**Lab Manual**

**Subject: Machine Learning**

**Course Code AI-414**

**By**

**Javed Mehmood 2023-BS-AI-029**

**Supervised by Sir Saeed**

**Degree name: BSAI-4A**

**Subject name: Machine Learning**

**DEPARTMENT OF COMPUTATIONAL SCIENCES FACULTY OF INFORMATION TECHNOLOGY**

The University of Faisalabad



|  |  |
| --- | --- |
|  | **Table of Content** |
| **Sr no** | **Regression Project** |
| **1** | Project Description |
| **2** | Data Description . |
| **3** | Code and Explanation |
| **4** | Introduction to Python and Libraries for Machine Learning, Environmental Setup |
| **5** | Handling missing values, data normalization, standardization, Data visualization |
| **6** | Data preprocessing |
| **7** | Implementing Linear Regression . |
| **8** | Implementing Decision Tree |
| **9** | Implementing Support Vector Machine |
| **10** | Evaluation Metrics |
| **11** | Training, Evaluating Model and Output of the Model |
|  | **Classification Project** |
| **12** | Project Description |
| **13** | Data Description |
| **14** | Introduction to Python and Libraries for Machine Learning, Environmental Setup |
| **15** | Handling missing values, data normalization, standardization, Data visualization |
| **16** | Data preprocessing |
| **17** | Implementing Logistic Regression |
| **18** | Implementing Decision Tree |
| **19** | Implementing Support Vector Machine |

**Regression Project**

**Salary management**

**Project Description:**

This dataset contains information about employees' years of experience and their corresponding salaries. It is commonly used for regression tasks, especially to build and evaluate models that predict salary based on experience.

### ****Columns:****

**YearsExperience**: Number of years the person has worked.

**Salary**: The corresponding annual salary in dollars.

This dataset is ideal for training a **Linear Regression** model to understand the relationship between experience and salary.

## ****Data Description:****

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| **YearsExperience** | Float | Number of years an individual has worked or has job experience. |
| **Salary** | Float | The annual salary (in dollars) corresponding to the experience. |

### ****General Notes:****

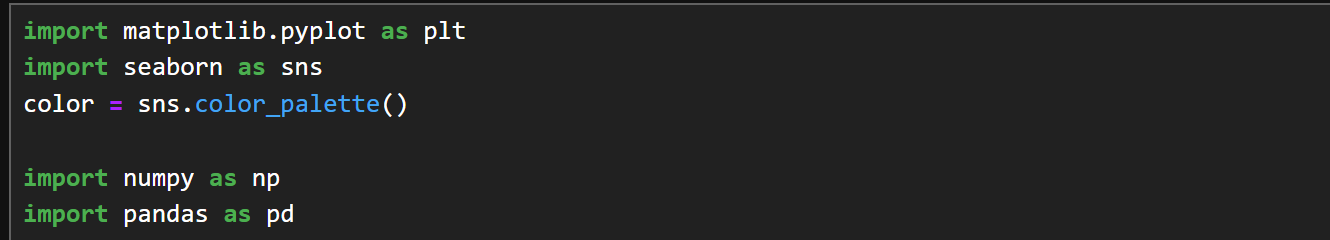
The dataset contains **continuous numerical data**.

There is a **linear relationship** expected between years of experience and salary.

Commonly used for **simple linear regression** analysis.

**Introduction to Python and Libraries for Machine Learning, Environmental Setup**

To introduce phyton basics and essential libraries used in machine learning and to setup the working enviroment.



**Code Explanation**:

| **Library** | **Explanation** |
| --- | --- |
| matplotlib.pyplot | Used for creating static, animated, and interactive plots. |
| seaborn | Built on matplotlib for easier and more attractive plotting. |
| numpy | Provides support for large, multi-dimensional arrays and math. |
| pandas | Used for data manipulation and analysis using DataFrames. |

**Summary of output:**  
  
The code imports four key Python libraries for data analysis and visualization:

* **Matplotlib** and **Seaborn** for plotting and visualizing data.
* **NumPy** for numerical computations
* **Pandas** for handling and analyzing structured data.

**Handling missing values, data normalization, standardization, Data visualization**

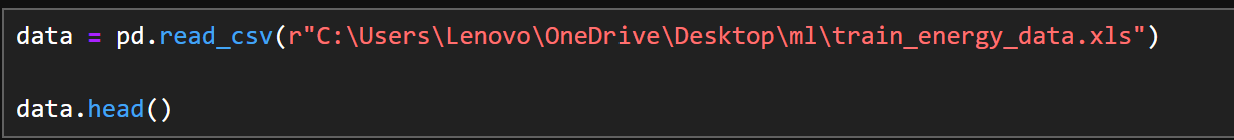
**Handling missing values** means filling in or removing data entries that are empty or not available, so they don’t affect analysis.

**Data normalization** scales data to a range, usually 0 to 1, to treat all features equally.

**Standardization** transforms data to have a mean of 0 and a standard deviation of 1, making it easier to compare.

**Data visualization** is the process of creating charts and graphs to understand patterns, trends, and insights in data.  
  
1. **Reading data:**

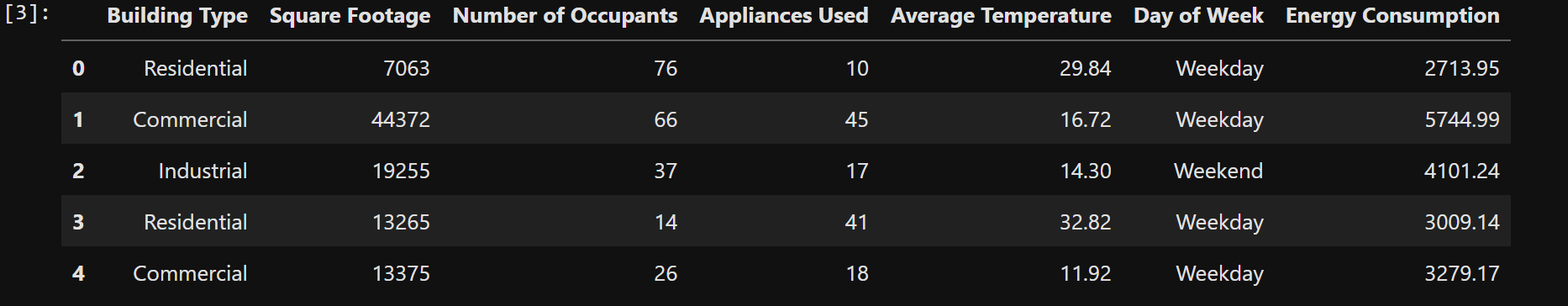
**Code:**



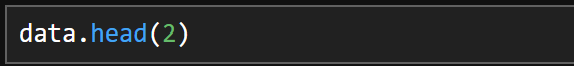
**Code explanation:**

**Reading data from dataset.**

**Output:**



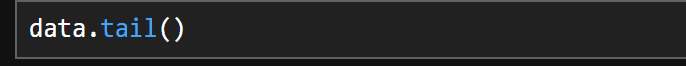
**Code:**



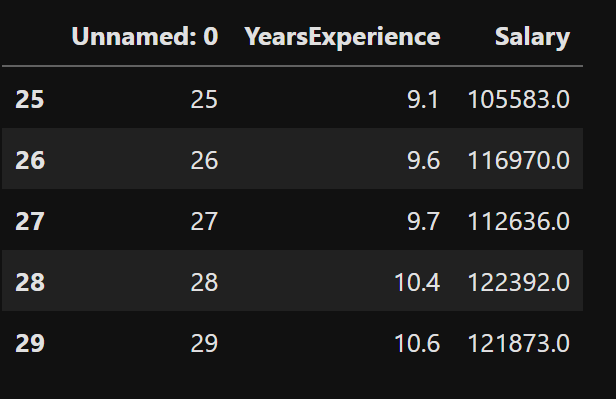
**Code Output:**



**Code:**



**Code output:**



#### 2.Data Overview and Summary:

#### CODE:

#### Screenshot 2025-06-02 212241

#### Explanation:

#### data.info()

Displays basic information about the dataset:

Total entries: 30

Columns: Unnamed: 0, YearsExperience, Salary

All columns have 30 non-null values

Data types: integers and floats

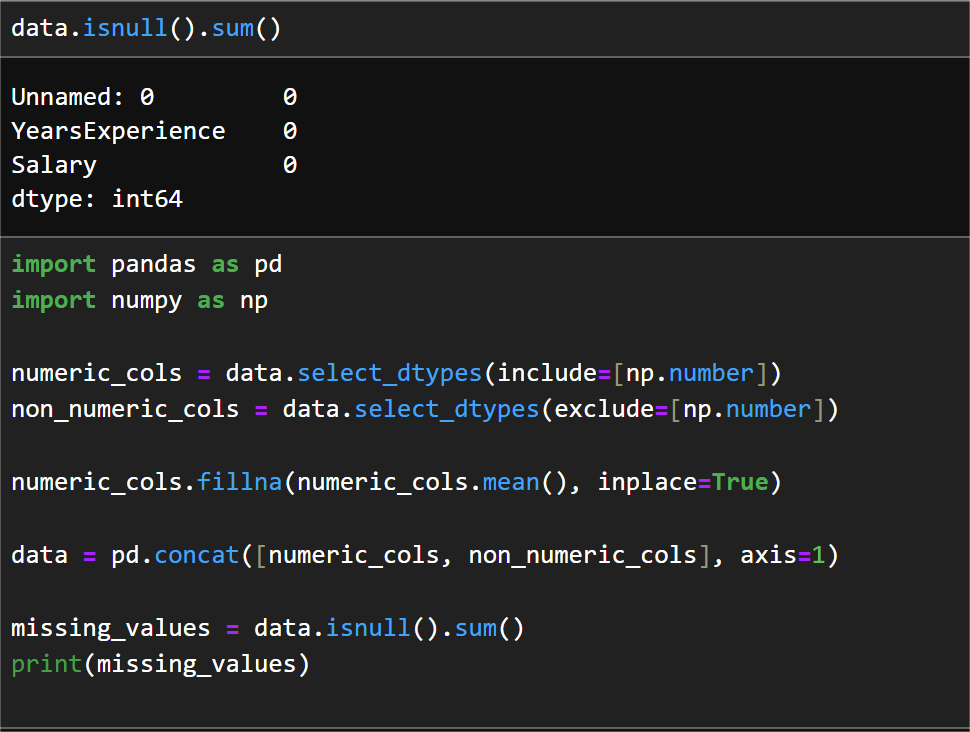
**data.describe()**

Generates summary statistics for numerical columns:

Includes count, mean, standard deviation, min, max, and quartiles

**3.Data Cleaning:**

**Code:**



**Code explanation:**

#### data.isnull().sum()

1. Checks for missing (null) values in each column.

Output shows all values are present (no missing data).

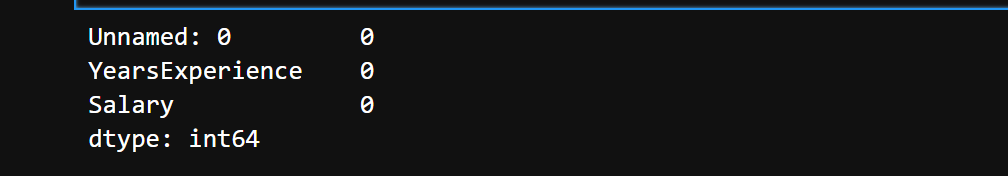
2 . Required libraries for data manipulation and numerical operations.

1. Separates numeric and non-numeric columns for targeted operations.
2. Replaces missing values in numeric columns with their respective column mean.
3. Merges numeric and non-numeric columns back into one DataFrame.

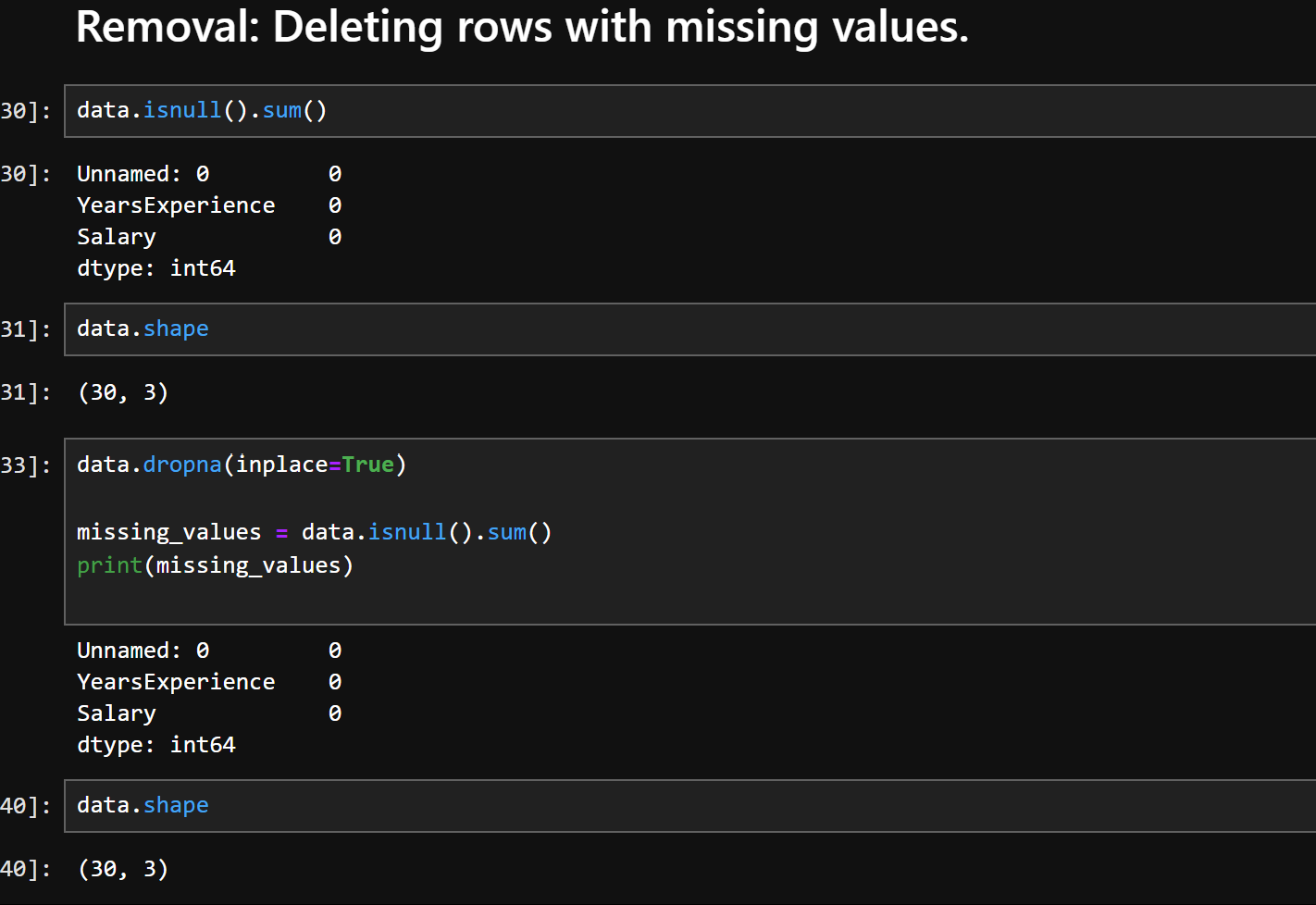
Verifies that all missing values have been filled.

**Purpose:**  
This step ensures the dataset is clean and ready for model training by handling any potential missing data using mean imputation.

Output:



**4: Removing Missing Data:  
Code:**



**Code Explanation:**

#### data.isnull().sum()

Checks for missing values in each column.

All columns have 0 missing values.

#### data.shape

Displays the shape (rows, columns) of the dataset: **(30, 3)**  
→ 30 rows and 3 columns.

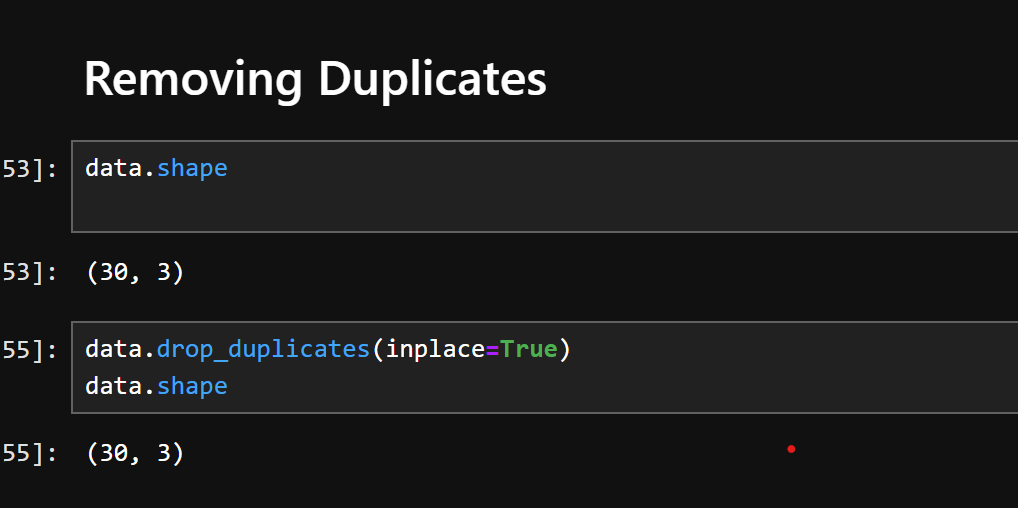
#### data.dropna(inplace=True)

Removes rows that contain any missing values.

inplace=True modifies the original DataFrame.

**5: Removing Duplicate Records**

**Code:**



### ****Code Explanation:****

#### data.shape

Displays the shape of the dataset before removing duplicates: **(30, 3)**

#### data.drop\_duplicates(inplace=True)

Removes any duplicate rows in the dataset.

