Implementing a Predictor from scratch

Brandon Isaac Pacheco Chan
Universidad Politécnica de Yucatán
Km. 4.5. Carretera Mérida — Tetiz
Tablaje Catastral 7193. CP 97357
Ucú, Yucatán. México
Email:2009102@upy.edu.mx

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

Abstract

This report presents a predictor for calculating the percentage of people who lack access to health services in the 2000s using a linear regression model. The predictor was implemented from scratch and trained on a modified data set from INEGI. The dataset was modified to focus on one target, which indicates whether a person has a lack of access to health services. The main objective of this report is to help students understand how to modify a dataset to manipulate only one target, how to implement a linear regression model from scratch, and how to compare two linear regression models, one implemented from scratch and the other using libraries like Scikit-learn. The predictor is able to accurately predict the percentage of people who lack access to health services in the 2000's. The results show that the percentage of people who lack access to health services varies depending on the state, with some states having a higher percentage than others. Linear regression models can be applied in robotics in a variety of ways. For example, they can be used to predict the movement of a robot arm, to classify objects in a scene, or to detect anomalies in sensor data.

Index Terms

Artificial intelligence, machine learning, linear regression, predictor, lack of health services, dataset modification, model comparison, robotics, Scikit-learn

Implementing a Predictor from scratch

I. Introduction

Access to health services is a fundamental human right and is essential for achieving good health and well-being. However, not everyone has equal access to health services. In Mexico, the percentage of people who lack access to health services varies depending on the state, with some states having a higher percentage than others.

Understanding the factors that influence access to health services is essential for developing effective policies and programs to improve access. Linear regression models can be used to identify the factors that are most important for predicting access to health services. This information can then be used to develop targeted interventions to improve access for those who need it most.

Linear regression is a statistical method that is used to model the relationship between a dependent variable and one or more independent variables. In this report, linear regression is used to model the relationship between the percentage of people who lack access to health services (the dependent variable) and a set of independent variables that are thought to influence access to health services.

We will implement a linear regression model from scratch and train it on a modified dataset from INEGI. The dataset was modified to focus on the target variable p-ser-sal-00, which indicates whether a person has access to health services. Also, compare the model to an other model implemented using the scikit-learn library.

Finally, Will discuss how linear regression models can be applied in robotics. Linear regression models can be used to solve a variety of problems in robotics, such as predicting the movement of a robot arm, classifying objects in a scene, or detecting anomalies in sensor data.

With the hope that this report will help people understand how to use linear regression models to solve real-world problems, both in the field of health services and in the field of robotics.

II. State of art

Linear regression is a statistical method that is used to model the relationship between a dependent variable and one or more independent variables. The dependent variable is the variable that we are trying to predict, and the independent variables are the variables that we think can be used to predict the dependent variable.

Linear regression is a very powerful tool that can be used to solve a wide variety of problems. For example, linear regression can be used to predict the price of a house, the number of customers who will visit a store on a given day, or the risk of a patient developing a certain disease.

In recent years, there has been a lot of research on new and improved methods for linear regression. Some of the most important advances in linear regression include:

Sparse linear regression: Sparse linear regression is a technique that is used to identify the most important features in a dataset and to remove the less important features. This can lead to more accurate and efficient linear regression models. Regularized linear regression: Regularized linear regression is a technique that is used to prevent overfitting. Overfitting occurs when a model learns the training data too well and is unable to generalize to new data. Regularized linear regression helps to prevent overfitting by penalizing complex models. Kernel-based linear regression: Kernel-based linear regression is a technique that is used to model non-linear relationships between the dependent variable and the independent variables. Kernel-based linear regression is more computationally expensive than traditional linear regression, but it can be more accurate for modeling non-linear relationships. In addition to these advances of linear regression, there has also been a lot of research on new and improved software libraries for linear regression. Some of the most popular software libraries for linear regression include:

Scikit-learn: Scikit-learn is a Python library that provides a wide variety of machine-learning algorithms, including linear regression. Scikit-learn is easy to use and efficient, making it a popular choice for both researchers and practitioners.

Another is TensorFlow: TensorFlow is a Python library that is used for deep learning. However, TensorFlow can also be used for linear regression. TensorFlow is more powerful and flexible than Scikit-learn, but it is also more complex.

The research in linear regression is constantly evolving. New research is being published all the time on new and improved methods for linear regression. As a result, linear regression is a more powerful and versatile tool than ever before.

