

**Department of Electrical and Computer Engineering**

**Linux lab**

**Project (1)**

**Dataset preprocessing**

**Student’s names & Id’s :**

**Rama Abdlrahman 1191344**

**Dana Hammad 1191568**

**Instructor: D. Mohammad Jubran**

**Date: 29/12/2022**

* **Main Idea**

Since the dataset is **“clean”.** We only need to take care of the logic of the operations requested such as **encoding and scaling** without having to worry about too many edge cases that a non-clean dataset would create. Both types of encoding follow the same logic except **one-hot encoding** has some extra steps. This means that implementing them together in the same function would shorten the code.

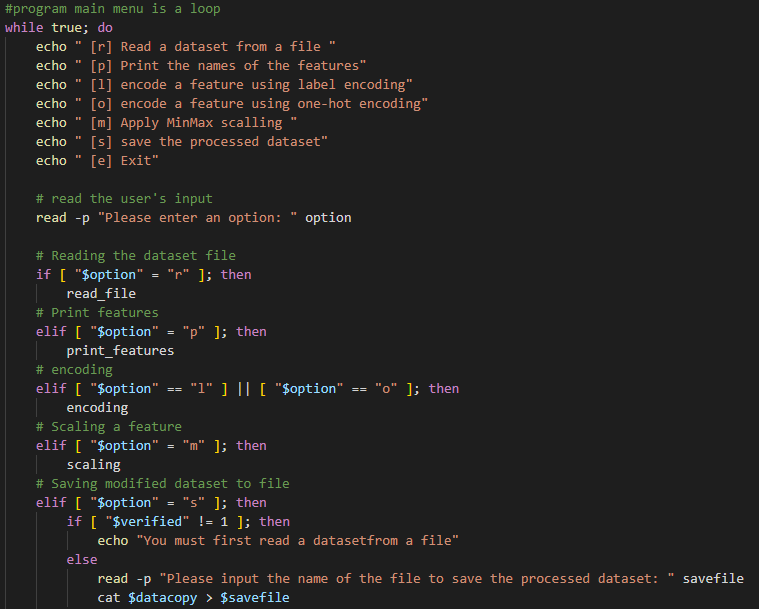
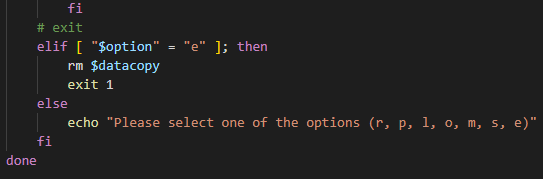
We can use **cut and paste commands** to extract columns of data from the dataset easily, do the encoding and scaling required and then put them back into the original dataset.

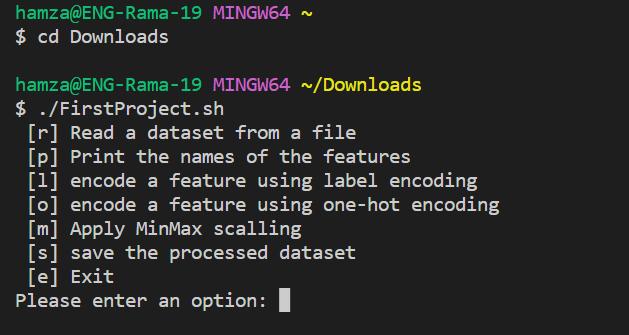
* **Explanation and test cases:**

1. **The main menu:**

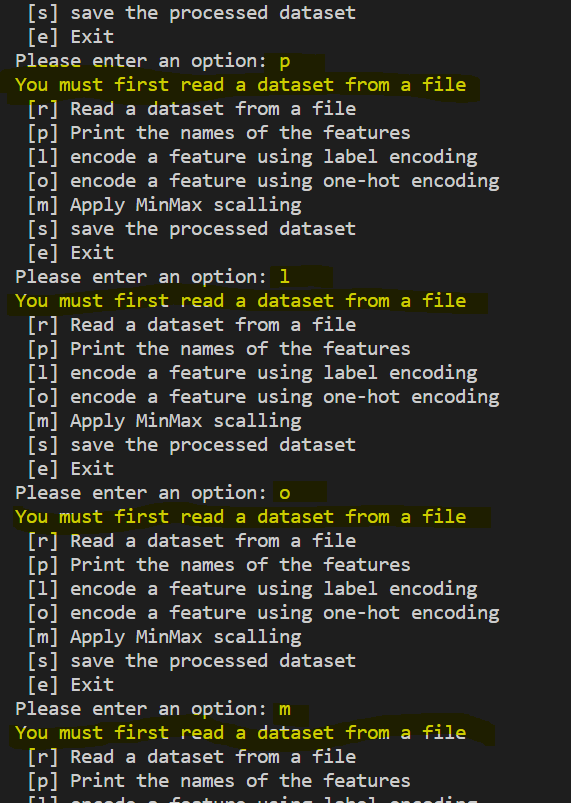
The main menu is a while loop that keeps on printing the menu lines and asking for user input. We check what the input is and run a certain function for each case. For example, if the option is “r” then we run the “read\_file” function.

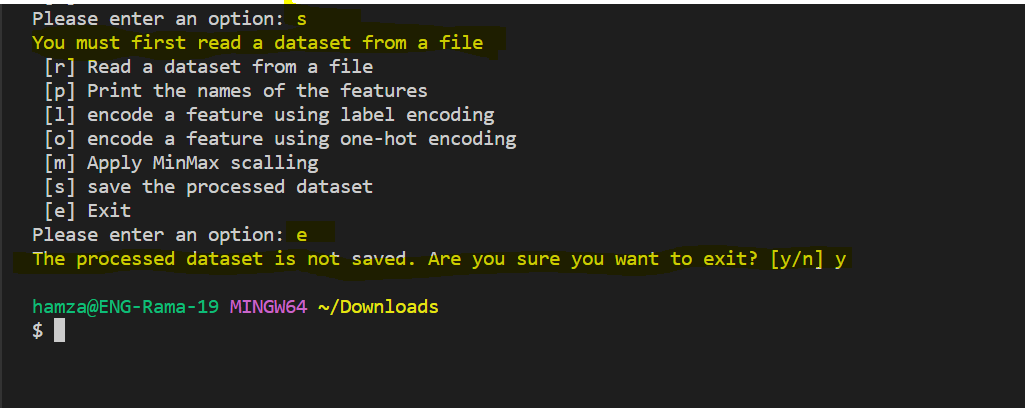
The encoding function holds the logic for both label encoding and one-hot encoding so we run it for both the “l” and “o” options (we check which one it is later inside the function).



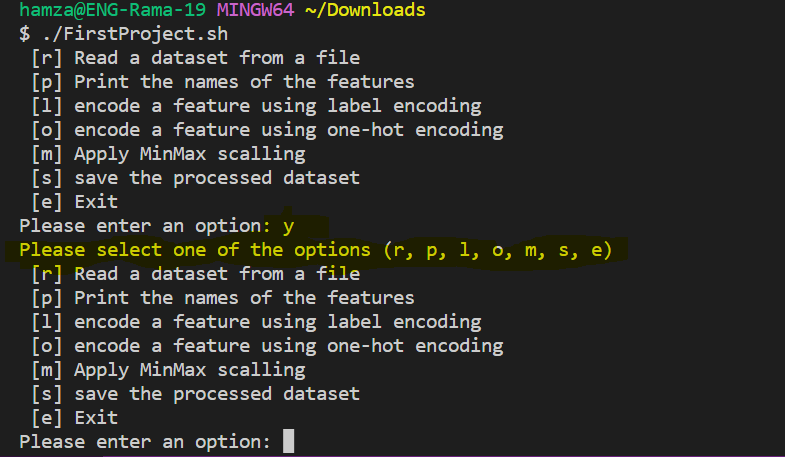


**At first , if you entered anything except ‘r’ or ‘e’ , program will print you a message that you should read file first and the menu will appear again:**



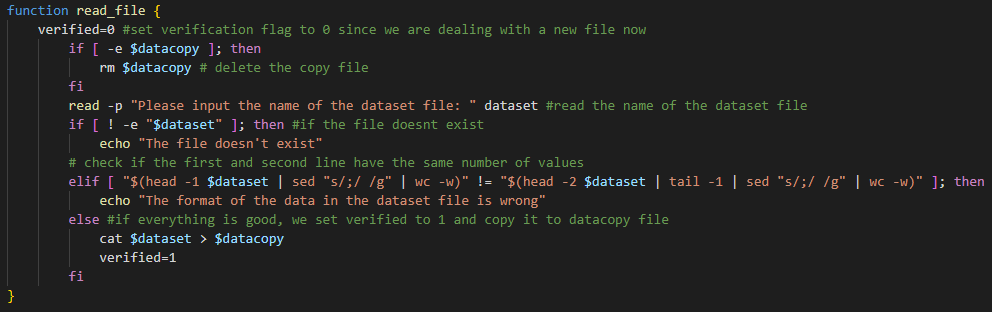


**If you entered wrong choice, a statement asks you to select one of the options in the menu will be printed , and the menu will appear again:**

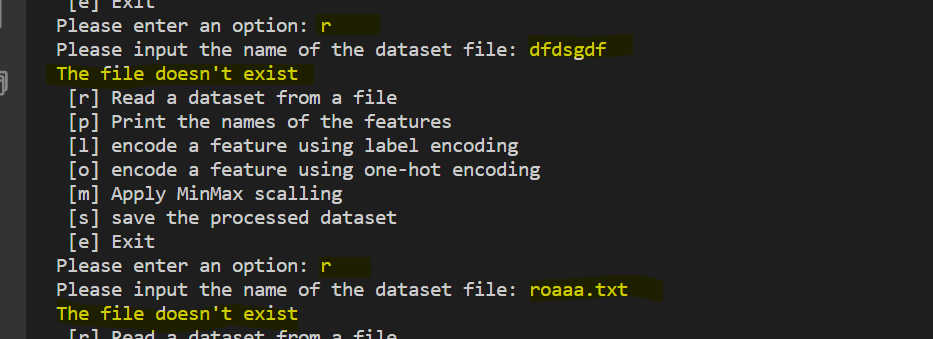


1. **Reading the file:**

To read the file, we first ask the user to input the name of the file. Then, we check if the file exists. After that, we check if the file is file is valid by comparing the second line’s number of columns first line’s. If they’re the same, we set the verified flag to 1 to indicate that the file has been verified. Then, we copy the file to a new file using cat which is where we will do all the other operations necessary (the original input file remains the same).

