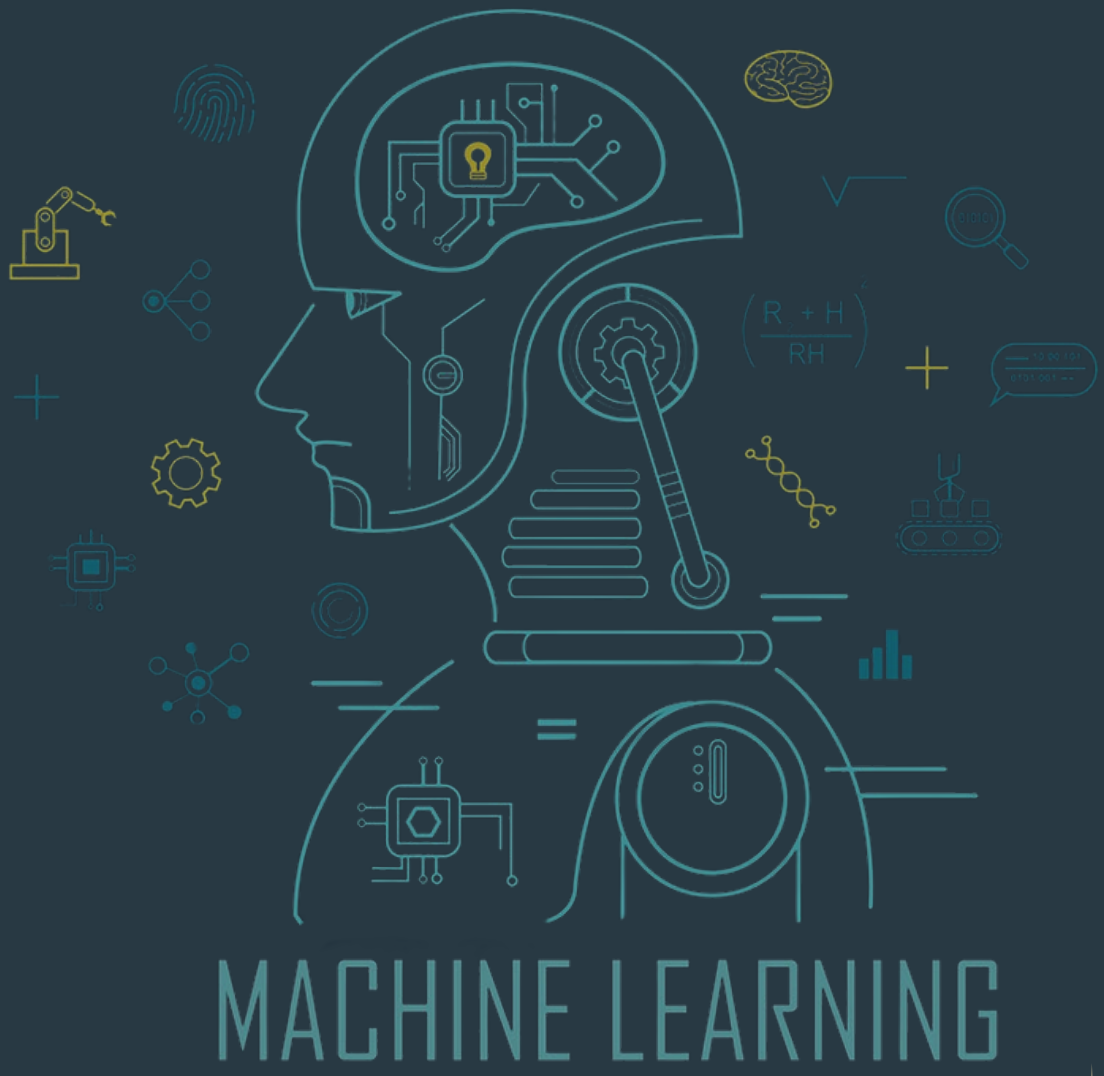
15. 11 . 2024

Project Report



Submitted by: **EFE EROL**
No: **200101066**

1. Introduction

The primary goal of this project is to implement a classification task using the Perceptron learning algorithm. The task involves training a model on a provided dataset (TRAINData) and testing the model on a separate dataset (TESTData). The classification labels are binary (class 2 and class 4), and the model will predict these classes based on the attributes provided in the dataset.

Objective

- Implement the Perceptron learning algorithm to classify data into two classes: 2 and 4.
- Train the algorithm on the training dataset and compute the hypothesis (weights and bias).

2. Dataset Description

The DataForPerceptron.xlsx file, which contains two sheets: TRAINData and TESTData, is read using the pandas.read_excel() function. The TRAINData is used to train the model, and TESTData is used to evaluate the performance of the trained model.

```
file = "DataForPerceptron.xlsx"

trainData = pd.read_excel(file, sheet_name='TRAINData')
testData = pd.read_excel(file, sheet_name='TESTData')
```

The dataset consists of the following attributes:

- Attributes: Attr1 to Attr9 (numeric values representing features).
- Class: The target variable (either 2 or 4 in the training data).

Example Data (TRAINData):

SubjectID	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Class
1	5	1	1	1	2	1	3	1	1	2
2	5	4	4	5	7	10	3	2	1	2

Example Data (TESTData):

SubjectID	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Class
551	5	7	10	10	5	10	10	10	1	
552	3	1	2	1	2	1	3	1	1	

3. Methodology

The implementation follows these steps:

- Preprocess the data by separating features (Attr1 to Attr9) and labels (Class).
- Split the training data into X_train, y_train, X_test, and y_test.
- Train the Perceptron algorithm using the training data (X_train, y_train).
- Predict the class labels for the test data (X_test) and evaluate the accuracy.

