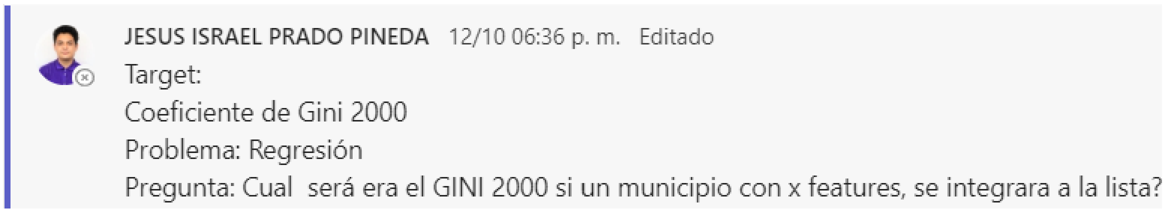
U2 - Implementing a Predictor from scratch

MACHINE LEARNING

JESUS ISRAEL PRADO PINEDA
UNIVERSIDAD POLITECNICA DE YUCATAN | 9 B ROBOTICS

A screenshot of the post in the general channel, where you state your Selected target, your selected problem (regression or classification), and the question that the predictor will answer.



JESUS ISRAEL PRADO PINEDA 12/10 06:36 p. m. Editado

Target:

Coeficiente de Gini 2000

Problema: Regresión

Pregunta: Cual será era el GINI 2000 si un municipio con x features, se integrara a la lista?

All the steps taken to get your dataset ready (including the rational over why to keep or drop features or observations).

First, it must be downloaded from the dataset on the website, in this case it was downloaded directly from [https://datos.gob.mx/busca/dataset/indicadores-de-pobreza-pobreza-por-ingresos-rezago-social-y- gini-at-municipal-level1990-200-2010](https://datos.gob.mx/busca/dataset/indicadores-de-pobreza-pobreza-por-ingresos-rezago-social-y-gini-at-municipal-level1990-200-2010)

Once the dataset was downloaded, with Pandas, we can manipulate the dataset, according to our convenience, The ISO decoder must be reviewed, in this case, a couple of lines must be written to be able to define that decoder, and open it based on it, to be able to load the dataset, in this case, the first thing we did was delete the columns that do not add value to my problem, such as Name of municipality, Municipality ID, State name (since it is already categorized in the same dataset), Municipality number

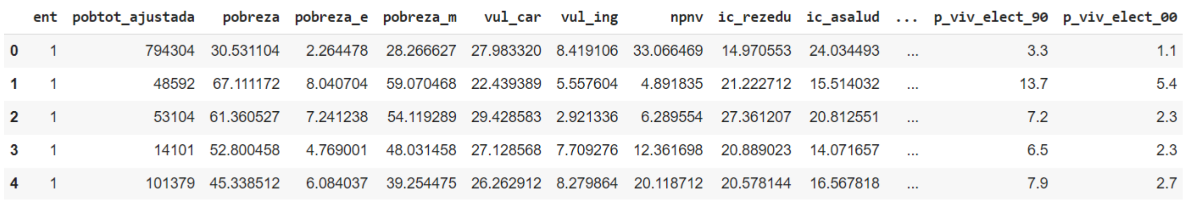
Subsequently, the objective is to calculate the GNI for the year 2000, therefore, we will not need the data obtained in 2005 and 2010 respectively, only those from 2000 as well as the data from 1990, and here the elimination is justified in more detail. of those features:

Year of interest: The main objective is to calculate the GNI for the year 2000. This suggests that you are working on a project or analysis focused on that particular year, and information from later years such as 2005 and 2010 is not needed, since it is not are relevant to that specific objective.

Data closest to the year of interest: Data from 2000 are the closest to the target year and, therefore, are the most relevant for calculating the GNI in that period. Data for 2005 and 2010 may be subject to significant changes and variations that may not accurately reflect the economic situation in 2000.

Historical background: Data from 1990 is important as historical background. This data can provide context and a historical baseline to better understand economic developments from 1990 to 2000. This is essential to analyze how the economy has changed and developed over the decade.

