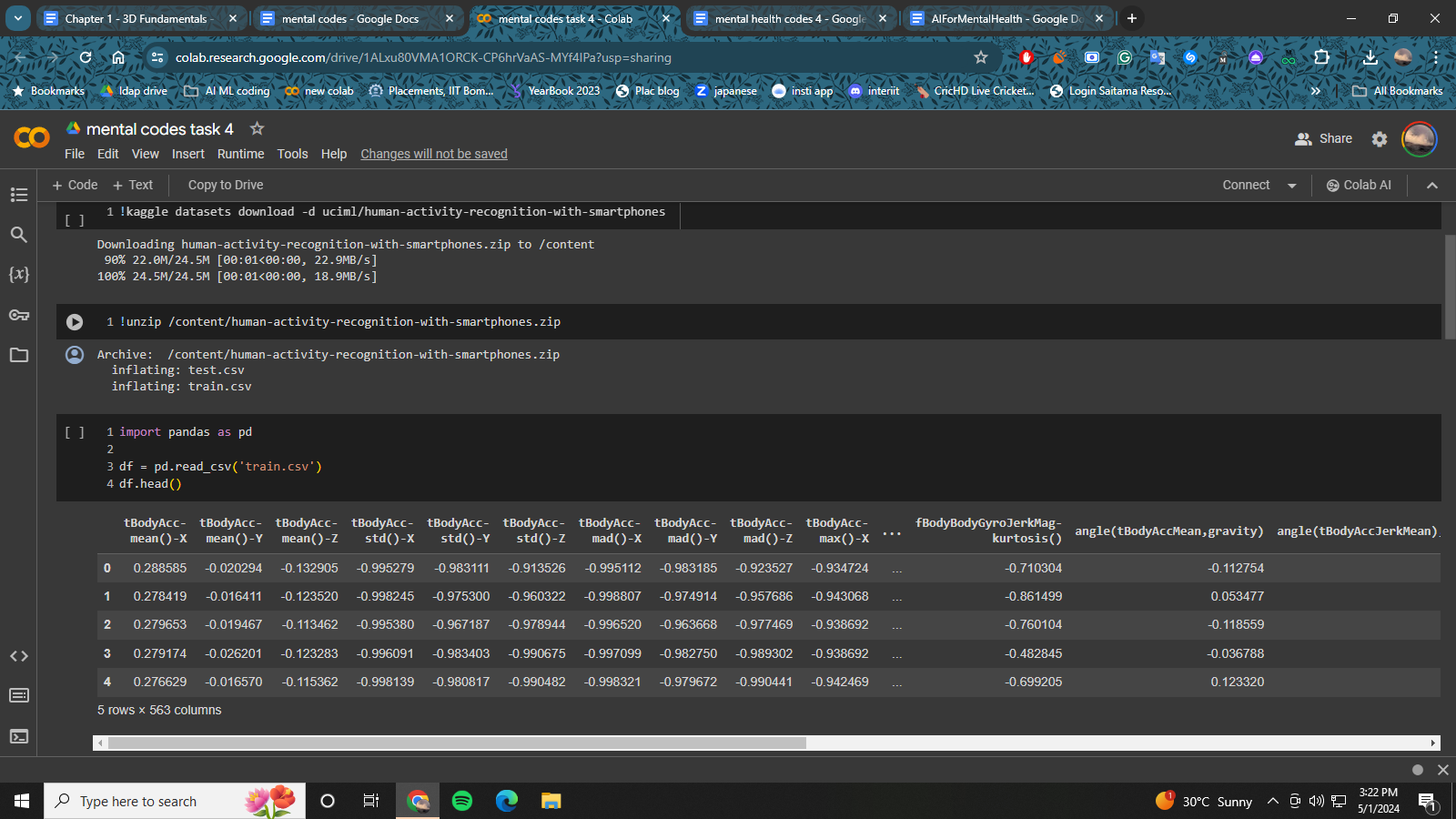
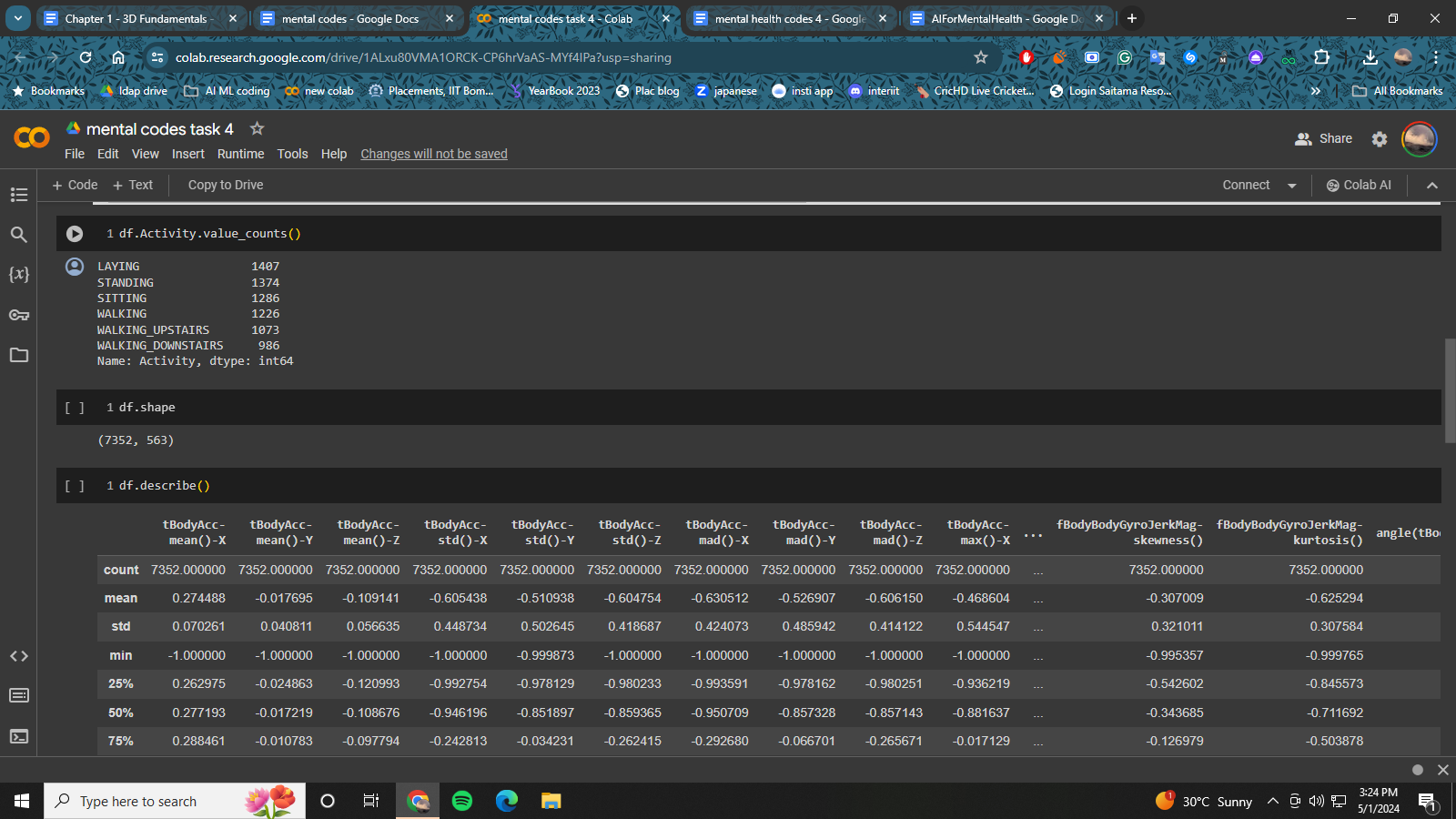
# Activity pattern detection from phone

