import pandas as pd

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler import tensorflow as tf

from tensorflow import keras

from keras.models import Sequential from keras.layers import Dense

from tensorflow.keras import layers, models import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix, classification\_report

df = pd.read\_csv(“/content/creditcard.csv")

df-

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Time | V1 | V2 | V3 | V4 | VS | U6 | V7 | U8 |  |
| 0 | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | 0.098698 |
| 1 | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | 0.085102 | -0.2' |
| 2 | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.247676 |  |
| 3 | 1.0 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.377436 | -1.3f |
| 4 | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.270533 |  |
| 284802 | 172786.0 | -11.881118 | 10.071785 | -9.834783 | -2.066656 | -5.364473 | -2.606837 | -4.918215 | 7.305334 |  |
| 284803 | 172787.0 | -0.732789 | -0.055080 | 2.035030 | -0.738589 | 0.868229 | 1.058415 | 0.024330 | 0.294869 | 0.51 |
| 284804 | 172788.0 | 1.919565 | -0.301254 | -3.249640 | -0.557828 | 2.630515 | 3.031260 | -0.296827 | 0.708417 |  |
| 284805 | 172788.0 | -0.240440 | 0.530483 | 0.702510 | 0.689799 | -0.377961 | 0.623708 | -0.686180 | 0.679145 | 0.3! |
| 284806 | 172792.0 | -0.533413 | -0.189733 | 0.703337 | -0.506271 | -0.012546 | -0.649617 | 1.577006 | -0.414650 |  |

284807 rows • 31 columns



df.drop(['Class'],axis=1)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Time | V1 | V2 | V3 | V4 | V5 | U6 |  |
| 0 | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.23959f |
| 1 | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.07880t |
| 2 | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.79146‘ |
| 3 | 1 0 | -0 966272 | -0 185226 | 1 792993 | -0 863291 | -0 010309 | 1.247203 | 0 23760! |
| 4 | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.59294‘ |

# Drop rows with NaN values in the target column

df df. Aopna(subse’t-[ C1ass }}

284802 172786.0 -11.881118 10.071785 -9.834783

# Standardize the data

scaler284g93ndl7868ZZén(- .732789 -0.055080 2.035030

-2.066656 -5.364473 -2.606837 -4.91821'

-0.738589 0.868229 1.058415 0.02433(

X = scaler.fit transform “Class” axi

y = 2788.0 -0.3012'54



-3 249640

-0.557828 2.630515 3.031260 -0.29682i

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 284805 172788.0 | -0.240440 | 0.530483 | 0.702510 | 0.689799 | -0.377961 | 0.623708 | -0.68618( |
| 284806 172792.0 | -0.533413 | -0.189733 | 0.703337 | -0.506271 | -0.012546 | -0.649617 | 1.57700f |

xtrain,xtest,ytrain,ytest = train\_test\_split(X,y,test\_size=0.2, random\_state=42)

284807 rows • 30 columns



input\_dim=xtrain.shape[1]

xtrain.shape

(227845, 30)

xtest.shape

(56962, 30)

encoder = models.Sequential([ layers.Input(shape=(input\_dim,)), layers.Dense(32, activation='relu'), layers.Dense(16, activation='relu')

decoder = models.Sequential([ layers.Input(shape=(16,)), layers.Dense(30,activation='relu'), layers.Dense(input\_dim,activation='linear')

autoencoder = models.Sequential([

encoder, decoder

autoencoder.compile(optimizer='adam', loss='mean\_squared\_error')

autoencoder.fit(xtrain, xtrain, epochs=10, batch\_size=32, shuffle=True, validation\_data=(xtest, xtest))

Epoch 1/10

7121/7121 [==============================] - 109s 3ms/step - loss: 0.3469 - val\_loss: 0.2050