From there on out, the other features are necessary to be able to perform the data analysis.



	ent	pobtot_ajustada	pobreza	pobreza_e	pobreza_m	vul_car	vul_ing	npnv	ic_rezedu	ic_asalud	...	p_viv_elect_90	p_viv_elect_00
0	1	794304	30.531104	2.264478	28.266627	27.983320	8.419106	33.066469	14.970553	24.034493	...	3.3	1.1
1	1	48592	67.111172	8.040704	59.070468	22.439389	5.557604	4.891835	21.222712	15.514032	...	13.7	5.4
2	1	53104	61.360527	7.241238	54.119289	29.428583	2.921336	6.289554	27.361207	20.812551	...	7.2	2.3
3	1	14101	52.800458	4.769001	48.031458	27.128568	7.709276	12.361698	20.889023	14.071657	...	6.5	2.3
4	1	101379	45.338512	6.084037	39.254475	26.262912	8.279864	20.118712	20.578144	16.567818	...	7.9	2.7

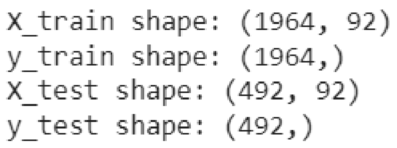
The image that was placed above is an image that allows us to see how the data was manipulated in a certain way, obtaining that result, which can also be seen when running the notebook and loading the csv file (once the data has been manipulated , you can download it from Google Collab and verify that it is truly what we need), in addition, the categorical variables had to be converted into numeric ones, and in addition, there were blank data within the excel, which, if left like this, there would be problems When training and predicting, to do so, it had to be filled in using a certain function, which allows taking the average of a feature, and thereby filling in the missing data.

Code for change the categorical variables to numeric ones	Code for fill the missing data	Code for erase columns

`df['gdo_rezsoc00'], _ = pd.factorize(df['gdo_rezsoc00'])`	`df.fillna(df.mean(), inplace=True)`	`columns_to_drop = ['nom_mun']`
		`df = df.drop(columns=columns_to_drop)`

All steps taken to train your predictor. Including an explanation on Why to choose such algorithm and an explanation on its characteristics and All steps taken to evaluate the performance of your predictor (using your test - sub dataset)

The first step before doing anything, was to first divide the dataset into X_train, X_test, y_train and y_test, and partition the dataset. in the famous "80/20", to later be able to work



```
X_train shape: (1964, 92)
y_train shape: (1964,)
X_test shape: (492, 92)
y_test shape: (492,)
```

Then, It takes a list of numbers and another number called "degree". Then, create a new list of numbers by multiplying the numbers in the original list with each other, as many times as indicated by the degree. This is done in order to better understand the relationships between the original numbers and make certain calculations more precise.

The degree allows to adjust the complexity of the model to the data set and the relationships you want to model. However, you should be careful not to increase the degree too much, as it can lead to overfitting of the model to the training data and worsen its ability to generalize to new data.

```
degree = 2
```

It then normalizes the polynomial features in both the training set and the test set, which is a common practice in preprocessing data before training a machine learning model.

Subsequently, a regularized linear regression model is trained using ridge regularization to prevent overfitting and improve the generalization ability of the model.

Adjusting the value of "lambda" allows to find a balance between properly fitting the training data and preventing overfitting. The choice of "lambda" depends on the nature of the data and the problem at hand, and is often done using cross-validation techniques to find the optimal value that maximizes model performance on unseen data.

