Scikit Learn

Professor: Gopinath Panda

Group No: 19

Kathan Vora - 202411073
Aniruddha Shinde - 202411065
Monson Reji Verghese - 202411039

Date: October 24, 2024

Contents

1 Introduction

This report provides a step-by-step analysis of an insurance dataset using scikit-learn. The aim is to predict insurance charges based on key features such as age, BMI, and children. The workflow includes data loading, outlier detection and treatment, numerical feature scaling, feature selection, and model training. Scikit-learn is used throughout the process for seamless machine learning tasks.

2 Importing Libraries & Data

2.1 Purpose of Packages

- **numpy**: Provides support for numerical computations and array manipulation.
- **pandas**: Used for data manipulation and analysis, including handling time-series data.
- **scikit-learn**: Used for building and evaluating the linear regression model for rainfall prediction.

2.2 How it Works

Libraries like pandas and numpy are used for data handling, while scikit-learn modules support data splitting, scaling, and modeling. The dataset is read from a URL into a DataFrame for further operations.

2.3 Key Parameters

- `pd.read_csv(url)`: Reads the dataset from a specified URL.
- `df.head()`: Displays the first five rows of the dataset.
- `df.describe()`: Provides summary statistics of the data.

3 Outlier Detection and Treatment Using IQR

3.1 Purpose

Outliers are detected using the Interquartile Range (IQR) method to improve model performance.

3.2 How it Works

IQR identifies outliers as data points that lie beyond 1.5 times the IQR from Q1 or Q3. After detection, rows with outliers are removed from the dataset. The following function is used for outlier removal:

```
def remove_outliers_iqr(df):
    numerical_columns = ['age', 'bmi', 'children', 'charges']
    for col in numerical_columns:
        q25, q75 = np.quantile(df[col], 0.25), np.quantile(df[col], 0.75)
        iqr = q75 - q25
        cut_off = iqr * 1.5
        lower, upper = q25 - cut_off, q75 + cut_off

        # Remove rows with outliers
        df = df[(df[col] >= lower) & (df[col] <= upper)]

    return df
```

```
# Apply the function to remove outliers
df_cleaned = remove_outliers_iqr(df)
```

3.3 Key Parameters

- Q1, Q3: Boundaries for the 25th and 75th percentiles.
- IQR: Difference between Q3 and Q1.
- Cut-off: 1.5 times the IQR for identifying outliers.

3.4 Result

- Total outliers detected: 148
- Outlier distribution:
  - Age: 0
  - BMI: 9
  - Children: 0
  - Charges: 139

After applying the outlier removal function, the dataset shape is `(1191, 7)`, and the updated outlier distribution shows:

- Total outliers across all columns: 0
- Age: 0, BMI: 0, Children: 0, Charges: 0

4 Numerical Feature Scaling Using Scikit-learn

4.1 Purpose

Feature scaling ensures all numerical features are on a similar scale to prevent bias during model training.

4.2 How it Works

The StandardScaler from scikit-learn standardizes the numerical features by subtracting the mean and dividing by the standard deviation.

4.3 Key Parameters

- `fit()`: Computes the mean and standard deviation.
- `transform()`: Scales the data using the computed statistics.

4.4 Code

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df[['age', 'bmi', 'children']] = scaler.fit_transform(
    df[['age', 'bmi', 'children']])
```

5 Feature Selection Using SelectKBest and f_regression

5.1 Purpose

SelectKBest identifies the top features correlated with the target variable using statistical tests.

5.2 How it Works

f_regression evaluates the relationship between each feature and the target, retaining only the top k features with the highest scores.

5.3 Key Parameters

- k: Number of features to select.
- Score function: f-statistics for feature relevance.

5.4 Code

```
from sklearn.feature_selection import SelectKBest, f_regression

X = df_cleaned[['age', 'bmi', 'children']]
y = df_cleaned['charges']

select_k_best = SelectKBest(score_func=f_regression, k=3)
X_selected = select_k_best.fit_transform(X, y)

# Get selected feature names
selected_features = X.columns[select_k_best.get_support()]
```

5.5 Result

- Selected features: `[age', 'bmi', 'smoker']`
- R2 score: 0.7776932310583374

6 Principal Component Analysis (PCA)

6.1 Purpose

PCA reduces the dimensionality of the dataset while preserving as much variance as possible.

6.2 How it Works

PCA transforms the original features into a new set of uncorrelated variables (principal components) ordered by the amount of variance they capture.

6.3 Code