Epoch 2/10

7121/7121 [==============================]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| - 20s | 3ms/step | - loss: | 0.1784 | - val\_loss: | 0.1475 |
| - 23s | 3ms/step | - loss: | 0.1445 | - val\_loss: | 0.1417 |
| - 20s | 3ms/step | - loss: | 0.1352 | - val\_loss: | 0.1368 |
| - 23s | 3ms/step | - loss: | 0.1284 | - val\_loss: | 0.1214 |
| - 19s | 3ms/step | - loss: | 0.1209 | - val\_loss: | 0.1056 |
| - 19s | 3ms/step | - loss: | 0.1056 | - val\_loss: | 0.0938 |
| - 22s | 3ms/step | - loss: | 0.0975 | - val\_loss: | 0.0902 |
| - 19s | 3ms/step | - loss: | 0.0924 | - val\_loss: | 0.0913 |
| - 25s | 3ms/step | - loss: | 0.0883 | - val\_loss: | 0.0789 |

Epoch 3/10

7121/7121 [==============================]

Epoch 4/10

7121/7121 [==============================]

Epoch 5/10

7121/7121 [==============================]

Epoch 6/10

7121/7121 [==============================]

Epoch 7/10

7121/7121 [==============================]

Epoch 8/10

7121/7121 [==============================]

Epoch 9/10

7121/7121 [==============================]

Epoch 10/10

7121/7121 [==============================]

<keras.src.callbacks.History at 0x7b5cf2af6cb0>

ypred = autoencoder.predict(xtest)

mse = np.mean(np.power(xtest-ypred,2),axis=1)

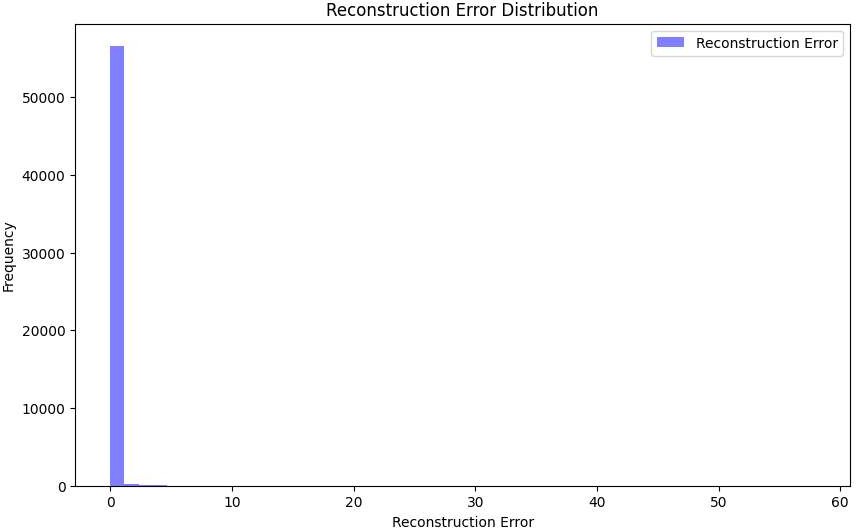
1781/1781 [==============================] - 3s 2ms/step

plt.figure(figsize=(10, 6))

plt.hist(mse, bins=50, alpha=0.5, color='b', label='Reconstruction Error') plt.xlabel(”Reconstruction Error“)

plt.ylabel(”Frequency”) plt.legend()

plt.title(“Reconstruction Error Distribution“) plt.show()



thresholds = np.arange(0.1,1.0,0.1)

for threshold in thresholds: anomalies=mse›threshold

num\_anomalies = sum(anomalies)

print(f“Threshold: {threshold:.1f}, Number of anomalies :{num\_anomalies}”)

Threshold: 0.9, Number of anomalies :511

print(”Confusion Matrix:”)

print(confusion\_matrix(ytest, anomalies))

print(”\nClassification Report:“) print(classification\_report(ytest, anomalies))

Confusion Matrix: [[56377

[ 74

487]

24]]

Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | fl-score | support |
| 0 | 1.00 | 0.99 | 1.00 | 56864 |
| 1 | 0.05 | 0.24 | 0.08 | 98 |
| accuracy |  |  | 0.99 | 56962 |
| macro avg | 0.52 | 0.62 | 0.54 | 56962 |
| weighted avg | 1.00 | 0.99 | 0.99 | 56962 |

import seaborn as sns

plt.figure(figsize = (6, 4.75))

