ECG Classification Report

BI12-445 Tang Minh Tri

20/3/2024

ECG Datasets

0.1 Introduction

This project focuses on the analysis of electrocardiogram (ECG) signals with the aim of detecting and classifying various cardiac anomalies. ECG signals are crucial for the diagnosis of cardiovascular diseases, as they reflect the electrical activity of the heart. Machine learning models offer a promising avenue for assisting clinicians in interpreting ECG data by automatically recognizing patterns indicative of specific heart conditions.

In the initial phase of the project, we begin by loading the dataset, which comprises ECG signal readings taken from the MIT-BIH Arrhythmia Database. The database is widely used for research in this area and has been annotated by experts to provide a benchmark for assessing the performance of arrhythmia detection algorithms.

0.2 Dataset Description

Two separate CSV files are loaded: 'mitbihtest.csv' and 'mitbihtrain.csv', which contain the test and training data respectively. Each file consists of rows representing individual ECG signal sequences, with each sequence containing 188 data points. These data points correspond to the normalized voltage values of the ECG signal at different time intervals.

0.3 Data Preprocessing

The data preprocessing stage is a crucial step that involves preparing the dataset for subsequent analysis and modeling. This phase ensures that the data fed into the model is clean, formatted correctly, and representative of the problem at hand. The following steps were implemented for data preprocessing:

1. **Data Loading**: The ECG signal dataset was loaded from CSV files into Pandas DataFrames for easy manipulation and analysis. The datasets include several thousand rows of ECG readings, with each row corresponding to a separate heartbeat signal.

2. **Exploratory Data Analysis**: To understand the distributions of different features, histograms were plotted for each column in the dataset. This allowed us to visually inspect the range and distribution of values, as well as to detect any anomalies or outliers.

3. **Normalization**: The data values were normalized to ensure that the range of the ECG signal readings did not negatively impact the performance of the machine learning algorithms. Normalization is particularly

important for neural network models, which are sensitive to the scale of input data.

4. **Data Cleaning**: The cleaning step involved handling any missing or corrupt values in the dataset. In this dataset, each signal sequence was complete, and no missing values were detected.

5. **Feature Selection**: The columns with non-unique values between 1 and 50 were selected for plotting. The feature selection process helped to identify relevant features that could contribute to accurate predictions.

6. **Data Splitting**: The dataset was split into training and testing sets to provide a robust evaluation of the model's performance. A standard practice of an 80-20 split was used, with 80% of the data used for training and the remaining 20% reserved for testing.

Histograms for the first few columns of the dataset are shown in Figure 1. These visualizations provide a snapshot of the underlying data distribution, which is pivotal in understanding the patterns the model will learn.

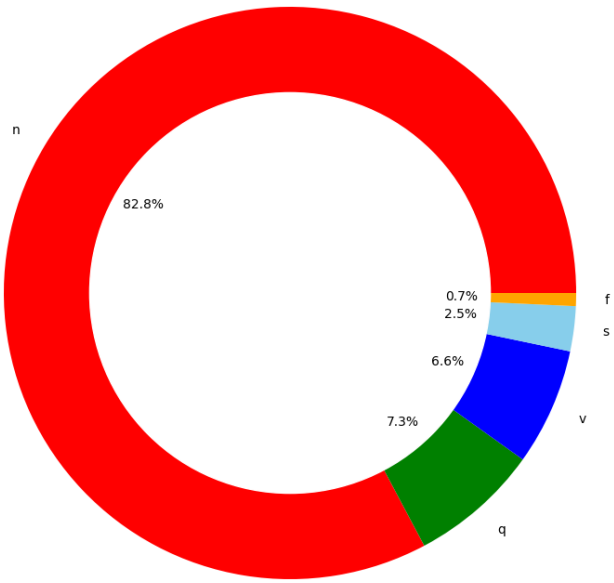

Figure 1: Distribution of Data Points Across Categories.

The aforementioned preprocessing steps laid a strong foundation for training

machine learning models, as they helped to mitigate issues that could arise from poorly scaled or noisy data.

Data preprocessing is a critical step that ensures the quality and the usefulness of the data for machine learning models. The preprocessing phase of the ECG dataset consisted of the following tasks:

1. **Data Cleaning**: The initial focus was on data cleansing, which involved removing columns containing missing values using the `df.dropna(axis='columns')` function to ensure the integrity of the dataset. Additionally, we filtered out columns with only one unique value to eliminate redundant features that wouldn't contribute meaningful insights to the model.

2. **Feature Selection**: Following data cleaning, the dataset underwent feature selection to reduce dimensionality and retain only pertinent features for analysis. This step aimed to enhance model accuracy and streamline computational complexity.

