Predictor from scratch report

Canul Cal, Jorge Carlos
Computational Robotics Engineering
Universidad Politécnica de Yucatán
Km. 4.5. Carretera Mérida — Tetiz
Tablaje Catastral 7193. CP 97357
Ucú, Yucatán. México
Email: 2009022@upy.edu.mx

Professor's Name
Victor Alejandro Ortiz Santiago
Universidad Politécnica de Yucatán
Km. 4.5. Carretera Mérida — Tetiz
Tablaje Catastral 7193. CP 97357
Ucú, Yucatán. México
Email: victor.ortiz@upy.edu.mx

Index Terms

Linear Regression Model, Machine Learning, Prediction.

Predictor from scratch report

I. Introduction

Machine learning is a topic that daily evolves, changing the perception of the world with automated processes yet it is important to consider the pre-process before start predicting possible outcomes in multiple fields such as meteorology, trending topics, fashion, income, etc., to do so, it is important to have a comprehensive dataset at disposal and most of the times it is not that way, making the user to manipulate the data to have something interpretable for the machine, or another possibility is that there are missing values, outliers, or feature selection. Finally, it is compared a predictor model from scratch and another model using the same algorithm but implementting libraries, optimazing the model in the precess.

II. Problem definition

The very first step before diving into the data analysis found in the *Indicadores de pobreza, pobreza por ingresos, rezago social y gini a nivel municipal* is to select one feature of the columns and rearrange it to transform it into the target desired to be predicted by either model according to the needs. In other words it is determined the name of the feature, type of problem to be solved, and question that is answered once completed the analysis at the end. To visualize it, check Fig. 1

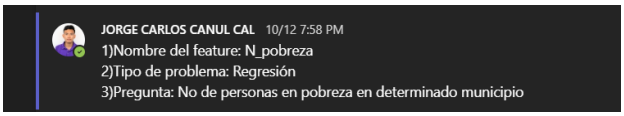

Fig. 1. Initial problem statement.

Despite it is not the complete screen, due to respect privacy and the problem statements of the rest of the class, the chooses are not shown.

III. Development

Machine learning is the computer's ability to make predictions using past experiences in different areas such as meteorology or products trending. The development of the model depends on the storage capacity and processing power of computers, meaning that it is highly recommended to have at hand specialized computers when working with immense datasets [1].

A. Dataset analysis

However, the problem presented to solve is not complicated compared to what could be seen in the industry but represents a challenge at the academic level, demonstrating skills and abilities acquired to manipulate and analyze data as preprocessing before performing the prediction as asked.

With that being said, and as a chronological document explaining all the process performed along the project is the following. First is to select the dataset to work with, which this time was predetermined to be information gathered by the *Instituto Nacional de Estadística y Geografía* (INEGI), which contains data of the Mexican republic segmented by states and then by municipalities determining the indicators of poverty principally in the years 1990, 2000, 2005, and 2010. Essentially the very first part enter the dataset and start scanning the features in order to select only one feature and transform it into the prediction target. Yet, the idea behind this is not only to define the characteristics to work with but also to identify the type of problem to be solved which can be either classification problem or linear regression problem giving an output in categories or numerical data. Once determined which problem problem could be solved in each feature was selected one among the whole dataset, in this case is 'N_pobreza' which faces a linear regression problem as can be seen in the figure below by looking at only the first five observations.

W	X	Y	Z	AA
rankin_p	rankin_pe	N_pobreza	N_pobreza_e	N_pobreza_m
2351	2385	242510	17987	224523
1340	1915	32611	3907	28704
1601	1989	32585	3845	28740
1930	2196	7445	672	6773
2108	2072	45964	6168	39796
2083	2255	20560	1813	18746
1824	1839	28789	4699	24090
1359	1978	5214	574	4639
1712	2079	12921	1329	11592

Fig. 2. Feature highlighted, denoting linear regression problem.

The definition of linear regression problem at this point is strictly important to set since it will represent the core of the work during the whole process. Finally, the question to be answered is *Number of people in condition of poverty in determined municipality* which is not only determining specifically one municipality but can be any depending on the user's need or even the client, when talking about a proper professional project condition. In simple words, the beginning was to quickly analyze the dataset, determine the feature to work with, identify the problem, and the question to be answered.