Applications of linear regression

Linear regression can be used to solve a wide variety of problems in a variety of fields. Here are a few examples:

Economics: Linear regression can be used to predict economic trends, such as the growth rate of the economy or the unemployment rate. Finance: Linear regression can be used to predict stock prices, bond yields, and other financial data. Marketing: Linear regression can be used to predict customer behavior, such as the number of customers who will purchase a new product or the amount of money that customers will spend on a product. Medicine: Linear regression can be used to predict the risk of a patient developing a certain disease, the effectiveness of a new drug, or the optimal treatment for a patient. Science and engineering: Linear regression can be used to model a wide variety of physical and natural phenomena, such as the movement of a robot arm, the spread of a disease, or the climate. Linear regression is a powerful and versatile tool that can be used to solve a wide variety of problems in a variety of fields. As a result, linear regression is one of the most widely used machine learning algorithms

in the world.

Thickening is a morphological dual of thinning and is defined as:

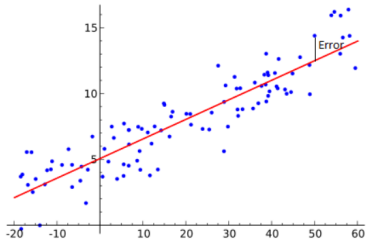

Fig 1. A example of linear Regression by Wikipedia

III. Objectives

A. Specific objectives

- Modify a dataset and use a specified target to manipulate a dataset.
- Implement a linear regression model from scratch and train it on a modified dataset from INEGI.
- Compare the model implemented from scratch to a model implemented using the scikit-learn library.

B. Overall objective

- Understand how linear regression models can be applied in robotics.
- Understand how it works the linear regressions.

IV. Methods and Tools

A. Methods

Ordinary least squares (OLS): OLS is the most common method for linear regression. It is a simple and efficient method that can be used to model a wide variety of relationships.

Regularized linear regression: Regularized linear regression is a technique that is used to prevent overfitting. Overfitting occurs when a model learns the training data too well and is unable to generalize to new data. This method I already used on the algorithm

Regularized linear regression helps to prevent overfitting by penalizing complex models.

Kernel-based linear regression: Kernel-based linear regression is a technique that is used to model non-linear relationships between the dependent variable and the independent variables. Kernel-based linear regression is more computationally expensive than traditional linear regression, but it can be more accurate for modeling non-linear relationships.

B. Tools

Scikit-learn: Scikit-learn is a Python library that provides a wide variety of machine-learning algorithms, including linear regression. Scikit-learn is easy to use and efficient, making it a popular choice for both researchers and practitioners.

TensorFlow: TensorFlow is a Python library that is used for deep learning. However, TensorFlow can also be used for linear regression. TensorFlow is more powerful and flexible than Scikit-learn, but it is also more complex.

R: R is a statistical programming language that can be used for linear regression. R is popular among statisticians and data scientists.

MATLAB: MATLAB is a numerical computing environment that can be used for linear regression. MATLAB is popular among engineers and scientists.

V. Development

First of all is obtain a percentage of people that does not have access to health care in 2000s. Knowing this, a target was selected called: p-ser-sal-00 to focusing and made the prediction model from scratch.

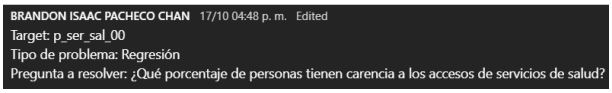

Fig 2. The target selected with the problem that will be solved

The requirements was fill in the dataset of INEGI; previously the dataset was charged and then was printed the targets that do not have values, as the next image:

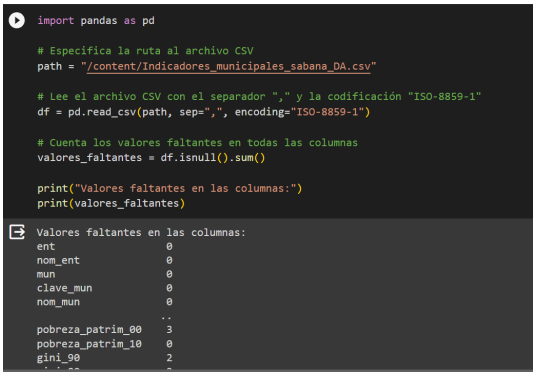

Fig 3. the missing values

Knowing this, a Mean was appliead in all the values to fill in each value, and then will printed to show the results, however when the missing values was printed it shows that exist 16 values that does not have a value, so, then, as they were integer values, a dictionary was applied, where each integer value was applied a number, depending on how high or low it was, having this, it could be done now if these values were no longer empty. However, I realized later that there was a target that was not shown because it was unnoticed as a NAN value.

