Here are my codes so far.

import pandas as pd

import numpy as np

from sklearn import preprocessing

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, r2\_score

from numpy import as array

df = pd.read\_csv(r'C:\Users\admin\Downloads\ehrdata.csv')

df.head()

# Check data types

df.dtypes

# Maake data the same type

df ['HAEMATOCRIT'] = df ['HAEMATOCRIT'].astype(int)

df ['HAEMOGLOBINS'] = df ['HAEMOGLOBINS'].astype(int)

df ['ERYTHROCYTE'] = df ['ERYTHROCYTE'].astype(int)

df ['LEUCOCYTE'] = df ['LEUCOCYTE'].astype(int)

df ['MCH'] = df ['MCH'].astype(int)

df ['MCHC'] = df ['MCHC'].astype(int)

y=df.MCV

X=df.drop('MCV',axis=1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

# Turning the categories into numbers

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

categorical\_features = ['AGE', 'SEX', 'SOURCE']

one\_hot = OneHotEncoder()

transformer = ColumnTransformer([('one\_hot', one\_hot, categorical\_features,)],remainder='passthrough')

transformed\_X = transformer.fit\_transform(X)

transformed\_X

# Create The model.

class LinModel:

def \_\_init\_\_(self):

pass

def health\_rec(self):

y=df.MCV

X=df.drop('MCV',axis=1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(transformed\_X, y, test\_size=0.2, random\_state=0)

reg = LinearRegression().fit(X\_train, y\_train)

y\_pred=reg.predict(X\_test)

RMSE=pow(mean\_squared\_error(y\_pred, y\_test),0.5)

R=r2\_score(y\_pred, y\_test)

return reg, y\_pred, RMSE, R

def coef(self):

return self.health\_rec()[0].coef\_

def main():

cof=LinModel().coef()

print(cof)

if \_\_name\_\_=='\_\_main\_\_':

main()

import phe as paillier

import json

def storedKeys():

public\_key, private\_key = paillier.generate\_paillier\_keypair()

keys={}

keys['public\_key'] = {'n': public\_key.n}

keys['private\_key'] = {'p': private\_key.p,'q':private\_key.q}

with open('recordskeys.json', 'w') as file:

json.dump(keys, file)

storedKeys()

def retrieveKeys():

with open('recordskeys.json', 'r') as file:

keys=json.load(file)

pub\_key=paillier.PaillierPublicKey(n=int(keys['public\_key']['n']))

priv\_key=paillier.PaillierPrivateKey(pub\_key,keys['private\_key']['p'],keys['private\_key']['q'])

return pub\_key, priv\_key

def serializeData(public\_key, data):

encrypted\_data\_list = [public\_key.encrypt(x) for x in data]

encrypted\_data={}

encrypted\_data['public\_key'] = {'n': public\_key.n}

encrypted\_data['values'] = [(str(x.ciphertext()), x.exponent) for x in encrypted\_data\_list]

serialized = json.dumps(encrypted\_data)

return serialized

pub\_key, priv\_key = retrieveKeys()

data = HAEMATOCRIT, HAEMOGLOBINS, ERYTHROCYTE, LEUCOCYTE, THROMBOCYTE, MCH, MCHC, AGE, SEX, SOURCE = [30, 9, 4, 22, 333, 23, 32, 1, 0, 0]

serializeData(pub\_key, data)

datafile=serializeData(pub\_key, data)

with open('data.json', 'w') as file:

json.dump(datafile, file)

from sklearn import linear\_model

def grabData():

with open('data.json', 'r') as file:

d=json.load(file)

data=json.loads(d)

return data

print(grabData())

def crunchData():

data=grabData()

thecoef=LinModel().coef()

pk=data['public\_key']

pubkey= paillier.PaillierPublicKey(n=int(pk['n']))

enc\_nums\_rec = [paillier.EncryptedNumber(pubkey, int(x[0], int(x[1]))) for x in data['values']]

results=sum([thecoef[i]\*enc\_nums\_rec[i] for i in range(len(thecoef))])

return results, pubkey

print(crunchData()[0].ciphertext())

def serializeData():

results, pubkey = crunchData()

encrypted\_data={}

encrypted\_data['pubkey'] = {'n': pubkey.n}

encrypted\_data['values'] = (str(results.ciphertext()), results.exponent)

serialized = json.dumps(encrypted\_data)

return serialized

print(serializeData())

print(sum([data[i]\*thecoef[i] for i in range(len(data))]))

def main():

datafile=serializeData()

with open('answer.json', 'w') as file:

json.dump(datafile, file)

if \_\_name\_\_=='\_\_main\_\_':

main()