Credit Card Fraud Detection

Using deep learning

Why is it interesting?

- In 2018, more than $24.2 billion was lost globally due to credit card fraud
- By 2027, huge losses from card fraud transactions are expected to reach $40 billion.

- The role of AI and deep learning in mitigation of fraud

Key Challenges

-RNNs and binary classification

-Highly Imbalanced dataset

- Accuracy vs Precision

-Struggle with long sequences, i.e Exploding gradients

-Data privacy concerns

-Update regularly

Design and Methodology

-Overall approach was experimental

-Why RNNs?

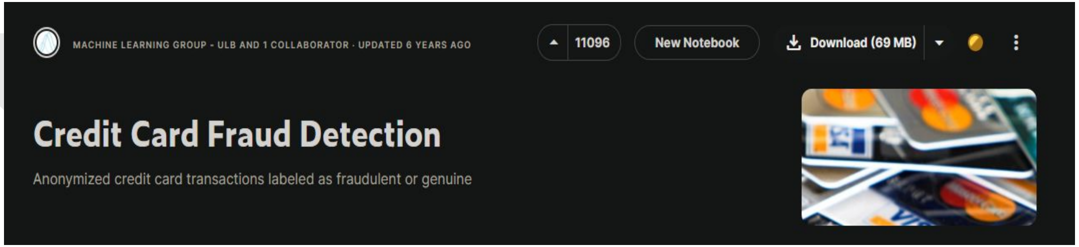
-Comparing performance of different Models

-Not using validation set

Design and methodology contd.

- Using Keras library
- With and without SMOTE
- Results:
    - Confusion matrix
    - Precision

Dataset and Training schedule



- Dataset: Kaggle- Credit Card Fraud Detection
- Transactions totalling 284,807 over span of two days during sept 2013
- Target variable is the 'class' column
- Sequential
- Highly imbalanced(only 492 fraudulent) - 0.172%

  - Artificial balancing and normalizing required,

  - SMOTE(Synthetic Minority Oversampling Technique)

- 75/25 train:test

- 284807 rows × 31 columns
- Time, Amount and Class
- Privacy
- Normal distribution for most of the classes

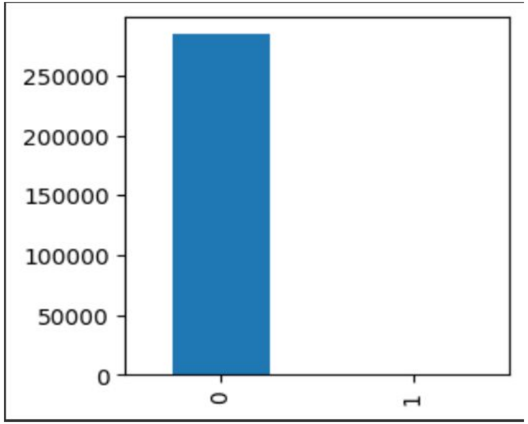
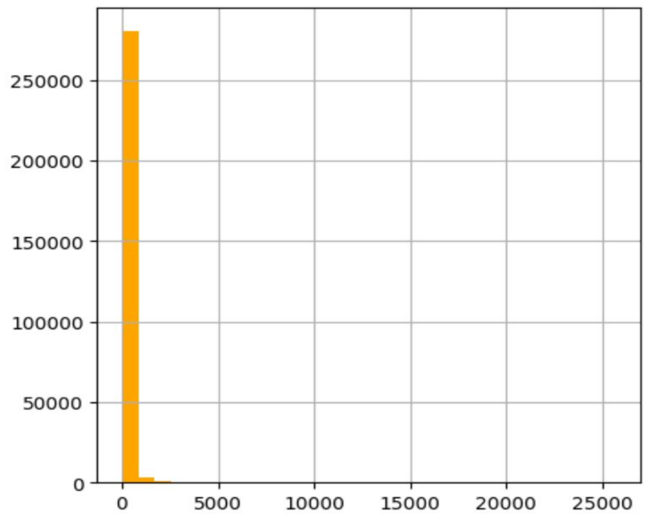


