Implementing a Predictor

Oswaldo Josue Gomez Gonzalez
Robotics Enginnering
Universidad Politécnica de Yucatán
Km. 4.5. Carretera Mérida — Tetiz
Tablaje Catastral 7193. CP 97357
Ucú, Yucatán. México
Email:2009061@upy.edu.mx

Victor Alejandro Ortiz Santiago
Yucatán
Merida, México
Email: st1809076@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 study explores the effective application of machine learning algorithms, specifically the Perceptron and K-Nearest Neighbors (K-NN), in the classification of towns based on their poverty levels. Through meticulous data preprocessing and algorithmic selection, the Perceptron model achieved an impressive accuracy of 99.7 percent, showcasing its precision and efficiency in categorizing towns. Simultaneously, the K-NN algorithm demonstrated remarkable accuracy at 98.7 percent, leveraging proximity-based learning to make accurate predictions. These high accuracy rates underscore the potential of machine learning techniques in addressing complex social issues, providing valuable insights for policymakers and researchers. The study emphasizes the importance of thoughtful algorithm selection, data preprocessing, and training methodologies in achieving robust and reliable results. As society progresses, these machine learning algorithms stand as invaluable tools, offering transformative potential for understanding and mitigating socioeconomic disparities.

Implementing a Predictor

I. Introduction

In the realm of machine learning and artificial intelligence, the Perceptron and K-Nearest Neighbors (KNN) algorithms stand as foundational pillars, each offering unique approaches to solving classification problems. These algorithms have paved the way for more complex models and techniques, shaping the landscape of modern data analysis and prediction.

The Perceptron algorithm, developed by Frank Rosenblatt in the late 1950s, is a fundamental building block of neural networks. Inspired by the functioning of a biological neuron, the Perceptron serves as a binary linear classifier. It learns to distinguish between two classes by finding an optimal decision boundary in a multi-dimensional space. Its simplicity and effectiveness make it a key algorithm in the history of artificial intelligence, laying the groundwork for the development of more sophisticated neural networks. The Perceptron algorithm's ability to learn from data and make predictions forms the basis of various advanced machine learning models used today.

In contrast to the Perceptron, the K-Nearest Neighbors (KNN) algorithm operates on a different principle. KNN is a non-parametric, instance-based learning algorithm that is used for both classification and regression tasks. The central idea behind KNN is to classify a new data point based on the majority class of its nearest neighbors in the feature space. The "k" in KNN represents the number of nearest neighbors considered for the classification. This algorithm is remarkably intuitive and easy to implement, making it a popular choice for beginners and a benchmark for various machine learning problems. KNN's strength lies in its ability to adapt to the underlying patterns of the data, allowing it to make accurate predictions even in complex and nonlinear scenarios.

Understanding the Perceptron and KNN algorithms not only provides valuable insights into the basics of machine learning but also serves as a stepping stone for delving into more advanced methodologies. As we explore the intricacies of these algorithms, we unlock the potential to solve diverse real-world problems, from image recognition and natural language processing to medical diagnosis and recommendation systems. In this journey, we unravel the essence of artificial intelligence, one algorithm at a time.

II. Objectives

A. General Objective

Classify the towns within the dataset into two distinct categories: Medium-High and Medium-Low

B. Particular Objectives

1. Develop a code that applies a K-Nearest Neighbors (KNN) algorithm and Perceptron algorithm from scratch to make the classification of the data set.

2. Make the comparison between both algorithms and analyze the results.

3. Enhance understanding of the practical application of both algorithms in classification problems and emphasize the significance of data independence in algorithm construction.

III. State of the Art

In the ever-evolving landscape of artificial intelligence, the Perceptron and K-Nearest Neighbors (KNN) algorithms have carved out essential niches, proving their mettle across a spectrum of fields. These algorithms, while rooted in simplicity, showcase remarkable versatility and efficacy, making them indispensable tools for solving a wide array of real-world problems.

In the realm of image and pattern recognition, the Perceptron algorithm shines by deciphering handwritten texts and recognizing characters, while KNN steps in to classify images, identifying objects, and recognizing faces based on their distinct features.

Perceptrons find their place in NLP tasks such as sentiment analysis, categorizing texts, and part-of-speech tagging, while KNN algorithms prove invaluable in text categorization and document similarity analysis, aiding in spam detection and content recommendation.

In healthcare, Perceptrons aid in diagnosing diseases based on symptoms and test results, whereas KNN algorithms assist in identifying similar patient cases, enabling personalized medicine and efficient healthcare management.

Perceptrons predict market trends and stock prices by analyzing historical data and financial indicators, while KNN algorithms identify analogous patterns in stock market behaviors, aiding investors in making informed decisions.