B. Data preparation

The analysis of the dataset dropped that there are null values, or empty cells but instead of working directly with the .csv file it was used the Google Colab platform that allows the implementation of libraries and accelerate the project performance, such as determining the number of cells

with null values or to pinpoint, in this case, the cells with categorical data and rearrange those values to convert them into numerical data so it is possible to work with a linear regression problem, as happen with the cells 'gdo_rezsoc00' 'gdo_rezsoc05' 'gdo_rezsoc10' as shown below.

```
categorical_columns = ['gdo_rezsoc00', 'gdo_rezsoc05', 'gdo_rezsoc10']
df_columns = df[categorical_columns]
print(df_columns)
```

```
    gdo_rezsoc00 gdo_rezsoc05 gdo_rezsoc10
ent
1       Muy bajo     Muy bajo     Muy bajo
1       Muy bajo     Muy bajo     Muy bajo
1       Muy bajo     Muy bajo     Muy bajo
1       Muy bajo     Muy bajo     Muy bajo
1       Muy bajo     Muy bajo     Muy bajo
..           ...          ...          ...
32         Medio        Bajo         Bajo
32      Muy bajo     Muy bajo     Muy bajo
32      Muy bajo     Muy bajo     Muy bajo
32          Bajo         Bajo     Muy bajo
32           NaN         Bajo     Muy bajo
```

Fig. 3. Categorical data in the dataset.

As can be observed, these values are not useful in that form right now so that are changed with functions so that can be understood by the model in the future. Once everything is set and analyzed, the next step is to fill the null values by using average value by state avoiding possible outliers or discrepancies when training the model in the future. To do so, and with the help of stack overflow [2], three are two main functions that essentially do the same but one for numerical data and the other one for categorical data. Fro example, the function *fill_na_with_group_mean* is designed to fill values with numeric data by replacing missing values with the mean of the group while the numeric_only=True ensures that only numeric columns are considered. On the other hand, *fill_na_non_numeric_with_group_mean* does exactly the same but for values with non-numeric columns by looping through the columns of the group and checks if a column's data type is numeric, then calculates the mean of that column and fills the missing values.

As stated before, the non-numerical data columns are 'gdo_rezsoc00' 'gdo_rezsoc05' 'gdo_rezsoc10', so by using pd.get_dummies() the categorical data are converted into a format suitable for the model. Finally, to prove that the functions work correctly are printed the total number of cells with null values.

However, since the dataset contains lots of informations than could be irrelevant for the predicting model, it is recommended to perform a correlation matrix that help visualize how the characteristics are related in between. For that, this is a general matrix correlation using a heat map.

The image was highly scaled to observe directly the heat map and how correlated are the values, yet it is impossible to see the numbers with a big size image with small letters. For that reason, and trying to be more specific with the target feature it is created another correlation image but considering 'N_pobreza', the highlighted feature to work with, giving the following picture,

```
def fill_na_with_group_mean(group): # numeric
    return group.fillna(group.mean(numeric_only=True))

def fill_na_non_numeric_with_group_mean(group): # non-numeric
    for col in group.columns:
        if pd.api.types.is_numeric_dtype(group[col].dtype):
            group_mean = group[col].mean()
            group[col] = group[col].fillna(group_mean)
    return group


grouped = df.groupby('nom_ent',group_keys=True)
df = grouped.apply(fill_na_with_group_mean) # Apply the function to the grouped da


df = pd.get_dummies(df, columns=['gdo_rezsoc00','gdo_rezsoc05', 'gdo_rezsoc10']) #
grouped = df.groupby(['ent'],group_keys=True)
df = grouped.apply(fill_na_non_numeric_with_group_mean)

print("Total cells with null values:", df.isnull().sum().sum()) # #total cells wit
```

```
Total cells with null values: 0
```

Fig. 4. Data processing for further predictor model.