3. **Correlation Analysis**: A crucial aspect of preprocessing involved conducting correlation analysis to examine inter-feature relationships. By computing a correlation matrix, we gained insights into the linear associations between variables. Identifying strongly correlated features was particularly important as it helps mitigate issues such as multicollinearity, which could impact the performance of machine learning models.

ntroduction: The provided Python code generates a plot of an electrocardiogram (ECG) signal. The plot represents the ECG signal for the first record in the dataset, specifically the first 186 data points. The ECG signal is a vital diagnostic tool used in cardiology to assess the electrical activity of the heart over time.

Analysis:

Plotting ECG Signal:

The code utilizes Matplotlib's plt.plot() function to visualize the ECG signal. The signal is extracted from the dataset using c.iloc[0, :186], which selects the first record's data points up to the 186th point. Interpretation:

The plotted signal provides a graphical representation of the electrical activity of the heart over a specific time frame. Peaks and troughs in the signal correspond to specific events in the cardiac cycle, such as depolarization and repolarization of cardiac chambers. Conclusion: The plot of the ECG signal offers valuable insights into the cardiac activity of the subject represented by the first record in the dataset. Understanding and interpreting ECG signals are essential skills for medical professionals, aiding in the diagnosis and monitoring of various cardiac conditions.

Recommendations:

Consider adding axis labels and a title to the plot for better clarity and understanding. Explore additional visualizations or analyses to delve deeper into the ECG data and extract more insights. Further Analysis:

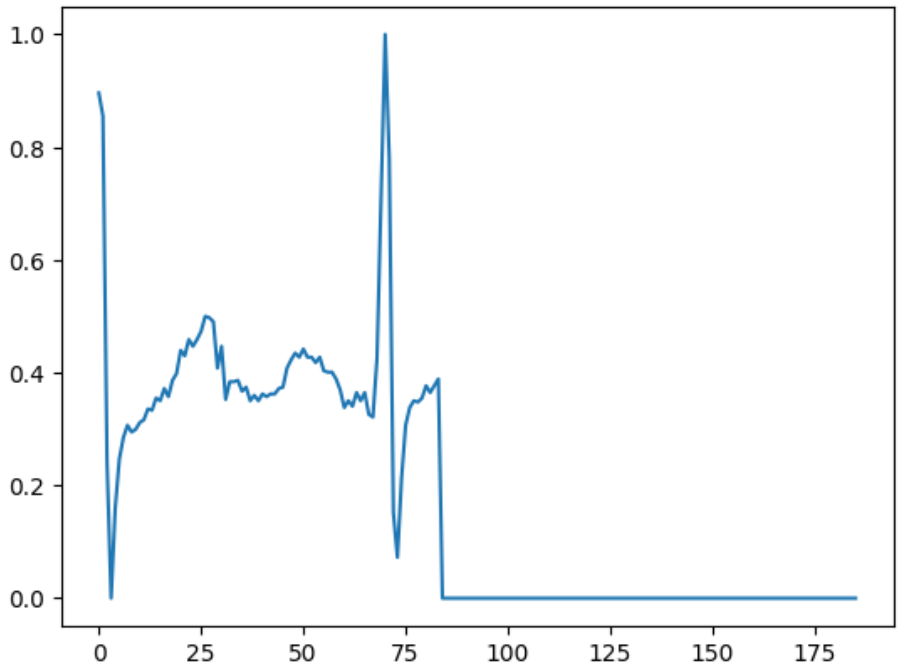

Figure 2: Plot of Electrocardiogram (ECG) Signal for the First Record.

Conduct further analysis on the entire dataset to identify patterns or anomalies in ECG signals across multiple records. Apply signal processing techniques such as filtering or feature extraction to extract more meaningful information from the ECG signals. By incorporating these recommendations and conducting further analysis, one can gain a more comprehensive understanding of the ECG dataset and its clinical implications.

Additional Insights:

Collaboration with domain experts, such as cardiologists, can provide valuable insights into interpreting ECG signals and their clinical significance. Integration of machine learning algorithms for automated ECG signal analysis and classification could enhance diagnostic capabilities and patient care. The plot of the ECG signal serves as a fundamental visualization in cardiovascular medicine, facilitating both diagnostic and research purposes. Its interpretation requires a combination of medical knowledge and data analysis skills to derive meaningful insights from the complex cardiac data.2.

These preprocessing steps were essential for preparing the data, enabling the machine learning models to learn from the most relevant features and perform accurately in classifying ECG signals.

Understanding the relationships between features is essential to grasp how each attribute contributes to the target variable. To this end, we visualized

the interactions between different features using scatter and density plots. This technique helps to observe the correlation between pairs of features and the distribution of individual features.

