Kaggle 신용카드 사기 검출 (Google Drive Mount)

https://www.kaggle.com/mlg-ulb/creditcardfraud

Credit Card Fraud Detection

- creditcard.csv (284,807 * 31)
- Class : 0 (정상), 1 (사기)
- 사기 검출(Fraud Detection), 이상 탐지(Anomaly Detection)

```
import warnings
warnings.filterwarnings('ignore')
```

I. Google Drive Mount

- 'creditCardFraud.zip' 파일을 구글드라이브에 업로드 후 진행

```
from google.colab import drive

drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

- 마운트 결과 확인

```
!ls -l '/content/drive/My Drive/Colab Notebooks/datasets/creditCardFraud.zip'
```

```
-rw------- 1 root root 69155672 Mar  4 04:46 '/content/drive/My Drive/Colab Notebooks/datasets/creditCardFraud.zip'
```

II. Data Preprocessing

1) Unzip 'creditCardFraud.zip'

- Colab 파일시스템에 'creditcard.csv' 파일 생성

```
!unzip /content/drive/My₩ Drive/Colab₩ Notebooks/datasets/creditCardFraud.zip
```

```
Archive:  /content/drive/My Drive/Colab Notebooks/datasets/creditCardFraud.zip
  inflating: creditcard.csv
```

- creditcard.csv 파일 확인

```
!ls -l
```

```
total 147304
-rw-r--r-- 1 root root 150828752 Sep 20  2019 creditcard.csv
drwx------ 5 root root      4096 Jul 20 06:04 drive
drwxr-xr-x 1 root root      4096 Jul 16 13:20 sample_data
```

2) 데이터 읽어오기

- pandas DataFrame

```
import pandas as pd

DF = pd.read_csv('creditcard.csv')

DF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

```
DF.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196

- 0 (정상) Class와 1 (사기) Class 개수

```
DF.Class.value_counts()
```

```
0    284315
1       492
Name: Class, dtype: int64
```

- 0 (정상) Class와 1 (사기) Class 비율

```
(DF.Class.value_counts() / DF.shape[0]) * 100
```

```
0    99.827251
1     0.172749
Name: Class, dtype: float64
```

3) Time 열(Column) 삭제

```
DF.drop('Time', axis = 1, inplace = True)

DF.head(1)
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.5516	-0.617801	-0.99139	-

4) train_test_split( )

- X (Input), y (Output) 지정

```
X = DF.iloc[:,:-1]
y = DF.iloc[:, -1]

X.shape, y.shape
```

```
((284807, 29), (284807,))
```

- With 'Stratify'

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.3,
                                                    random_state = 2045,
                                                    stratify = y)

X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((199364, 29), (199364,), (85443, 29), (85443,))
```

- Train_Data와 Test_Data의 1 (부정) 비율이 균형

```
print('Train_Data :','₩n', (y_train.value_counts() / y_train.shape[0]) * 100)
print('Test_Data :','₩n', (y_test.value_counts() / y_test.shape[0]) * 100)
```

```
Train_Data :
 0    99.827451
1     0.172549
Name: Class, dtype: float64
Test_Data :
 0    99.826785
1     0.173215
Name: Class, dtype: float64
```

I. Keras Modeling

1) Import Tensorflow

- Tensorflow Version 확인

```
import tensorflow

tensorflow.__version__
```

```
'2.5.0'
```

2) Model Define

- 모델 신경망 구조 정의

```
from tensorflow.keras import models
from tensorflow.keras import layers

ccfd = models.Sequential()
```

```
ccfd.add(layers.Dense(128, activation = 'relu', input_shape = (29,)))
ccfd.add(layers.Dense(64, activation = 'relu'))
ccfd.add(layers.Dense(32, activation = 'relu'))
ccfd.add(layers.Dense(1, activation = 'sigmoid'))
```

- 모델 구조 확인

```
ccfd.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 128)               3840
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 14,209
Trainable params: 14,209
Non-trainable params: 0
_________________________________________________________________
```

3) Model Compile

- 모델 학습방법 설정