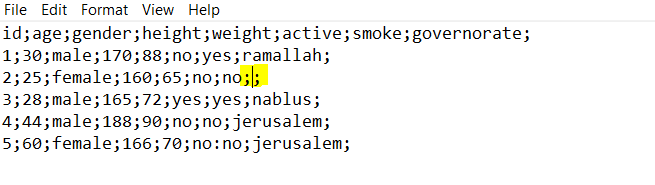


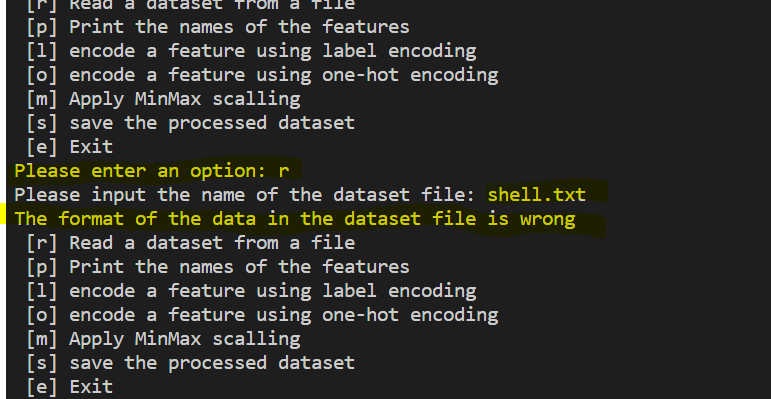
**verifying if the file exists:**



**checking the format of the data in the dataset file. In case of any format problems, the program should print on the screen “The format of the data in the dataset file is wrong” and then return to the main menu:**

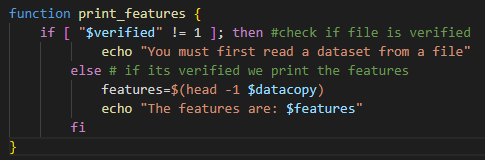
**example of wrong format:**

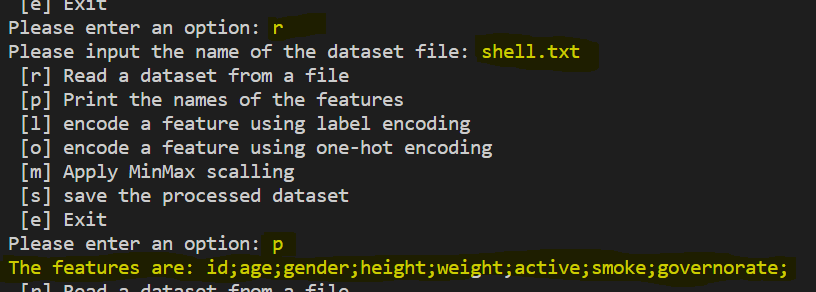




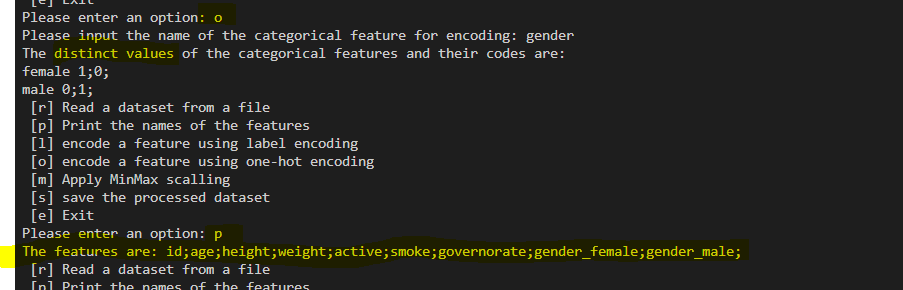
1. **Printing the features:**

Printing the features is simple. We check if the verified flag is set to 1 (if we already read a file or not) and we print the first line using the head command of it if so. Otherwise, we print an error message and return to main menu.





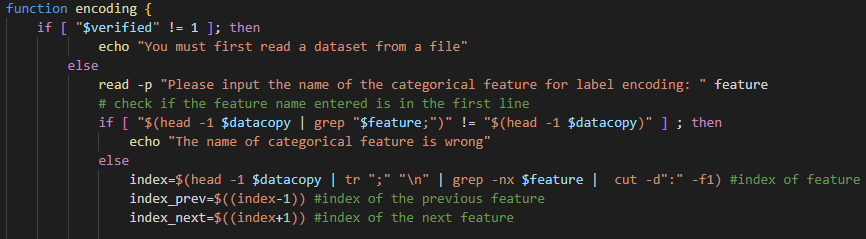
**Also printing features after one hot encoding on feature ‘gender’:**



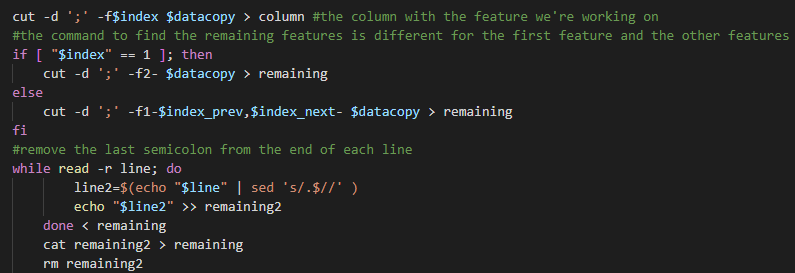
1. **Encoding:**

There are two kinds of encoding but like we mentioned earlier, they follow the same logic up to a certain point. That’s why they are included in the same function.

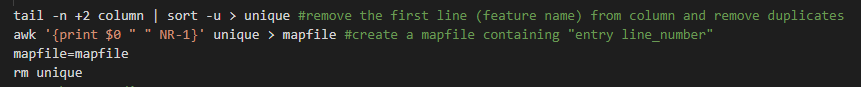
As before, we first check if the file is verified. Then, we ask the user for the name of the feature they want to encode and check if such feature exists. This is done by using grep and head to extract the feature name from the first line. If everything is good, we find the index of the feature by first using head to get the feature line, replacing the semicolons with newline characters, using grep -nx to print “linenumber:value” and finally cut using colon as a delimiter to extract the first value (line number) which is the index. Then, we calculate the index of the next feature and the one before it as seen below. This will be used in the cut command to extract the feature’s column.



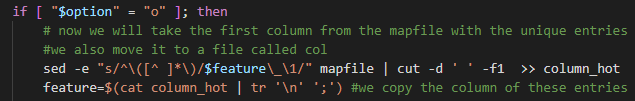
After that, we cut the feature column using semicolon as a delimiter and the index into a new file called column and we cut the remaining features into a file called remaining. We also remove the last semicolon from each line in the file since it caused us problems.



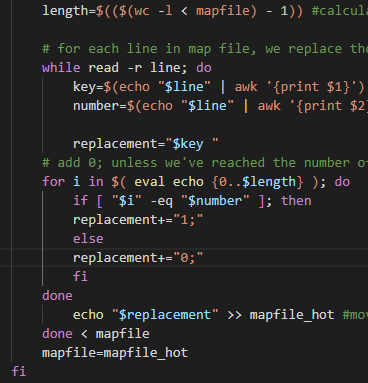
To encode the file, we create a file called mapfile which holds the unique entries of the feature’s column and their respective code. This is done by first removing the first line from the column file which holds the name of the feature, then sorting using sort -u which removes duplicate values and placing the result in a file called unique. Then, we use awk to print the values from the unique file with their line number into the mapfile. In the case of label encoding, the code is just the number of the line they’re in.



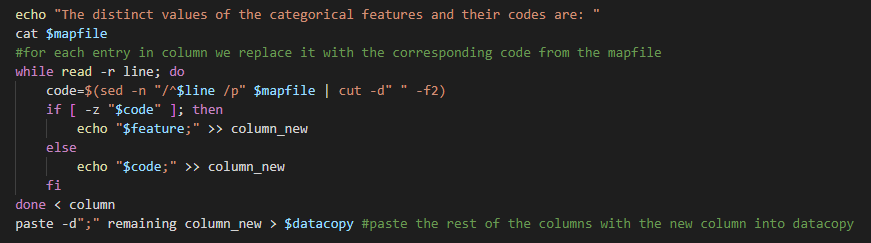
But in the case of one-hot encoding, we first need to modify the mapfile to hold the values of the unique entries of the feature’s column but in the form “feature\_value”. This is done by using the sed command as seen below to the value which is cut from the mapfile’s first column. After that, we remove the newline characters from the column and replace them with semicolons and store the value in the feature variable.



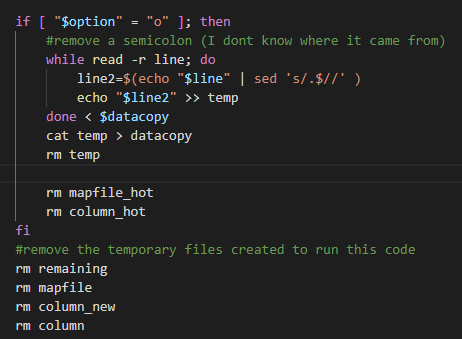
Then, we calculate the length of the mapfile and we read it line by line, storing the value using awk from the first column (distinct value) into the key variable and the second column (code) into the number variable. We also create a variable called replacement which holds the key with a space. After that, we loop through a for loop that is as long as the length of the mapfile and we add 1; or 0; to replacement depending on the number of line we’re on from the while loop and we store the result into a file called mapfile\_hot which would now contain the one-hot code instead of the label code for the distinct values.



Following that, we print the data inside the mapfile using cat and we loop through the column file and create a new column replacing the value in the column with the corresponding value from the mapfile using sed to replace and cut to get the code from the mapfile. This works for both label encoding or one-hot encoding. We also check if the value of the code variable and if its null (we can’t find a code for a certain value). This means that we are on the feature name line and we just print it into the new column file. Finally, we paste the remaining file which contains the other columns with the new column into the datacopy file creating our new dataset with the encoded feature.

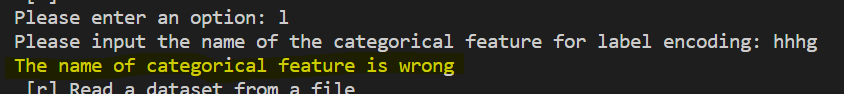


In addition, we remove an extra semicolon we get during one-hot encoding and we remove all of the files that we used for encoding.