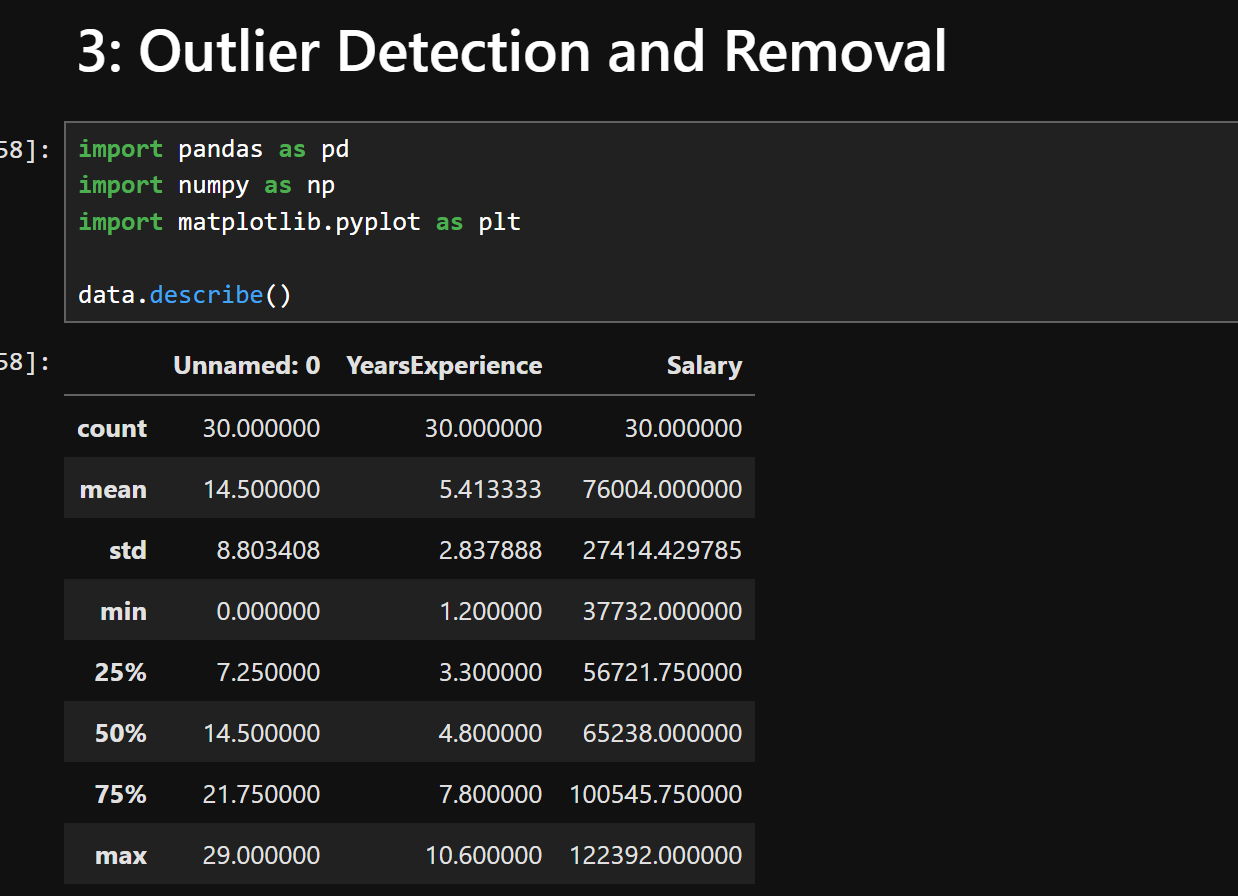
inplace=True updates the DataFrame directly.

#### data.shape (again)

Confirms that the dataset shape remains **(30, 3)**, indicating no duplicate rows were found.

**6: Outlier Detection and Removal (Initial Summary)**

**Code:**



### ****Code Explanation:****

#### data.describe()

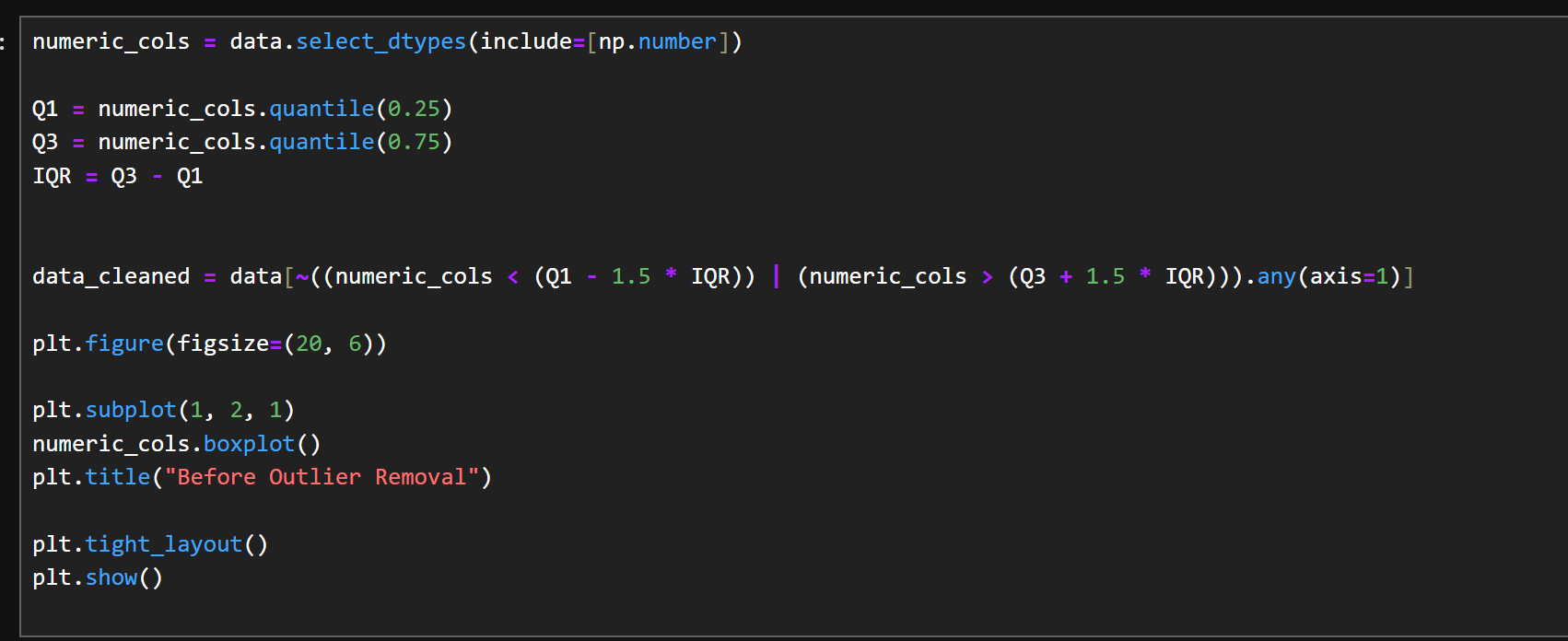
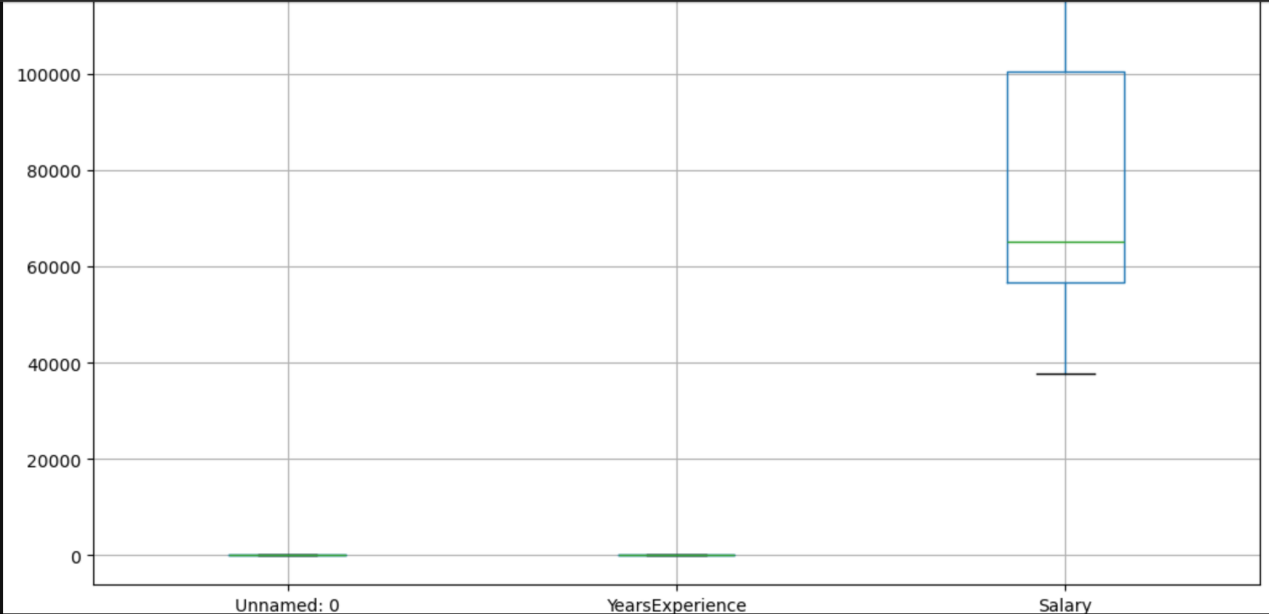
Provides a statistical summary of the dataset.

Helps identify potential outliers by examining min, max, and quartile values.