```
# @title PCA
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.decomposition import PCA
from sklearn.metrics import r2_score
import pandas as pd

# Initialize PCA for feature reduction
n_components = [1, 2, 3, 4, 5, 6, 7]  # Number of principal components you want to keep
for n in n_components:
  pca = PCA(n_components=n)

  # Fit PCA on the training data and transform both train and test sets
  X_train_pca = pca.fit_transform(X_train)
  X_test_pca = pca.transform(X_test)

  # Print the explained variance ratio for each component
  print(f"Explained variance ratio by each component (n = {n}): {pca.explained_variance_ratio_}")

  # Initialize and fit the regression model
  model = LinearRegression()
  model.fit(X_train_pca, y_train)

  # Make predictions
  predictions = model.predict(X_test_pca)

  # Evaluate the model
  r2 = r2_score(y_test, predictions)
  print(f'R2 score: {r2}')
```

6.4 Result

- Explained variance ratio by each component (n = 1): [0.58654434]
  R2 score: 0.0019256243566018183
- Explained variance ratio by each component (n = 2): [0.58654434 0.1033703]
  R2 score: 0.024027130631683047
- Explained variance ratio by each component (n = 3): [0.58654434 0.1033703 0.10101865]
  R2 score: 0.05374806905540119
- Explained variance ratio by each component (n = 4): [0.58654434 0.1033703 0.10101865 0.09768599]
  R2 score: 0.05139156881011908
- Explained variance ratio by each component (n = 5): [0.58654434 0.1033703 0.10101865 0.09768599 0.06356361]
  R2 score: 0.6515352353173205
- Explained variance ratio by each component (n = 6): [0.58654434 0.1033703 0.10101865 0.09768599 0.06356361 0.0240791]
  R2 score: 0.6515442490500358
- Explained variance ratio by each component (n = 7): [0.58654434 0.1033703 0.10101865 0.09768599 0.06356361 0.0240791 0.01910562]
  R2 score: 0.7607243036470185

Based on the experiments we can say that the optimal number of principal components is 5 as there is a drastic difference in R2 score when we go from 4 to 5.

7 Recursive Feature Elimination (RFE)

7.1 Purpose

RFE selects the most important features by recursively removing the least important ones.

7.2 How it Works

A model is trained on the dataset, and features are ranked based on their importance. The least important features are removed iteratively.

7.3 Code

```
# @title RFE
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
from sklearn.metrics import r2_score

# Initialize the regression model
model = LinearRegression()

# Initialize RFE to select the top k features
k = 3  # Number of top features you want to select
rfe = RFE(estimator=model, n_features_to_select=k)

# Fit RFE on the training data
X_train_rfe = rfe.fit_transform(X_train, y_train)

# Transform the test set to keep only the selected features
X_test_rfe = rfe.transform(X_test)

# Print the selected feature names
selected_features = X.columns[rfe.get_support()]
print(f"Selected features: {selected_features.tolist()}")

# Fit the regression model using the selected features
model.fit(X_train_rfe, y_train)

# Make predictions
predictions = model.predict(X_test_rfe)

# Evaluate the model
r2 = r2_score(y_test, predictions)
print(f'R2 score: {r2}')
```

7.4 Result

- Selected features: `['age', 'bmi', 'smoker']`
- R2 score: 0.7776932310583374

8 Feature Importance

8.1 Purpose

Feature importance indicates the contribution of each feature in predicting the target variable.

8.2 How it Works

Various models (like decision trees) provide built-in feature importance metrics, indicating the significance of each feature in the decision-making process.

8.3 Code

```
model = ExtraTreesRegressor(n_estimators=100, random_state=42)

# Fit the model to the training data
model.fit(X_train, y_train)

# Use SelectFromModel to select important features based on the fitted model
selector = SelectFromModel(model, threshold="mean")
X_train_selected = selector.fit_transform(X_train, y_train)
X_test_selected = selector.transform(X_test)

# Print the selected feature names
selected_features = X.columns[selector.get_support()]
print(f"Selected features: {selected_features.tolist()}")
```

8.4 Result

- Selected features: `['age', 'bmi', 'smoker']`
- R2 score: 0.8073190928332266

You can access our work here :GitHub Repository
Google Colab Link (please access using college ID): Colab Link