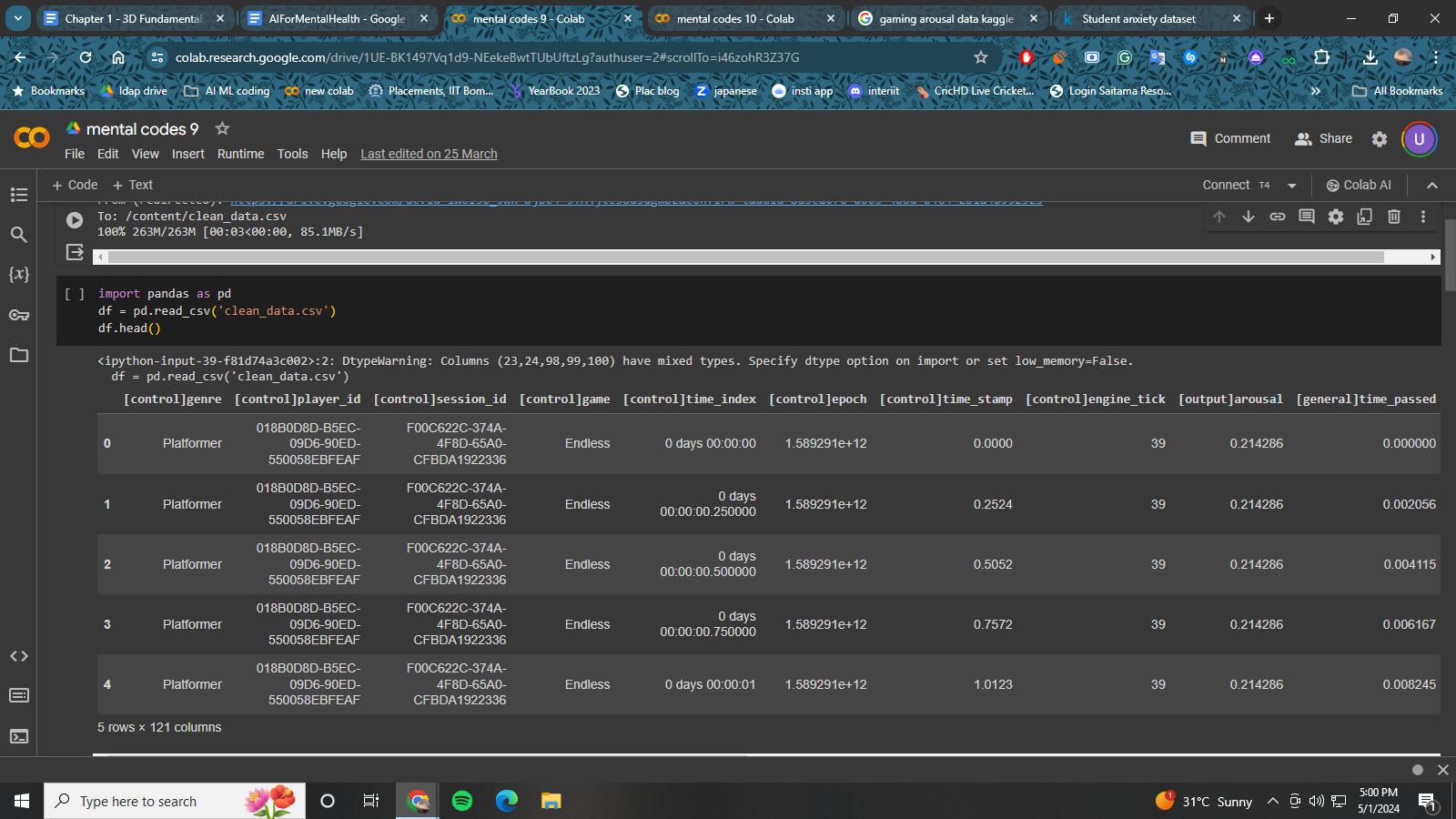
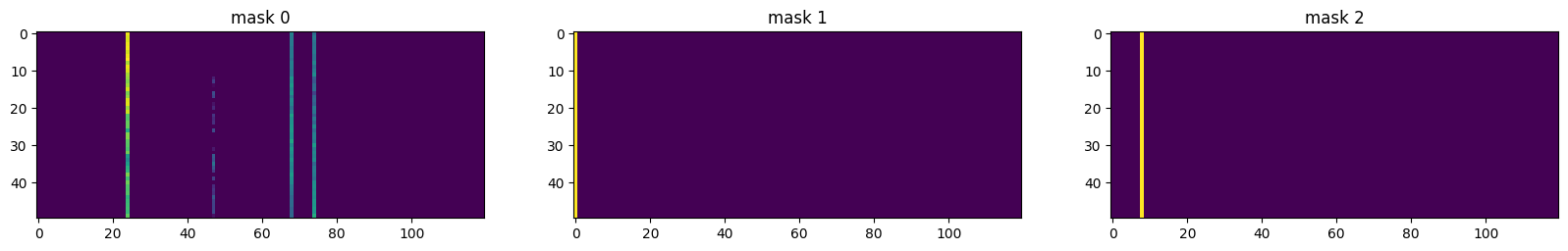
# Gaming to detect the real-time emotional state of a player during various stages in gameplay