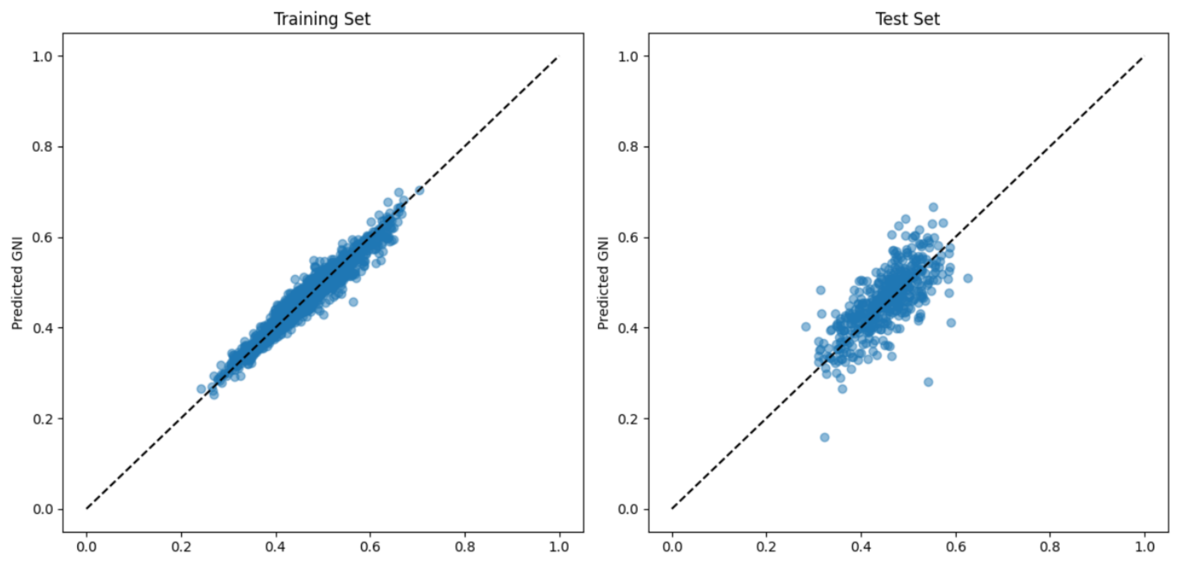
```
lambda_ = 1
```

Almost finally, a function is created to quantitatively evaluate how well a model's predictions fit the true values, by calculating the average of the squared errors between the predictions and the true values. A lower MSE value indicates a better fit of the model to the data.

Finally, It takes a set of features and a set of coefficients from the linear regression model and calculate the model predictions for the given input features. It is useful for obtaining estimates based on a previously trained model.

```
Train MSE with regularization: 0.000274544239438095
Test MSE with regularization: 0.0023769761732889555
```

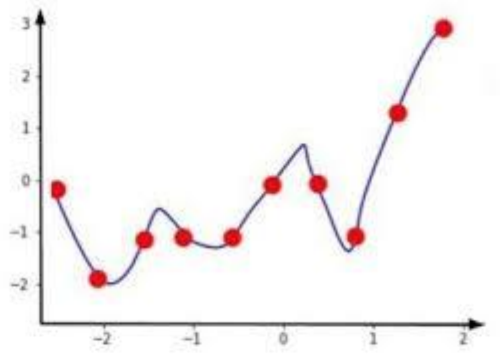
Additionally, a graph was created with two subgraphs that show the comparison between the actual values and the model predictions, both in the training set and in the test set, allowing the performance of the model to be visually evaluated.



The choice of this algorithm is considered the optimal option for the prediction of the Gini Index (GNI) in the year 2000, using 92 characteristics

Linear Regression: Linear regression, a widely used machine learning model for regression problems, is used.

Regularization: Linear regression is performed with L2 regularization (known as Ridge). Regularization helps prevent model overfitting by penalizing large coefficients.



Polynomial Characteristics: A process for generating second-degree polynomial characteristics has been applied from the original characteristics. This can help the model capture non-linear relationships between the features and the target variable.

Normalization: Normalization of polynomial features has been performed, a fundamental procedure to ensure that all features have the same scale.

Performance Evaluation: The performance evaluation of the model has been carried out by calculating the mean square error (MSE) on the training and test sets.

In order to precede something, first we load the data from a csv file, then the Y column is ignored, and if it does not have Y, we proceed, and finally, the prediction is printed

Predicted Data	Real data
`Predicciones: [0.42522934]`	gini_00 0.445

All steps taken to train your predictor using libraries (sklearn?). And a comparison of results

Also, the first step before doing anything, was to first divide the dataset into X_train, X_test, y_train and y_test, and partition the dataset. in the famous "80/20", to later be able to work

Subsequently, the scikit-learn library is used to scale training and testing data sets using the standard scaling technique. This means that it fit a scaling model to the training data and then apply the same transformation to the test data.

