Heart Failure Prediction using Classification

Project Overview

This project focuses on predicting the likelihood of heart failure in patients using a classification-based machine learning approach. The primary dataset used is `heart.csv`, which contains various medical attributes of patients.

The project follows a structured machine learning workflow:

1. **Data Collection**: Loading the dataset.
2. **Data Visualization**: Exploring data distributions, correlations, and the target variable's balance.
3. **Data Cleaning**: Checking for missing values.
4. **Data Transformation**: Encoding categorical features (One-Hot Encoding for nominal, Label Encoding for binary) and scaling numerical features.
5. **Initial Model Training**: Building a baseline Logistic Regression model.
6. **Model Evaluation**: Assessing the model's performance using metrics like accuracy, classification report, confusion matrix, and ROC AUC score.
7. **Hyperparameter Tuning**: Optimizing a `RandomForestClassifier` using `GridSearchCV` and `RandomizedSearchCV` to improve predictive performance.
8. **Feature Selection and Model Interpretation**: Identifying the most influential features for predicting heart failure using the best-performing model.

Dataset

- **Name**: `heart.csv`
- **Source**: (The notebook does not explicitly state the original source of the `heart.csv` dataset. It's assumed to be a common dataset for heart disease prediction tasks.)

The dataset includes features such as:

- `Age`: Age of the patient.
- `Sex`: Sex of the patient (M/F).
- `ChestPainType`: Type of chest pain (e.g., TA, ATA, NAP, ASY).
- `RestingBP`: Resting blood pressure.
- `Cholesterol`: Serum cholesterol in mg/dl.
- `FastingBS`: Fasting blood sugar > 120 mg/dl (1 = true; 0 = false).
- `RestingECG`: Resting electrocardiographic results (e.g., Normal, ST, LVH).
- `MaxHR`: Maximum heart rate achieved.
- `ExerciseAngina`: Exercise-induced angina (Y/N).
- `Oldpeak`: ST depression induced by exercise relative to rest.
- `ST_Slope`: The slope of the peak exercise ST segment (e.g., Up, Flat, Down).
- `HeartDisease`: Target variable (1 = heart disease, 0 = normal).

Methodology

1. Data Collection

The dataset was loaded into a pandas DataFrame. Initial checks included `df.head()`, `df.info()`, and `df.describe()` to understand its structure and basic statistics.

2. Data Visualization

Several visualizations were generated to gain insights:

- **Correlation Heatmap**: To understand the linear relationships between numerical features.
- **Heart Disease Distribution**: A count plot to visualize the balance of the target variable (`HeartDisease`).
- **Distribution of Features**: Histograms for all numerical features to observe their distributions.

3. Data Cleaning

- Missing values were checked using `df.isnull().sum()` . The notebook implies no significant missing values that required imputation or removal, as no explicit steps were taken.
- Unique values in categorical columns were examined to understand their diversity.

4. Data Transformation (Encoding + Scaling)

- **One-Hot Encoding**: Applied to nominal categorical features (`ST_Slope` , `RestingECG` , `ChestPainType` , `ExerciseAngina`) using `pd.get_dummies()` with `drop_first=True` to avoid multicollinearity.
- **Label Encoding**: Applied to the binary categorical feature `Sex` using `LabelEncoder()` .
- **Scaling**: Numerical features were scaled using `StandardScaler` to standardize their ranges, which is beneficial for many machine learning algorithms.
- The transformed features were stored in `X` and the target variable in `y` .

5. Initial Model Training

- The data was split into training (80%) and testing (20%) sets using `train_test_split` with `random_state=42` .
- A `LogisticRegression` model (with `max_iter=1000`) was trained as a baseline.

6. Model Evaluation (Baseline Model)

Predictions were made on the test set. The Logistic Regression model was evaluated using:

- `accuracy_score`
- `classification_report` (providing precision, recall, F1-score for each class)
- `confusion_matrix` (visualized with a heatmap)
- `roc_auc_score` and ROC curve plot

7. Hyperparameter Tuning (Random Forest Classifier)

A `RandomForestClassifier` was chosen for further optimization.

a. GridSearchCV

- **Hyperparameters Tuned**:
  - `n_estimators` : [100, 200]
  - `max_depth` : [5, 10, None]
  - `min_samples_split` : [2, 5, 8]
  - `min_samples_leaf` : [1, 2]
- `GridSearchCV` with 5-fold cross-validation and `accuracy` scoring was used.

- The best estimator from Grid Search was evaluated using `classification_report` and `accuracy_score`.

b. RandomizedSearchCV

- **Hyperparameters Tuned (distributions)**:
  - `n_estimators`: `randint(100, 300)`
  - `max_depth`: [None, 10, 20, 30, 40]
  - `min_samples_split`: `randint(2, 11)`
  - `min_samples_leaf`: `randint(1, 5)`
  - `max_features`: ['sqrt', 'log2']
- `RandomizedSearchCV` with 50 iterations, 5-fold cross-validation, `accuracy` scoring, `verbose=1`, and `n_jobs=-1` was used.
- The best estimator from Randomized Search was evaluated using `classification_report` and `accuracy_score`.
- **Selected Best Model**: The notebook indicates that the `GridSearchCV` best estimator yielded the best metric scores. This model's confusion matrix and ROC curve were also plotted for a more detailed evaluation.

8. Feature Selection and Model Interpretation

- Feature importances were extracted from the `best_grid_model` (the Random Forest Classifier tuned with GridSearchCV).
- A bar plot visualized the importance of each feature, sorted in descending order.

Models Used

- Logistic Regression (Baseline)
- Random Forest Classifier (Optimized with GridSearchCV and RandomizedSearchCV)

Evaluation Metrics

- **Accuracy Score**: Proportion of correctly classified instances.
- **Classification Report**: Includes Precision, Recall, F1-score, and Support for each class.
- **Confusion Matrix**: Shows the number of true positives, true negatives, false positives, and false negatives.
- **ROC AUC Score (Area Under the Receiver Operating Characteristic Curve)**: Measures the model's ability to distinguish between classes.

Results Summary (Illustrative - actual values are in the notebook)

The notebook shows that hyperparameter tuning, particularly with GridSearchCV for the Random Forest Classifier, led to improved performance over the baseline Logistic Regression model.

Model Configuration	Accuracy	Other Key Metrics (Precision, Recall, F1, ROC AUC)
Logistic Regression (Baseline)	0.85	0.86, 0.85, 0.85, 0.93

Random Forest (GridSearchCV Best Estimator)	0.875	*0.88, 0.88, 0.88, 0.93*
Random Forest (RandomizedSearchCV Best Estimator)	0.86	*0.87, 0.87, 0.87, 0.93*

Feature importance analysis highlighted the most significant predictors for heart failure according to the trained Random Forest model.

Conclusion

This project successfully implemented a machine learning pipeline for heart failure prediction. Through data preprocessing, model selection, and hyperparameter tuning, a Random Forest Classifier was developed that demonstrated good predictive accuracy. The feature importance analysis provided valuable insights into the key factors contributing to heart failure risk as identified by the model.

How to Run

1. Ensure you have Python installed.
2. Install the required libraries:

   ```
   pip install pandas scikit-learn matplotlib seaborn
   ```

3. Ensure the dataset `heart.csv` is in the same directory as the notebook `classification.ipynb`.
4. Open and run the `classification.ipynb` notebook in a Jupyter environment (e.g., Jupyter Notebook, JupyterLab, VS Code with Python extension).