4. Perceptron()

The Perceptron function is defined to take trainData and a learning rate alfa as input. We extract the features (Attr1 to Attr9) and the class label from the trainData. The features are stored in X, and the class labels are stored in y.

```
# Extracting features from columns (ignoring subject id and class)
X = trainData.iloc[:, 1:-1].values

# Extracting class labels (the last column)
y = trainData.iloc[:, -1].values
```

A weight vector weights is initialized to zeros, and the bias is also initialized to zero.

```
weights = np.zeros(X.shape[1])  # Initialize weights to zero for each feature
bias = 0
```

The algorithm iterates for a fixed number of epochs (1000 in this case). For each sample in the dataset, the Perceptron makes a prediction by calculating the weighted sum of the input features plus the bias.

```
  # Calculate the prediction by multiplying the features with weights and adding
  prediction = np.dot(X[i], weights) + bias
```

5. Perceptron()

If the prediction is less than 0, the predicted class is assigned to 2, and if greater than or equal to 0, it is assigned to 4.

```
if prediction < 0:
    predictionClass = 2
else:
    predictionClass = 4
```

If the predicted class is different from the actual class (y[i]), the weights and bias are updated based on the error, which is the difference between the true label and the predicted label.

```
# If the predicted class is different from the actual class
if predictionClass != y[i]:
    error = y[i] - predictionClass
    weights += alfa * error * X[i]
    bias += alfa * error
```

Once the training process completes, the function returns the final weight vector and bias, which will be used for making predictions on the test data.

6. Prediction()

The Predict function is responsible for using the trained model (weights and bias) to predict the class label for new data.

```
def Predict(X, weights, bias):

    predict = np.dot(X, weights) + bias

    if predict < 0:
        return 2
    else:
        return 4
```

The input X is multiplied by the weight vector, and the bias is added. If the result is negative, the function predicts class 2; otherwise, it predicts class 4.

7. Test()

After training the model, the Test function evaluates the performance on the TESTData. The features from the testData are passed through the Predict function to obtain the predicted class labels.

```
def Test(testData, weights, bias):

    # Extract features from the test data (excluding subject id and class label)
    testX = testData.iloc[:, 1:-1].values

    # List to store predictions for each test sample
    testPredictions = []

    # For each sample in the test set, use the Predict function to make predictions
    for i in range(len(testX)):
        testPrediction = Predict(testX[i], weights, bias)
        testPredictions.append(testPrediction)

    return testPredictions
```

The list testPredictions stores the predicted class labels for each sample in the TESTData.

8. Save()

The Save function is designed to update the Class column in the testData DataFrame with the predictions made by the Perceptron model, and then give the user an option to save the updated DataFrame to an Excel file.

The function starts by making a copy of the testData DataFrame to ensure that the original data remains unmodified.

Next, it updates the Class column of the copied DataFrame with the predicted class labels stored in the predictions array.

After the DataFrame is updated, the function prompts the user with a message asking whether they would like to save the updated file to an Excel file named Updated_TESTData.xlsx.

If the user enters y, the updated DataFrame is saved to the specified Excel file, otherwise the function simply prints a message stating that the file was not saved.

```
def Save(testData, predictions):

    updatedTESTData = testData.copy()

    updatedTESTData['Class'] = predictions

    save = input("\nDo you want to save the updated data to 'Updated_TESTData.xlsx'? (y/n): ").strip().lower()

    if save == 'y':
        updatedTESTData.to_excel("Updated_TESTData.xlsx", index=False)
        print("\nFile saved successfully as 'Updated_TESTData.xlsx'.\n")
    else:
        print("File was not saved.\n")
```

9. main()

The main() function orchestrates the entire process by reading the data, training the model, and testing it.

```python
def main():

    file = "DataForPerceptron.xlsx"

    trainData = pd.read_excel(file, sheet_name='TRAINData')
    testData = pd.read_excel(file, sheet_name='TESTData')

    """
    print("Train Data:")
    print(trainData.head())

    print("\nTest Data:")
    print(testData.head())
    """

    weights, bias = Perceptron(trainData, alfa = 0.1, epochs=1000)
    #print(f"Weights {weights} ---- Bias {bias}")

    predictions = Test(testData, weights, bias)
    print("\nPredictions:", predictions)

    Save(testData, predictions)
```

10. Predictions

Once the model is trained, the weights and bias are printed. The test predictions are displayed as a list of class labels predicted for each row in the TESTData.

```
Predictions: [4, 2, 2, 4, 4, 4, 4, 2, 2, 4, 2, 2, 2, 2
, 2, 2, 4, 4, 2, 2, 2, 4, 2, 4, 2, 4, 4, 4, 2, 4, 2, 2
, 2, 2, 2, 2, 2, 2, 4, 4, 4, 2, 2, 4, 2, 4, 4, 4, 2, 2
, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 4
, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2
, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4
, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 2, 2, 2, 2, 2, 2
, 2, 2, 2, 4, 2, 2, 2, 2, 4, 4, 4]
```

11. Conclusion

This report outlines the step-by-step implementation of the Perceptron algorithm for binary classification using a provided dataset. The model successfully learns to classify data based on the attributes provided in TRAINData and then predicts the class labels for the TESTData.

15 . 11 . 2024

Thank you

for taking the time to read this report.

Submitted by: **EFE EROL**
No: **200101066**