```
import pandas as pd
from math import *
from math import sqrt
from sklearn import metrics
from sklearn import neighbors
from sklearn.model_selection import train_test_split




#based on release coivd cases for each country for one day, preict whethetr in the
next day there will be a profit in investment
#lets asssume it costs 0.1% ot buy and sell the data, there mut be at least a 101%
stock value the following day
# to make single day profits
# stock price precentage current/previous * 100

findat = pd.read_csv('DATA1002 Spreadsheets - DATA1002 Spreadsheets - Financials
CLEANED.csv')
headat = pd.read_csv('DATA1002 Spreadsheets - DATA1002 Spreadsheets - Health Care
CLEANED.csv')
itdat = pd.read_csv('DATA1002 Spreadsheets - DATA1002 Spreadsheets - IT
CLEANED.csv')
resdat = pd.read_csv('DATA1002 Spreadsheets - DATA1002 Spreadsheets - Resources
CLEANED.csv')
dfindex = pd.DataFrame()

#below we merge the covid data with the stock data for easy use in the ml training

dfindex['Date'] = findat['Date']
dfindex['FinClose'] = (findat['Close'].diff()/findat['Close'])*100
dfindex['HeaClose'] = (headat['Close'].diff()/headat['Close'])*100
dfindex['ItClose'] = (itdat['Close'].diff()/itdat['Close'])*100
dfindex['ResClose'] = (resdat['Close'].diff()/resdat['Close'])*100

## now if there is a larger than 1% increase, it is worth single day investing
## i will set values at 1 if it is worth investing, and 0 if it is not because
# this will make the k-nearest neighbours algorithm cleaner to compute and
understand

dfindex.loc[dfindex['FinClose'] >1 , 'FinClose'] = 1
dfindex.loc[dfindex['FinClose'] <1 , 'FinClose'] = 0
dfindex.loc[dfindex['HeaClose'] >1 , 'HeaClose'] = 1
dfindex.loc[dfindex['HeaClose'] <1 , 'HeaClose'] = 0
dfindex.loc[dfindex['ItClose'] >1 , 'ItClose'] = 1
dfindex.loc[dfindex['ItClose'] <1 , 'ItClose'] = 0
dfindex.loc[dfindex['ResClose'] >1 , 'ResClose'] = 1
dfindex.loc[dfindex['ResClose'] <1 , 'ResClose'] = 0
bigdf = pd.read_csv('DATA1002 Spreadsheets - Merged Covid and Stocks.csv')
coviddata = pd.DataFrame()
coviddata['Cases India'] = bigdf['Cases India']
coviddata['Cases UK'] = bigdf['Cases UK']
coviddata['Cases US'] = bigdf['Cases US']
coviddata['Cases STRAYA'] = bigdf['Cases STRAYA']

print(dfindex)
print(coviddata)

largedata = dfindex.merge(coviddata, left_index=True, right_index=True)
```

```
largedata.dropna(inplace=True)

# above we double check that somehow no NA values where created during the merge

print(largedata)

# change value so buy = 1, dont buy = 0
#below we build an input loop so that the index could be chosen by the user/ the
automatic trading machine
# this input is then converted into a value that can reference the respective
column in the code for the training

qo = True
while qo:
    p = input('select Australian index investment option: Financials(f), Health(h),
It(i), Resources(r)')
    if p == 'f':
        g = 2
        qo = False
    elif p == 'h':
        g = 3
        qo = False
    elif p == 'i':
        g=4
        qo = False
    elif p == 'r':
        g = 5
        qo = False
    else:
        print('invalid')

## in the end we decided not to use the covid data from india because it had such a
different volatilty to
## the other three countries and we didnt want it to potentially misconstrue our
predictions

X = largedata.values[:, 6:9]      # slice dataFrame for input variables
y = largedata.values[:, g]        # slice dataFrame for target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.9,
random_state=30)
neigh = neighbors.KNeighborsRegressor(n_neighbors=4).fit(X_train, y_train)

#below we create a few loops for inputting the reported covid cases for the day, we
make sure these
# are numeric so that there is no errors in the test

us_scap = True
uk_scap = True
au_scap = True
while us_scap:
    inUS = input('US cases?')
    if inUS.isnumeric():
        us_scap = False
while uk_scap:
    inUK = input('UK cases?')
    if inUK.isnumeric():
        uk_scap = False
while au_scap:
    inAu = input('Au cases?')
    if inAu.isnumeric():
        au_scap = False

# below we feed the cleaned cases into the algorithm to predict a 1 or a 0

sample = [int(inUK) ,int(inUS),int(inAu)]
print('~~~~~~~\nWill you make profit for a single day investment?\n')
for column, value in zip(list(largedata)[6:9], sample):
    print(column + ' today -  ' + str(value))
sample_pred = neigh.predict([sample])
if sample_pred ==1:
    a = 'Invest today'
else:
    a = 'Not today'
print("verdict: " + a+ "\n")
print('~~~~~~~')

y_pred = neigh.predict(X_test)
# Root mean squared error
mse = metrics.mean_squared_error(y_test, y_pred)
print('Root mean squared error (RMSE):', sqrt(mse))
print('R-squared score:', metrics.r2_score(y_test, y_pred))

#this code could be extremely useful for an automatic Low-Frequency trading machine
because it could be easily
# alltered to simply 'return' a True value that could initiate an investment.
However, because
#most outcomes from the data suggest that there should be no investment made, this
would require high value
#investments which further require high reliabily from the code, trust in the
process could be improved by
# widening the %lossrisk from 1%(line in code 27-34) to a higher percentage(eg 1.5
or 2)
```