Afterwards, a neural network model is built with three layers: an input layer with 64 neurons, a hidden layer with 32 neurons and an output layer with 1 neuron. The model will be used to solve a machine learning problem, where the input is expected to be pre-scaled and the model makes predictions based on these layers.

```
model = tf.keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    layers.Dense(32, activation='relu'),
    layers.Dense(1)
```

The neural network model is then configured to use the Adam optimizer and the mean square error (MSE) loss function during the training process. These are key aspects for effective training of regression models.

In addition, the neural network model is trained using scaled training and validation data. It is run for 50 epochs, using batches of 16 examples at a time, and is evaluated on a separate validation set to monitor its performance.

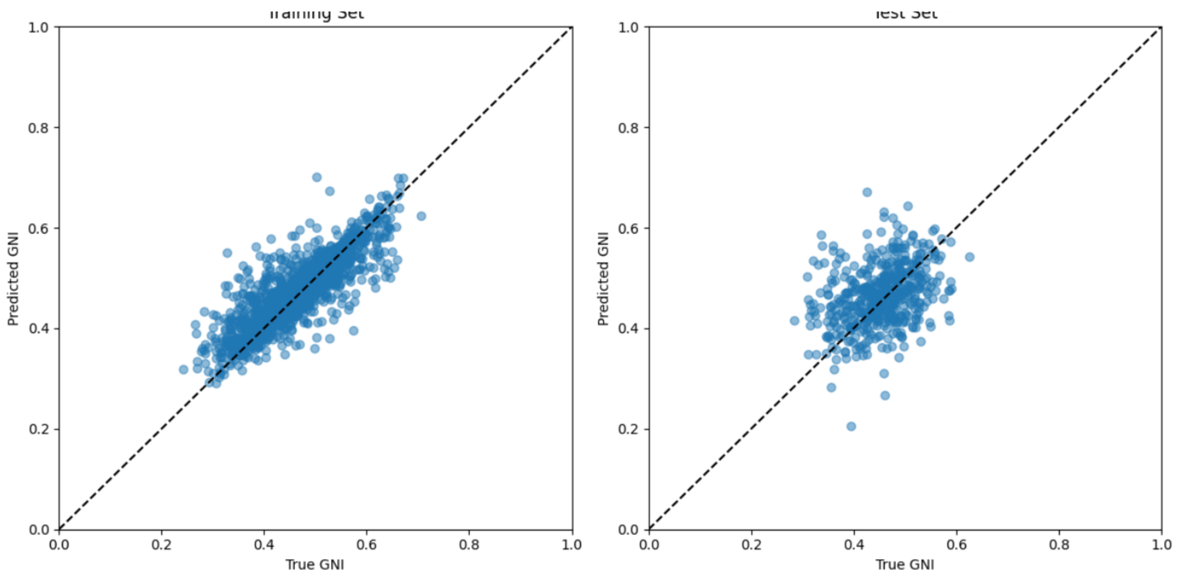
```
Epoch 49/50
99/99 [==============================] - 0s 3ms/step - loss: 0.0023 - val_loss: 0.0100
Epoch 50/50
99/99 [==============================] - 0s 3ms/step - loss: 0.0013 - val_loss: 0.0076
```

Finally, the model's loss on the test data is calculated and displayed, providing a measure of its performance on data it has not seen during training.

```
Test Loss: 0.008677328936755657
```

A Test Loss of 0.008677328936755657 indicates that the model is making very accurate predictions on the test data set and that its generalization ability is strong for this problem.

Additionally, a graph was created with two subgraphs that show the comparison between the actual values and the model predictions, both in the training set and in the test set, allowing the performance of the model to be visually evaluated.

