CLASSIFICATION FINAL REPORT

Data Mining

PROFESSOR:
Arturo Raymundo

STUDENTS:
Fatima Martínez Torres
Mario Morales Zapata

DATE:
November 16th, 2020

GROUP:
Data 6A

FINAL REPORT

Classification

To begging with this report, it is essential for us to describe a little bit about the process needed in order to extract our data and the mental process we passed to decide on one type of classification.

Dataset

First of all, we must say we decided to work on the Instituto Nacional de Estadística y Geografía (INEGI)'s data on Households and Housing Units, in its subsection called Household Income and Expenditure, because we thought we could extract more information and we discussed the fact that there are hundreds of thousands of houses in the Mexican territory, so we were expecting to have a good amount of information.

Downloading the information was not precisely intuitive.
From the INEGI's Home Page, we tried to download the Tabular Data. Nonetheless, there was no intuitive way to do so. Therefore, we explored the Related Programs tab which leaded us to the National Survey of Household Income and Expenditure (Encuesta Nacional de Ingresos y Gastos de los Hogares ENIGH). This survey has the purpose of providing a statistical overview of the behavior of household income and expenditure in terms of its amount, source and distribution; plus, the information it gives on the occupational and socio-demographic characteristics of the household members, as same as the infrastructure and equipment.

Once we were in this page, we used the Open Data option to download the latest data from 2018 in a CSV format. On the zip file downloaded, we ended up with several folders, each of which had inside (at least) five sub-folders with the needed information to get to know the data stored in those CSV files.

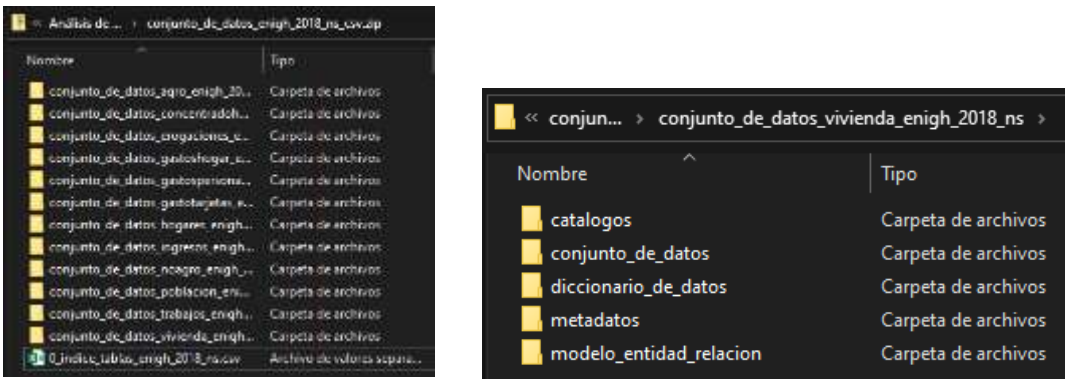

Figure 1 Overview of the data

(You can see what the folders contain in the link)

Even though all of the files in there were house-related, we decided to work just on the one that was called *conjunto_de_datos_viviendas_enigh_2018_ns* which gives us the characteristics of the properties by the household members. That one only focuses on the physical characteristics of the house (type of floor, material, number of rooms, etc.).

That dataset contained more than fifty columns and more than seventy thousand rows, so it was a good one to work with. One of those fifty columns had the information of, so to speak, the socioeconomic status of the house with four different results:

- Bajo
- Medio bajo
- Medio alto
- Alto

Methodology

We decided then to train a classification algorithm to identify (and classify of course) a house given its characteristics into one of these categories.

Now, following the idea of we having to classify an input into a category, a logistic regression was our best option; nonetheless, the output is not binary but polychotomous, so following that constrain, the most accurate model we could possibly use, is a multivariable logistic regression.