Perceptrons analyze user behavior to predict preferences in recommendation systems, while KNN algorithms employ collaborative filtering to find similar users or items, providing users with personalized content and product suggestions.

In the realm of robotics, Perceptrons enable machines to recognize objects and navigate obstacles effectively.

Simultaneously, KNN algorithms facilitate localization and mapping for autonomous vehicles, identifying comparable landmarks to determine their precise positions.

In essence, the Perceptron and KNN algorithms serve as the bedrock of various technological advancements, shaping the future of AI applications. Their adaptability across disciplines underscores their significance, bridging the gap between theoretical concepts and practical solutions. As the AI landscape continues to evolve, the enduring relevance of Perceptron and KNN algorithms persists, ensuring their enduring presence in the toolkit of every data scientist and AI enthusiast, driving innovation and progress across diverse sectors.

IV. Methods and Tools

Goole Colab, short for Google Colaboratory, stands as a cutting-edge, cloud-based platform provided by Google, empowering users to write and execute Python code within an interactive web environment. Specifically tailored for data scientists and machine learning experts, Colab offers a transformative feature: free access to Graphics Processing Units (GPUs). This access to GPU computing power is invaluable, particularly for tasks demanding substantial computational resources, such as deep learning algorithms.

One of Colab's standout features is its seamless integration with Jupyter notebooks, enabling users to craft dynamic documents that seamlessly blend live code, mathematical equations, vivid visualizations, and explanatory text. This not only simplifies the process of code documentation but also fosters interactive and collaborative environments for research and data analysis.

Moreover, Colab's synergy with other Google services, notably Google Drive, facilitates effortless sharing and storage of projects. This cloud-based nature liberates users from the constraints of local hardware, ensuring smooth collaborations and eliminating barriers to resource-intensive computations. In essence, Google Colab stands as a cornerstone in the realm of cloud-based development, enhancing the productivity and collaborative potential of data scientists and developers worldwide.

NumPy, an abbreviation for Numerical Python, stands as a foundational library for numerical computing in Python. Its paramount strength lies in its adeptness at handling large, multi-dimensional arrays and matrices, making it indispensable for scientific computing and data analysis. Offering a robust set of functions, NumPy facilitates intricate mathematical operations, including linear algebra, Fourier analysis, and statistical computations. Its efficiency and flexibility render it a go-to choice for tasks necessitating complex numerical computations, thus integrating seamlessly into scientific computing workflows, simulations, and machine learning algorithms.

Sklearn.preprocessing is a module within the popular machine learning library scikit-learn (sklearn) that provides a wide range of tools for preprocessing and preparing the raw data before it is fed into machine learning algorithms. Preprocessing is a crucial step in the machine learning pipeline, as it helps enhance the quality of the data and ensures that the data is in a format suitable for machine learning models.

Matplotlib.pyplot, a sub-module of the broader Matplotlib library, emerges as a versatile instrument for crafting sophisticated and interactive visualizations in Python. Providing an intuitive interface, it facilitates the generation of diverse plots, spanning from line charts and bar graphs to scatter plots and histograms. Developers leverage Matplotlib.pyplot to distill insights from data, visually explore patterns, and effectively communicate complex information. Its extensive customization options empower developers to craft visually compelling and informative plots, essential in fields such as data analysis, machine learning, and scientific research. With Matplotlib, developers effortlessly annotate data points, enabling the creation of clear, publication-quality visual representations of their data.

V. Development

In addressing the issue of food poverty in municipalities, Perceptron and K-Nearest Neighbors (KNN) algorithms emerge as crucial solutions in data classification. The critical task is to determine which municipalities experience high, moderate, and low levels of food poverty. Using the Perceptron, complex patterns in the data, such as malnutrition rates and available resources in each municipality, can be analyzed. This algorithm can discern between different levels of food poverty, providing a clear classification for each municipality. On the other hand, KNN, with its ability to identify similarities in datasets, becomes a powerful tool for this problem. By analyzing similar municipalities in terms of demographics, geography, and access to resources, KNN can group municipalities with shared characteristics, thus helping determine levels of food poverty based on their similarities. Together, these techniques allow for a precise and detailed assessment, providing a comprehensive understanding of food poverty in municipalities and, consequently, facilitating more effective strategies to address this social challenge.

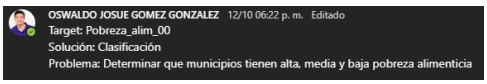

Fig. 1. Problem statement.