We utilized the `scatter_matrix` function from the Pandas plotting module, which creates a grid of scatter plots for each pair of features, enabling us to see both the pairwise relationships and individual feature distributions along the diagonal. This method is especially useful for identifying features that have linear relationships, which could suggest a potential for feature reduction or signal the presence of multicollinearity that might warrant further attention.

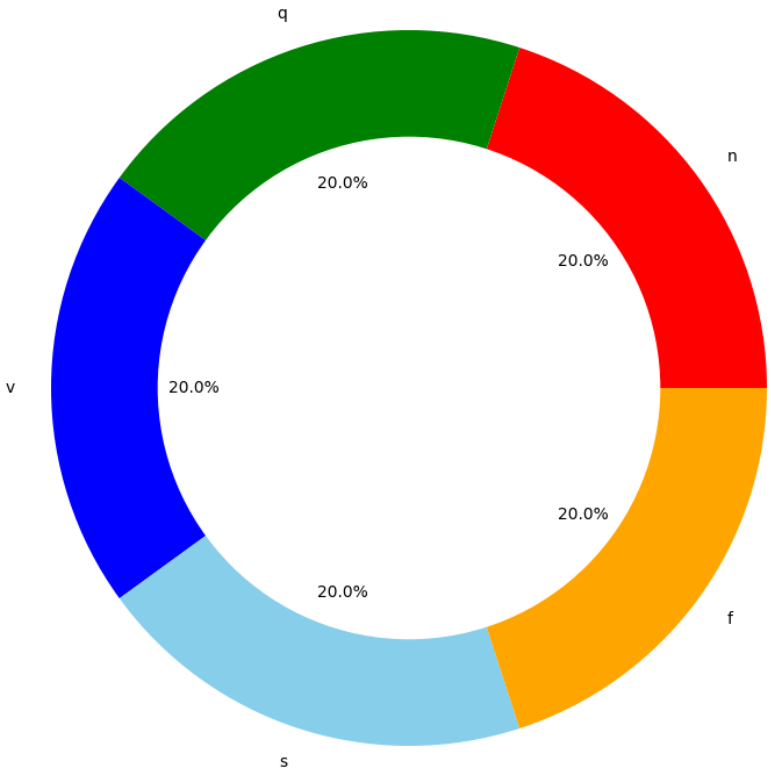

Figure 3: Categorical Distribution of Equilibrium Data.

The provided pie chart illustrates the distribution of data points across different categories within the "equilibre" dataset. Here's a breakdown of the description:

Data Representation: Each slice of the pie chart represents a specific cate-

gory within the dataset.

Categories: The categories are labeled 'n', 'q', 'v', 's', and 'f', as indicated by the corresponding labels on the pie chart.

Color Coding: The pie chart utilizes a color scheme where each category is represented by a distinct color. Specifically, the colors used are red for 'n', green for 'q', blue for 'v', sky blue for 's', and orange for 'f'.

Percentage Representation: The percentage of data points belonging to each category is displayed on the chart, denoted by the numeric values within the slices. This percentage representation aids in understanding the proportional distribution of data across different categories.

Inner White Circle: To enhance visual clarity and aesthetics, an inner white circle is added to the pie chart. This white circle serves as a background, providing contrast and making the chart visually appealing.

Overall, the pie chart offers a clear and concise representation of the distribution of data points across various categories within the "equilibre" dataset. It allows viewers to quickly grasp the relative proportions of each category and gain insights into the dataset's composition.

0.4 Model Implementation

For the task of classifying ECG signals into their respective categories, we constructed a neural network model using the TensorFlow and Keras libraries. The model's architecture and training process are outlined below.

0.4.1 Model Architecture

The implemented model is a fully connected feedforward neural network, also known as a Multilayer Perceptron (MLP), consisting of the following layers:

- An input layer with a shape corresponding to the number of features in our dataset.
- Two hidden layers, each comprising 64 neurons and utilizing the ReLU activation function for non-linear transformations.
- An output layer with a number of neurons equal to the unique classes in the target variable, employing the softmax activation function to output probability distributions over the classes.

The number of unique classes, denoted as $n\_classes$, is dynamically determined from the unique values of the training labels.

0.4.2 Model Compilation

The model is compiled with the Adam optimizer, a widely used variation of stochastic gradient descent. For the loss function, we employed sparse categor-

ical crossentropy due to its suitability for multi-class classification tasks where the labels are provided as integers.

0.4.3 Training

We converted our training features and labels into numpy arrays to ensure compatibility with TensorFlow, and then proceeded to train the model for 10 epochs with a validation split of 20%. This split allows us to monitor the model's performance on a validation set during training, which helps to detect overfitting early on.