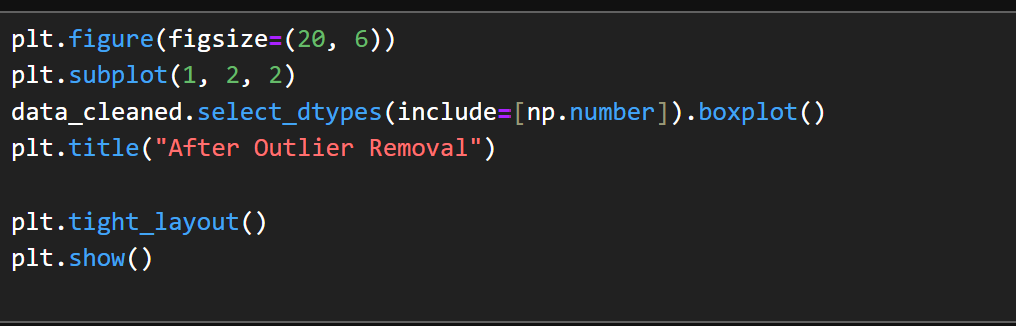
**OUTPUT:**



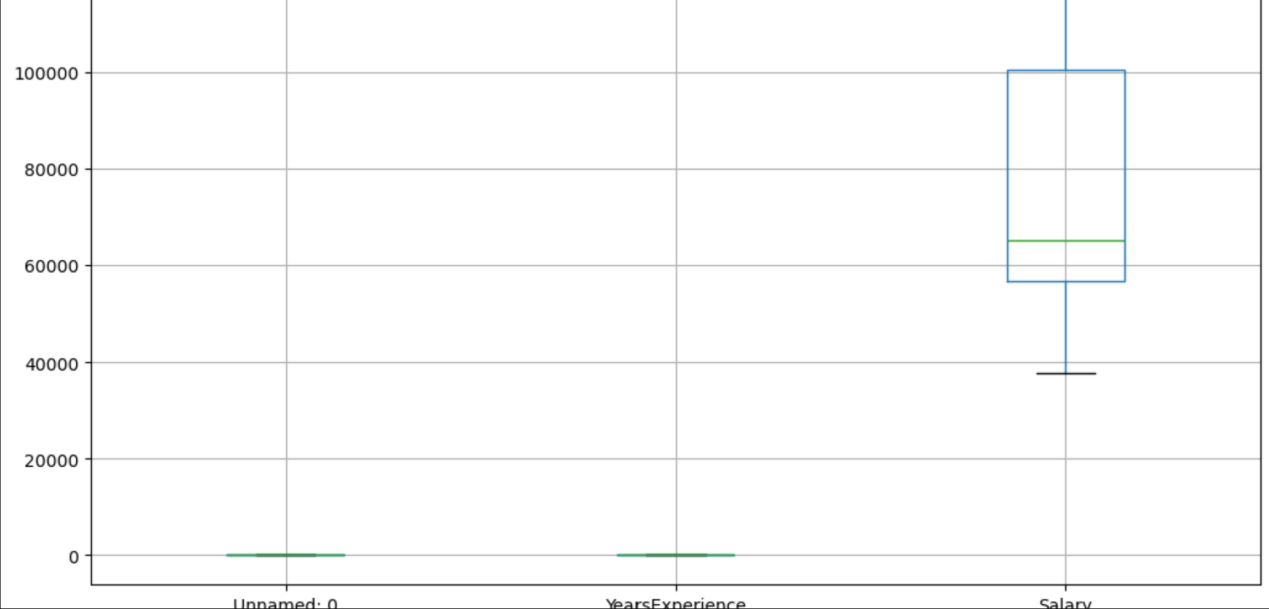
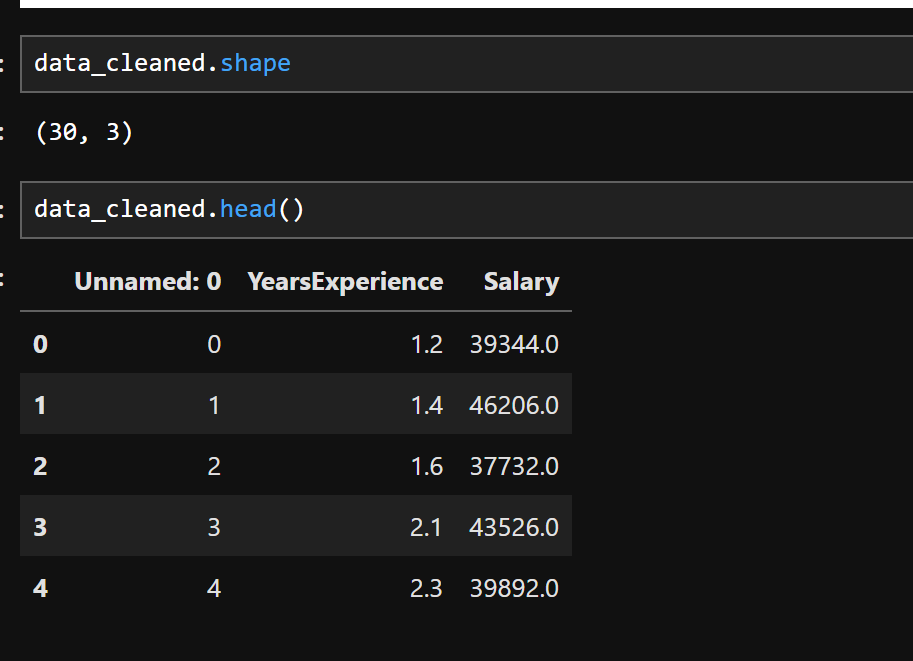
**Before Outlier Removal:**

  
**Output:**  


**After Outlier Removal:**



**output:**

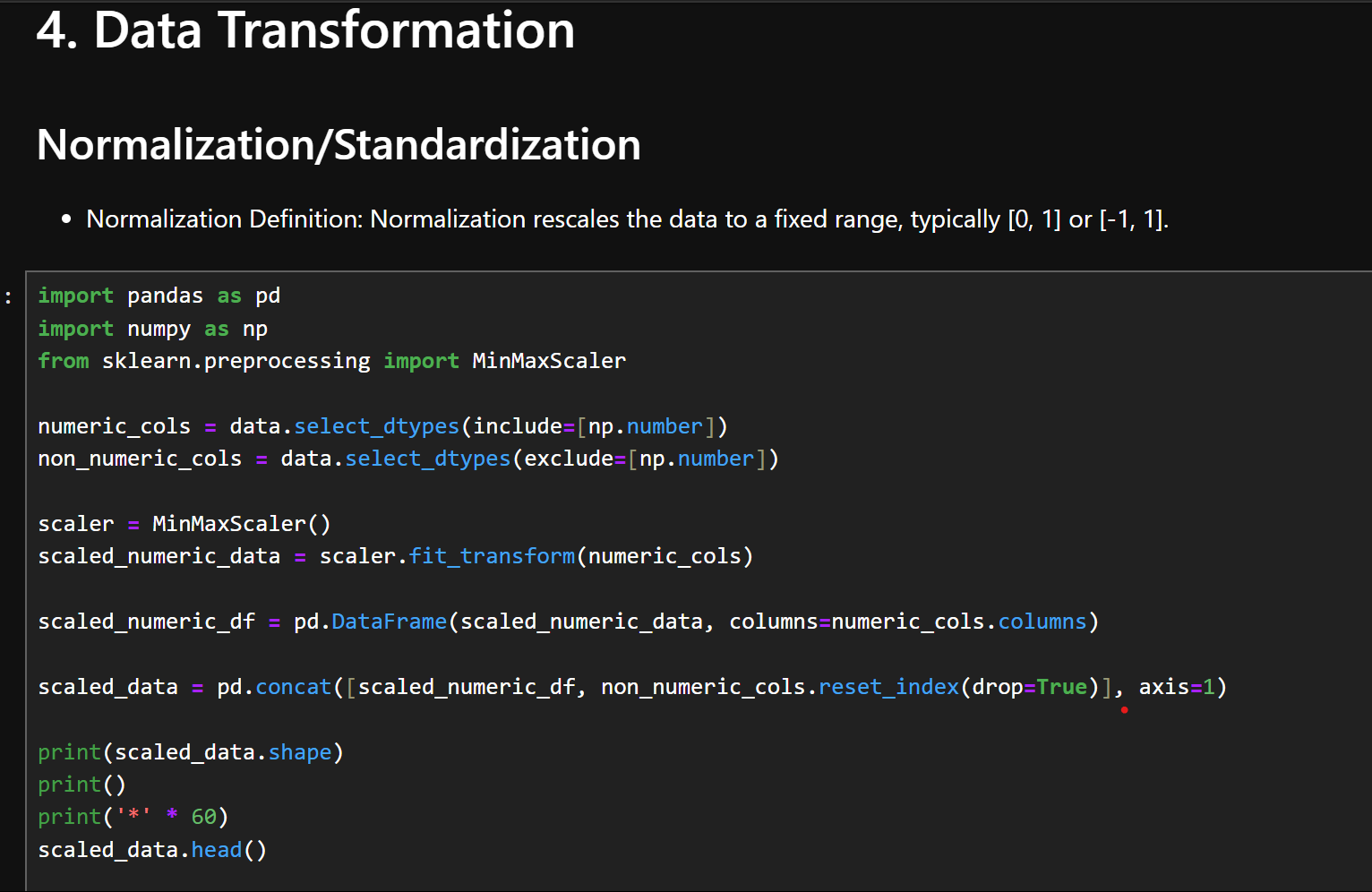
**output:**

**data\_cleaned.shape → (30, 3):**  
**No rows were removed** after applying the outlier detection using the IQR method.**data\_cleaned.head()** shows normal-looking data.  
This suggests **no values were considered outliers** by your IQR

**6 Data Transformation – Normalization using MinMaxScaler**

Normalization rescales the data to a fixed range, typically [0, 1] or [-1, 1].