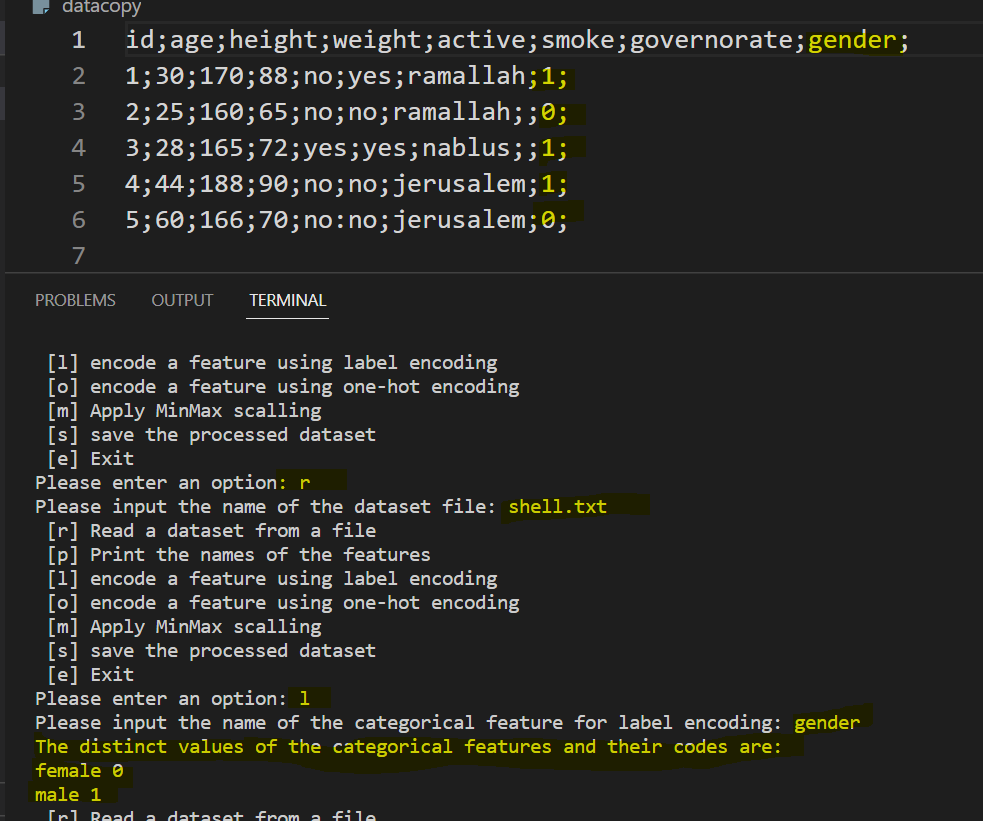


* **Running for label encoding:**

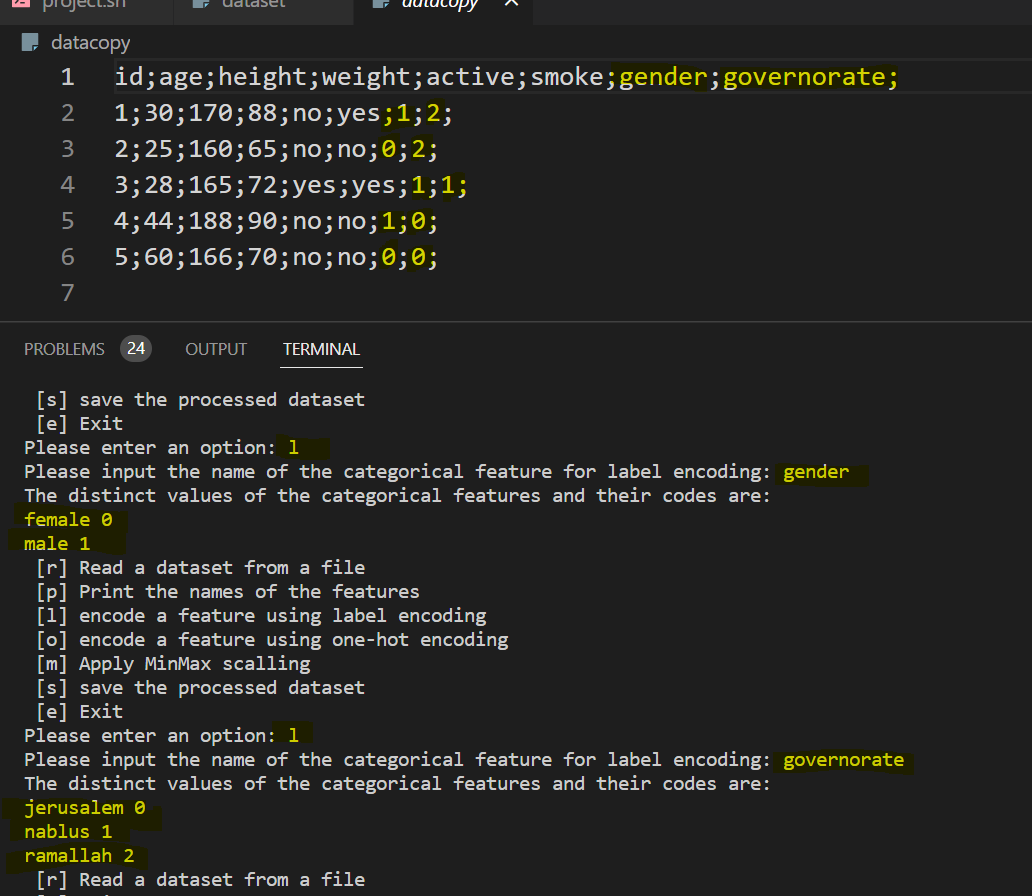
**program verifying that the entered name of the categorical feature exists in the dataset or not :**



**Label encoding for ‘gender’ feature:**

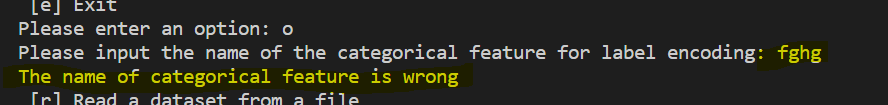


**Label encoding for another feature:**

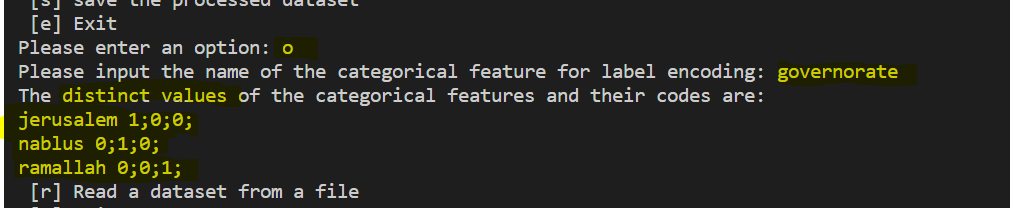


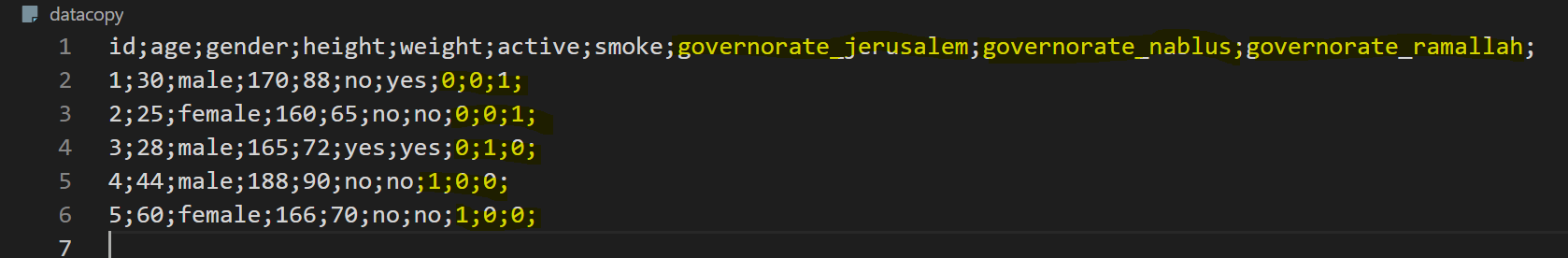
* **For one-hot encoding:**

**program verifying that the entered name of the categorical feature exists in the dataset or not :**

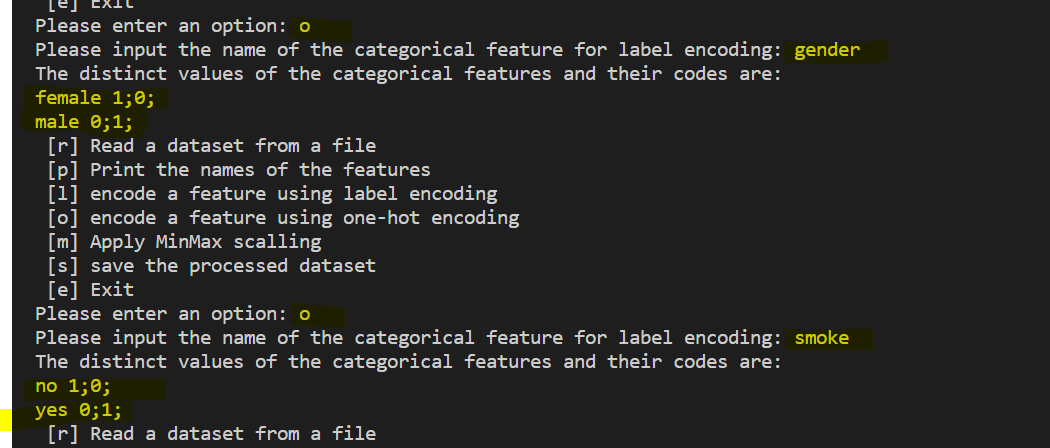


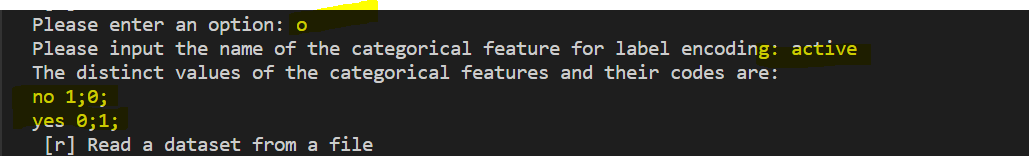
**One-hot encoding for ‘governorate’ feature:**



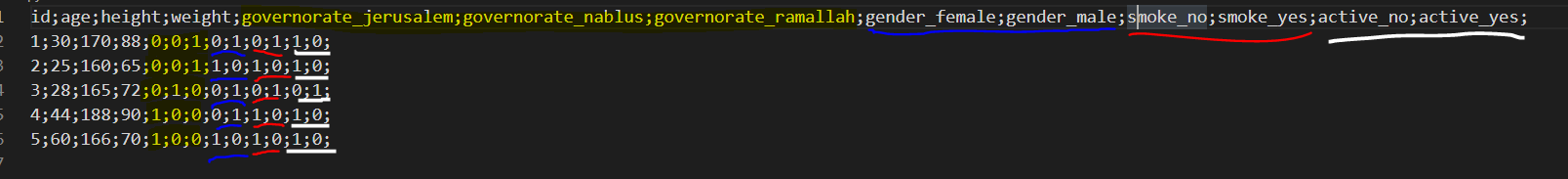


**Testing one-hot encoding for other features:**





**After Appling one-hot encoding for 4 features:**

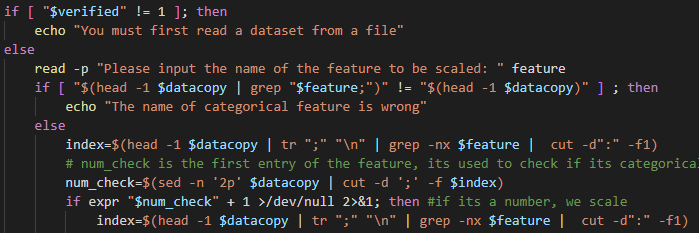


1. **Scaling:**

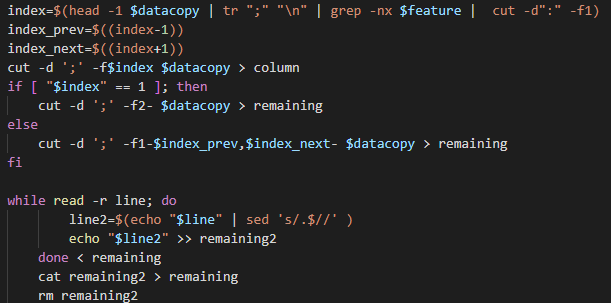
To scale a feature, we do all of the checks we usually do for the previous operations like checking If we read a file already and checking if the feature name is correct. But apart from all of that, we need to check if the feature’s values are numeric or not. We do this by finding the value on the 2nd line of that feature and checking if its numeric or not by doing

expr "$num\_check" + 1 >/dev/null 2>&1;

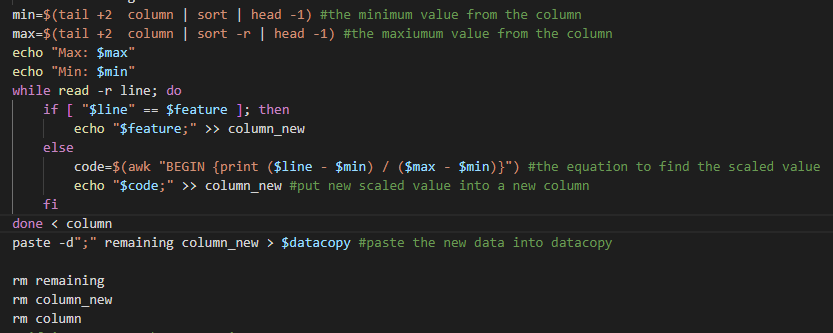
If this line returns true, then num\_check which is the variable that holds the value on the 2nd line is numeric, otherwise we print an error statement.