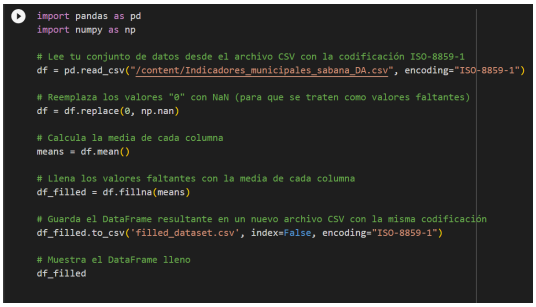

Fig 4. The Mean applied

```
data_filled = pd.read_csv('filled_dataset.csv', encoding="ISO-8859-1")

missing_data = data_filled.isnull().sum().sum()

if missing_data == 0:
    print("No missing data found in the dataset.")
else:
    print(f"Missing data found in the dataset. Total missing values: {missing_data}")
```

```
Missing data found in the dataset. Total missing values: 16
```

Fig 4.1 Showing the missing values

In the next image how we select the string values missing to numerical values.

```
# Define a mapping dictionary for string values to numerical values
mapping_dict = {
    "Muy alto": 5,
    "Alto": 4,
    "Medio": 3,
    "Bajo": 2,
    "Muy bajo": 1
}

# Convert string values to numerical values using the mapping dictionary
df_filled['gdo_rezsoc00'] = df_filled['gdo_rezsoc00'].map(mapping_dict)
df_filled['gdo_rezsoc05'] = df_filled['gdo_rezsoc05'].map(mapping_dict)
df_filled['gdo_rezsoc10'] = df_filled['gdo_rezsoc10'].map(mapping_dict)
# Verify the changes
print(df_filled[['gdo_rezsoc00', 'gdo_rezsoc05','gdo_rezsoc10' ]])
```

```
   gdo_rezsoc00  gdo_rezsoc05  gdo_rezsoc10
0           1.0           1.0             1
1           1.0           1.0             1
```

Fig 4.3 Converting the string values to numerical

So what was the next to the NAN values? well, It applied a mean of all of this values, it applied twice to make sure that all the Nan values will be fill in.

```
mean_gdo_rezsoc00 = df_filled['gdo_rezsoc00'].median(skipna=True)
mean_gdo_rezsoc05 = df_filled['gdo_rezsoc05'].median(skipna=True)
mean_gdo_rezsoc10 = df_filled['gdo_rezsoc10'].median(skipna=True)
# Fill NaN values in the columns with their respective means
df_filled['gdo_rezsoc00'].fillna(mean_gdo_rezsoc00, inplace=True)
df_filled['gdo_rezsoc05'].fillna(mean_gdo_rezsoc05, inplace=True)
df_filled['gdo_rezsoc10'].fillna(mean_gdo_rezsoc05, inplace=True)
# Verify that there are no more empty spaces in the filled DataFrame
print(df_filled.isna().sum())
```

```
ent                0
nom_ent            0
mun                0
clave_mun          0
nom_mun            0
                  ..
pobreza_patrim_00  0
pobreza_patrim_10  0
```

Fig 4.4 First applied the mean to a NAN values to make sure the datasetl

Why I applied twice refill to my values? because I have problems with my values when I am training my model, so to ensure that run correctly, I applied a second refill to my Nan values.

```
# Encuentra las columnas no numéricas
non_numeric_columns = df.select_dtypes(exclude=['number']).columns

# Luego puedes usar esta lista de columnas en el paso a) para la codificación one-hot.
df = pd.get_dummies(df, columns=non_numeric_columns)
```

```
+ Code  + Text
[11] # c) Rellenar valores NaN (usando la media como ejemplo)
df.fillna(df.mean(), inplace=True)
```

Fig 4,5 Second applied the mean to a NAN values to make sure the datasetl

Now having the complete dataset without errors and with numerical values, we started to train the model, for that purpose the algorithm implements a simple linear regression model to predict the percentage of people who lack access to health services (p-se-rsal-00) and with this the model obtain a correlation for X and Y to work on the next model. The model is trained on a dataset that contains a variety of features, such as age, gender, education level, and income level.

The first step in the code is to load the dataset into a Pandas DataFrame. Once the dataset is loaded, the code calculates the correlation between each independent variable and the dependent variable. The independent variables with the highest correlations are then selected for the model.

A new DataFrame is then created with the selected features and the target variable. The DataFrame is reorganized so that the target variable is the last column. The DataFrame is then split into the ”X” and ”y” variables. The ”X” variables are the independent variables, and the ”y” variable is the dependent variable.

