Machine Learning

Project 02

08.07.2024

Seyedehsadaf Edalat (30372928)
Aziz Fahmi (30461097)

Table of Content

Introduction

Goal: Develop a predictive model for gas consumption using various weather and solar-related variables. By applying Principal Component Analysis (PCA) and Linear Regression, we aim to determine the most significant factors influencing gas consumption and predict gas usage.

Techniques:

- PCA: Simplifies data by identifying key variables.
- Linear Regression: Predicts gas consumption based on significant predictors.

Data Description

Data source and period

Heating data from an energy management system

Key variables

Gas consumption [kWh/day] (Target)
Solar pump [h/day]
Solar yield [kWh/day]
Outdoor temperature [°C]
Sunshine duration [h/day]
Valve [h/day]

Data cleaning

Encoded categorical data using LabelEncoder. Standardized numerical features with StandardScaler.

Data Exploration

Variable	Mean	Median	Std Dev
Sunshine duration [h/day]	4.56	3.85	3.33
Outdoor temperature [°C]	10.41	9.55	5.85
Solar yield [kWh/day]	43.52	34.35	37.73
Solar pump [h/day]	2.83	2.70	2.12
Valve [h/day]	10.09	9.05	8.11
Gas consumption [kWh/day]	278.19	288.80	152.62

Principal Component Analysis (PCA)

Purpose: Reduce the dimensionality of the data while retaining most of the variance.

Process:

- **Standardization:** Scale the features to have mean=0 and variance=1.
- **Covariance Matrix Computation:** Calculate the covariance matrix to understand how features vary together.
- **Eigen Decomposition:** Identify principal components by computing eigenvalues and eigenvectors.
- **Selection of Principal Components:** Choose the number of components that explain a significant portion of the variance.

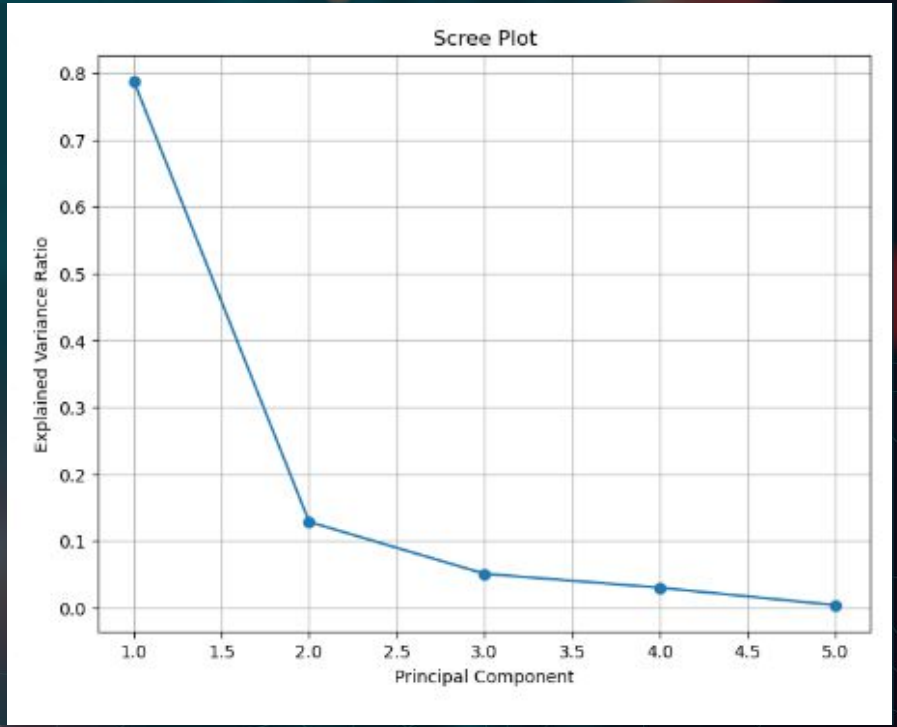
```
# Standardize numerical features
scaler = StandardScaler()
encoded_data[numerical_cols] = scaler.fit_transform(encoded_data[numerical_cols])

# Split the data into features (X) and target variable (y)
X = encoded_data.drop('Gas consumption [kWh/day]', axis=1)
y = encoded_data['Gas consumption [kWh/day]']

# Perform PCA
pca = PCA()
X_pca = pca.fit_transform(X)
explained_variance_ratio = pca.explained_variance_ratio_
```

Principal Components Scree plot

Explained Variance Ratio by Components:

- PC1: 78,751883%
- PC2: 12,825661%
- PC3: 5,066179%
- PC4: 2,978784%
- PC5: 0,377493%

The first two principal components explained approximately 91.58% of the total variance in the dataset, with the first component alone explaining 78.75%.

