IT - 462 Exploratory Data Analysis

Assignment: Scikit - Learn Package

Group number – 4

Group members:

- Ghori Zeel Jivrajbhai (202201287)
- Aryankumar Panchasara (202201056)
- Nischay Agrawal (202411031)

Professor: Gopinath Panda

October 22, 2024

Scikit-Learn Package

Overview

Scikit-learn is a highly regarded library for machine learning in Python, known for its user-friendly interface and comprehensive range of algorithms. This guide will walk you through the steps to utilize Scikit-learn effectively, covering data loading, preprocessing, model training, evaluation, and making predictions.

To install library - *pip install scikit-learn*

Data Preprocessing

Data preprocessing is essential in the machine learning workflow. Scikit-learn provides various tools to clean, transform, and prepare your dataset:

1. **Handling Missing Values**
   Missing data can adversely affect your model's performance. Scikit-learn offers methods for addressing this:
   - *SimpleImputer*: Substitutes missing values using strategies like mean, median, most frequent, or a constant value.

   ```
   from sklearn.impute import SimpleImputer imputer =
   SimpleImputer(strategy='mean')
   X_imputed = imputer.fit_transform(X)
   ```

   - *IterativeImputer*: Utilizes advanced imputation methods that leverage the relationships among features.
2. **Encoding Categorical Variables**
   Many machine learning algorithms require numerical inputs. For categorical data, Scikit-learn includes several encoding methods:
   - *OneHotEncoder*: Converts categorical features into a format that can be provided to ML algorithms to do a better job in prediction.

   ```
   from sklearn.preprocessing import OneHotEncoder
   encoder = OneHotEncoder(sparse=False)
   X_encoded = encoder.fit_transform(X_categorical)
   ```

   - *OrdinalEncoder*: Assigns an integer to each unique category.
   - *LabelEncoder*: Encodes target labels to values ranging from 0 to n_classes-1.
3. **Feature Scaling**
   Standardizing features can be crucial for many algorithms. Scikit-learn offers several scaling techniques:

- *StandardScaler*: Normalizes features by removing the mean and scaling to unit variance.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

- *MinMaxScaler*: Scales features to a specified range, typically [0, 1].
- *RobustScaler*: Scales features based on statistics that are resistant to outliers.

4. **Feature Selection**
   Identifying the most relevant features can enhance model performance and mitigate overfitting. Scikit-learn provides various techniques for feature selection:
   - *VarianceThreshold*: Eliminates low-variance features.

```
from sklearn.feature_selection import VarianceThreshold
selector = VarianceThreshold(threshold=0.1)
X_reduced = selector.fit_transform(X)
```

   - *SelectKBest*: Chooses the k best features based on statistical tests.
   - *RFE (Recursive Feature Elimination)*: Iteratively considers smaller sets of features.
5. **Dimensionality Reduction**
   For datasets with many features, reducing dimensionality can be advantageous. Scikit-learn includes several methods for this:
   - *PCA (Principal Component Analysis)*: A linear technique to reduce dimensions by transforming to a lower-dimensional space.

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(X)
```

   - *t-SNE*: A non-linear technique useful for visualizing high-dimensional data.

Training Models

Scikit-learn provides a uniform interface for training various machine learning models:

1. **Supervised Learning**
   For labelled data, the following types of tasks are available:
   - **Classification** (predicting categories):
     - Logistic Regression
     - Support Vector Machines (SVM)
     - Decision Trees and Random Forests

        - K-Nearest Neighbours
        - Naive Bayes

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
```

    - **Regression** (predicting continuous values):
        - Linear Regression
        - Ridge and Lasso Regression
        - Support Vector Regression
        - Decision Tree and Random Forest Regressors

2. **Unsupervised Learning**
   For unlabelled data, you can perform:
    - **Clustering** (grouping similar data points):
        - K-Means
        - DBSCAN
        - Hierarchical Clustering
    - **Dimensionality Reduction**:
        - PCA
        - t-SNE
3. **Model Training Workflow**
   The typical steps for training a model in Scikit-learn are:
        - Initialize the model with the desired parameters.
        - Use the *fit()* method on your training data.
        - Make predictions or perform further analysis with the trained model.
4. **Hyperparameter Tuning**
   Most models include hyperparameters that need configuration prior to training. Scikit-learn provides several tools for this:
        - *GridSearchCV*: Conducts an exhaustive search over specified parameter values.
        - *RandomizedSearchCV*: Performs a random search on hyperparameters.
        - *Cross-validation*: Implements methods like k-fold cross-validation to evaluate model performance.

```
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [0.1, 1, 10]}
grid_search = GridSearchCV(LogisticRegression(), param_grid)
grid_search.fit(X_train, y_train)
```

Model Evaluation

Evaluating the performance of your model is vital for understanding its effectiveness and for comparing different models:

1. **Classification Metrics**
    - *Accuracy*: The ratio of correct predictions.
    - *Precision*: The ratio of true positives to total positive predictions.
    - *Recall*: The ratio of true positives to total actual positives.
    - *F1 Score*: The harmonic mean of precision and recall.
    - *ROC-AUC*: The area under the Receiver Operating Characteristic curve.
2. **Regression Metrics**
    - Mean Squared Error (MSE)
    - Root Mean Squared Error (RMSE)
    - Mean Absolute Error (MAE)
    - R-squared (Coefficient of Determination)
3. **Cross-Validation**
   Cross-validation assesses how well a model generalizes to independent datasets. Scikit-learn provides functions for various cross-validation methods:
    - K-Fold Cross-Validation
    - Stratified K-Fold (for classification tasks)
    - Leave-One-Out Cross-Validation
4. **Learning Curves**
   Learning curves visualize a model's performance as a function of the training set size, helping identify overfitting or underfitting.

Making Predictions

After training and evaluating your model, you can make predictions on new, unseen data:

1. **Prediction Methods**
    - *predict()*: Returns predicted class labels for classification or predicted values for regression.
    - *predict_proba()*: For classification, gives probability estimates for each class.
2. **Interpreting Predictions**
   Understanding the rationale behind a model's predictions is crucial, especially in sensitive contexts. Techniques for interpreting models include:

- Feature importances (for tree-based models)
- Coefficient values (for linear models)
- SHAP (SHapley Additive exPlanations) values

Conclusion

Scikit-learn offers an extensive array of tools to support the entire machine learning workflow, from preprocessing to model evaluation and predictions. By grasping these concepts and applying them through Scikit-learn, you can effectively address a variety of machine learning challenges.
While Scikit-learn provides powerful capabilities, successful machine learning projects also rely on a thorough understanding of your data, careful feature engineering, and thoughtful interpretation of outcomes.

Google Drive link:

https://drive.google.com/drive/folders/1ewK9FrEj5IU8slg5qT73FJZFvfc1I1AW?usp=sharing

Github link:

Exploratory_Data_Analysis/Group4_sklearn at main · zeelghori27012004/Exploratory_Data_Analysis