```
ccfd.compile(loss = 'binary_crossentropy',
             optimizer = 'adam',
             metrics = ['Recall'])
```

4) Model Fit

- 모델 학습 수행

```
%%time

Hist_ccfd = ccfd.fit(X_train, y_train,
                     epochs = 50,
                     batch_size = 1024,
                     validation_data = (X_test, y_test))
```

```
Epoch 1/50
195/195 [==============================] - 3s 9ms/step - loss: 0.0464 - recall: 0.4331 - val_loss: 0.0099 - val_recall: 0.8311
Epoch 2/50
195/195 [==============================] - 1s 8ms/step - loss: 0.0087 - recall: 0.7326 - val_loss: 0.0036 - val_recall: 0.6959
Epoch 3/50
195/195 [==============================] - 1s 8ms/step - loss: 0.0084 - recall: 0.7093 - val_loss: 0.0114 - val_recall: 0.8378
Epoch 4/50
195/195 [==============================] - 1s 8ms/step - loss: 0.0246 - recall: 0.7035 - val_loss: 0.0499 - val_recall: 0.5405
Epoch 5/50
195/195 [==============================] - 1s 8ms/step - loss: 0.0580 - recall: 0.6366 - val_loss: 0.0073 - val_recall: 0.8243
Epoch 6/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0059 - recall: 0.7471 - val_loss: 0.0061 - val_recall: 0.8514
Epoch 7/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0037 - recall: 0.7820 - val_loss: 0.0032 - val_recall: 0.8378
Epoch 8/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0035 - recall: 0.7965 - val_loss: 0.0048 - val_recall: 0.8243
Epoch 9/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0134 - recall: 0.7267 - val_loss: 0.0035 - val_recall: 0.8514
Epoch 10/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0065 - recall: 0.7529 - val_loss: 0.0031 - val_recall: 0.8176
Epoch 11/50
195/195 [==============================] - 1s 8ms/step - loss: 0.0046 - recall: 0.7733 - val_loss: 0.0033 - val_recall: 0.8514
Epoch 12/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0046 - recall: 0.7965 - val_loss: 0.0036 - val_recall: 0.8446
Epoch 13/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0032 - recall: 0.8110 - val_loss: 0.0031 - val_recall: 0.7770
```

