

Exploratory Data Analysis - IT 462
Assignment - 3
Scikit-Learn

Group:10

Team Members:

- 1. Hammad Farid - 202312108**
- 2. Anmol Rangwani - 202312027**
- 3. Yuvraj Bhadoriya - 202411002**

Scikit-Learn (sklearn) Overview

Scikit-learn is an open-source machine-learning library in Python built on top of libraries like NumPy, SciPy, and Matplotlib. It provides simple and efficient tools for predictive data analysis, emphasizing ease of use, versatility, and a consistent interface for various machine-learning models.

Key Features of Scikit-Learn

1. **Supervised Learning Algorithms:** Scikit-learn offers a wide range of algorithms for supervised learning, including classification and regression:
 - Classification: SVM, Decision Trees, Random Forests, k-Nearest Neighbors (k-NN), Logistic Regression, and Naive Bayes.
 - Regression: Linear Regression, Lasso, Ridge Regression, and ElasticNet.
2. **Unsupervised Learning Algorithms:** Scikit-learn also supports unsupervised learning methods for tasks like clustering and dimensionality reduction:
 - Clustering: K-Means, DBSCAN, and hierarchical clustering.
 - Dimensionality Reduction: PCA, LDA, and t-SNE.
3. **Model Selection and Evaluation:** Tools for model selection, tuning, and evaluation make it easy to assess and compare different models:
 - Cross-validation techniques, such as K-fold cross-validation.
 - Metrics for evaluating models (accuracy, F1 score, confusion matrix, and ROC-AUC for classification; R^2 and mean squared error for regression).
4. **Data Preprocessing and Transformation:** Scikit-learn includes modules for preparing and transforming data, essential for handling real-world datasets:
 - Data Scaling: StandardScaler, MinMaxScaler, and RobustScaler.
 - Encoding: LabelEncoder for categorical variables, and OneHotEncoder for one-hot encoding.
 - Imputation: SimpleImputer for filling missing values with mean, median, or mode.
5. **Pipeline API:** Scikit-learn's Pipeline feature allows chaining multiple steps in data processing and modeling, making workflows seamless and reproducible. It's especially useful for hyperparameter tuning, as Pipeline works well with GridSearchCV and RandomizedSearchCV for parameter optimization.

Label Encoding with Scikit-Learn:

Label Encoding is a technique used to convert categorical text labels into numerical values, enabling their use in machine learning models.

Key Steps:

- **Import the LabelEncoder Class:** Start by importing the `LabelEncoder` class from the scikit-learn library. `(from sklearn.preprocessing import LabelEncoder)`
- **Initialize the Encoder:** Create an instance of the `LabelEncoder`. `(le = LabelEncoder())`
- **Fit and Transform the Data:** Use the encoder to fit and transform a specific categorical column, converting the categorical labels into numerical values.

Standard Scaler in Scikit-Learn:

Standard Scaler is a preprocessing technique in scikit-learn standardize the features, ensuring they have a mean of 0 and a standard deviation of 1, which improves model performance.

Steps to Use Standard Scaler in Scikit-Learn

1. **Import StandardScaler:** Import the `StandardScaler` class from scikit-learn.
2. **Initialize the Scaler:** Create an instance of the `StandardScaler`.
3. **Fit the Scaler:** Fit the scaler on the training data to compute the mean and standard deviation.
4. **Transform the Data:** Apply the scaler to transform both the training and test datasets.

SelectKBest method in scikit-learn:

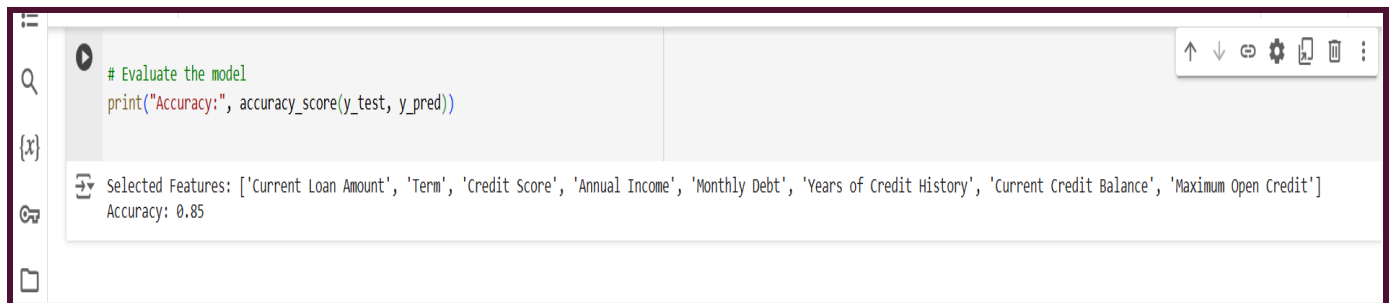
The **SelectKBest** method in scikit-learn is a feature selection technique that helps identify the top k features in a dataset based on their relevance to the target variable. It evaluates each feature individually using a scoring function and selects the best k features, enhancing model performance by reducing irrelevant or redundant features.

Using SelectKBest with f_regression

1. **Selector Creation:** `SelectKBest(score_func=f_regression, k=k)` creates a selector that scores features based on their correlation with the target variable (`f_regression` is used for regression tasks and computes the F-statistic).
2. **Feature Selection:**
 - `fit_transform()`: Trains the selector on the training data and reduces it to the top k features.

- **get_support():** Retrieves the indices of selected features, allowing them to be viewed or used in further analysis.
3. **Model Training:** Once the top k features are selected, they can be transformed, scaled, and used for model training, as demonstrated with logistic regression in this example.