****

Then, we do many things that are similar to what we did before such as creating the column and remaining files using the index.

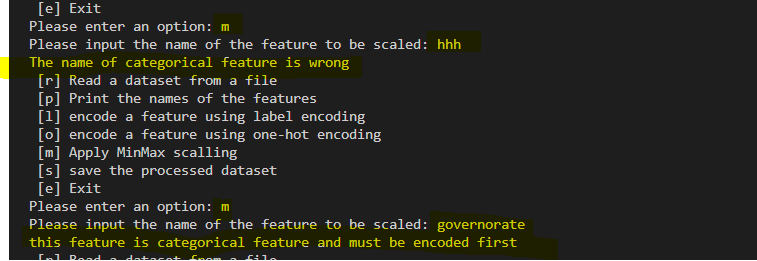
****

Finally, we find the minimum and maximum values in the column file by removing the first line which contains the feature name using tail and then sorting the file and keeping only the first line using head. Then we use the awk command to find the new scaled code using the equation provided in the project description and add it to a new column then paste the new column with the remaining columns into the datacopy file. At the end, we remove any temporary files we’ve created.

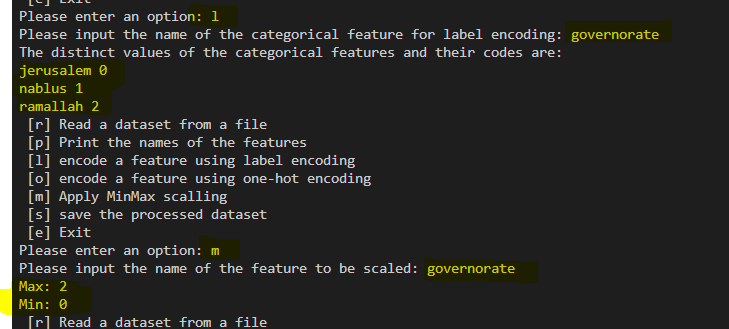
****

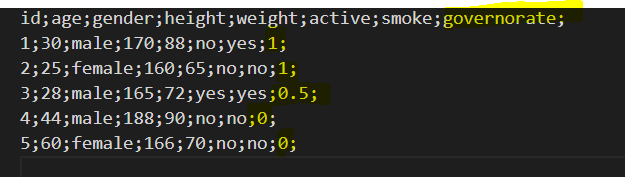
* **Running For scaling :**

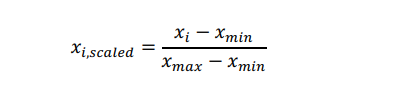
**Trying to enter wrong feature name and trying to enter a categorical feature:**



**Encoding the categorical feature then scaling it:**







**So:**

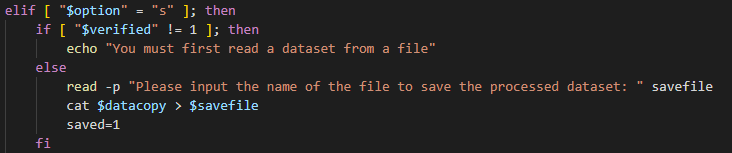
**Scaling for Ramallah= 2-0/2-0=1**

**Scaling for Nablus= 1-0/2-0=0.5**

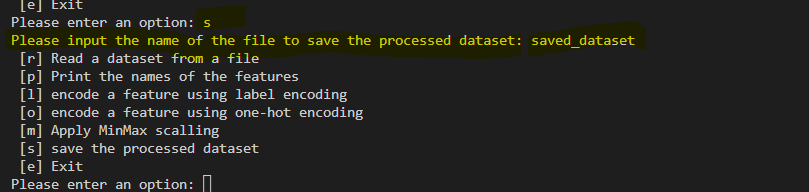
**Scaling for Jerusalem= 0-0/2-0=0**

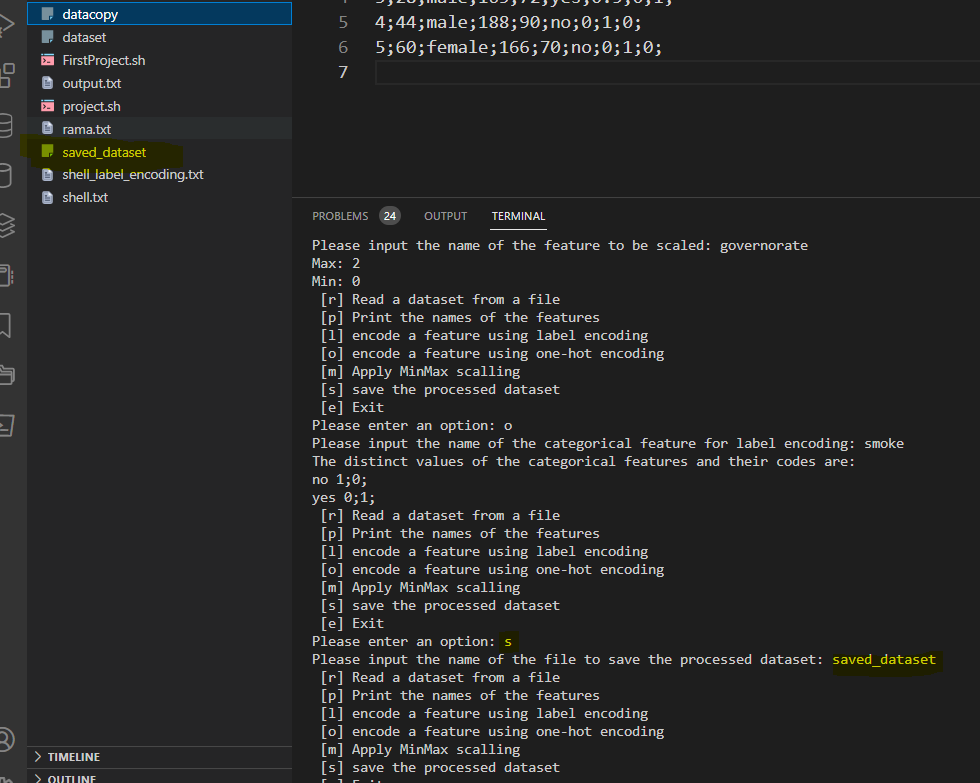
1. **Saving:**

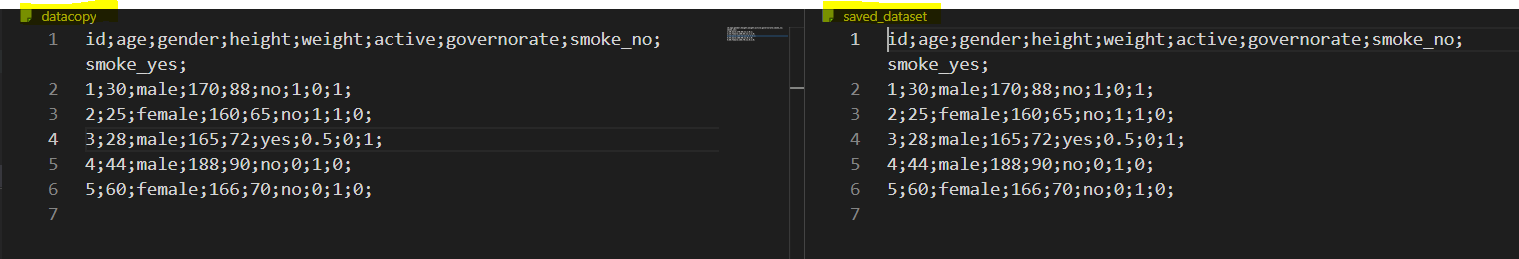
To save a file, we simple use cat to put the contents of the file into a new file defined by the user, we also set the saved flag to 1.



* **Running saving:**

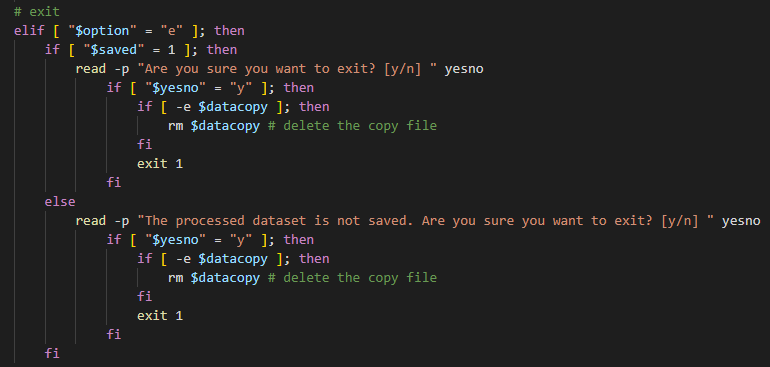




**saving ‘datacopy’ file to ‘saved\_dataset’ file. ‘datacopy’ file removed after finishing the program, ‘saved\_dataset’ file will be saved in my computer:**

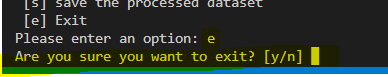
1. **Exit:**

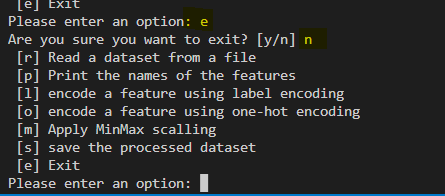
When the user wants to exit the program, we check if the file has been saved first using the saved flag in order to determine what to print to the user.