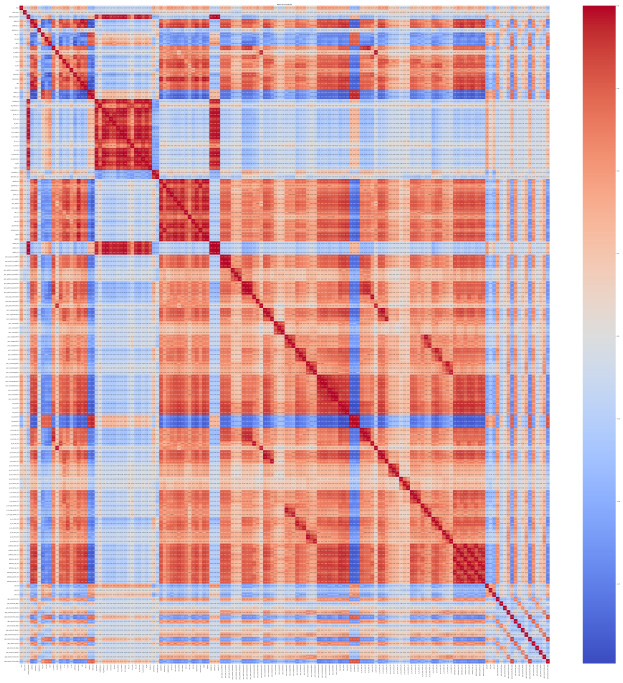

Fig. 5. Heat map, correlation matrix

The code part will be further explained in the proper section. Yet, it is possible to find anomalies within the features, so to avoid the any incongruence when realizing the predicting model it is designed one more correlation figure but using a bar graph, that somehow it is more easy to read instead of a heat map when using limited characteristic. In this case, the related variables.

However, this step can be skipped but to be sure that the dataset has not been corrupted or missed any value it is done. Once the process is done, it is recommended to identify if there are any outlier in the current dataset. An outlier is an observation that lies an abnormal distance from other values in a random sample from a population [3]. So those values can lead to wrong prediction when analyzing and performing the model ending up with errors, which is something to be

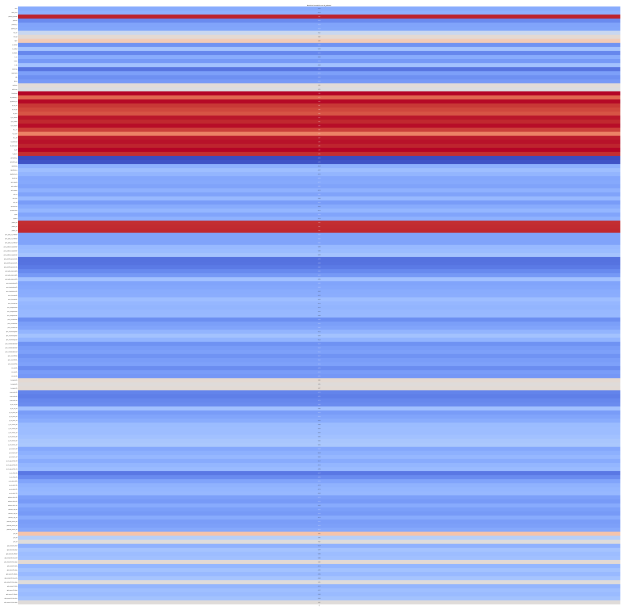

Fig. 6. Correlation image regarding N_pobreza.

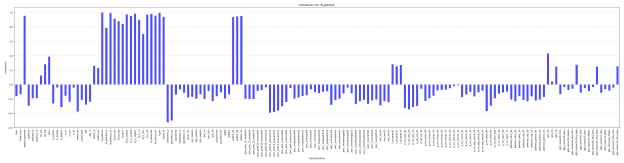

Fig. 7. Correlation bar graph regarding N_pobreza.

avoided if want to be applied the model in any field, even though it is an academical project. With that being said, it was decided to eliminate those outliers.

The mentioned outliers are decided to be dropped from the data because:

- The numbers are way far from the average as seen in between.
- It is already taken out irrelevant data to train the model. In other words, the data that is not highly correlated to the target feature.
- The outcome and predictions will be easier to visualize with relevant data.
- Noise will be reduced leading to a more stable and interpretable model.

With the reasons explained, it is presented the code fragment that focuses on removing

```
numeric_columns = df.select_dtypes(include=np.number).columns

# Parameter inputation
Q1 = df[numeric_columns].quantile(0.25)
Q3 = df[numeric_columns].quantile(0.75)
IQR = Q3 - Q1

# Using the parameters, determine the outliers
outliers = (df[numeric_columns] < (Q1 - 1.5 * IQR)) | (df[numeric_columns] > (Q3 + 1.5 * IQR))

# Remove outliers
df = df[~outliers.any(axis=1)]
```

Fig. 8. Outliers elimination process.