Data Preprocessing

We started our data preprocessing by analyzing and looking for NaN and Null values, but we were not able to find any; nonetheless that did not mean that everything in the dataset was worth the shoot of using it into our model. Keeping that in mind, we dropped a lot of columns because the content of those columns did not make sense if you are trying to find the house's socioeconomic tag (let us remind you that this category does not refer to the people living in the house but to the physical characteristics of it, so for example, we did not care if that house has to pay a million dollars on electricity, because even the poorest house can consume that same energy if needed).

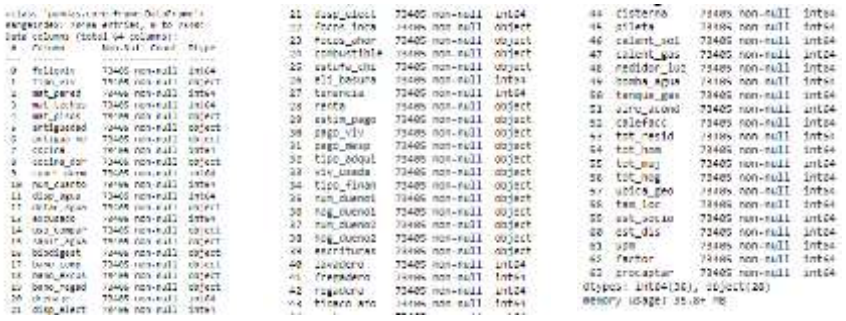

Figure 2 Non-Null values

Once we stayed with just the columns that did make sense for our classification goal, we noticed that all the data was represented numerically, but everything was explained in the data dictionary provided by the INEGI. The thing with this fact, is that they were supposed to represent the information through numbers (check the dictionary of the data here), so most of the columns were type integer (int64), but some others were type object.

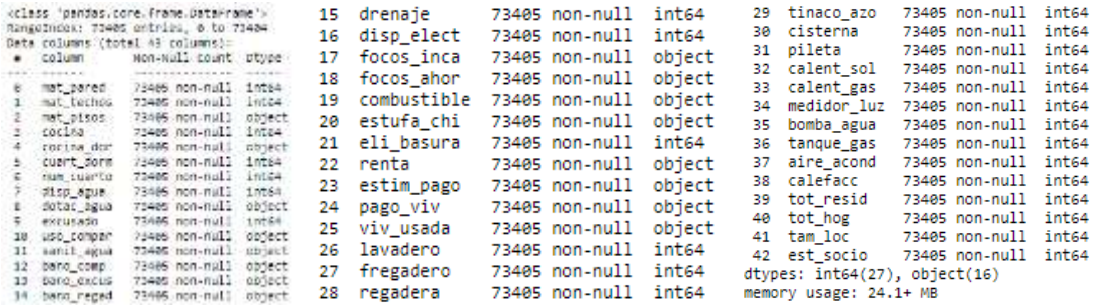

Figure 3 Cleaned

Further analysis on those columns proved them to have errors in the information saved in there (for example, they were supposed to be completely numeric, but some of them also had "&" or empty spaces), giving us the reason why the data type were of the object type. Unfortunately, all rows had at least one mistake in some column, so instead of dropping all of our dataset, we decided to drop the columns that had at least one mistake. This could sound extreme, but let us consider the fact that the treatment required to replacing this errors must be exhaustive, because we would change at least one characteristic on the row in all the rows in our data… As this task is mainly focused on the classification process, we decided not to go deep into that part and just remaining with the columns that were saved correctly.

```
1  # Eliminamos todas las columnas que sean objetos
2  df.drop(objeto, axis=1,inplace=True)
3  df.head()
```

	mat_pared	mat_techos	cocina	cuart_dorm	num_cuarto	disp_agua	excusado	drenaje	disp_
0	8	10	1	4	6	1	1	1	
1	8	10	1	5	6	1	1	1	
2	8	10	1	2	7	1	1	1	
3	8	10	1	2	5	1	1	1	
4	8	10	1	3	4	1	1	1	

5 rows × 27 columns

Figure 4 Table with no object-datatype

Classification system

After all the cleaning, we decided to have a first approach to our classification in order to see if all the data were ready to be classified, or at least strong enough.