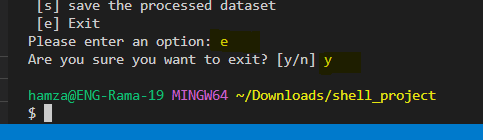


* **Using ‘e’ option:**

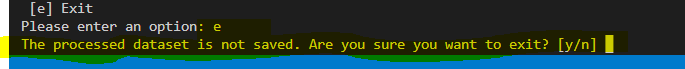
**When changes are saved:**

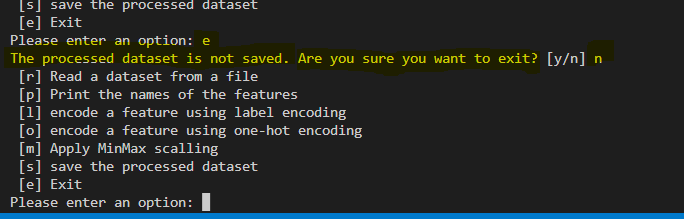


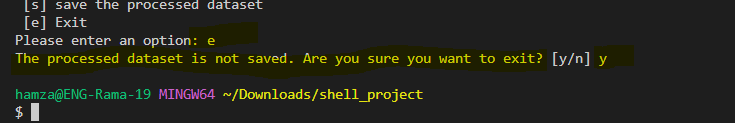




**When changes are not saved:**

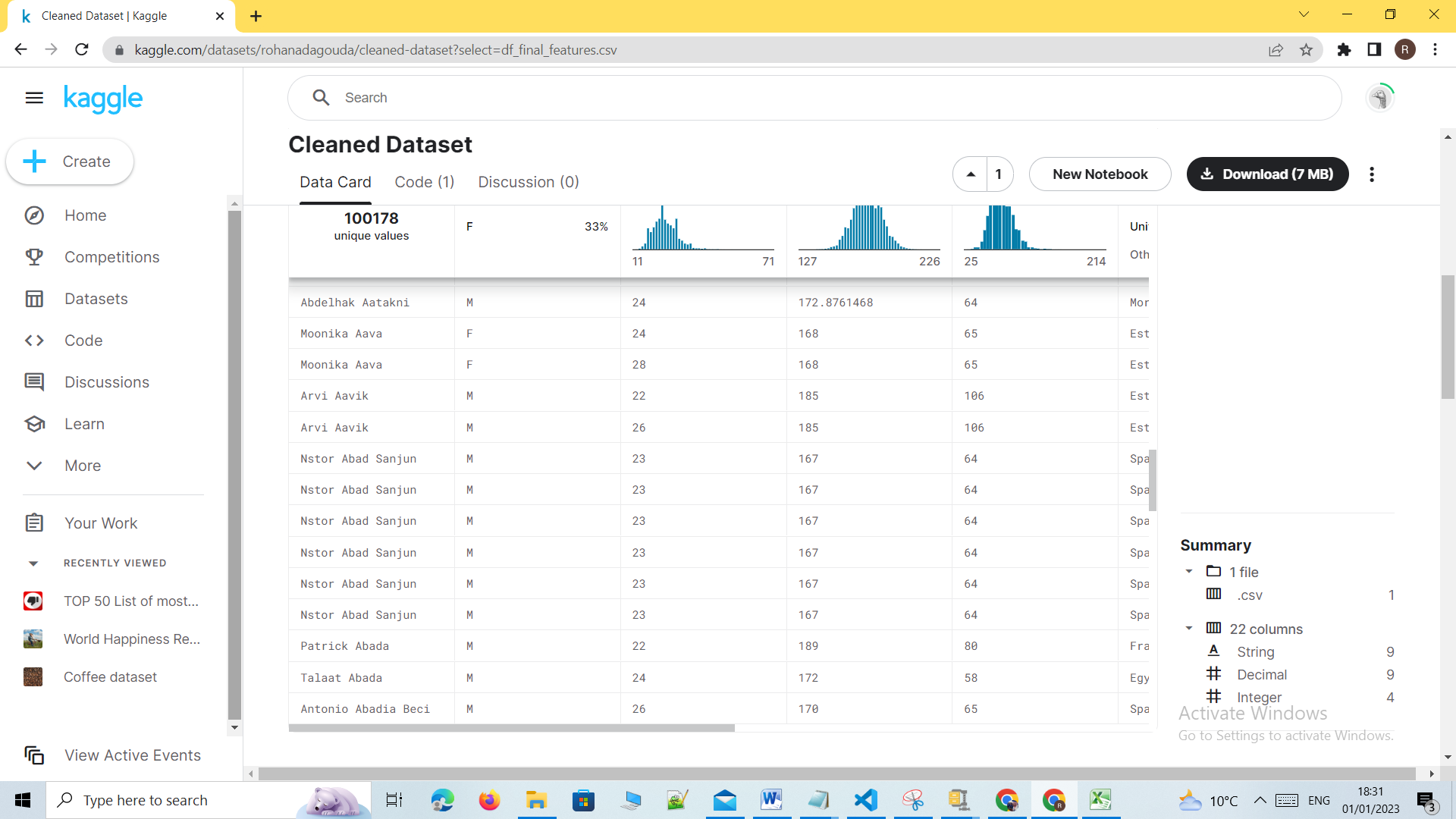




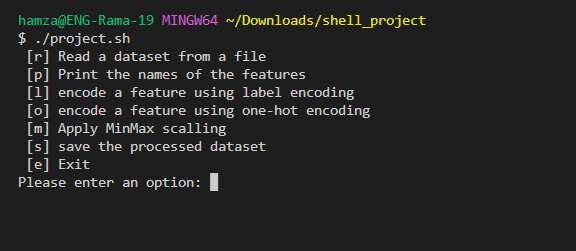


* **Testing for another dataset:**

**I have chosen a cleaned dataset from kaggle site:**

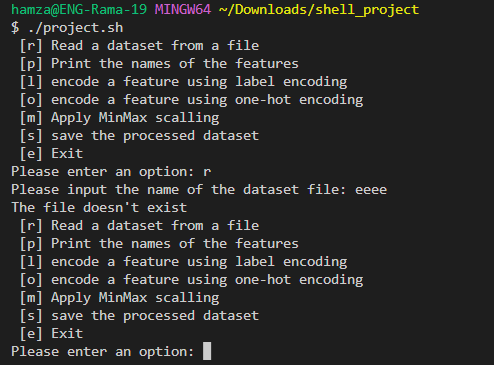


1. **The program should print on the screen the main menu and ask the user to select an option**

****

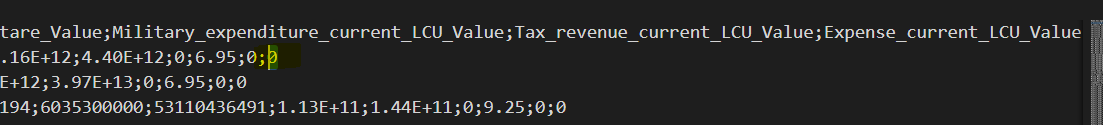
1. **If the user enters ‘r’:**

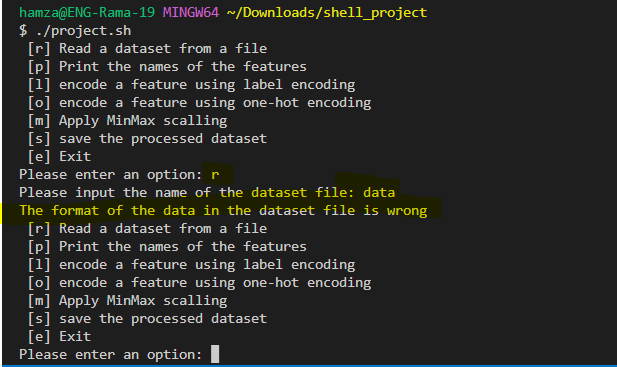
**a. The program should print on the screen “Please input the name of the dataset file”. As it should verify that the file exists.**

****

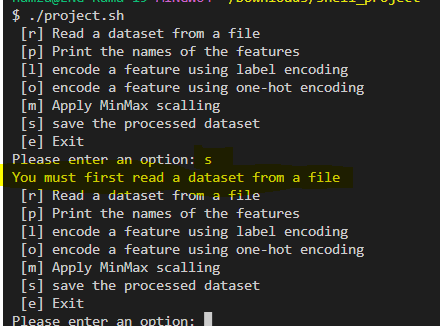
**b. The program should then check the format of the data in the dataset file. In case of any format problems, the program should print on the screen “The format of the data in the dataset file is wrong” and then return to the main menu.**

**Causing error in format by deleting the value of last column for the first row:**

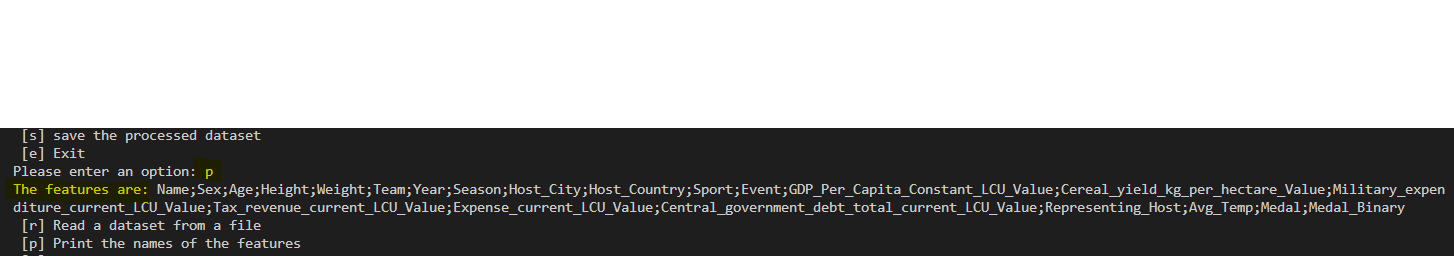
****

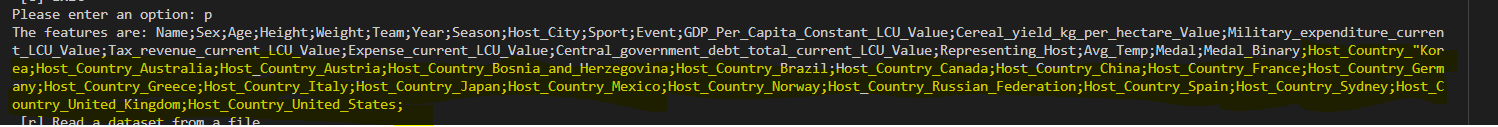
****

**c. If the person selects any option other than ‘r’ or ‘e’ before the format of the data in the dataset file is verified correctly, the program should print on the screen “You must first read a dataset from a file” and then return to the main menu.**

****

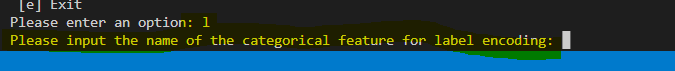
1. **If the user enters ‘p’, the program should print on the screen the names of all features of the dataset file and then return to the main menu.**