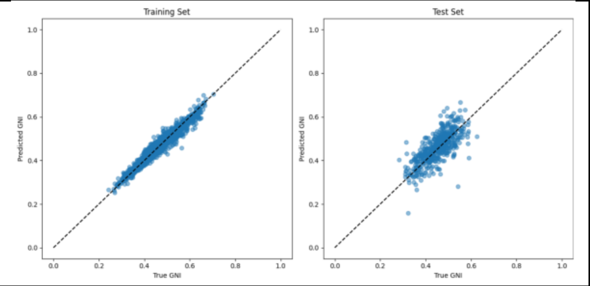
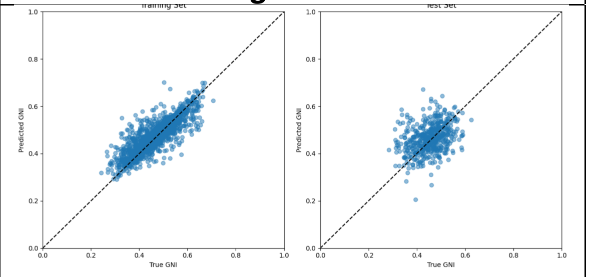


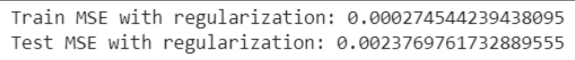
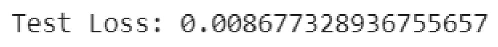
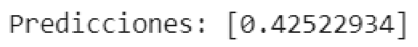
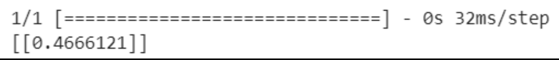
In order to precede something, first we load the data from a csv file, then the Y column is ignored, and if it does not have Y, we proceed, and finally, the prediction is printed

Predicted Data	Real data
`1/1 [==============================] - 0s 32ms/step` `[[0.4666121]]`	gini_00 0.445

Comparison

We can compare the results in 2 ways: using the graphs provided above, and the GNI predictions of a municipality, as well as the displacement values obtained throughout the work, then the results will be compared

Not using libraries	***Using libraries***
	

Here It can see that when trained with the NumPy model, it is seen that the lines that cross between the predictions and reality fit too well, likewise, with the tests that are obviously new data, they fit in a fairly good way.	Although creating a neural network is quite simple, it requires some computing power to do so, however, having the test and train data, the predictions are a little more complicated to make for the network, although If we had more data and could train more, it would be better
 `Train MSE with regularization: 0.000274544239438095` `Test MSE with regularization: 0.0023769761732889555`	 `Test Loss: 0.008677328936755657`
The margin of error when providing data is between 2% and 3%	The margin of error when providing data is between 8% and 9%
 `Predicciones: [0.42522934]`	 `1/1 [==============================] - 0s 32ms/step` `[[0.4666121]]`
Taking the real data to be 0.445, an error of 4% was obtained.	Taking the real data to be 0.445, an error of 5% was obtained.

A link to a GitHub project where you publish your code.

https://github.com/JesusIsraelPradoPineda/U2---Implementing-a-Predictor-from-scratch.git

A reflexion over what were your main challenges during this project, how do you see this could be applied to the field of Robotics.

One of the biggest challenges of the project was the development of a learning algorithm without the use of training libraries. While attempting this, numerous failures were experienced, leading to the need for extensive internet research to resolve the issue and obtain the desired results.

The regression model applied to robotics can be used for various applications, such as object detection by robots or decision making in response to specific events. This is particularly relevant in the context of mobile robotics and advanced surveillance, among others.

In relation to the regression model based on the GNI 2000, in the field of robotics, it could be used to identify the areas of greatest vulnerability in Mexico. This would allow the targeting of support resources according to the GNI of the corresponding populations. To achieve this, drones could be used that, by taking a simple photograph of the region, can predict the GNI and provide the necessary support. Furthermore, considering that new municipalities are created daily in the states of Oaxaca, Guerrero and Chiapas, this tool would help in the documentation of the newly created municipalities and in the evaluation of their needs in comparison with others. It should be noted that the situation of a municipality like Texoloc is very different from that of the capital of Nuevo León.