Hence, we start with our training data for the Logistic Regression using as parameter to compare the *est_socio* (estatus socioeconómico) column, and since the values were given by integers from 1 to 4, those were the data compared on our chosen Confusion Matrix, so that we can visualize the precision of the predicted data.

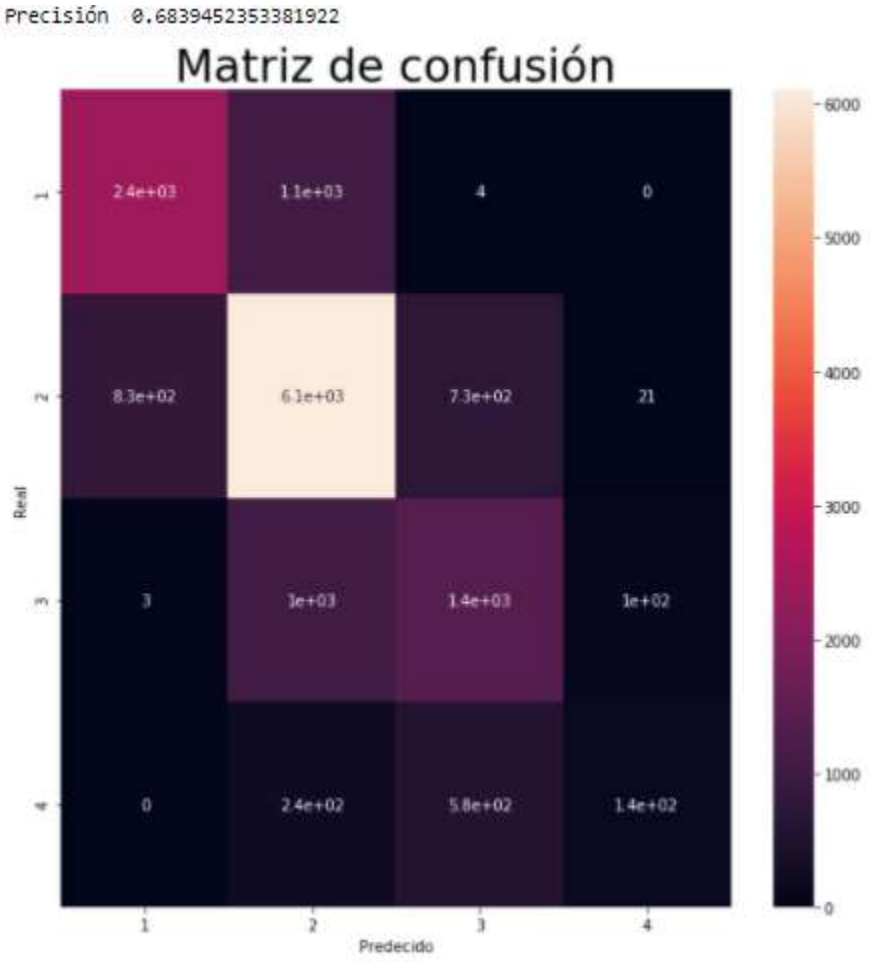

Figure 5 Confusion Matrix

We could notice, then, that the accuracy on our predicted data was not enough since there were thousands of data that showed so.

It is obvious that our model gives very poor predictions; We could not really trust it, therefore, we had to implement some modifications in our analysis in order to improve the accuracy of the model.

Most of the data found in the column represents data of nominal type, for example, in the column *viv_type* represents the type of house on which the information was captured; the possible values within this column are the numeric values from 1 to 5 where:

1. Independent house
2. Apartment in building
3. Neighborhood housing
4. Living in the roof room
5. Local not built for room

Once knowing this information, it is useless for us to think of a logical order for this column just as suggested by its numerical values, we cannot think that 1 < 2 as it simply does not make sense and is a potential threat to our algorithm to apply later.

In that way, we decided to replace the nominal values on some columns, which might have show us important information on our analysis, into their real meaning. So that we could use the data as other type but numerical, since this type does not give a quantity but information.