The method used to remove outliers is based on the Interquartile Range method by segmenting the columns that contains only numeric data, so by looking at the formula that creates a boolean DataFraame where each cell corresponds to whether a value is an outlier or not by comparing each numeric value to the lower bound and the upper one (defined as Q1 and Q3), so the when the value is lower or greater than the bounds will be marked as outliers and removed from the dataset.

The IQR method is commonly used to identify and remove outliers because it is robust and less sensitive to extreme values compared to other methods. It is based on the spread of the middle 50% of the data, making it less influenced by extreme values in the dataset.

All this process correspond to prepare the dataset by setting all the data into numerical values eliminating, for example the very first columns that show the multiple states and municipalities in the Mexican republic which are irrelevant for the model, so by performing this processing it is highly optimized by reducing the dataset to only relevant variables defined by the correlation matrix resulting in a prediction strictly for the target selected.

C. Predictor model preparation

To begin with, the model decided to work with was the Ordinary Least Squares (OLS) method which in simple words calculates unknown parameters in a linear regression model minimizing the sum of squared vertical distances between the observed responses in the dataset [4]. The objective of the model is predicting a continuous target variable that is N_pobreza, deliberately chosen by the user to understand better the usage of the different multiple predictors in machine learning.

To do so, and considering all the pre-process and analysis before making the model it is divided into two characteristics X and y, which X corresponds to the correlated variables to N_pobreza, the target to be predicted and by looking at the multiple images done before, there are only 18 different features that help predict the target. Then, in the figure below can be seen how the model is divided for the training and test parts which is simply done by splitting the dataset into 80% training and 20% for testing. Following the sequence of the code, the next step decided is that the X_train variable adds a column of ones to the training data for the intercept term. In other words, perform operations in matrix form to compute predictions and coefficients efficiently, then the model makes predictions with the rest of the data, with the testing data.

```
X = df[['pobtot_ajustada','N_pobreza_e', 'N_pobreza_m',
y = df['N_pobreza'].values

# Traning and testing division
split_ratio = 0.8
split_index = int(len(X) * split_ratio)
X_train, X_test = X[:split_index], X[split_index:]
y_train, y_test = y[:split_index], y[split_index:]
```

Fig. 9. Model division training.

The next step to develop the algorithm is to train the model using the Ordinary Least Squares method that calculates the coefficients that minimizes the sum of squared differences between predicted and actual values. After that the model tries to make predictions using the testing data once it is trained properly with the very first part of the data set and to do so, the OLS method calculates the Mean Square Error measuring the average squared difference between predicted and actual values in the testing set. In simple words, when the error closes up to zero it indicates that the performance is great and the value will be close to the reference line indicating the actual values.

```
# To add a column of ones for the bias term.
X_train = np.c_[np.ones(X_train.shape[0]), X_train]

# Calculate coefficients using OLS (Ordinary Least Squares)
coefficients = np.linalg.inv(X_train.T.dot(X_train)).dot(X_train.T).dot(y_train)

# Predictions using the testing values
X_test = np.c_[np.ones(X_test.shape[0]), X_test]
y_pred = X_test.dot(coefficients)

# Calculate Mean Square Error (MSE)
mse = np.mean((y_pred - y_test) ** 2)
```

Fig. 10. Model algorithm parameters.

Now, regarding to build the same model but using libraries implemented on Colab environment will demonstrate the difference between the predictions made from scratch and the ones performed by an optimized linear regression model. To do so, the algorithm must be the same which is Ordinary Least Squares method that draws a reference line and put dots according to the Mean Square Error. In simple words, the steps taken to train the model is exactly the same as performed for the scratch version, meaning that the code does not present any significant change but the libraries that perform all the mathematics within the library.

```
# Data division
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Linear Regresion Model
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Mean Square Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Coefficient of Determination (R^2)
r2 = r2_score(y_test, y_pred)

print('Mean Square Error:', mse)
print('Coefficient of Determination (R^2):', r2)
```

Fig. 11. Optimized model implementation.