**Code:**



**Explanation:**

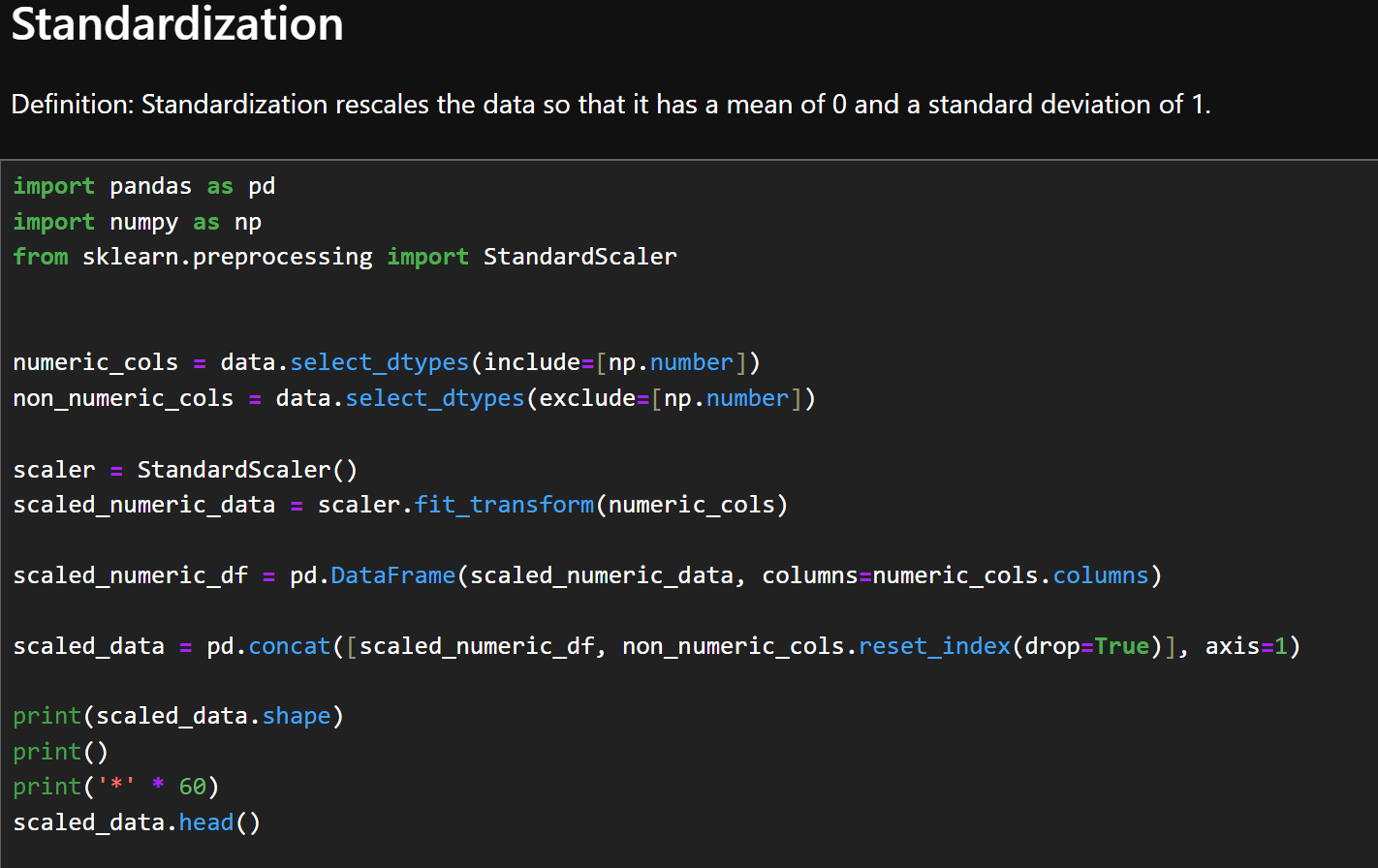
This section performs **normalization** of the numeric features in the dataset using the MinMaxScaler from sklearn.preprocessing. Normalization rescales the values of numeric features to a fixed range, usually [0, 1]. This helps improve the performance and training stability of machine learning models, especially those involving gradient-based optimization.

**Output**

**:**

1. **Data Transformation – Standardization using StandardScaler**
2. **Definition:** Standardization rescales the data so that it has a mean of 0 and a standard deviation of 1.

**Code:**



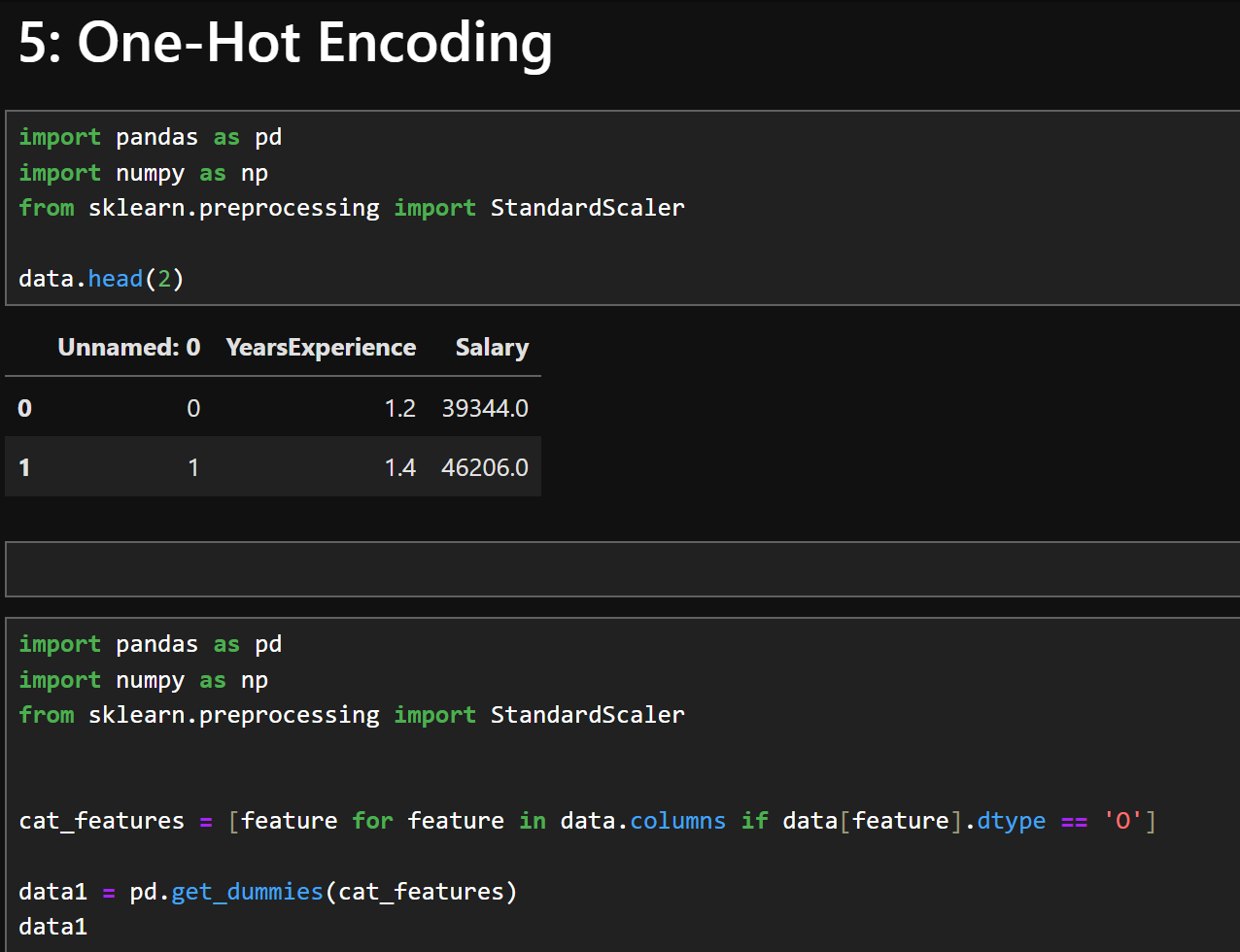
**Explanation:**

This section performs **standardization** on the numeric columns of the dataset using StandardScaler from sklearn.preprocessing. Standardization transforms the data such that the mean of each numeric feature becomes 0 and the standard deviation becomes 1. This is useful for machine learning algorithms that assume normally distributed data or are sensitive to the scale of input features, such as linear regression, which we're using for salary prediction.