However, since we got to the idea of working with the table as dummy values, we found out that those columns that shows Boolean data will be unnecessary to change their values of 1 or 2, so we kept those columns as we originally have it.

```
# Creo una lista de todos los datos ordinales. Por cuestiones de la forma en que el INEGI guarda los datos, también
# tengo que incluir en la lista todos los nombres de las columnas que representen información binaria (si, no)
ordinales = ['num_cuarto','cuart_dorm','bano_comp','bano_excus','bano_regad','focos_inca','focos_ahor',
             'renta','estim_pago','pago_viv','tot_resid','tot_hog','cocina','cocina_dor','excusado',
             'uso_compar','estufa_chi','lavadero','fregadero','regadera','tinaco_azo','cisterna','pileta',
             'calent_sol','calent_gas','medidor_luz','bomba_agua','tanque_gas','aire_acond','calefacc','viv_usada']

# Creo una lista con todas las columnas del dataframe
nominales = df.columns.tolist()

# Elimino todos los nombres de columnas nominales de la lista total de columnas
for element in ordinales:
    if element in nominales:
        nominales.remove(element)
```

Figure 6 Omitting Boolean Columns

```
# Reemplazo todos los valores nominales por su significado real
for i in range (len(nominales)):
    csv = 'Data\\'+nominales[i]+'.csv'
    # Acceso al dataframe que contiene el diccionario de los datos de la respectiva columna
    temp = pd.read_csv(csv,encoding = 'latin')
    for index,row in temp.iterrows():
        # Leo la combinacion de valores de cada diccionario de datos
        numero = temp[nominales[i]][index]
        valor = temp['descripcion'][index]
        df[nominales[i]].replace(numero,valor,inplace=True)
```

```
# Dataframe a trabajar
df.head(1)
```

	mat_pared	mat_techos	cocina	cuart_dorm	num_cuarto	disp_agua	excusado	drenaje	disp_elect	eli_basura	...	calent_gas	medidor_luz	bomba_agua
0	Tabique, ladrillo, block, piedra, cantera, cem...	Losa de concreto o viguetas con bovedilla	1	4	6	Agua entubada dentro de la vivienda	1	La red pública	Del servicio público	La traen en un contenedor o depósito	...	1	1	1
1	Tabique, ladrillo, block, piedra, cantera, cem...	Losa de concreto o viguetas con bovedilla	1	5	6	Agua entubada dentro de la vivienda	1	La red pública	Del servicio público	La traen en un contenedor o depósito	...	1	1	1

2 rows × 27 columns

Figure 7 Replacing nominal values

We finally got the data frame as needed to convert our nominal columns to a set of dummy columns, which, indeed, gave us a higher precision on our Confusion Matrix.

```
# Creo un dataframe con todas las características dummy
dummy = pd.get_dummies(df[nominales[:-1]])
#dummy_1 = pd.get_dummies(df[si_no])
# Elimino las columnas "originales" exceptuando columa de resultados. La razón es que crearía redundancia con los dummies
df.drop(nominales[:-1],axis=1,inplace=True)
#df.drop(si_no,axis=1,inplace=True)
# Hago un merge con mi dataframe original
df = df.merge(dummy,left_index=True,right_index=True)
#df = df.merge(dummy_1,left_index=True,right_index=True)
```

```
# Dataframe listo para utilizarse
df.head()
```

	cocina	cuart_dorm	num_cuarto	excusado	lavadero	fregadero	regadera	tinaco_azo	cisterna	pileta	...	eli_basura_La recoge un camión o carrito de basura	eli_basura_La tiran al río, lago o mar	eli_basura_La tiran en barranca o grieta	eli
0	1	4	6	1	1	1	1	1	1	2	...	0	0	0	
1	1	5	6	1	1	1	1	1	1	2	...	0	0	0	
2	1	2	7	1	1	1	1	1	1	2	...	0	0	0	
3	1	2	5	1	1	1	1	1	1	2	...	0	0	0	
4	1	3	4	1	1	1	1	1	1	1	...	0	0	0	