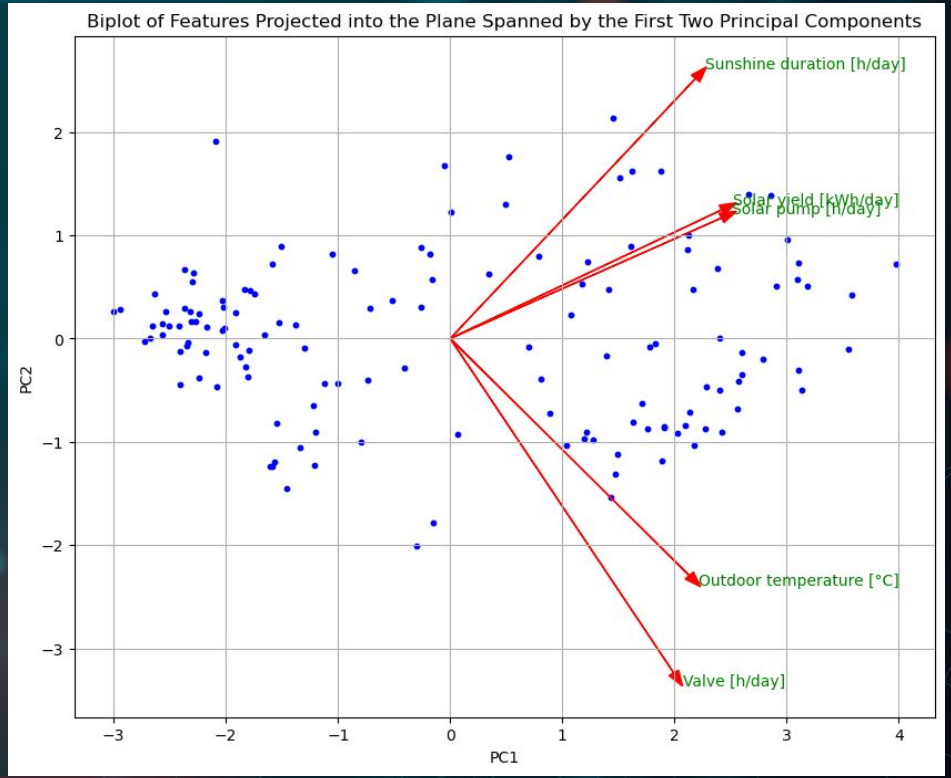


Principal Components Biplot

The blue points on the plot represent the samples or observations in the dataset, projected onto the new principal component space.

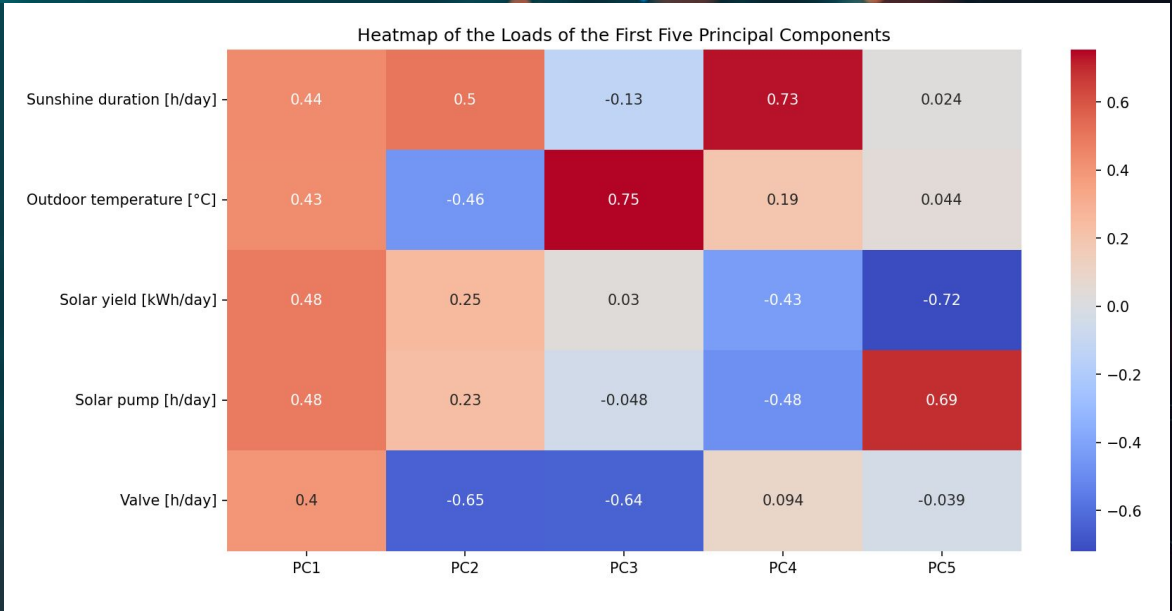
The arrows (vectors) represent the original features.

