UNIVERSIDAD POLITECNICA DE YUCATAN

UNIT 2

MACHINE LEARNING

TEACHER

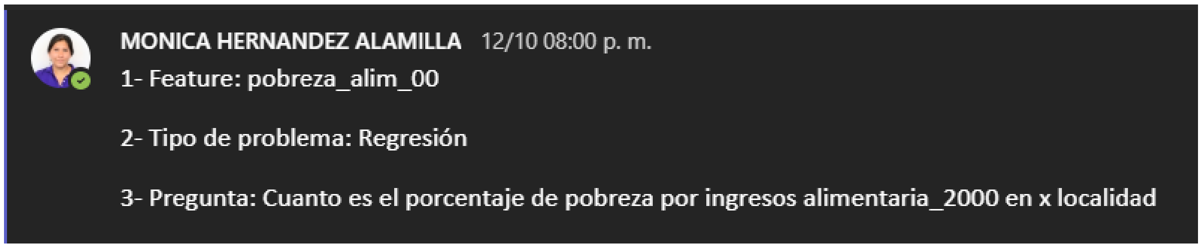
Víctor Alejandro Ortiz Santiago

Implementing a Predictor from scratch

by

Monica Hernández Alamilla

A. 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.



MONICA HERNANDEZ ALAMILLA 12/10 08:00 p. m.

1- Feature: pobreza_alim_00

2- Tipo de problema: Regresión

3- Pregunta: Cuanto es el porcentaje de pobreza por ingresos alimentaria_2000 en x localidad

B. A chronological document or journal describing in detail:

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

Dataset Preparation:

1. Data Loading:

Loaded the dataset using pandas from the file '/content/Indicadores_municipales_sabana_DA (1).csv'.

Encoded the data using 'ISO-8859-1'.

```
import pandas as pd
df = pd.read_csv('/content/Indicadores_municipales_sabana_DA (1).csv', encoding='ISO-8859-1')
```

2. Initial Exploration:

Explored the dataset's structure, features, and data types.

```
# Displaying basic information about the dataset
df.info()
```

3. Feature Selection:

Dropped columns 'nom_mun' and 'nom_ent' as they were not deemed relevant for the predictive task.

```
# Dropping irrelevant columns
df_procesado = df.drop(columns=['nom_mun', 'nom_ent'])
```

4. Target Variable:

Identified the target variable as 'pobreza_alim_00'.

```
# Setting the target column
target_column = 'pobreza_alim_00'
```

5. Handling Missing Values:

Calculated the global mean of the target column and filled missing values with this mean.

```
# Filling missing values with the global mean
promedio_global = np.nanmean(df_procesado[target_column])
df_procesado[target_column] = df_procesado[target_column].fillna(promedio_global)
```

6. Categorical Columns:

Identified categorical columns for one-hot encoding.

```
# Identifying categorical columns
columnas_categoricas = df_procesado.select_dtypes(include=['object']).columns
```

7. Data Transformation:

Constructed a Column Transformer to apply one-hot encoding to categorical columns.

```
# Constructing a ColumnTransformer for one-hot encoding
transformador = ColumnTransformer(
    transformers=[
        ('onehot', OneHotEncoder(), columnas_categoricas)
    ],
    remainder='passthrough'
)
```

- <u>All steps taken to train your predictor. Including an explanation on Why to choose such algorithm and an explanation on its characteristics.</u>

Predictor Training:

1. Model Selection:

Chose Linear Regression as the predictive model due to its simplicity and interpretability.

Considered the one-hot encoding for categorical variables and handled missing values using mean imputation.

```
# Creating a pipeline with data transformation and Linear Regression
modelo_regresion_lineal = Pipeline(steps=[
    ('transformador', transformador),
    ('imputador', SimpleImputer(strategy='mean', missing_values=np.nan)),
    ('regresion_lineal', LinearRegression())
])
```

2. Model Pipeline:

Created a sklearn Pipeline for the entire modeling process, including data transformation and model training.

```
# Model training pipeline
modelo_regresion_lineal.fit(X_train, y_train)
```

3. Train-Test Split:

Split the dataset into training and testing sets (70% training, 30% testing).

```
# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

4. Model Training:

Trained the Linear Regression model using the training set.

```
# Training the model
modelo_regresion_lineal.fit(X_train, y_train)
```

- <u>All steps taken to evaluate the performance of your predictor (using your test - sub dataset)</u>

Performance Evaluation:

1. Prediction on Test Set:

Made predictions on the test set using the trained model.

```
# Making predictions on the test set
y_pred_regresion_lineal = modelo_regresion_lineal.predict(X_test)
```

2. Evaluation Metric:

Calculated the R-squared (precision) as the evaluation metric using sklearn's metrics module.

```
# Calculating R-squared
precision_regresion_lineal = metrics.r2_score(y_test, y_pred_regresion_lineal)
print(f"Precisión del modelo de regresión lineal: {precision_regresion_lineal}")
```

3. Real vs. Predicted Comparison:

- Selected the last ten real values of the target variable.
- Removed the target variable and prepared the data for prediction.
- Predicted values using the trained Linear Regression model.

```
# Comparing real vs. predicted values for the last ten instances
ultimas_diez_real = df_procesado[target_column].iloc[-10:]
indices_ultimas_diez = ultimas_diez_real.index
X_ultimas_diez = df_procesado.loc[indices_ultimas_diez].drop(columns=[target_column])
y_pred_regresion_lineal_ultimas_diez = modelo_regresion_lineal.predict(X_ultimas_diez)
```

4. Results Presentation:

Displayed the Data Frame with real and predicted values for the last ten instances.

```
# Displaying the results
df_resultados_ultimas_diez = pd.DataFrame({
    'Real (pobreza_alim_00)': ultimas_diez_real.values,
    'Regresión Lineal': y_pred_regresion_lineal_ultimas_diez
}, index=indices_ultimas_diez)
print(df_resultados_ultimas_diez)
```

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

Libraries and Comparison:

1. Library Usage:

Utilized sklearn libraries extensively for data preprocessing, model training, and evaluation.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn import metrics
```

2. Comparison:

- Chose sklearn's Linear Regression due to its simplicity and effectiveness for this regression task.
- Compared the model's precision on the test set using R-squared.

```
# Comparing model precision using R-squared
print(f"Precisión del modelo de regresión lineal: {precision_regresion_lineal}")
```

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

https://github.com/monhdz23/IRC9A-MACHINELEARNING/tree/main/MHA_MachineLearning_U2/MHA_PredictorFromScratch

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

At the time of doing this project, I think the main challenges were navigating a dataset with a large data set, data quality, model selection. The balance between interpretability and complexity of the models, especially evident when opting for linear regression, required careful thought. Processing categorical data through one-time coding and assessing model performance using appropriate metrics added complexity. Looking ahead, these challenges have resonance in the field of robotics. Regression models can be useful for predicting maintenance needs, optimizing route planning, calibrating sensors, and improving resource allocation in robotic systems.