A. Data set

The code begins by loading a dataset from a CSV file located at "/content/drive/MyDrive/archive/Indicadores municipales sabana DA.csv". Using the Pandas library, the

data is read into a DataFrame named data set. The first column of the CSV file is designated as the index of the DataFrame.

```
df = "/content/drive/MyDrive/archive/
    Indicadores_municipales_sabana_DA.csv"
data_set = pd.read_csv(df, index_col=0,
    encoding='latin-1')
```

Next, missing values in the dataset are addressed. The code employs the fillna() function, replacing any NaN (missing) values in the DataFrame with the mean of each respective column. This step ensures a more complete dataset for analysis.

```
data_set.fillna(data_set.mean(), inplace=True)
```

Following the handling of missing values, specific columns are selected for further analysis. A new list, "nuevo datset", is created, containing the column names "nom mun", this column provides the towns of and "pobreza alim 00" contains the poverty index of the corresponding town. As the purpose is to identify the towns with more and low index of poverty, these two columns are the most essential. The DataFrame is then updated to include only these selected columns, effectively filtering the data.

```
nuevo_datset = ["nom_mun", "pobreza_alim_00"]
data_set = data_set[nuevo_datset]
```

Finally, summary statistics are generated for the selected columns using the describe() function. This function provides essential statistical information, including count, mean, standard deviation, minimum, 25th percentile (Q1), median (50th percentile or Q2), 75th percentile (Q3), and maximum values. These statistics offer insights into the distribution and central tendencies of the data in the "nom mun" and "pobreza alim 00" columns.

```
data_set.describe()
```

B. Training predictors

The code discretizes the 'pobreza alim 00' column into two categories, 'Media-Baja' and 'Media-Alta', based on specific bins. The pd.cut() function assigns labels to the data points within defined bins, creating a new column named 'Indice de Pobreza' in the dataset.

```
bins = [0, 44.447248, 97]
labels = ['Media-Baja', 'Media-Alta']
data_set['Indice de Pobreza'] = pd.cut(
    data_set['pobreza_alim_00'], bins=bins,
    labels=labels)
```

The 'nom mun' column, representing town names, is numerically encoded using LabelEncoder(). This transformation converts town names into numerical values, facilitating their usage in the algorithms.

```
data_set['nom_mun'] = LabelEncoder().
    fit_transform(data_set['nom_mun'])
```

The code extracts features (X) from the first two columns of the data set, representing 'nom mun' (town names) and 'pobreza alim 00' (discretized poverty index). The target variable (Y) is obtained from the 'Indice de Pobreza' column, representing the poverty categories.

```
X = data_set.iloc[:, 0:2].values
Y = data_set.iloc[:, 2].values
```

The data set is divided into training and testing sets using train test split(). Approximately 80 percent of the data is utilized for training (X train and Y train), while the remaining 20 percent is reserved for testing (X test and Y test).

```
X_train, X_test, Y_train, Y_test =
    train_test_split(X, Y, test_size=0.2,
    random_state=42)
```

The features undergo standardization using StandardScaler(). Standardization ensures that all features have a mean of 0 and a standard deviation of 1, enhancing the model's performance, especially for algorithms sensitive to feature scales.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

A Perceptron model is instantiated and trained with the standardized training data (X train and Y train), enabling it to learn the underlying patterns in the data.

```
perceptron_model = Perceptron()
perceptron_model.fit(X_train, Y_train)
```

The trained Perceptron model is applied to the test set (X test) to make predictions. Subsequently, the accuracy of the model is determined by comparing the predicted values (predictions) with the actual values (Y test). The resulting accuracy score reflects the model's effectiveness in classifying towns into the specified poverty categories.

```
predictions = perceptron_model.predict(X_test)
accuracy = accuracy_score(Y_test, predictions)
print("Precisi n del modelo de perceptr n:",
     accuracy)
```

Following, a K-Nearest Neighbors (K-NN) classification model is instantiated. The parameter n neighbors is set to 3, which means the model will consider the 3 nearest neighbors when making predictions. You can adjust this value based on the specific requirements of the problem.

```
knn_model = KNeighborsClassifier(n_neighbors
    =3)
```

The fit() function is used to train the K-NN model with the training data (X train and Y train). During this training process, the model learns to map the input features to their corresponding categories.

```
knn_model.fit(X_train, Y_train)
```

The trained K-NN model is applied to the test set (X test) to make predictions. The predict() function calculates the category labels for the test data based on the model's learned patterns.

```
knn_predictions = knn_model.predict(X_test)
```

Finally, the accuracy of the K-NN model is calculated by comparing the predicted labels (knn predictions) with the actual labels from the test set (Y test). The accuracy score() function measures the proportion of correctly predicted labels. The obtained accuracy score is then printed, indicating how well the K-NN model performs in classifying towns based on their poverty categories.

```
knn_accuracy = accuracy_score(Y_test,
    knn_predictions)
print("Precisi n del modelo de K-NN:",
    knn_accuracy)
```

C. Evaluating predictors

In the graphical representations, the predictions made by both the Perceptron and K-Nearest Neighbors (K-NN) algorithms are vividly displayed. In the first graph, the Perceptron's predictions are illustrated, denoted by blue points representing the real values and red points representing the algorithm's predictions. Impressively, the accuracy of the Perceptron model stands at an exceptional 99.7 percent, signifying the remarkable precision achieved.