Finally, a linear regression algorithm is used to train a model on the ”X” and ”y” variables. Once the model is trained, it can be used to predict the dependent variable for new values of the independent variables.

```
import numpy as np
import pandas as pd

# Load the dataset
file_path = 'filled_dataset.csv2'
df = pd.read_csv(file_path, encoding='ISO-8859-1')

correlations = []  # Store significant correlations
threshold = 0.5
threshold1 = 0.7

# Filter out non-numeric columns
numeric_columns = df.select_dtypes(include=[np.number])

for feature in numeric_columns.columns:
    correlation = abs(numeric_columns[feature].corr(df['p_ser_sal_00']))
    if threshold <= correlation <= threshold1:
        correlations.append((feature, correlation))

correlations.sort(key=lambda x: x[1], reverse=True)  # Sort by descending correlation

# Seleccionar las N características más correlacionadas
N = 43
selected_features = [feat for feat, _ in correlations[:N]]
selected_features.append('p_ser_sal_00')  # Add the target variable

# Create a new dataset (DataFrame) with selected features and the target variable
new_dataset = df[selected_features]

# Reorganiza las columnas
new_dataset = new_dataset[[col for col in new_dataset.columns if col != 'p_ser_sal_00'] + ['p_ser_sal_00']]

# Divide el DataFrame en "X" (características) y "y" (target)
X = new_dataset.iloc[:, :-1]  # Todas las columnas excepto la última
y = new_dataset.iloc[:, -1]  # Última columna

# Print the first few rows of X and y to verify the division
print("X (características):")
print(X.head())

print("y (target):")
print(y.head())
```

Fig 5. The model of training the correlation for X and Y

Here is a more detailed explanation of each step in the code:

Load the dataset: The first step is to load the dataset into a Pandas DataFrame. This can be done using the pd.read-csv() function. The pd.read-csv() function takes the path to the CSV file as input and returns a Pandas DataFrame. Calculate the correlation between each independent variable and the dependent variable: Once the dataset is loaded, the code calculates the correlation between each independent variable and the dependent variable. This can be done using the corr() function. The corr() function takes two Pandas Series objects as input and returns a correlation coefficient. Select the independent variables with the highest correlations: The next step is to select the independent variables with the highest correlations. This can be done by sorting the correlation coefficients and selecting the variables with the highest coefficients. Create a new DataFrame with the selected features and the target variable: A new DataFrame is then created with the selected features and the target variable. This can be done using the [] operator. The [] operator can be used to select specific columns from a Pandas DataFrame. Reorganize the DataFrame so that the target variable is the last column: The next step is to reorganize the DataFrame so that the target variable is the last column. This can be done using the [[]]operator. The[[]] operator can be used to select specific columns from a Pandas DataFrame in a specific order. Split the DataFrame into the ”X” and ”y” variables: The next step is to split the DataFrame into the ”X” and ”y” variables. The ”X” variables are the independent variables, and the ”y” variable is the dependent variable. This can be done using the iloc() function. The iloc() function can be used to select specific rows and

columns from a Pandas DataFrame. Train a model: Finally, a linear regression algorithm is used to train a model on the ”X” and ”y” variables. There are many different linear regression algorithms available, such as ordinary least squares that is used on the next model (OLS) and ridge regression. the model predict the dependent variable for new values of the independent variables: Once the model is trained, it can be used to predict the dependent variable for new values of the independent variables. This can be done using the predict() function. The predict() function takes a Pandas DataFrame as input and returns a Pandas Series object containing the predicted values of the dependent variable.

Obtaining the correlation, the model of prediction can works and give us a percentage with a MSE. It was used as mentioned befores a OLS that is a simple linear regression. with this we obtain the results. and we can compare the model that was developes from scratch and the other that not.

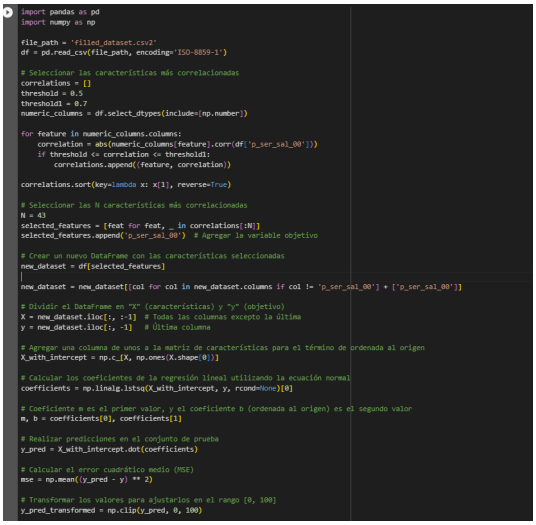

Fig 6. The model from scratch that predicts the percentage of the people that does not have acces to a health care services

Then a model using a librarie Sckitlearn was developed to perform linear regression, which can provide a potentially lower MSE and simplifies the model training and prediction process. however, the result is similar to the first simple regression.