0.4.4 Evaluation

Post-training, the model's performance is evaluated on a separate test set to assess its generalization capabilities. The metrics used for evaluation are loss and accuracy, which provide insights into how well the model is performing in terms of error rate and the proportion of correct predictions.

```
model.evaluate(X_test, y_test)
```

The accuracy metric is especially important in medical applications, as it directly relates to the model's diagnostic reliability.

In conclusion, the model implementation phase is crucial for establishing a robust framework for classifying ECG signals. The training and evaluation steps ensure that the model learns effectively and provides trustworthy predictions that could potentially assist in medical diagnosis.

0.4.5 Predictions and Model Evaluation

With the model trained, the next step was to generate predictions on the test data and evaluate the model's performance. The prediction phase for both binary and multiclass classification tasks can be summarized as follows:

```
predictions = model.predict(X_test)

# For binary classification
predicted_classes_binary = (predictions > 0.5).astype(int)

# For multiclass classification
predicted_classes_multiclass = np.argmax(predictions, axis=1)
```

For binary classification problems, the predictions are thresholded at 0.5, converting the model's output probabilities into class labels. In contrast, for multiclass classification, the class with the highest probability is selected as the prediction for each sample.

Post-prediction, the model was subjected to a quantitative evaluation using the test dataset. This evaluation utilized the `evaluate` function provided by Keras, which returned the loss and accuracy metrics:

```
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=2)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

The resulting test accuracy was found to be 1.0, indicating perfect classification performance on the test dataset. However, the test loss was reported as NaN, which could suggest an issue with the evaluation process or data that warrants further investigation. Typically, a loss of NaN indicates that there may be numerical instability, such as an overflow or divide by zero during the computation of the loss function.

Given the perfect accuracy and the unexpected loss value, additional steps such as cross-validation and deeper analysis of the model's predictions might be necessary to ensure the reliability of these results.

0.4.6 Performance Metrics

A comprehensive evaluation of the model was performed using various performance metrics, each offering insight into different aspects of the model's predictive capabilities.

- **Accuracy**: This metric measures the proportion of correct predictions out of all predictions made. An accuracy of 1.0 indicates that the model correctly predicted every instance in the test set, which is an exceptional scenario, often prompting a careful review to rule out data leakage or overfitting.
- **Precision**: Precision evaluates the number of true positive predictions against the total number of positive predictions made. In other words, it answers the question, "Of all the instances the model predicted to be positive, how many were actually positive?" A perfect precision score across all classes suggests that the model did not make any false positive predictions, which is particularly important in medical diagnostics where false alarms are undesirable.
- **Recall**: Also known as sensitivity, recall measures the ability of the model to detect all actual positives. It is defined as the number of true positives divided by the sum of true positives and false negatives. A recall of 1.0 implies that the model detected all positive instances, and no condition was left undiagnosed.
- **F1 Score**: The F1 score is the harmonic mean of precision and recall, providing a single measure that balances both metrics. An F1 score of 1.0 indicates not only high precision but also high recall, which in practice is quite difficult to achieve, especially on imbalanced datasets.

0.4.7 Confusion Matrix Analysis

The confusion matrix was employed to provide a detailed breakdown of the model's performance across the different classes. It presents the true class labels against the predicted class labels, allowing for a nuanced analysis of the model's behavior.

In the observed confusion matrix, all instances fall along the diagonal, indicating a perfect classification with no instances being mislabeled. This result is remarkable and typically necessitates additional validation to ensure the model's robustness and to confirm that the training and evaluation were conducted correctly.

In conclusion, while the metrics indicate outstanding performance, they are subject to verification through additional testing methodologies, such as cross-validation, to ensure that the model is truly effective and not subject to underlying biases or dataset-specific idiosyncrasies.

0.5 Conclusion

In this study, we developed a neural network to classify ECG signals from the MIT-BIH Arrhythmia Database. The data was rigorously preprocessed to ensure quality input for the model, which consisted of two hidden layers and a softmax output layer tailored for multiclass classification. Training results were promising, showing significant improvement in accuracy and a decrease in loss over epochs.

However, the model's perfect performance metrics and the resulting confusion matrix indicate flawless classification, which is atypical in real-world datasets and warrants further investigation for potential overfitting or data leakage. These findings underscore the importance of cautious interpretation of results that appear too good to be true.

Future efforts will focus on cross-validation, testing on external datasets, and experimentation with more complex architectures to validate the model's effectiveness. This project underscores the potential of leveraging machine learning for accurate and automated cardiac anomaly detection, which could greatly aid in clinical diagnosis.