IV. Results

Finally, the results of both models are practically identical with a insignificant difference among the decimals and this outcome is due to the variables selected to perform the training and predictions and to make it clear, the accuracy is presented in the following figures.

As the title of the figure suggests, the results are shown from 0 to 1, and the accuracy is almost perfect but it does not, indicating that the group of variables put together define the total number of people in poverty situation.

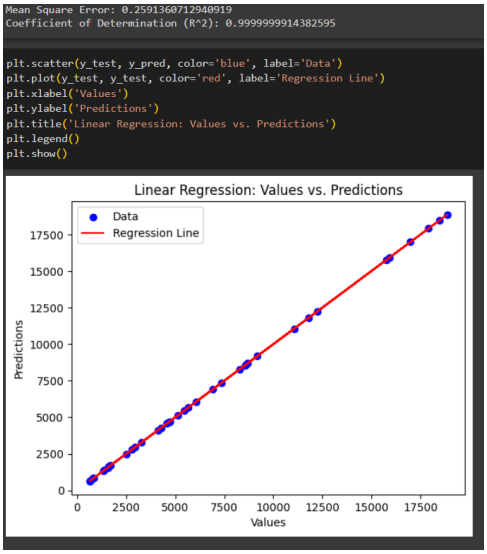

Fig. 12. Results, algorithm from scratch.

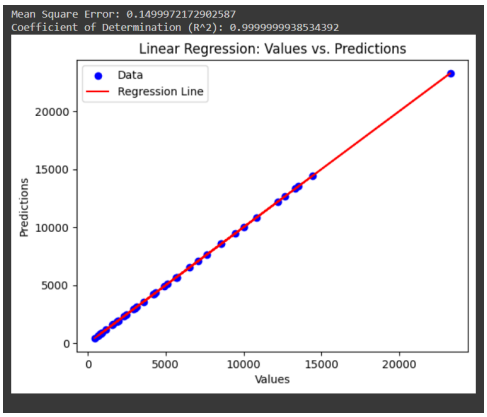

Fig. 13. Results, algorithm using libraries.

On the other hand, the optimized model increases a bit the accuracy but it is practically meaningless since the only changes happens in the seventh decimal. However, it demonstrates that it the scratch model could have been a little bit more optimized.

V. Github link

https://shorturl.at/djqrN

That URL corresponds to the repository where it is allocated the predictor model. However, to reduce the amount of letters, it is presented a shorten link but redirecting to the same place.

VI. Conclusion

Probably the most challenging part of the project was to implement a linear regression model that could fit with the

requirements and expected outcome for the predictions despite that the algorithm used it is absolutely different to what was seen in class, so the part of research and understanding the mathematics behind the model was significantly challenging but that is not all since manipulating huge amount of data resulted unknown for the very first time. In other words, dealing with missing values, outliers, and selecting the most relevant features was a time-consuming task without mentioning the research behind the process to understand the optimal methodology to be followed. Yet, the outliers represented a topic to be considered since those anomalies in the dataset may affect the performance and outcome of the predictions and a decision had to be made, remove the outliers, impute missing values, or transform the data to obtain standardized values.

On the other hand, these kind of projects can be applied to robotics since robots normally generate large volumes of sensor data and the pre-process might help create optimized models or even calibrate sensors when working with robots. But also linear regression can help in object recognition and localization by learning the relationships between sensor inputs and objects positions. Finally, other possible application is the so desired human-robot interaction possible by understanding the behavior of the model so recognizing and predicting human gestures and intentions as happens with artificial vision.

REFERENCES

[1] Mustafa Özuysal Yalin Baştanlar. Introduction to machine learning. *Springer Link*, 1107(1):105, 2013.

[2] Stack overflow. Pandas: filling missing values by mean in each group. https://stackoverflow.com/questions/19966018/pandas-filling-missing-values-by-mean-in-each-group, 2014. Online; Last accessed 20 October 2023.

[3] National Institute of Standards and Technology. What are outliers in the data? https://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm, n. d. Online; Last accessed 20 October 2023.

[4] David Anderson Burnham, Kenneth P. *Model Selection and Multi-Model Inference*. Springer, ISBN 0-387-95364-7., 2nd edition edition, 2002.