****

 **Also printing features after one hot encoding on feature ‘Host\_Country':**

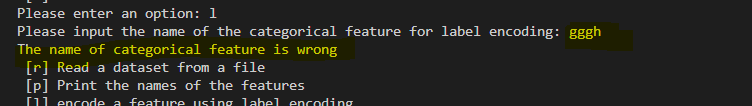
1. **If the user enters ‘l’:**

**a. The program should ask for the name of the feature to be encoded using label encoding by**

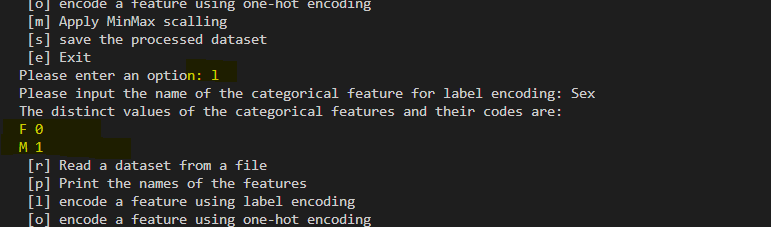
****

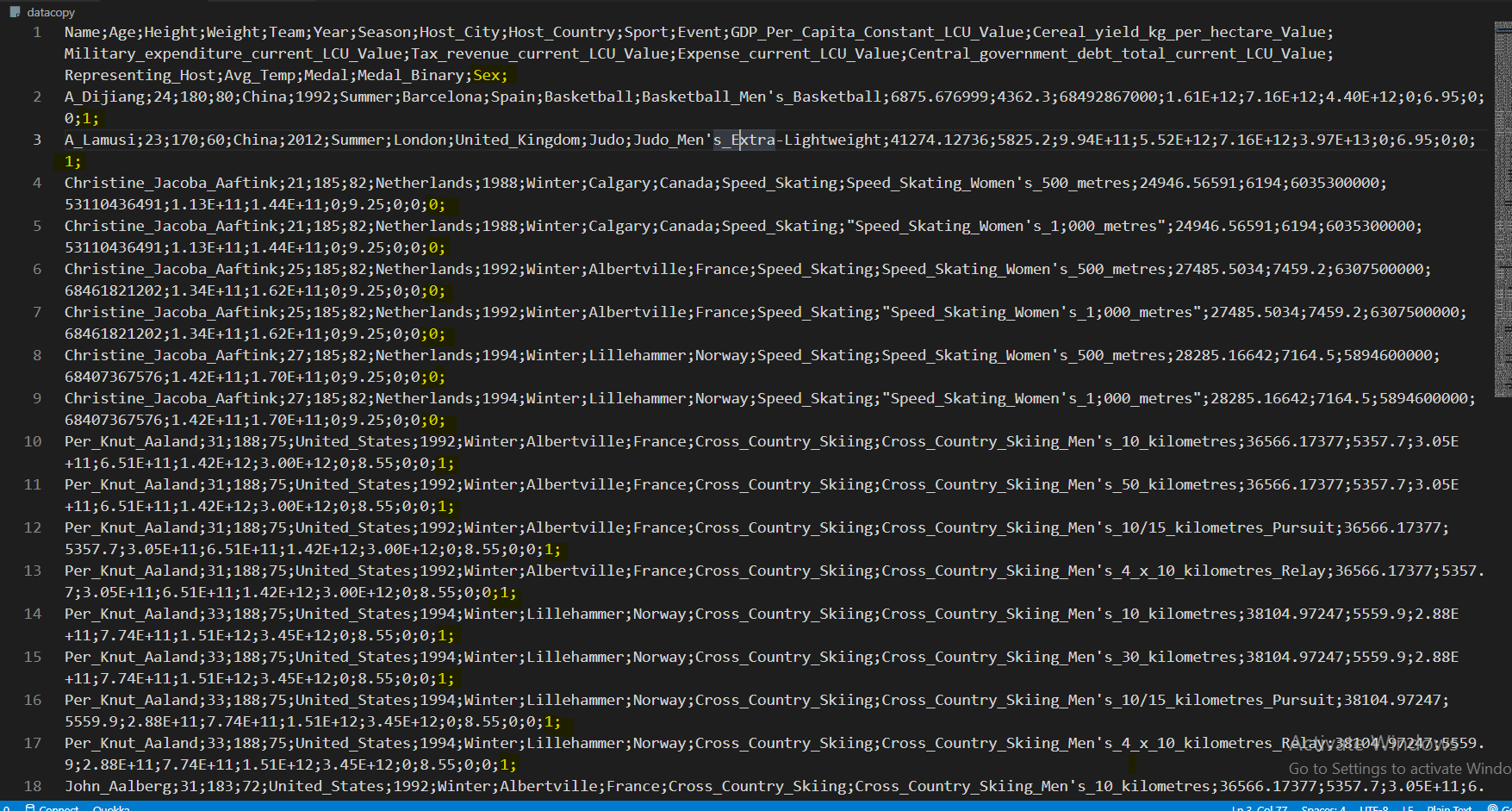
**b. The program should verify if the entered name of the categorical feature exists in the**

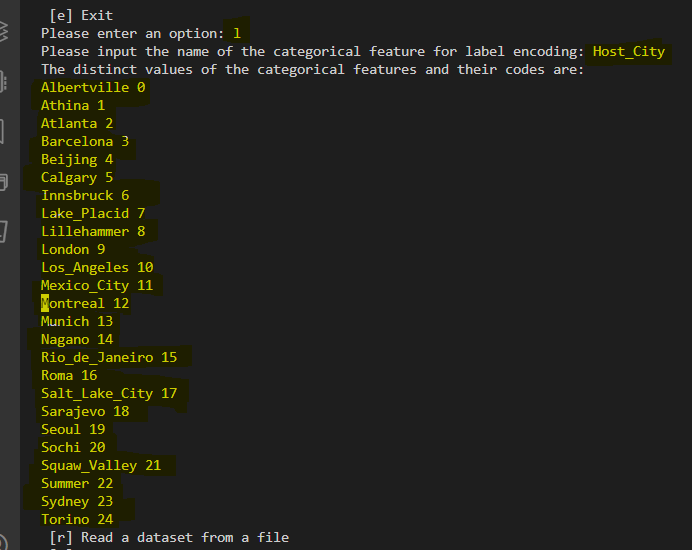
**dataset or not.**

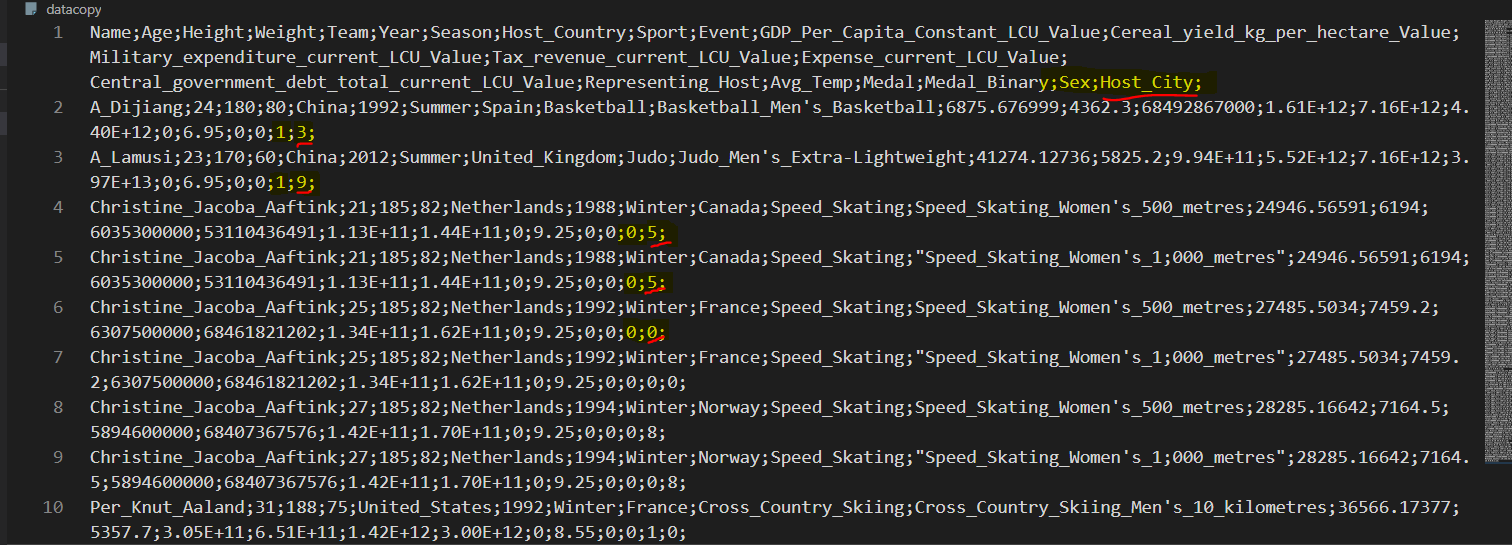
****

**c. If the entered name of the categorical feature exists, the program should print on the screen the distinct values of the categorical feature and the code of each value. And also, to encode the categorical feature in the dataset using label encoding and then return to the main menu.**

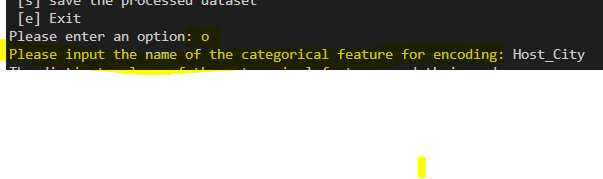
****

****

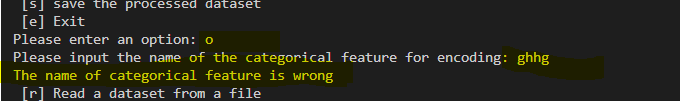
****

****

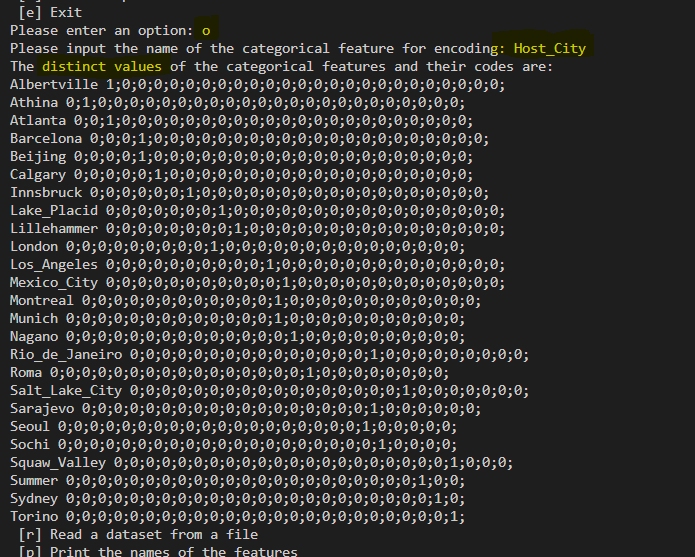
1. **If the user enters ‘o’:**
2. **The program should ask for the name of the feature to be encoded using one-hot encoding by printing on the screen “Please input the name of the categorical feature for one-hot encoding”.**