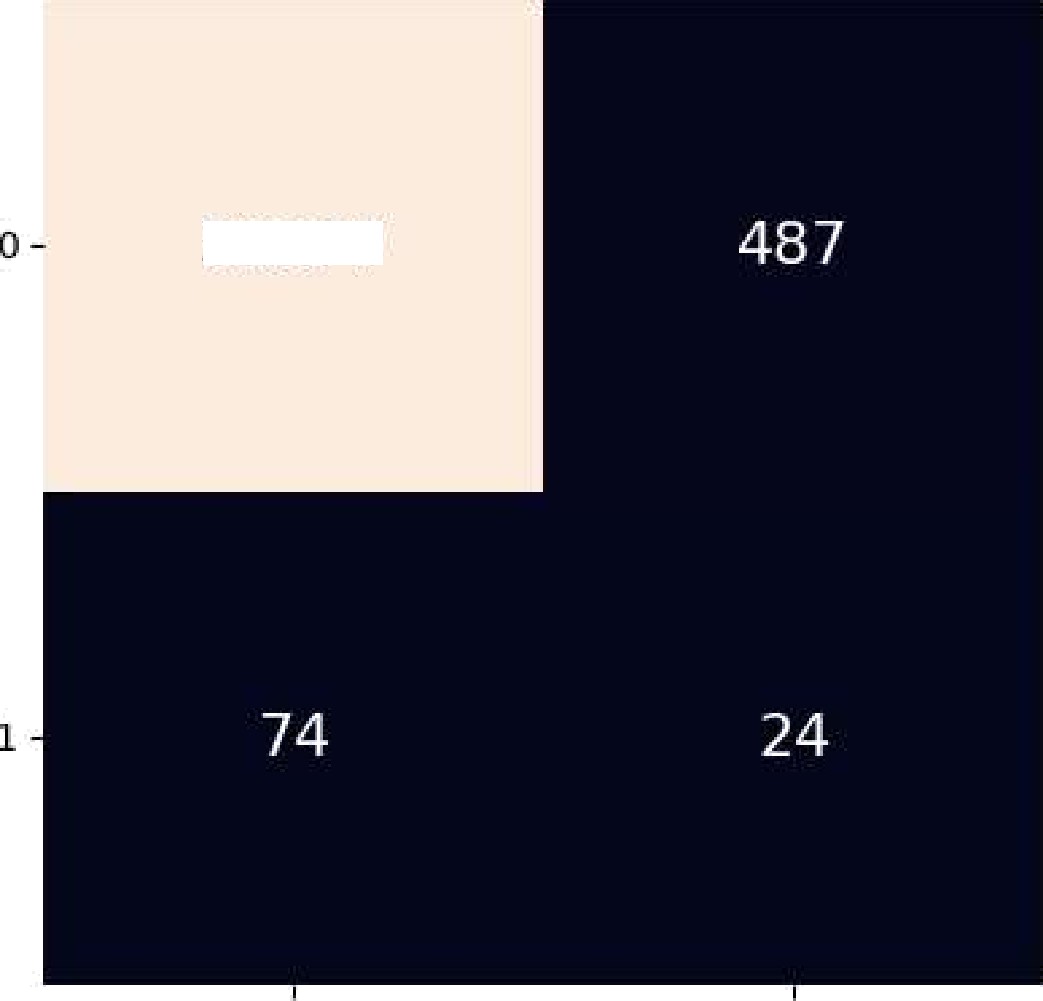
sns.heatmap(confusion\_matrix(ytest, anomalies), annot = True, annot\_kws = {”size“: 16}, fmt = 'd') plt.xticks([0.5, 1.5], rotation = 'horizontal')

plt.yticks([0.5, 1.5], rotation = 'horizontal') plt.xlabel(”Predicted label”, fontsize = 14) plt.ylabel(”True label”, fontsize = 14) plt.title(”Confusion Matrix", fontsize = 14) plt.grid(False)

plt.show()

Confusion Matrix

# -50000



56377

L

4D000



—

True label

3D000

# 20000

10000

Predicted label