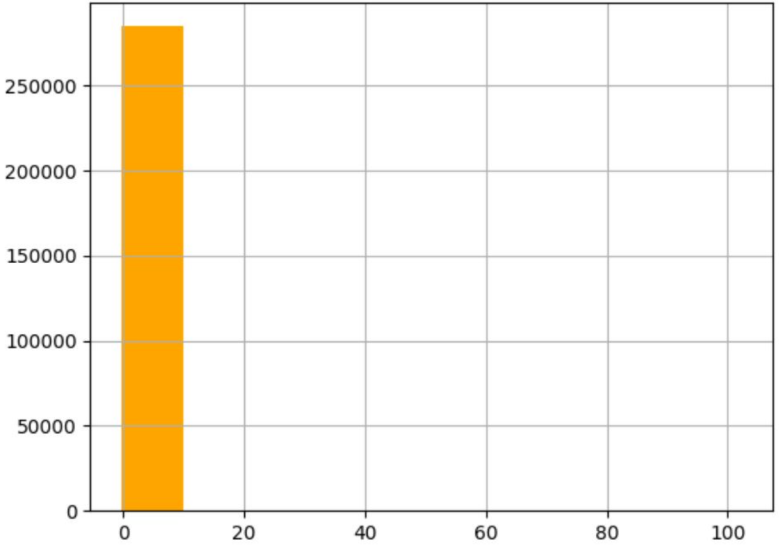
Figure of our target class variable

Before and after scaling our data

```
count    284807.000000
mean         88.349619
std         250.120109
min           0.000000
25%           5.600000
50%          22.000000
75%          77.165000
max       25691.160000
Name: Amount, dtype: float64
```




```
count    2.848070e+05
mean    -8.981359e-18
std      1.000002e+00
min     -3.532294e-01
25%     -3.308401e-01
50%     -2.652715e-01
75%     -4.471707e-02
max      1.023622e+02
Name: Amount, dtype: float64
```



Results

Logistic Regression - baseline model

```
[[71065    11]
 [   56    70]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71076
           1       0.86      0.56      0.68       126

    accuracy                           1.00     71202
   macro avg       0.93      0.78      0.84     71202
weighted avg       1.00      1.00      1.00     71202
```

Logistic regression w/smote

```
[[69349  1727]
 [   13   113]]
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     71076
           1       0.06      0.90      0.11       126

    accuracy                           0.98     71202
   macro avg       0.53      0.94      0.55     71202
weighted avg       1.00      0.98      0.99     71202
```

Used Standardscaler from sklearn

MLP

```
2226/2226 [==============================] - 3s 1ms/step
[[71044    32]
 [   22   104]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71076
           1       0.76      0.83      0.79       126

    accuracy                           1.00     71202
   macro avg       0.88      0.91      0.90     71202
weighted avg       1.00      1.00      1.00     71202
```

Using SMOTE

```
2226/2226 [==============================] - 3s 1ms/step
[[70914   162]
 [   16   110]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71076
           1       0.40      0.87      0.55       126

    accuracy                           1.00     71202
   macro avg       0.70      0.94      0.78     71202
weighted avg       1.00      1.00      1.00     71202
```

-1 dense hidden layer(64 neurons w/relu)

-Used BatchNormalization on the output of that layer

-output layer is a single neuron sigmoid dense layer

-lr = 0.001

CNN w/out SMOTE

```
[[69349  1727]
 [   13   113]]
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     71076
           1       0.06      0.90      0.11       126

    accuracy                           0.98     71202
   macro avg       0.53      0.94      0.55     71202
weighted avg       1.00      0.98      0.99     71202
```

- 2 1d convolutional layers
- 32/64 filters and kernel size 2
- relu activation
- Batch Normalization
- sigmoid layer for output
- SGD
- binary-cross entropy

Lr =0.001

CNN W/SMOTE

```
2226/2226 [==============================] - 3s 1ms/step
[[70935   141]
 [   16   110]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71076
           1       0.44      0.87      0.58       126

    accuracy                           1.00     71202
   macro avg       0.72      0.94      0.79     71202
weighted avg       1.00      1.00      1.00     71202
```

Results contd.

RNN with SMOTE

```
2226/2226 [==============================] - 13s 6ms/step
[[65306  5770]
 [   58    68]]
              precision    recall  f1-score   support

           0       1.00      0.92      0.96     71076
           1       0.01      0.54      0.02       126

    accuracy                           0.92     71202
   macro avg       0.51      0.73      0.49     71202
weighted avg       1.00      0.92      0.96     71202
```

RNN w/o SMOTE

```
2226/2226 [==============================] - 13s 6ms/step
[[71061    15]
 [   29    97]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71076
           1       0.87      0.77      0.82       126

    accuracy                           1.00     71202
   macro avg       0.93      0.88      0.91     71202
weighted avg       1.00      1.00      1.00     71202
```

Double-click (or enter) to edit

-1 LSTM hidden layer with 64 neurons
-Using Dropout of 0.5
-Batch normalization
-Sigmoid output layer
-lr = 0.001
Epochs = 5

Accuracy table

	Logistic Regression	MLP	CNN	RNN
Without SMOTE	1.00	1.00	0.98	1.00
With SMOTE	0.98	1.00	1.00	1.00

Precision Table

	Logistic Regression	MLP	CNN	RNN
Without SMOTE	1/0.86	1.00/0.76	1/0.06	1/0.87
With SMOTE	1/0.06	1.00/0.40	1/0.44	1/0.01

References

https://legaljobs.io/blog/credit-card-fraud-statistics

https://www.johnmarshallbank.com/resources/security-center/fraud-facts-and-statistics/

https://factnight.com/credit-card-fun-facts/

https://mint.intuit.com/blog/planning/credit-card-fraud-statistics/