**Output:**



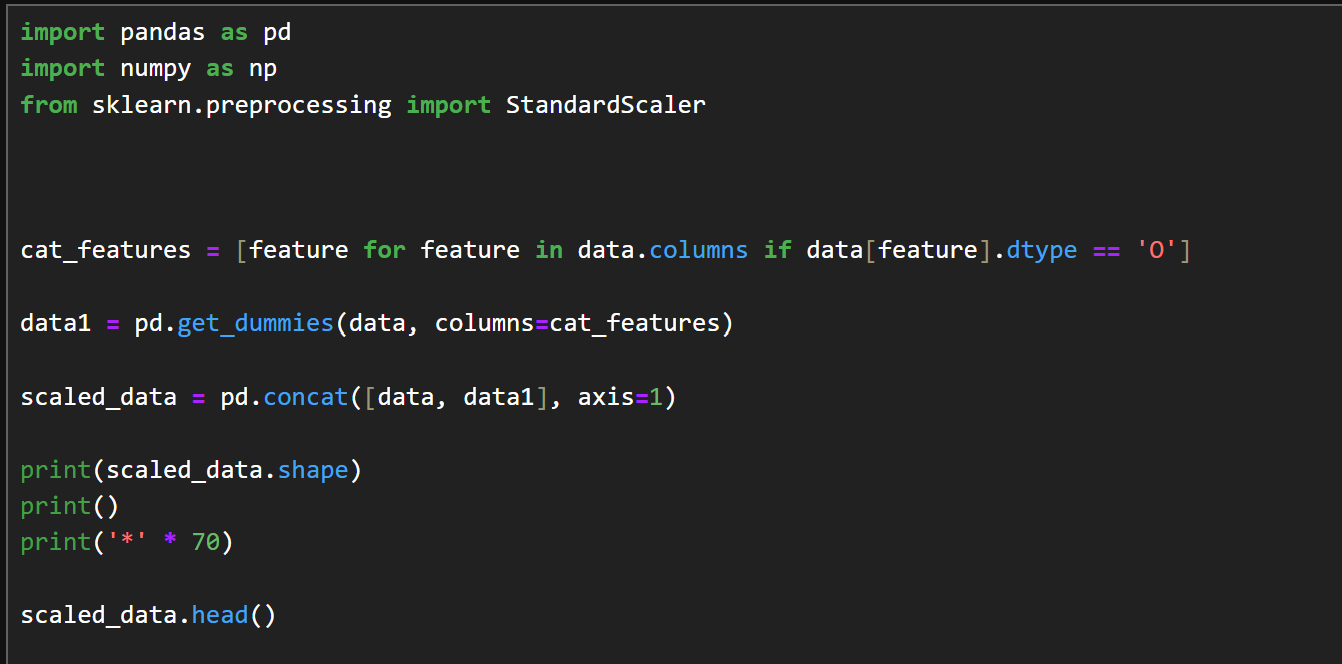
**8. One-Hot Encoding for Categorical Features**



**Explanation:**

One-Hot Encoding is a technique used to convert categorical (non-numeric) variables into a numerical format that can be used by machine learning models. It creates a new binary column for each category in a categorical feature, assigning 1 or 0 depending on the presence of the category.

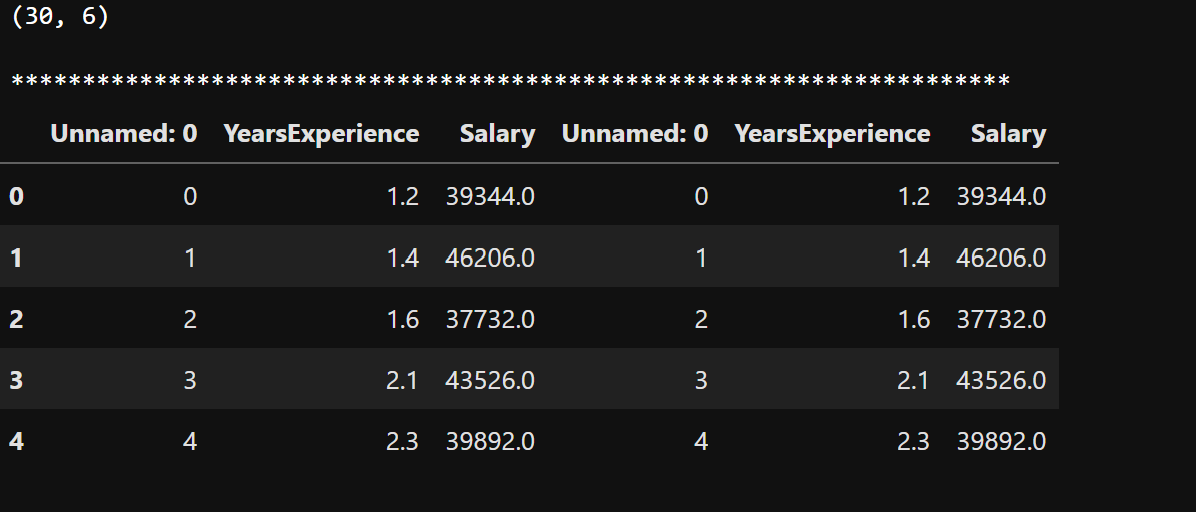
**Code:**



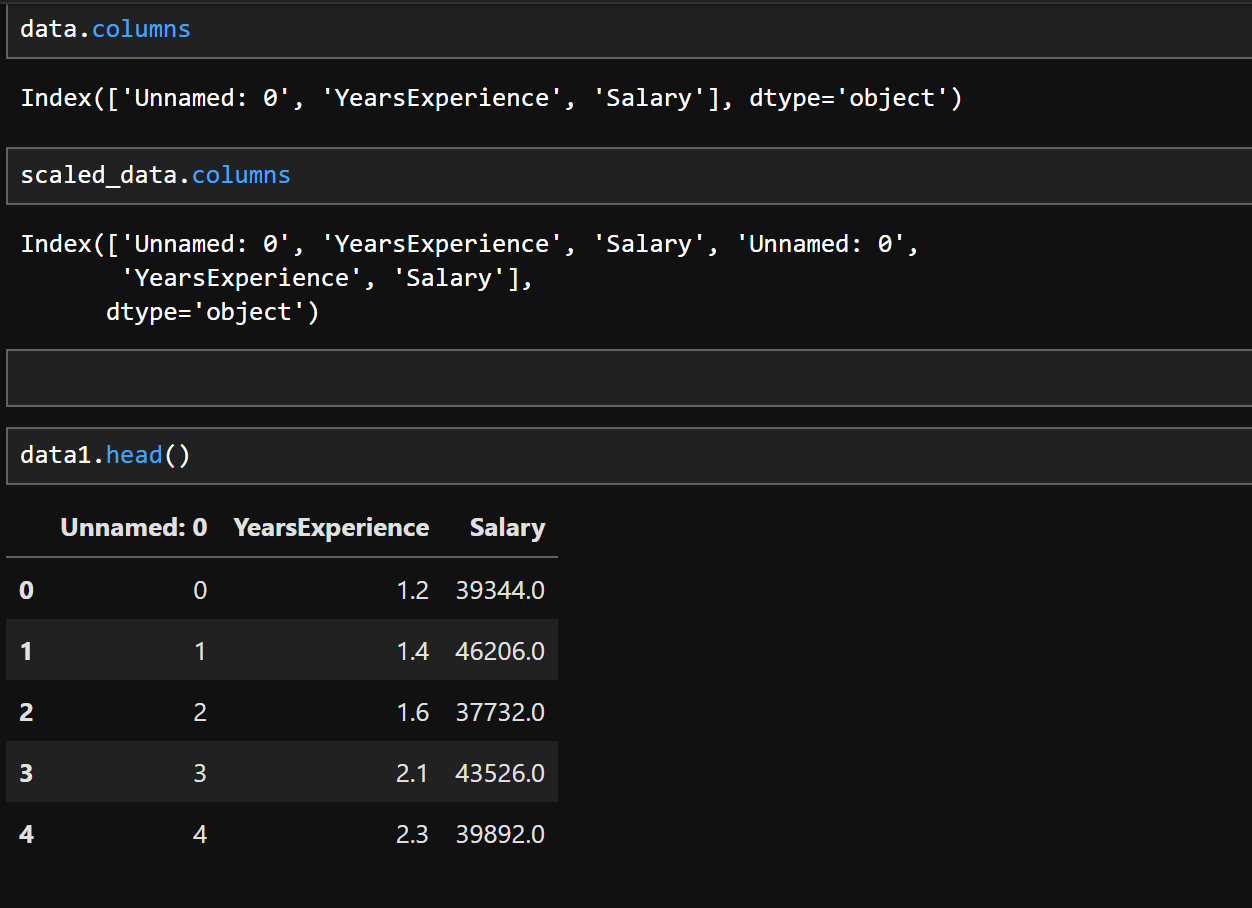
**Explanation:**

This code block improves upon the previous one-hot encoding process by integrating the encoded data back into the original dataset. One-hot encoding is used to convert categorical variables into binary vectors, and this version correctly applies it and combines the result.