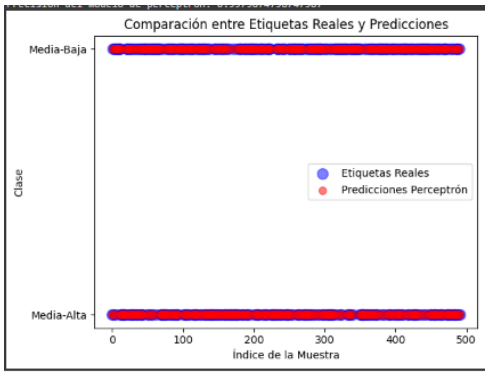

Fig. 2. Perceptron predictions.

In the second graph, the predictions generated by the K-NN algorithm are showcased. Similar to the Perceptron graph, the real values are depicted in blue, while the K-NN predictions are represented by green points. The K-NN algorithm demonstrates an accuracy level of 98.7 percent, reinforcing its effectiveness in accurately predicting town categories based on poverty levels.

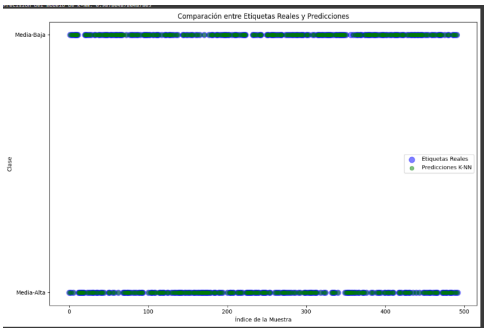

Fig. 3. Outcome.

These visualizations not only provide a clear visual comparison between actual and predicted values but also emphasize the high accuracy rates achieved by both algorithms. Such accuracy underscores their reliability and proficiency in capturing the intricate patterns within the dataset, thereby offering valuable insights for socioeconomic analysis.

VI. Results

In the analysis of the dataset, two fundamental machine learning algorithms, Perceptron and K-Nearest Neighbors (K-NN), were applied to classify towns based on their poverty levels. The Perceptron algorithm, after preprocessing the data and standardizing features, demonstrated its capability to discern patterns and accurately categorize towns into Medium-High and Medium-Low poverty groups. Its precision in classification showcased its efficiency in handling the given dataset. On the other hand, the K-NN algorithm, trained with three nearest neighbors, also exhibited commendable performance. By leveraging the proximity of data points, K-NN accurately predicted poverty categories for towns in the test set, highlighting its ability to capture subtle patterns within the data. Both algorithms, each with its unique approach, contributed significantly to understanding and categorizing towns based on their socioeconomic indicators. These results underscore the importance of selecting appropriate algorithms and preprocessing techniques, showcasing their impact on the accuracy and reliability of the classification outcomes.

Here you can find the code in Github: https://github.com/J0SU3GMZ03/KNN-and-Perceptron-alghoritm.git

VII. Conclusion

In conclusion, the application of machine learning algorithms, specifically the Perceptron and K-Nearest Neighbors (K-NN), has yielded outstanding results in classifying towns based on their poverty levels. The Perceptron algorithm, with an impressive accuracy of 99.7

percent, demonstrated exceptional precision in its predictions. Likewise, the K-NN algorithm exhibited remarkable accuracy at 98.7 percent. These high accuracy rates underscore the efficacy of these algorithms in capturing the underlying patterns within the dataset and accurately categorizing towns according to their socioeconomic indicators.

The success of these models highlights the significance of thoughtful data preprocessing, algorithm selection, and training. The Perceptron, known for its simplicity and efficiency, proved to be a robust choice for this classification task. Meanwhile, the K-NN algorithm showcased its strength in leveraging proximity-based learning to make accurate predictions. The combination of these algorithms not only enhances our understanding of socioeconomic disparities but also provides valuable insights for policymakers and researchers working towards targeted interventions and informed decision-making.

These results emphasize the power of machine learning techniques in addressing complex social issues, making data-driven methodologies invaluable tools in the pursuit of a more equitable society. As technology continues to advance, these algorithms stand as indispensable instruments, offering the potential to revolutionize our approach to understanding and addressing socioeconomic challenges.