```
Epoch 14/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0101 - recall: 0.7762 - val_loss: 0.0098 - val_recall: 0.8311
Epoch 15/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0053 - recall: 0.8023 - val_loss: 0.0037 - val_recall: 0.8446
Epoch 16/50
195/195 [==============================] - 2s 8ms/step - loss: 0.0033 - recall: 0.8081 - val_loss: 0.0034 - val_recall: 0.8378
Epoch 17/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0148 - recall: 0.7238 - val_loss: 0.0032 - val_recall: 0.8041
Epoch 18/50
195/195 [==============================] - 1s 8ms/step - loss: 0.0030 - recall: 0.8110 - val_loss: 0.0031 - val_recall: 0.8243
Epoch 19/50
195/195 [==============================] - 1s 8ms/step - loss: 0.0027 - recall: 0.8140 - val_loss: 0.0032 - val_recall: 0.7973
Epoch 20/50
195/195 [==============================] - 1s 8ms/step - loss: 0.0029 - recall: 0.8140 - val_loss: 0.0031 - val_recall: 0.8649
Epoch 21/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0031 - recall: 0.8081 - val_loss: 0.0067 - val_recall: 0.7027
Epoch 22/50
195/195 [==============================] - 1s 8ms/step - loss: 0.0050 - recall: 0.7994 - val_loss: 0.0033 - val_recall: 0.8514
Epoch 23/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0024 - recall: 0.8169 - val_loss: 0.0032 - val_recall: 0.8649
Epoch 24/50
195/195 [==============================] - 2s 8ms/step - loss: 0.0024 - recall: 0.8110 - val_loss: 0.0031 - val_recall: 0.8716
Epoch 25/50
195/195 [==============================] - 2s 8ms/step - loss: 0.0024 - recall: 0.8227 - val_loss: 0.0035 - val_recall: 0.7635
Epoch 26/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0023 - recall: 0.8256 - val_loss: 0.0033 - val_recall: 0.8514
Epoch 27/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0022 - recall: 0.8372 - val_loss: 0.0033 - val_recall: 0.8108
Epoch 28/50
195/195 [==============================] - 1s 8ms/step - loss: 0.0023 - recall: 0.8459 - val_loss: 0.0058 - val_recall: 0.8311
Epoch 29/50
195/195 [==============================] - 1s 7ms/step - loss: 0.0063 - recall: 0.8110 - val_loss: 0.0045 - val_recall: 0.8243
Epoch 30/50
```

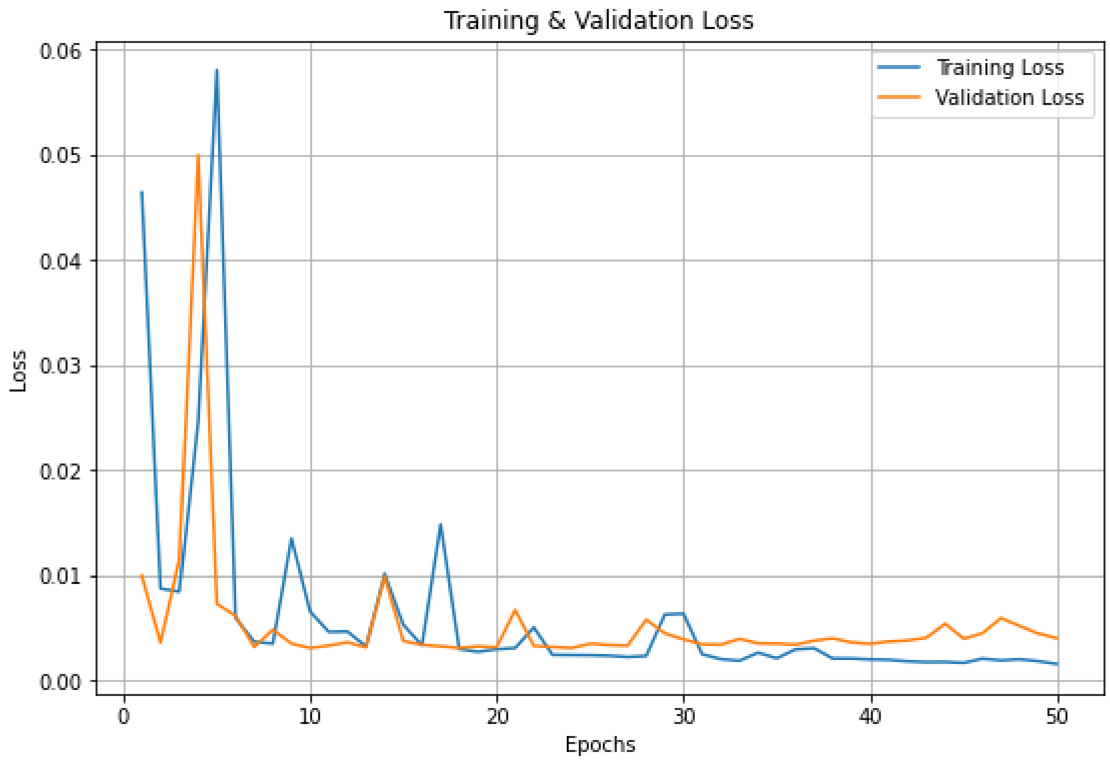
5) 학습 결과 시각화

- Loss Visualization