**Output:**



**Observing the Effect of One-Hot Encoding on Data Columns**

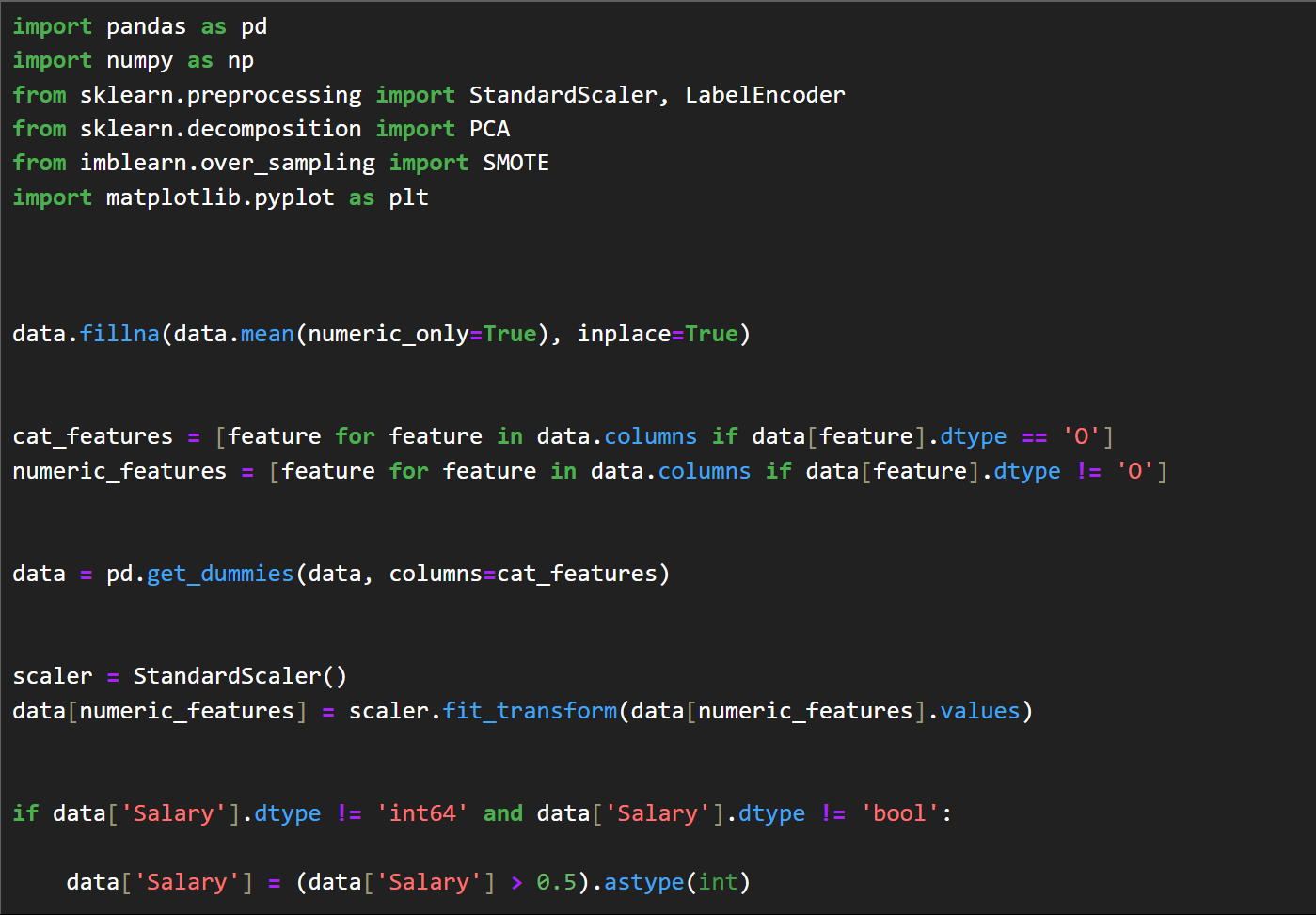
**Code:**  
  


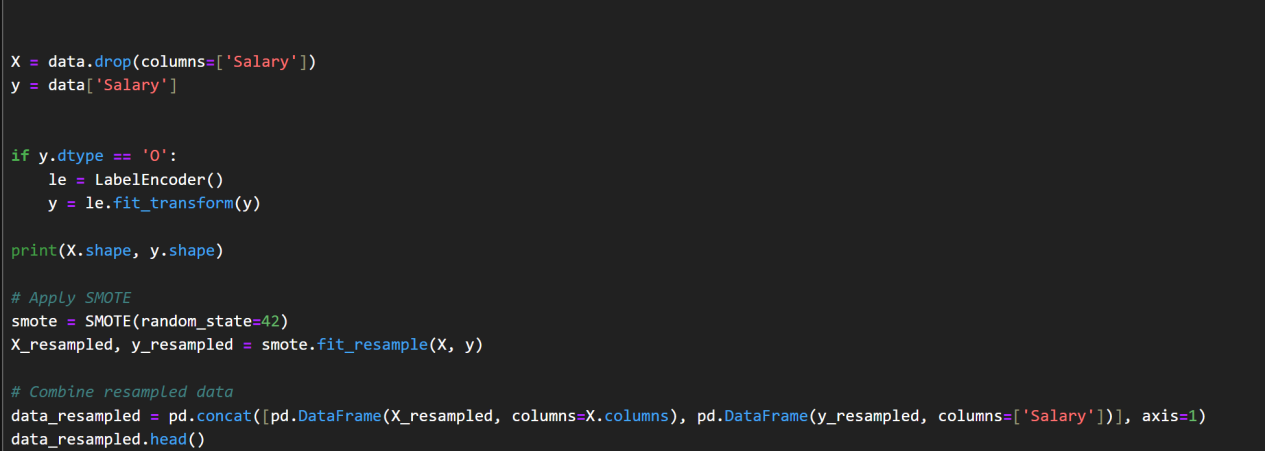
**Explanation:**

This section inspects how one-hot encoding has affected the structure of the dataset.

**9.Handling Imbalanced Data Using SMOTE**

**Code:**



**  
Explanation:**

Reads a dataset with YearsExperience and Salary.

Drops duplicate or unnecessary columns like Unnamed: 0.

Splits data:  
X = features (excluding Salary),  
y = target (Salary).

Converts categorical y to numeric (if needed) using LabelEncoder.

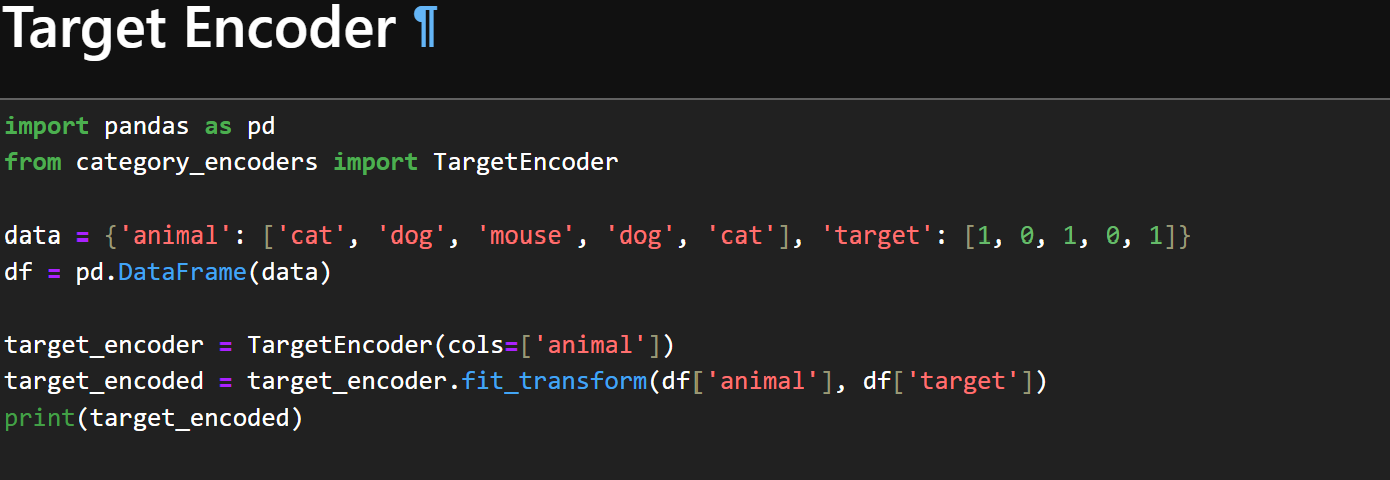
Applies **SMOTE** to create synthetic samples for the minority class, balancing the dataset.

**Output:**



1. **Encoding Categorical Variables Using Target Encoding**

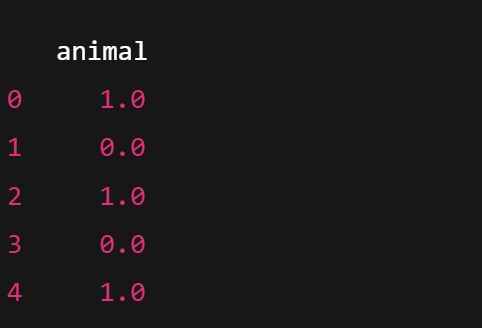
**Code:**



**Explanation:**

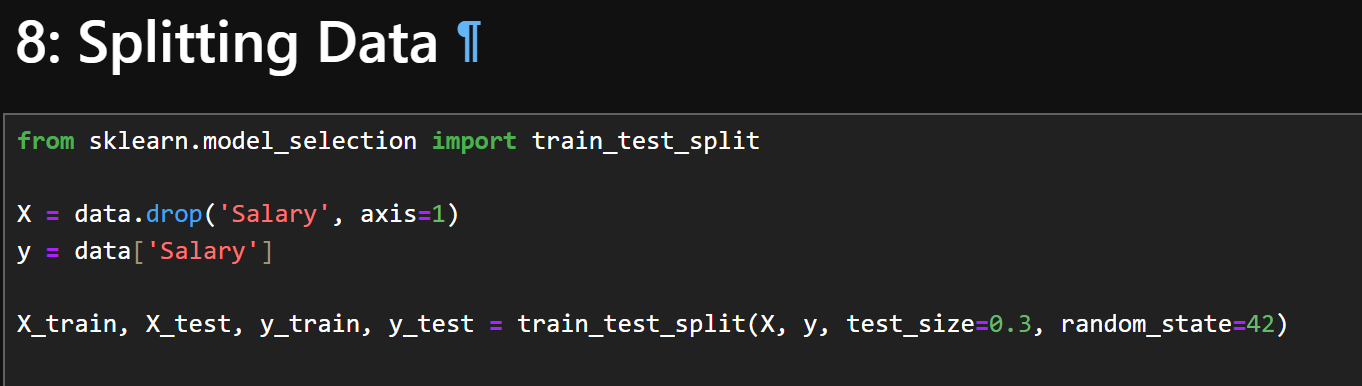
This code demonstrates how to convert categorical values into numeric form using **Target Encoding**. This encoding is useful in machine learning tasks where categorical variables have predictive power based on the target variable.