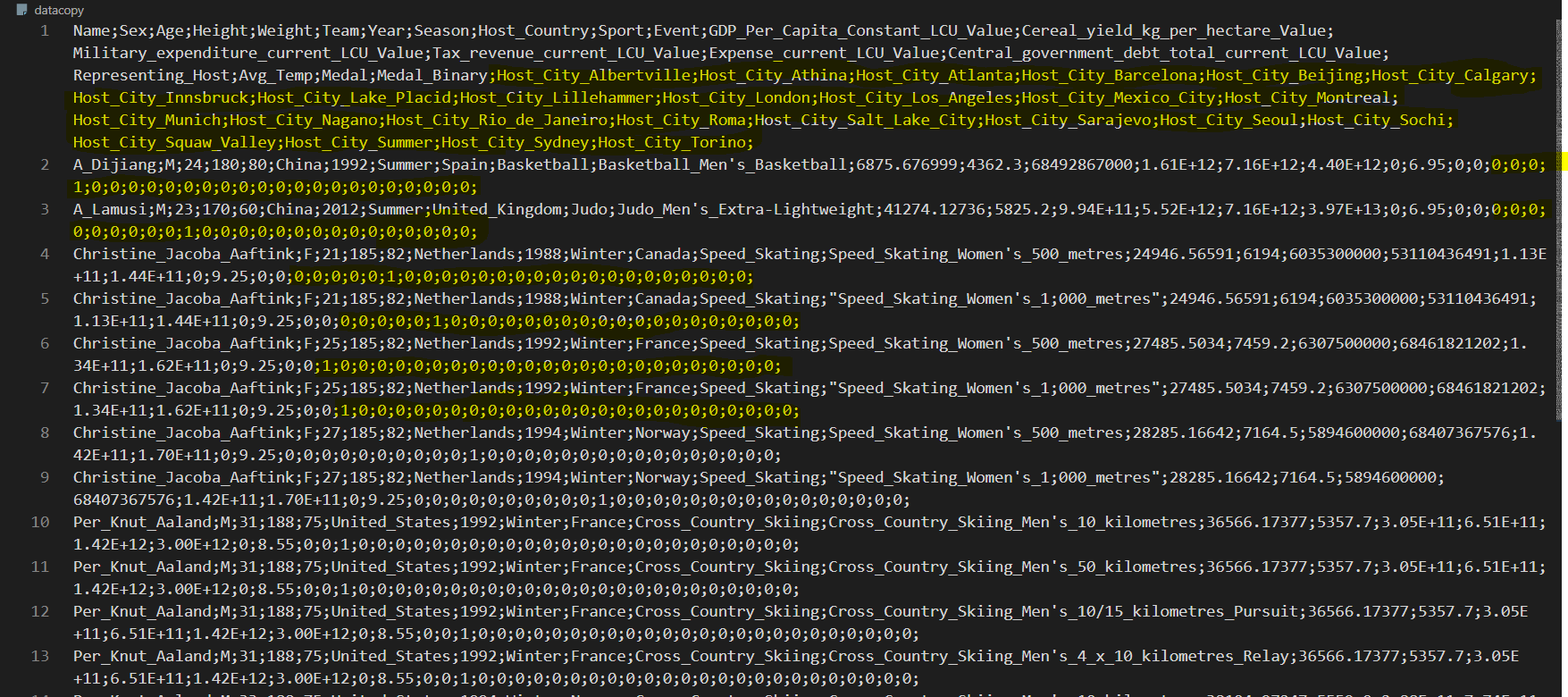
****

1. **The program should verify that the entered name of the categorical feature exists in the dataset, or not.**

****

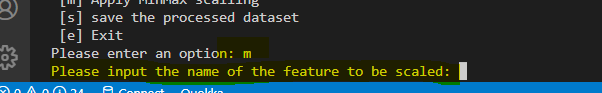
1. **If the entered name of the categorical feature exists, the program should then print on the screen the distinct values of the categorical feature. And also, to encode the categorical feature in the dataset using one-hot encoding and then return to the main menu.**

****

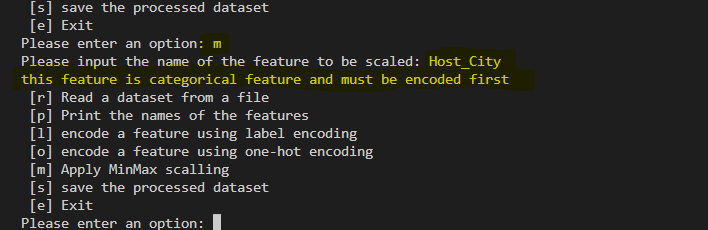
****

1. **If the user enters ‘m’:**

**a. The program should ask for the name of the feature to be scaled using MinMax scaling by printing on the screen “Please input the name of the feature to be scaled”.**

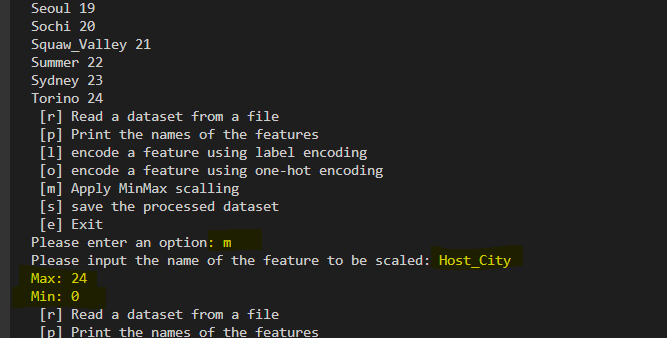
****

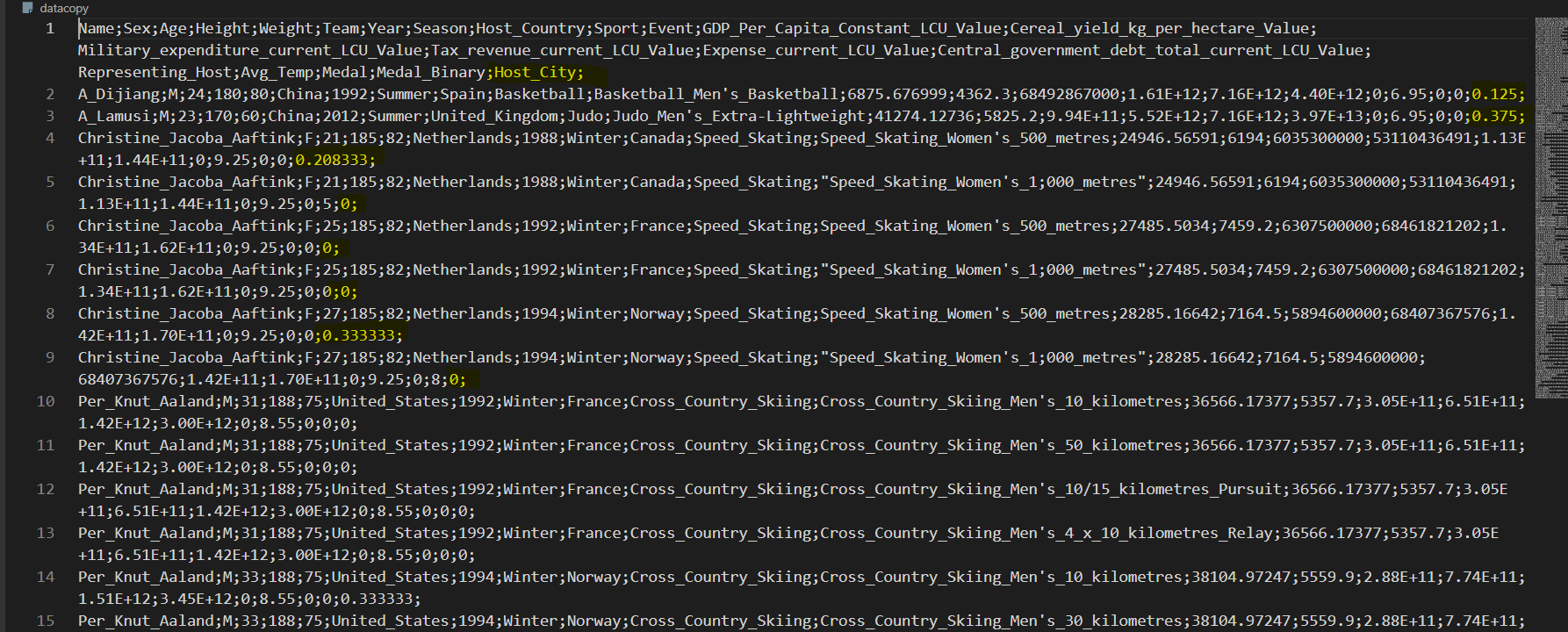
**b. If the entered feature is a categorical feature, the program should verify that this feature is encoded, otherwise, the program should print on screen “this feature is categorical feature and must be encoded first” and then return to the main menu.**

****

1. **If the feature is numeric or encoded categorical feature, the program should print on the screen the minimum and maximum values of the feature and apply the Min-Max scaling to the feature vector and then return to main menu.**

**After applying label encoding on ‘Host\_City’ feature, We were able to use scaling on this feature, as following:**

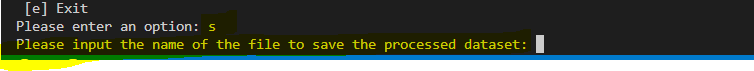




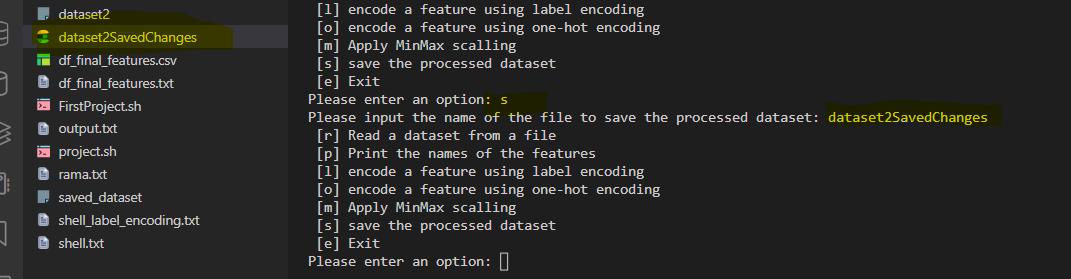
1. **If the user enters ‘s’:**

**a. The program should print on the screen “Please input the name of the file to save the**

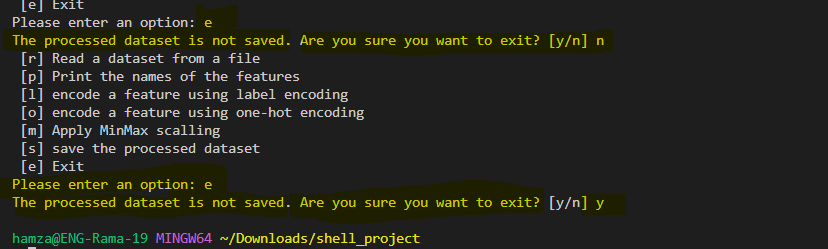
**processed dataset”.**



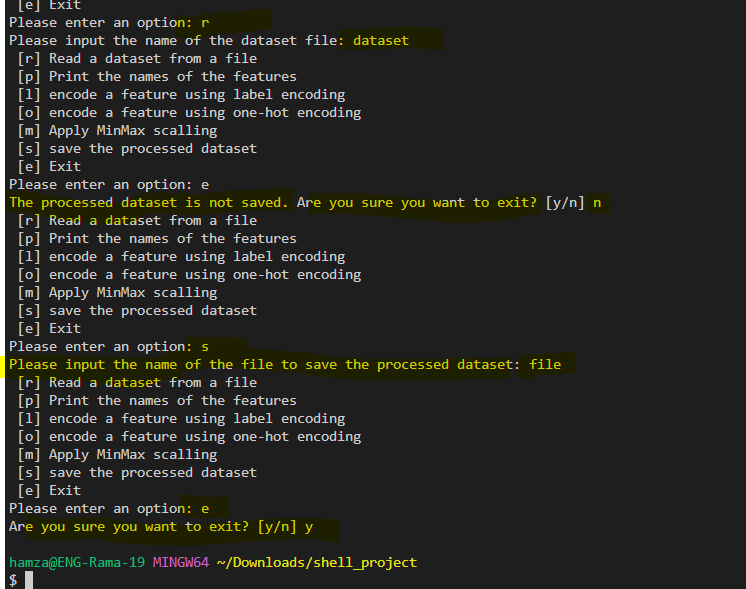
**b. The program should save the processed dataset into the entered filename and then return to the main menu.**