5 rows × 67 columns

Figure 8 Data Frame as Dummy columns

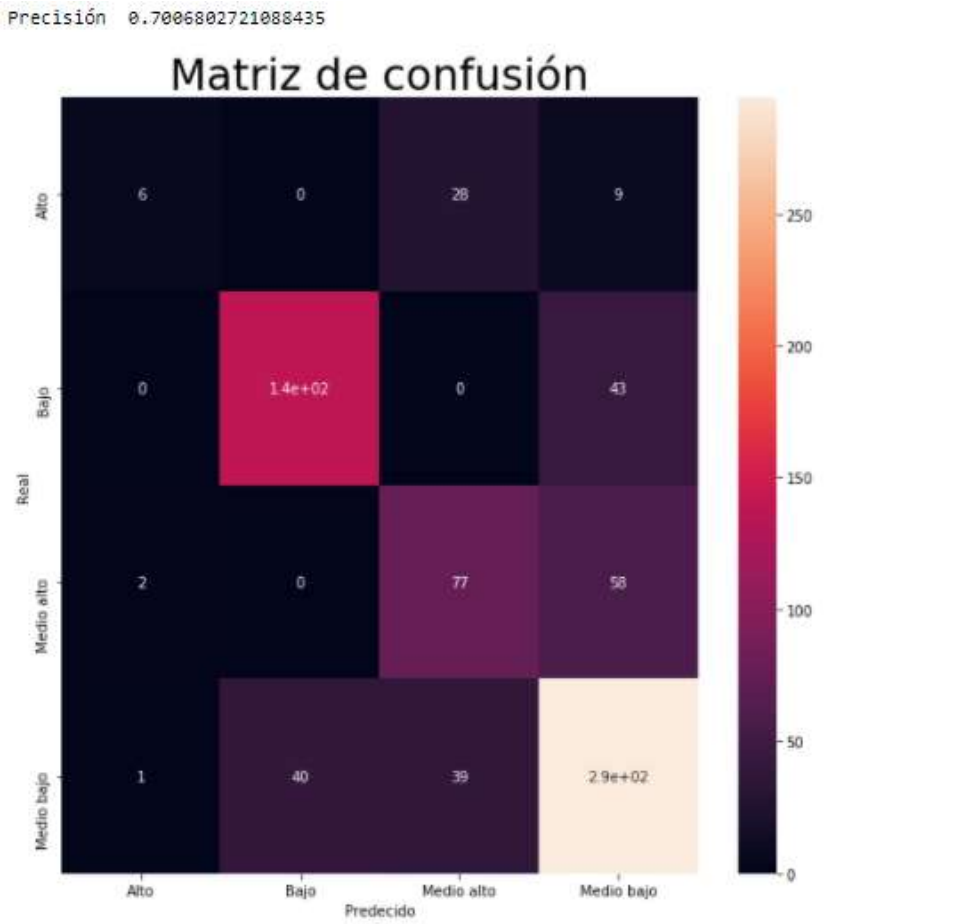

Figure 9 More accurate Confusion Matrix

Conclusion

As a conclusion, we would like to focus on the fact that the accuracy of our model is not even close of being reliable; nonetheless we could indeed find some improvement between our first approach and the second one with the categorical data expressed as it supposed to.

Another thing to take into consideration is the fact that we kept less than half of the original number of columns, so even though our model does not perform the best way it could, it is not that bad given the amount of information we lost. With further treatment in the columns that were not saved in a right way, it is very likely that our model's performance could be really good.

References

INEGI. (2018). *Encuesta Nacional de Ingresos y Gastos de los Hogares (ENIGH). 2018 Nueva serie*. INEI. https://www.inegi.org.mx/programas/enigh/nc/2018/#Documentacion

Martínez Torres, F., & Morales Zapata, M. A. (2020). *DataMining*. GitHub. https://github.com/fatimamt/Data-Mining