Using SelectKBest improves model interpretability and performance by reducing the dataset to its most informative features.



```
# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Selected Features: ['Current Loan Amount', 'Term', 'Credit Score', 'Annual Income', 'Monthly Debt', 'Years of Credit History', 'Current Credit Balance', 'Maximum Open Credit']
Accuracy: 0.85

Here **k=8** specifies that the method should select the top 8 features from the dataset based on their relevance to the target variable(['Loan_status']).

PCA(Principal Component Analysis):

Principal Component Analysis (PCA) is a dimensionality reduction technique in scikit-learn that simplifies a dataset with many features by projecting it onto a smaller set of uncorrelated components, called **principal components**. **Key Parameters in PCA (scikit-learn):**

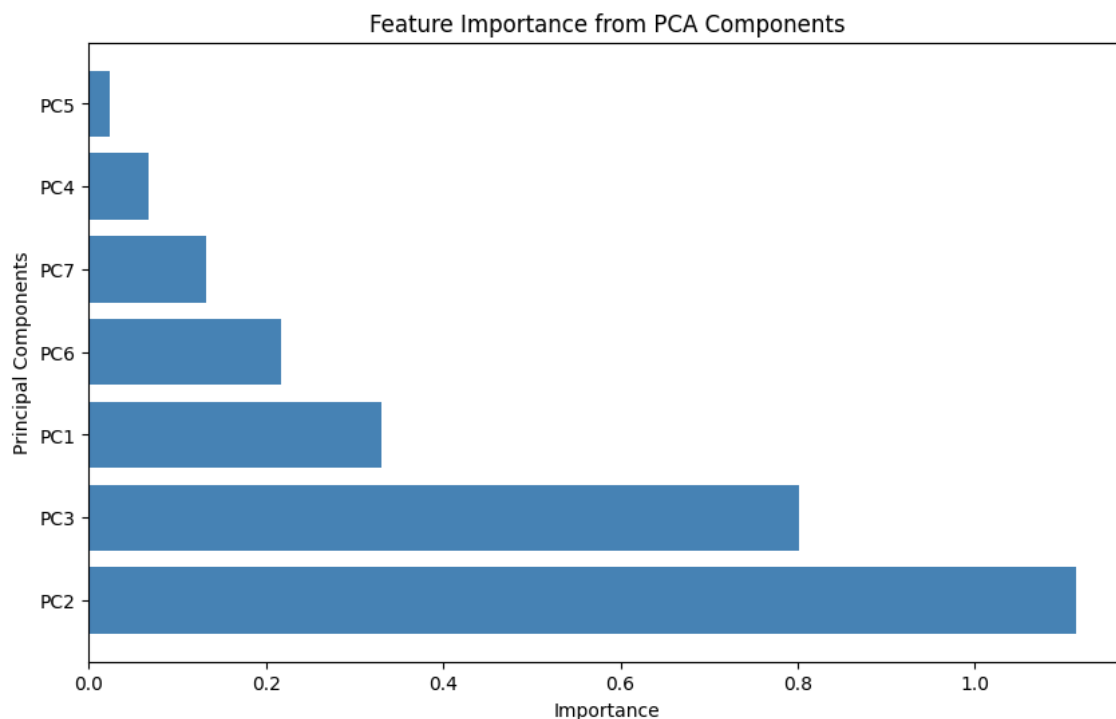
1. **n_components:** Specifies the number of principal components to retain.
2. **explained_variance_ratio_:** After fitting PCA, this attribute provides the variance ratio explained by each principal component, helping decide how much information is captured by each one.
3. **svd_solver:** Defines the algorithm for computing PCA (e.g., auto, full, arpack). auto lets PCA choose the best method based on input.

We have used PCA to reduce dimensionality and capture 95% of the dataset's variance, then fits a **Logistic Regression** model on these components to classify data.

Key Steps:

1. **Determine Components:** Fit **PCA** on `X_train_scaled` and select the number of components needed to reach **95% explained variance**.
2. **Transform Data:** Apply PCA with the selected components to reduce `X_train` and `X_test` to `X_train_pca` and `X_test_pca`.
3. **Model Training:** Train **Logistic Regression** on `X_train_pca` and evaluate on `X_test_pca` using accuracy, confusion matrix, and classification report.
4. **Feature Importance:** Extract the absolute values of logistic regression coefficients to assess the importance of each PCA component, and visualize these in a bar plot.

This approach identifies the principal components most relevant for classification, enhancing interpretability and model efficiency.



Classification Report

The **Classification Report** provides metrics for a classification model, including:

- **Precision:** True positives / Predicted positives
- **Recall:** True positives / Actual positives
- **F1-Score:** Harmonic mean of precision and recall
- **Support:** Actual occurrences of each class

Confusion Matrix

The **Confusion Matrix** displays prediction outcomes:

- **True Positives (TP)**: Correct positive predictions
- **True Negatives (TN)**: Correct negative predictions
- **False Positives (FP)**: Incorrect positive predictions
- **False Negatives (FN)**: Incorrect negative predictions

It visually summarizes model performance and helps identify misclassifications.

```
Confusion Matrix:
[[ 2 14]
 [ 0 84]]

Classification Report:

```

	precision	recall	f1-score	support
0	1.00	0.12	0.22	16
1	0.86	1.00	0.92	84
accuracy			0.86	100
macro avg	0.93	0.56	0.57	100
weighted avg	0.88	0.86	0.81	100

Colab link : [s k learn colab link](#)

Github link:

<https://github.com/yuvraj-daiict/exploratory-data-analysis/tree/main/assignment-3>