score. The accuracy is high at 86%, but this is heavily influenced by the accurate predictions for class 0, suggesting that the model is good at identifying instances of class 0 but performs poorly in identifying instances of class 1.

K-Nearest Neighbor-Smote

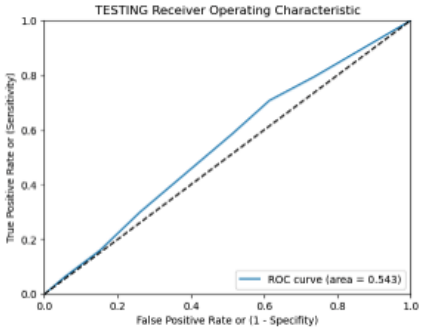
```
TESTING METRICS

    Test Confusion Matrix:
    [[1269  838]
 [ 175  146]]

    Test Report:
                  precision    recall  f1-score   support

           0       0.88      0.60      0.71      2107
           1       0.15      0.45      0.22       321

    accuracy                           0.58      2428
   macro avg       0.51      0.53      0.47      2428
weighted avg       0.78      0.58      0.65      2428
```



The model shows moderate performance in predicting class 0, with relatively high precision but moderate recall. The model's performance in predicting class 1 is poor, thus the overall accuracy of the model is relatively low at 58%, suggesting moderate performance.

K-Fold Validation

We performed a K-Fold Validation on the Logistic Regression model to further evaluate the accuracy of the model by running the data through 5 different splits.

```
# K-Fold Validation
#Implementing cross validation

k = 5
kf = KFold(n_splits=k, random_state=None)
model = LogisticRegression(solver= 'liblinear')

acc_score = []

for train_index , test_index in kf.split(X):
    X_train , X_test = X.iloc[train_index,:],X.iloc[test_index,:]
    y_train , y_test = y[train_index] , y[test_index]

    model.fit(X_train,y_train)
    pred_values = model.predict(X_test)

    acc = accuracy_score(pred_values , y_test)
    acc_score.append(acc)

avg_acc_score = sum(acc_score)/k

print('accuracy of each fold - {}'.format(acc_score))
print('Avg accuracy : {}'.format(avg_acc_score))
```

```
accuracy of each fold - [0.870236869207003, 0.88259526261586, 0.8872296601441813, 0.8789907312049433, 0.8201957753735188]
Avg accuracy : 0.8678496597091012
```

The accuracy of each fold was fairly similar, ranging from 0.820 to 0.887, with the average accuracy being 0.868.

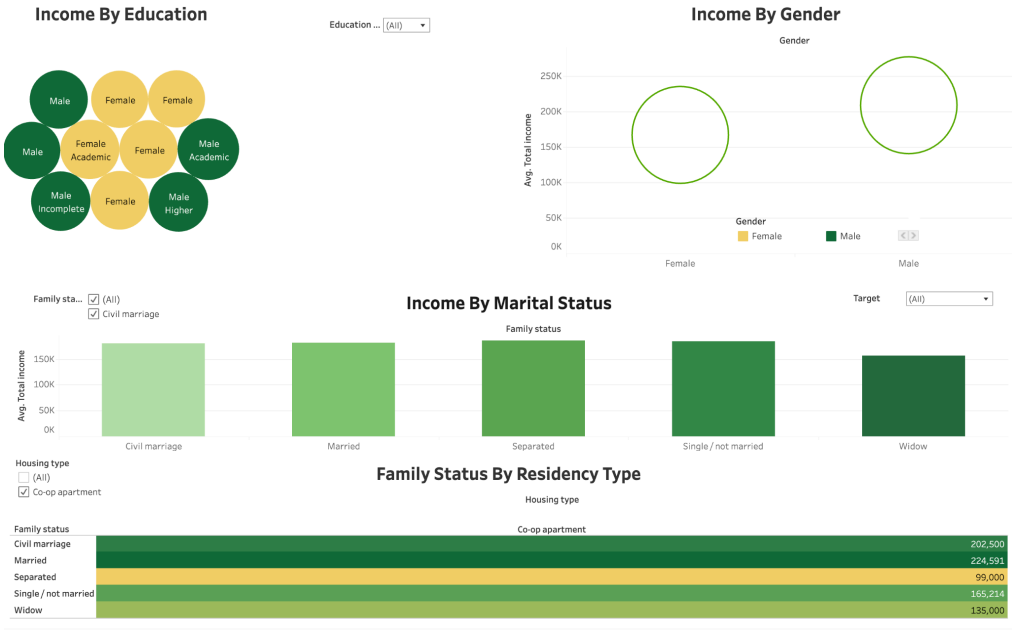
Model Conclusions

In the loan business, the main goal of predicting someone's risk is to avoid providing a high risk individual with a loan. Thus our main priority when choosing a model was to minimize the number of false negatives. That being said, we did also take the number of false positives into account, because the models with the lowest false negatives had extremely high false positives.

Our three best models were SVC-Smote with 106 false negatives and 1305 false positives, Logistic Regression-Smote with 210 false negatives and 571 false positives, and KNN-Smote with 175 false negatives and 838 false positives. We chose the Logistic Regression-Smote model.