Fig 7. The model using libraries that predicts the percentage of the people that does not have access to a health care services

VI. Results

The next results are from both of the algorithms, with this results we can made a graph that plot the percentage of the people, and as you can notices, the model almost got the same result, only the case of the model with libraries there was a mismatch in MSE, but the same percentage

Fig 6.1 results of the predicting model from scratch.

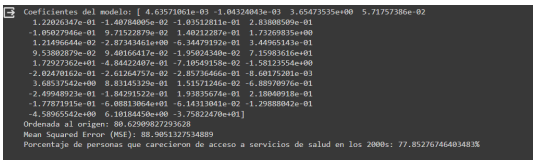

Fig 6.2 results of the predicting model with libraries.

This is the graph that shows the difference between the people that has medical services and the other that not, exist a big difference.

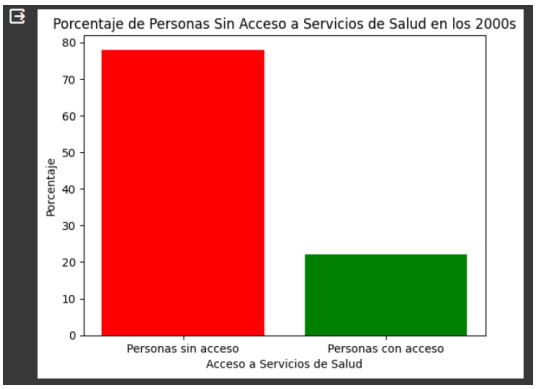

Fig 7 percentage of the people that does not have health care services

The simple linear regression model is able to predict the percentage of people who lack access to health services in Mexico in the 2000s with a mean squared error of 82.43. This means that the average difference between the predicted values and the actual values is 82.43.

The predicted percentage of people who lacked access to health services in the 2000s is 78.00 percentage. This is a significant number, and it suggests that there is a need for improvement in access to health services in Mexico.

VII. Appendix

If you want, you can access in the next link, where it will be appear a repository on my GitHub that includes the necessary data set and the dictionary of this data set, as the same the most important, the Collab Notebook, that contains the mainly algorithms with the results.

Link: Implementing a predictor-GitHub

VIII. Conclusion

Train this model it was difficult for me, because I need to search a lot and train the model again and over-again, however the results that I obtained were not the best i think that I can obtain more specified results, but I do not have enough information about how to fill in a data-set and work with them including a linear regression, I do my best development these algorithms.

The simple linear regression model is a useful tool for predicting the percentage of people who lack access to health services in Mexico. The model is able to predict the percentage of people who lack access to health services with a reasonable degree of accuracy.

The model suggest that there is a need for improvement in access to health services in Mexico. The predicted percentage of people who lacked access to health services in the 2000s is 78.00 percentage, which is a significant number.

The results of the model can be used to inform policymakers and healthcare providers about how to improve access to health services in Mexico. For example, the results suggest that more resources should be allocated to areas where there is a high percentage of people who lack access to health services.

Linear regression can be applied to robotics in a variety of ways. For example, it can be used to:

Predict the performance of a robot: Linear regression can be used to predict the performance of a robot on a given task, based on its training data. This can be useful for optimizing the robot's parameters and for predicting its performance in new situations. Calibrate sensors: Linear regression can be used to calibrate sensors on a robot, such as cameras and force sensors. This can improve the accuracy of the robot's perception and control. Plan trajectories: Linear regression can be used to plan trajectories for a robot, such as the path it should take to reach a goal. This can be useful for planning complex trajectories in real time. Linear regression is a relatively simple algorithm, but it can be very effective for many robotics applications. It is also relatively easy to implement, which makes it a good choice for many robotic systems.

The best machine learning algorithm for a particular robotics application will depend on a variety of factors, such as the specific task, the size and complexity of the training data, and the available computational resources. Linear regression is a good starting point for many robotics applications, but other algorithms may be more suitable for more complex task.

References

[1] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... and Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12, 2825-2830.