Machine Learning for Binary Classification

Tai Luan Nguyen

Overview

This project performs various machine learning tasks on a dataset with 19020 samples and 11 attributes. Demonstrates the application of various machine learning algorithms to a dataset, allowing the comparison of their performance using classification reports.

Additionally, it conducts a hyperparameter search for a neural network model.

The project including:

- Importing Libaries
- Dataset Loading and Preprocessing
- Split dataset
- Data Scaling and Oversampling
- k-Nearest Neighbour (kNN) Classifier
- Naive Bayes Classifier
- Logistic Regression Classifier
- Support Vector Machine (SVM) Classifier
- Neurol Network (Deep Learning) Classifier

Project link: https://github.com/luan30092000/binaryClassification

Dataset

Data are MC generated to simulate registration of high energy gamma particles in an atmospheric Cherenkov telescope.

Reference: Bock,R.. (2007). MAGIC Gamma Telescope. UCI Machine Learning Repository. https://doi.org/10.24432/C52C8B.

Importing Libaries

Necessary libraries and its purpose for this project:

- `numpy`: data manipulation and array operation.
- `pandas`: load and manipulate dataset, as well as for data preprocessing and analysis.
- `matplotlib`: `matplotlib.pyplot`, create histograms for data visualization.
- `sklearn.preprocessing`: `StandardScaler` from this library is used for feature scaling, which standardizes the data to have a mean of 0 and a standard deviation of 1.

- `imblearn.over_sampling`: `RandomOverSampler` is used to oversample the minority class to balance the class distribution.
- `sklearn.neighbors`: `KNeighborsClassifier` to create and evaluate a kNN model.
- `sklearn.naive_bayes`: `GaussianNB` to create and evaluate a Naive Bayes model.
- `sklearn.linear_model`: `LogisticRegression` to create and evaluate a logistic regression model.
- `sklearn.svm`: `SVC` to create and evaluate an SVM model
- `tensorflow`: used to create and train a neural network for classification

Dataset Loading and Preprocessing

- Loads a dataset from a file named "magic04.data" using pandas and defines column names for the dataset.
- Converts the "class" column to binary values (0 or 1) by mapping "g" to 1 and "h" to 0.
- Plots histograms for each feature, comparing the distributions for class 0 (hadron) and class 1 (gamma).

Train, Validation, and Test Datasets:

- The dataset is split into train, validation, and test sets using the np.split function, with a 60-20-20 split ratio.

Data Scaling and Oversampling

- Scale the features and, optionally, oversample the dataset.
- `StandardScaler` is used to scale the features

k-Nearest Neighbours (kNN) Classifier

- Uses the scikit-learn library to create a k-Nearest Neighbors (kNN) model with 5 neighbors.
- Fits the model to the training data and makes predictions on the test data.
- **Accuracy: 81%**

Naive Bayes Classifier

- Uses `scikit-learn`'s Gaussian Naive Bayes classifier to create a Naive Bayes model.
- Fits the model to the training data and makes predictions on the test data.
- **Accuracy: 73%**

Logistic Regression Classifier

- Logistic regression model is created using `scikit-learn`
- The model is trained on the training data and used to predict on the test data.
- **Accuracy: 78%**

Support Vector Machine (SVM) Classifier

- Support Vector Machine (SVM) classifier is created using `scikit-learn`.
- Model is trained on the training data and used to predict on the test data.
- **Accuracy: 85%**

Neural Network (Deep Learning) Classifier

- Neural network model is defined using `TensorFlow/Keras`
- Performs a hyperparameter grid search, iterating over various combinations of the number of nodes, dropout probability, learning rate, and batch size.
- Trains multiple neural network models with different hyperparameters on the training data and selects the model with the lowest validation loss.
- The chosen neural network model has the **accuracy of 87%**