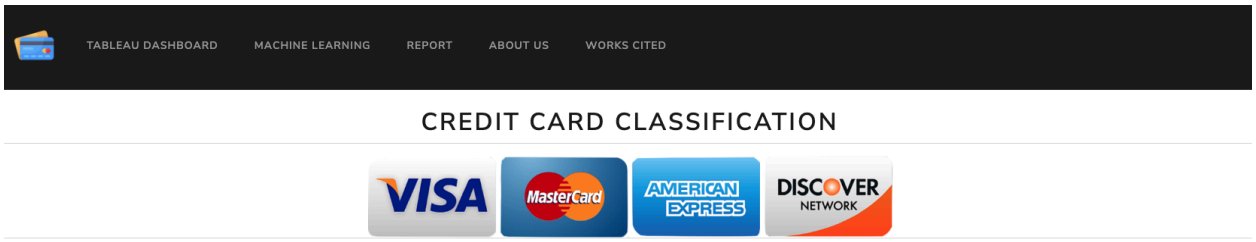
Visualizations

To create our visualizations, we used Tableau. We applied the "Target" column as a filter across all the dashboards.

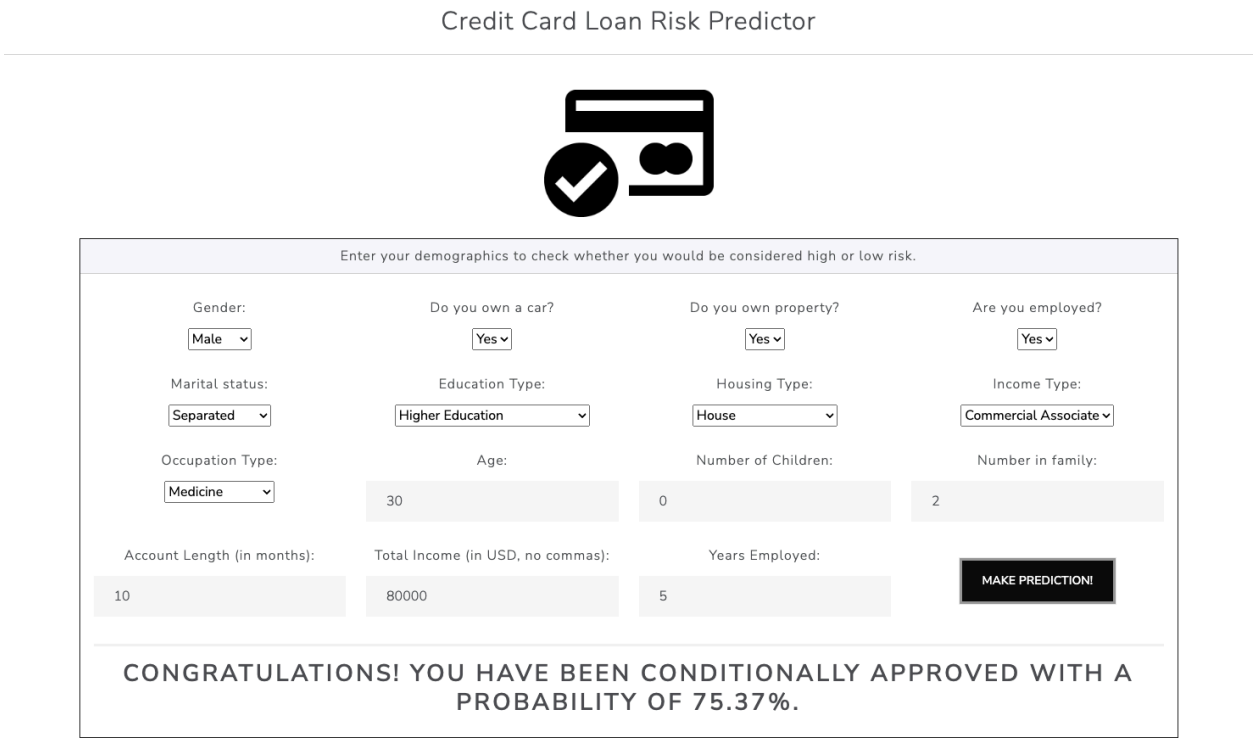


Creating the Web App

We created six HTML files corresponding to six different pages in our web app: a home page, a Tableau/visualization page, an ML form, our report, an about us page, and a works cited page.



The ML form has 15 user inputs: 9 dropdowns and 6 fill-ins for each of the demographics in our model.



We set up a Flask application deployed to PythonAnywhere to provide access to the site.

Findings/Conclusion

Our study highlights the profound impact of class imbalance on the performance of classification models. Through rigorous experimentation and analysis, it's evident that an imbalanced dataset significantly deteriorates the effectiveness of the classification model. In particular, when one class vastly outnumbers the other, creates a skewed learning environment where the model tends to favor the majority class, often at the expense of misclassifying minority samples. Consequently, the classification model's ability to accurately discern between classes diminishes, resulting in low precision, recall, and F1 scores. In conclusion, addressing class imbalance is imperative for developing robust classification models capable of delivering reliable predictions across all classes, thereby enhancing their utility and efficacy in various domains.

Resources

https://www.kaggle.com/datasets/samuelcortinhas/credit-card-classification-clean-data/data

https://www.kaggle.com/datasets/itssuru/loan-data

https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html

https://datascience.stackexchange.com/questions/47852/visualizing-decision-tree-with-feature-names

https://www.askpython.com/python/examples/k-fold-cross-validation

https://www.turintech.ai/what-is-imbalanced-data-and-how-to-handle-it/