In this part we take an activity recognition dataset- [data](https://www.kaggle.com/datasets/uciml/human-activity-recognition-with-smartphones)  and perform human activity recognition. The dataset was created by recording 30 participants performing activities of daily living (ADL) while wearing a waist-mounted smartphone with embedded inertial sensors. The objective is to classify activities into one of the six activities performed namely WALKING, WALKING\_UPSTAIRS, WALKING\_DOWNSTAIRS, SITTING, STANDING, LAYING. The dataset consists of more than 7000 datapoints.



This task can be viewed as a tabular classification task where we can leverage the tubular techniques like decision models. First, we begin by loading and understanding the distribution of the data.



After this we use the labelencoder to convert the string labels into integer classes so that it can be fed into the models for training. Then we move onto to use Principal Component Analysis(PCA). This is because the original data has too many features which can hamper the model training. So, by using PCA we project the higher dimension data into lower dimension so as to decrease the number of features. Finally we use the the Random Forest classifier, a tree based algorithm from sklearn to make the predictions.

## CODE:

## Installation

Pip install pandas sklearn

## Imports

Import pandas as pd

from sklearn.preprocessing import OneHotEncoder, LabelEncoder

from sklearn.decomposition import PCA

from sklearn.ensemble import RandomForestClassifier

## Data Loading

df = pd.read\_csv('train.csv')

print(df.Activity.value\_counts())

X\_train = df.iloc[:,0:len(df.columns)-1]

Y\_train = df.iloc[:,-1]

## Data processing

le = LabelEncoder()

Y\_train = le.fit\_transform(Y\_train)

le\_name\_mapping = dict(zip(le.classes\_, le.transform(le.classes\_)))

print(le\_name\_mapping)

## PCA

pca = PCA(0.95)

pca.fit(X\_train)

train\_x\_pca = pca.transform(X\_train)

print(pca.n\_components\_)

print(pca.explained\_variance\_)

## Model Building

clf = RandomForestClassifier()

clf.fit(train\_x\_pca, Y\_train)

clf.score(train\_x\_pca,Y\_train)