```
import matplotlib.pyplot as plt

epochs = range(1, len(Hist_ccfd.history['loss']) + 1)

plt.figure(figsize = (9, 6))
plt.plot(epochs, Hist_ccfd.history['loss'])
plt.plot(epochs, Hist_ccfd.history['val_loss'])
plt.title('Training & Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['Training Loss', 'Validation Loss'])
plt.grid()
plt.show()
```

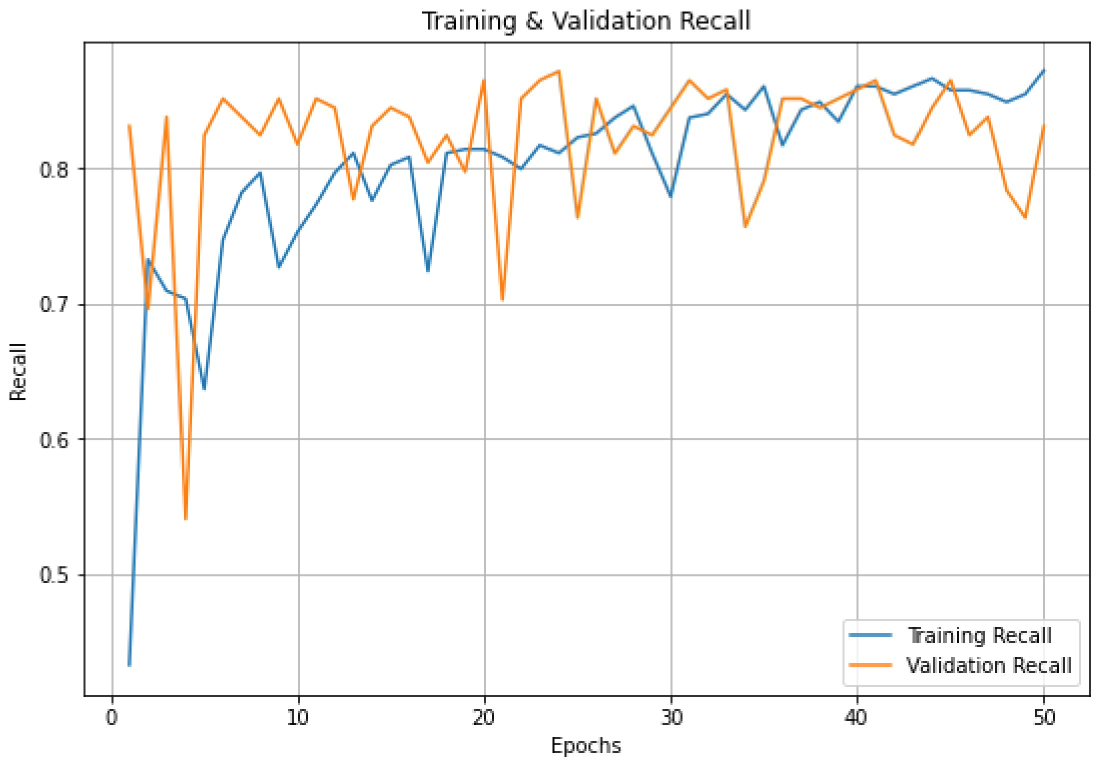


- Recall Visualization

```
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt

epochs = range(1, len(Hist_ccfd.history['recall']) + 1)

plt.figure(figsize = (9, 6))
plt.plot(epochs, Hist_ccfd.history['recall'])
plt.plot(epochs, Hist_ccfd.history['val_recall'])
plt.title('Training & Validation Recall')
plt.xlabel('Epochs')
plt.ylabel('Recall')
plt.legend(['Training Recall', 'Validation Recall'])
plt.grid()
plt.show()
```



6) Model Evaluate

- Loss & Accuracy

```
loss, recall = ccfd.evaluate(X_test, y_test)

print('Loss = {:.5f}'.format(loss))
print('Recall = {:.5f}'.format(recall))
```

```
2671/2671 [==============================] - 3s 1ms/step - loss: 0.0040 - recall: 0.8311
Loss = 0.00401
Recall = 0.83108
```

7) Model Predict

```
y_hat = ccfd.predict_classes(X_test)
```

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test, y_hat)
```

```
array([[85280,    15],
       [   25,   123]])
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score

print(accuracy_score(y_test, y_hat))
print(precision_score(y_test, y_hat, pos_label = 1))
print(recall_score(y_test, y_hat, pos_label = 1))
```

```
0.9995318516437859
0.8913043478260869
```

```
    0.831081081081081
```

#

#

#

The End

#

#

#