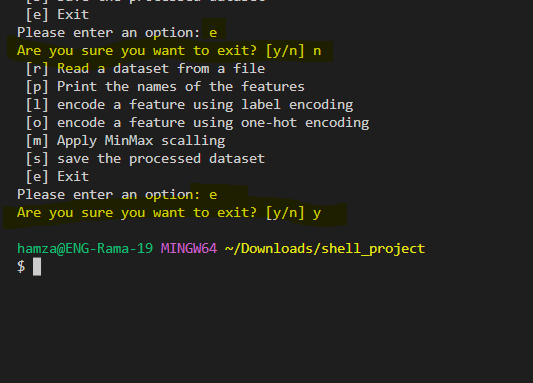
1. **If the user enters ‘e’:**
2. **The program should check if the processed dataset is saved using option “s”. if not, the program should print on the screen “The processed dataset is not saved. Are you sure you want to exist”. If the person inputs “yes”, the program ends. Otherwise, the program should return to main menu.**



**After we returned to main menu we can choose to save file , then exit:**



1. **However, if the dataset is saved, the program should print on the screen “Are you sure you want to exist”. If the person inputs “yes”, the program ends. Otherwise, the program should return to the main menu.**



**Appendix :**

**The full code:**

#!/bin/bash

function read\_file {

    verified=0 #set verification flag to 0 since we are dealing with a new file now

    if [ -e $datacopy ]; then

        rm $datacopy # delete the copy file

    fi

    read -p "Please input the name of the dataset file: " dataset #read the name of the dataset file

    if [ ! -e "$dataset" ]; then                                  #if the file doesnt exist

        echo "The file doesn't exist"

        # check if the first and second line have the same number of values

    elif [ "$(head -1 $dataset | sed "s/;/ /g" | wc -w)" != "$(head -2 $dataset | tail -1 | sed "s/;/ /g" | wc -w)" ]; then

        echo "The format of the data in the dataset file is wrong"

    else #if everything is good, we set verified to 1 and copy it to datacopy file

        cat $dataset >$datacopy

        verified=1

        saved=0 #set saved to 0 since its a new file now

    fi

}

function print\_features {

    if [ "$verified" != 1 ]; then #check if file is verified

        echo "You must first read a dataset from a file"

    else # if its verified we print the features

        features=$(head -1 $datacopy)

        echo "The features are: $features"

    fi

}

function encoding {

    if [ "$verified" != 1 ]; then

        echo "You must first read a dataset from a file"

    else

        read -p "Please input the name of the categorical feature for encoding: " feature

        # check if the feature name entered is in the first line

        if [ "$(head -1 $datacopy | grep "$feature;")" != "$(head -1 $datacopy)" ]; then

            echo "The name of categorical feature is wrong"

        else

            index=$(head -1 $datacopy | tr ";" "\n" | grep -nx $feature | cut -d":" -f1) #index of feature

            index\_prev=$((index - 1))                                                    #index of the previous feature

            index\_next=$((index + 1))                                                    #index of the next feature

            cut -d ';' -f$index $datacopy >column #the column with the feature we're working on

            #the command to find the remaining features is different for the first feature and the other features

            if [ "$index" == 1 ]; then

                cut -d ';' -f2- $datacopy >remaining

            else

                cut -d ';' -f1-$index\_prev,$index\_next- $datacopy >remaining

            fi

            #remove the last semicolon from the end of each line

            while read -r line; do

                line2=$(echo "$line" | sed 's/.$//')

                echo "$line2" >>remaining2

            done <remaining

            cat remaining2 >remaining

            rm remaining2

            tail -n +2 column | sort -u >unique       #remove the first line (feature name) from column and remove duplicates

            awk '{print $0 " " NR-1}' unique >mapfile #create a mapfile containing "entry line\_number"

            mapfile=mapfile

            rm unique

            # one-hot encoding

            if [ "$option" = "o" ]; then

                # now we will take the first column from the mapfile with the unique entries and insert feature\_ to its beginning

                #we also move it to a file called col

                sed -e "s/^\([^ ]\*\)/$feature\\_\1/" mapfile | cut -d ' ' -f1 >>column\_hot

                feature=$(cat column\_hot | tr '\n' ';') #we copy the column of these entries and create a line of them seperated by semicolons since theyre our new features

                length=$(($(wc -l <mapfile) - 1))       #calculate the number of lines in the mapfile

                # for each line in map file, we replace the number of the line with ones are zeros accordingly

                while read -r line; do

                    key=$(echo "$line" | awk '{print $1}')    #value from the first column of the mapfile (name of the entry)

                    number=$(echo "$line" | awk '{print $2}') #number of the line (entry code using label encoding)

                    replacement="$key "

                    # add 0; unless we've reached the number of the line we're on we add 1;

                    for i in $(eval echo {0..$length}); do

                        if [ "$i" -eq "$number" ]; then

                            replacement+="1;"

                        else

                            replacement+="0;"

                        fi

                    done

                    echo "$replacement" >>mapfile\_hot #move the new values to a new mapfile

                done <mapfile

                mapfile=mapfile\_hot

            fi

            echo "The distinct values of the categorical features and their codes are: "

            cat $mapfile

            #for each entry in column we replace it with the corresponding code from the mapfile

            while read -r line; do

                code=$(sed -n "/^$line /p" $mapfile | cut -d" " -f2)

                if [ -z "$code" ]; then

                    echo "$feature;" >>column\_new

                else

                    echo "$code;" >>column\_new

                fi

            done <column

            paste -d";" remaining column\_new >$datacopy #paste the rest of the columns with the new column into datacopy

            if [ "$option" = "o" ]; then

                #remove a semicolon (I dont know where it came from)

                while read -r line; do

                    line2=$(echo "$line" | sed 's/.$//')

                    echo "$line2" >>temp

                done <$datacopy

                cat temp >datacopy

                rm temp

                rm mapfile\_hot

                rm column\_hot

            fi

            #remove the temporary files created to run this code

            rm remaining

            rm mapfile

            rm column\_new

            rm column

        fi

    fi

}

function scaling {

    if [ "$verified" != 1 ]; then

        echo "You must first read a dataset from a file"

    else

        read -p "Please input the name of the feature to be scaled: " feature

        if [ "$(head -1 $datacopy | grep "$feature;")" != "$(head -1 $datacopy)" ]; then

            echo "The name of categorical feature is wrong"

        else

            index=$(head -1 $datacopy | tr ";" "\n" | grep -nx $feature | cut -d":" -f1)

            # num\_check is the first entry of the feature, its used to check if its categorical or not

            num\_check=$(sed -n '2p' $datacopy | cut -d ';' -f $index)

            if expr "$num\_check" + 1 >/dev/null 2>&1; then #if its a number, we scale

                index=$(head -1 $datacopy | tr ";" "\n" | grep -nx $feature | cut -d":" -f1)

                index\_prev=$((index - 1))

                index\_next=$((index + 1))

                cut -d ';' -f$index $datacopy >column

                if [ "$index" == 1 ]; then

                    cut -d ';' -f2- $datacopy >remaining

                else

                    cut -d ';' -f1-$index\_prev,$index\_next- $datacopy >remaining

                fi

                while read -r line; do

                    line2=$(echo "$line" | sed 's/.$//')

                    echo "$line2" >>remaining2

                done <remaining

                cat remaining2 >remaining

                rm remaining2

                min=$(tail +2 column | sort -n | head -1)    #the minimum value from the column

                max=$(tail +2 column | sort -n -r | head -1) #the maxiumum value from the column

                echo "Max: $max"

                echo "Min: $min"

                while read -r line; do

                    if [ "$line" == $feature ]; then

                        echo "$feature;" >>column\_new

                    else

                        code=$(awk "BEGIN {print ($line - $min) / ($max - $min)}") #the equation to find the scaled value

                        echo "$code;" >>column\_new                                 #put new scaled value into a new column

                    fi

                done <column

                paste -d";" remaining column\_new >$datacopy #paste the new data into datacopy

                rm remaining

                rm column\_new

                rm column

            else #if its not a number, we print an error mesasge

                echo "this feature is categorical feature and must be encoded first"

            fi

        fi

    fi

}

datacopy="datacopy" # the name of the file where we store a copy of the dataset

verified=0          # flag to indicate if a file has been verified already or not

saved=0             # flag to indicate if a file has been saved already or not

#program main menu is a loop

while true; do

    echo " [r] Read a dataset from a file "

    echo " [p] Print the names of the features"

    echo " [l] encode a feature using label encoding"

    echo " [o] encode a feature using one-hot encoding"

    echo " [m] Apply MinMax scalling "

    echo " [s] save the processed dataset"

    echo " [e] Exit"

    # read the user's input

    read -p "Please enter an option: " option

    # Reading the dataset file

    if [ "$option" = "r" ]; then

        read\_file

        # Print features

    elif [ "$option" = "p" ]; then

        print\_features

    # encoding

    elif [ "$option" == "l" ] || [ "$option" == "o" ]; then

        encoding

        # Scaling a feature

    elif [ "$option" = "m" ]; then

        scaling

        # Saving modified dataset to file

    elif [ "$option" = "s" ]; then

        if [ "$verified" != 1 ]; then

            echo "You must first read a dataset from a file"

        else

            read -p "Please input the name of the file to save the processed dataset: " savefile

            cat $datacopy >$savefile

            saved=1

        fi

        # exit

    elif [ "$option" = "e" ]; then

        if [ "$saved" = 1 ]; then

            read -p "Are you sure you want to exit? [y/n] " yesno

            if [ "$yesno" = "y" ]; then

                if [ -e $datacopy ]; then

                    rm $datacopy # delete the copy file

                fi

                exit 1

            fi

        else

            read -p "The processed dataset is not saved. Are you sure you want to exit? [y/n] " yesno

            if [ "$yesno" = "y" ]; then

                if [ -e $datacopy ]; then

                    rm $datacopy # delete the copy file

                fi

                exit 1

            fi

        fi

    else

        echo "Please select one of the options (r, p, l, o, m, s, e)"

    fi

done