Gaming is very common nowadays amon youngsters. This part aims to detect the emotional state of the people during gaming. We make use of the popular gaming stimulation dataset AGAIN affect gaming annotation dataset- [data](https://again.institutedigitalgames.com/). This data contains realtime annotation of gamers and their arousal levels.



First we begin by loading the data and separating it into categorical and continuous features. Categorical feature are those which have classes or which are quantised whereas continuous features are not quantised as the name suggests. We convert the categorical features to integers using the labelencoder from sklearn. Finally, we use the popular TabNet model to perform regression. TabNet is one of the most widely used and a highly accurate model for both tabular classification and regression. In the end we also plot out the explanation masks from the trained model.



## CODE:

## Installation

Pip install pandas pytorch-tabnet sklearn torch numpy matplotlib

## Imports

Import pandas as pd

from pytorch\_tabnet.tab\_model import TabNetRegressor

import torch

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import mean\_squared\_error

import numpy as np

from matplotlib import pyplot as plt

%matplotlib inline

## Data Loading

df = pd.read\_csv('clean\_data.csv')

## Data processing

categorical\_columns = []

categorical\_dims = {}

for col in df.columns[df.dtypes == object]:

print(col, df[col].nunique())

l\_enc = LabelEncoder()

df[col] = df[col].fillna("NA")

df[col] = l\_enc.fit\_transform(df[col].values)

categorical\_columns.append(col)

categorical\_dims[col] = len(l\_enc.classes\_)

for col in df.columns[df.dtypes == 'int64']:

df.fillna(df.loc[:, col].mean(), inplace=True)

for col in df.columns[df.dtypes == 'float64']:

df.fillna(df.loc[:, col].mean(), inplace=True)

## Model building

target = '[output]arousal'

features = [ col for col in df.columns if col not in [target]]

cat\_idxs = [ i for i, f in enumerate(features) if f in categorical\_columns]

cat\_dims = [ categorical\_dims[f] for i, f in enumerate(features) if f in categorical\_columns]

cat\_emb\_dim = [2, 64, 128, 6, 128, 1024, 16, 3,8,4,5]

clf = TabNetRegressor(cat\_dims=cat\_dims, cat\_emb\_dim=cat\_emb\_dim, cat\_idxs=cat\_idxs)

X\_train = df[features].values

y\_train = df[target].values.reshape(-1, 1)

clf.fit(

X\_train=X\_train, y\_train=y\_train,

eval\_metric=['rmsle', 'mae', 'rmse', 'mse'],

max\_epochs=25,

patience=50,

batch\_size=1024,

virtual\_batch\_size=128,

num\_workers=0, drop\_last=False)

## Prediction

preds = clf.predict(X\_train)

y\_true = y\_train

test\_score = mean\_squared\_error(y\_pred=preds, y\_true=y\_true)

explain\_matrix, masks = clf.explain(X\_train)

fig, axs = plt.subplots(1, 3, figsize=(20,20))

for i in range(3):

axs[i].imshow(masks[i][:50])

axs[i].set\_title(f"mask {i}")