TAXI PRICE PREDICTION AND LIFE EXPECTANCY

PART 1:

● Import files using the kaggle.json file that is, using the kaggle API to download the data and extracting it.

● importing necessary libraries. We import some libraries later in the code as per our use.

Cleaning DATA:

● Removing all the null values then resetting the index

● Checking and removing the negative value of fare amount

● Removing the latitude and longitude according to the facts(latitude extend from -90 to 90 and longitude extend from -180 to 180

● Removing a very small value of fare amount from the dataset.

● Removing data points which have very large value of passengers taking only 6 passengers at max.

● New york coordinates are around 41-latitude to (-73)- longitude so removing the data points far away from these coordinates range(40 - 42 and -73 to -75)

● Also some of the latitude and longitude value were exchanged which i replaced back in the dataset as well

Pre-processing:

● Finding correlation between various parameters with the fare amount to see which relates with the fare.

● Converting the date-time into the readable format

● Scaling the data and plotting various graphs like fare vs Haversine distance, fare vs passenger count, fare vs day of week ,etc.

● I also plot manhattan distance against the fare to observe the output, it also showed that there is high relation between the distance and fare amount which is expected.

Models:

● Linear regression:

Cross validation with user defined K and self implemented.

First I created my own linear regression model using sklearn library and the converted it into k fold cross validation where I split the data into k parts then splitting the data into training and test and finally calculating the score.

(everything is done on scaled data)

● Pseudo inverse:

np.dot(np.dot(np.linalg.inv(np.dot(A.transpose(),A)),A.transpose()),y_tra
in)

This is the algorithm to calculate the weight vector. Here A is the numpy matrix of the inputs. The error is quite low in this (~0.01) hence it is a good metric to calculate our outcome.

I tried to create K fold validation for pseudo inverse as well but a “UFuncTypeError” pop up which i was unable to resolve for a very long time(~8 hours) I also tried to use sympy which uses symbols for all calculations but still could not resolve the error.

● I also tried to use a support vector machine for it. But it returns only labels so we have to convert out y_train to a label (using labelEncoder()) but still it wasn’t good enough hence

I choose not to show it and comment on it.
Finally I created a pipeline for scaling training and validating the data.
PART 2:
● Importing necessary library
● Reading the file
● Removing the null values
● Finding the correlation matrix
● Plotting the correlation matrix using plotly and seaborn
● This metric is very helpful in understanding the relation between the features as high correlation implies highly related. we can see (infant_death,under-5 death), (GDP,percentage expenditure),(thinnes 1-19 years,thinness 5-9 years) these are some which have high correlation. This can also be seen intuitively as in each tuple the mentioned parameters are almost similar in nature.
● Plotting histogram for life expectancy. This shows that the average life expectancy is between 70 to 80 years
● we see that in developing nations the mortality is present below 70 years as well but in developed nations the mortality is zero below 50 in this dataset.
● Calculation of RFE was not working for me(it kept on running without giving any output. There might be many reasons for it may be my data is not cleaned enough, maybe there are way too many parameters and I tried to shrink it into very small values)
● Applying OLS to generate p-values for each of the data parameters for classification.
● Scaling the data using the standardScaler().(all calculations will be done on this dataset).
● Converted the Country and status columns into labels for better computation.(strings can not be passed into the models.)
● Splitting the data.
● Applying linear regression with K fold validation(almost similar to part 1,some small changes based on the requirements)
● Applied lasso regression with cross validation and finding the best fit.
● Applied Rigg regression with cross validation and finding the best fit.
● Applying pseudo inverse methods as well.
● Creating a pipeline for scaling training and validating the data