Their directions and lengths indicate their contributions to the principal components.



Heatmap of loads

The heatmap, also known as the **loading matrix** or eigenvectors of the principal components, visualizes the contributions of original features to each principal component. Each row corresponds to an original feature, and each column represents a principal component. Warm colors indicate a positive loading (or weight), while cool colors indicate an inverse relationship.



Coefficients (First Two Components): [-0.44631241 0.36994004]

Linear Regression Model

Purpose: Predict gas consumption using other features in the dataset.

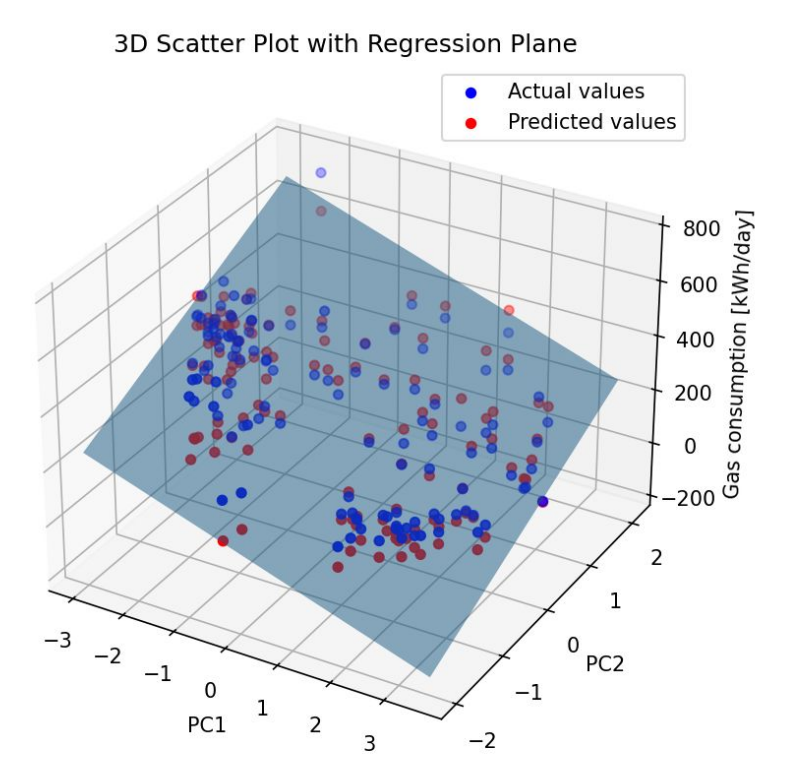
Process:

- Data Preparation: Features were standardized, and PCA was performed to reduce dimensionality.
- Model Training: Trained on original data, Trained on PCA-transformed data.
- Model Evaluation: Mean Squared Error (MSE), R-squared (R^2).

```
# Select the first two principal components
X_pca_two_components = X_pca[:, :2]

# Split the data into training and testing sets using the first two principal components
X_train, X_test, y_train, y_test = train_test_split(X_pca_two_components, y, test_size=0.2, random_state=42)

# Fit a linear regression model
lr = LinearRegression()
lr.fit(X_train, y_train)

# Evaluate the model's performance
y_pred = lr.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
model_score = lr.score(X_test, y_test)
```

Linear Regression Model

	Model Score	MSE
Linear Regression (first 2 components) Training	0.89	0.10
Linear Regression (first 2 components) Testing	0.87	0.15
Linear Regression (original data) Training	0.92	0.07
Linear Regression (original data) Testing	0.91	0.10



THANKS FOR YOUR ATTENTION!