Contents

Documentation for Genetic and Environmental Data Classification Program

Overview

This program is designed to classify data based on genetic, environmental, and demographic features using machine learning algorithms. It loads datasets from CSV files, preprocesses the data, trains multiple classification models (Decision Tree, Random Forest, and Support Vector Machine), evaluates their performance, and visualizes the results.

Requirements

To run this program, you need to have the following Python libraries installed:

- `numpy`
- `pandas`
- `matplotlib`
- `seaborn`
- `scikit-learn`

Data Files

The program requires the following CSV files to be present in the working directory:

1. `genetic_data_train.csv`: Contains genetic data.
2. `EnvironmentalDataSet.csv`: Contains environmental data.
3. `world_population_data.csv`: Contains demographic data.

Usage

1. Load the Data: The program loads the datasets from the specified CSV files.

2. Preprocess the Data: It selects numeric columns, handles missing values, and combines the datasets into a single feature set.

3. Train-Test Split: The data is split into training and testing sets.

4. Model Training: Three classification models (Decision Tree, Random Forest, and Support Vector Machine) are trained on the training data.

5. Model Evaluation: The performance of each model is evaluated using classification metrics and confusion matrices.

6. Visualization: The program visualizes feature importances for the Random Forest model.

Code Structure

Step 1: Data Loading

```
genetic_data = pd.read_csv('genetic_data_train.csv')

environmental_data = pd.read_csv('EnvironmentalDataSet.csv')

demographic_data = pd.read_csv('world_population_data.csv')
```

Step 2: Data Preprocessing

```
genetic_data_numeric = genetic_data.select_dtypes(include=np.number)

environmental_data_numeric = environmental_data.select_dtypes(include=np.number)

demographic_data_numeric = demographic_data.select_dtypes(include=np.number)


min_rows = min(len(genetic_data_numeric), len(environmental_data_numeric),
len(demographic_data_numeric))



df = df.dropna()  # Drop rows with NaN in any column
```

```
df = df.fillna(df.mean())  # Fill NaN values with the mean of each column
```

Step 3: Data Exploration

```
print("Genetic Data Shape:", genetic_data_numeric.shape)
print("Environmental Data Shape:", environmental_data_numeric.shape)
print("Demographic Data Shape:", demographic_data_numeric.shape)
```

Step 4: Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Step 5: Model Training

```
dt_model = DecisionTreeClassifier(random_state=42)
rf_model = RandomForestClassifier(random_state=42)
svm_model = SVC(random_state=42)

dt_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)
svm_model.fit(X_train, y_train)
```

Step 6: Model Evaluation

```
def evaluate_model(y_true, y_pred, model_name):
    print(f"Evaluation for {model_name}:")
    print(classification_report(y_true, y_pred))
    cm = confusion_matrix(y_true, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'Confusion Matrix for {model_name}')
```

```
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

Step 7: Visualization of Results

```
importances = rf_model.feature_importances_
plt.bar(range(X.shape[1]), importances[indices], align="center")
```

Conclusion

This program provides a comprehensive approach to classifying data based on genetic, environmental, and demographic features. By utilizing various machine learning models, it allows for effective evaluation and visualization of the results, making it a valuable tool for data analysis in related fields.

Future Work

- Hyperparameter Tuning: Implement techniques such as Grid Search or Random Search to optimize model parameters for better performance.

- Cross-Validation: Use cross-validation to ensure the robustness of the model evaluations.

- Additional Models: Explore other classification algorithms like Gradient Boosting or Neural Networks for potentially improved accuracy.

- Feature Engineering: Investigate the creation of new features or transformations of existing features to enhance model performance.

License