Outout:



1. **Splitting the Dataset into Training and Testing Sets**

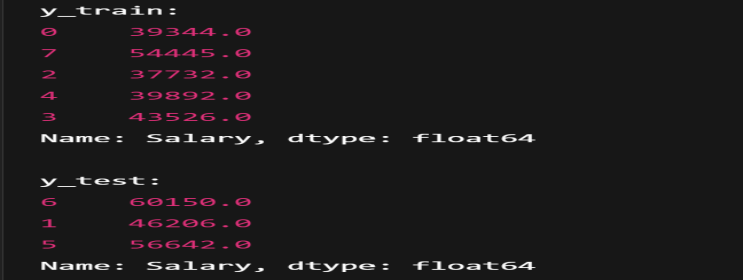
**Code:**

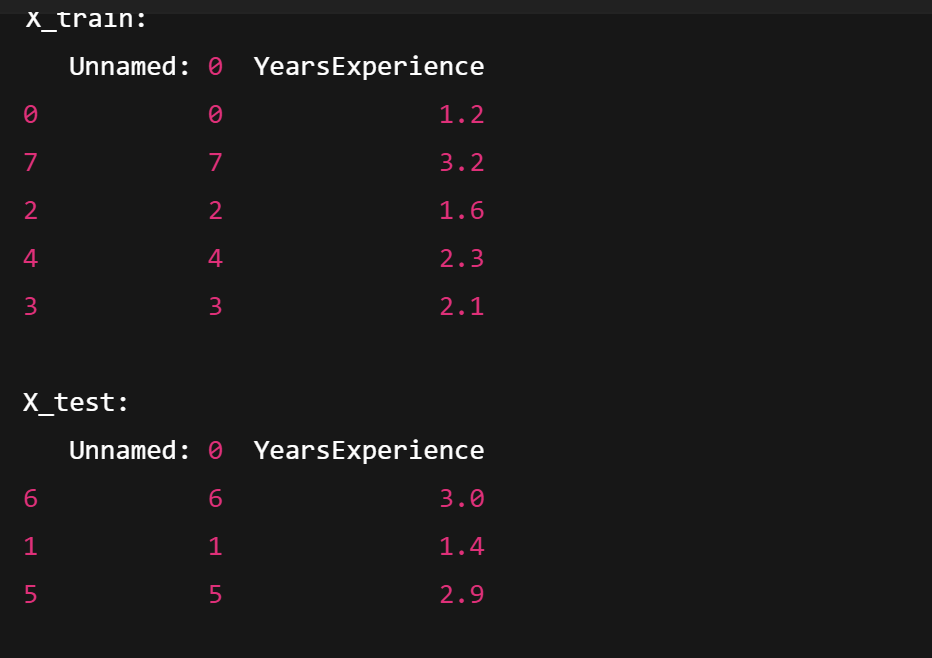


**Explanation:**

In this step, the dataset is divided into two parts: a **training set** and a **testing set** using the train\_test\_split() function from the sklearn.model\_selection module.

**Output**





**Project NO 2**

**Classification Project**

**Fake news Detection**

**Project Description:**

This project focuses on detecting fake news using a **Logistic Regression** model. It processes textual data from news headlines, converts it into numerical features using **TF-IDF Vectorization**, and trains a classifier to distinguish between real and fake news. The model can predict the authenticity of new headlines, helping combat misinformation.

## ****Data Description:****

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| **text** | String | The news headline or short news content to be classified. |
| **label** | Integer | The target label: 0 for real news, 1 for fake news. |

### ****General Notes:****

The text column contains unstructured natural language data.

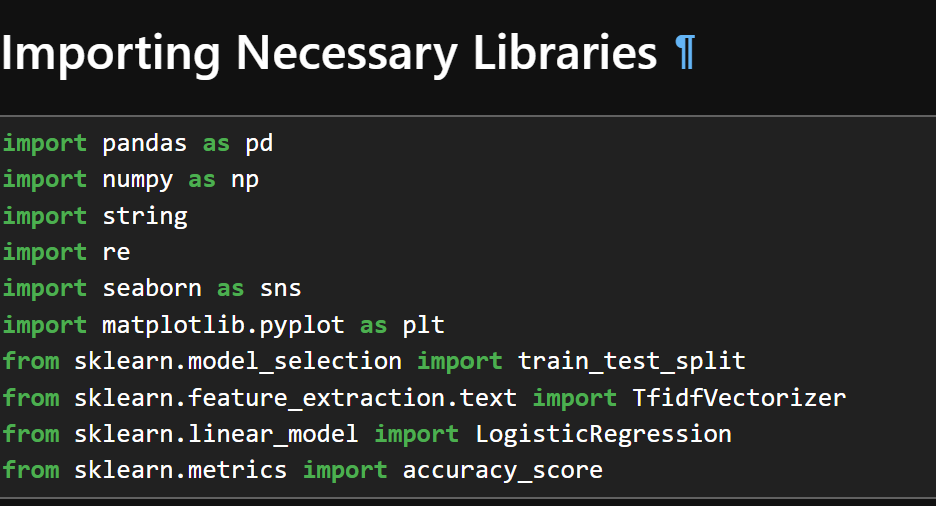
The label column is binary, used for supervised classification.

Suitable for **Natural Language Processing (NLP)** and text classification task

**Classification**

**Introduction to Python and Libraries for Machine Learning, Environmental Setup**

**1.Importing Necessary Libraries**



**2.Reading and Previewing the Fake News Dataset**

**Code:**



### Explanation:

pd.read\_csv("News.csv"): Reads the dataset named News.csv and stores it in a DataFrame called df.

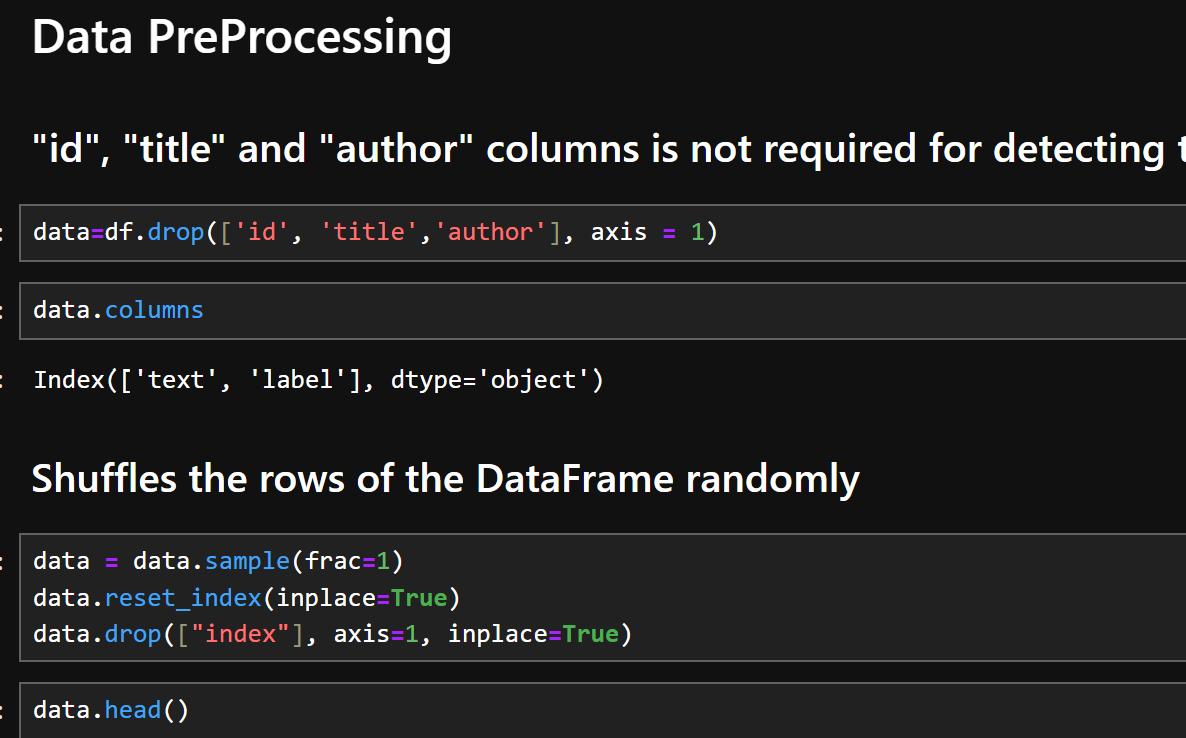
df.head(): Displays the first 5 rows to get a quick preview of the data.

df.shape: Returns the shape of the dataset. In this case, it has 20,800 rows and 5 columns.

df.columns: Lists the column names: 'id', 'title', 'author', 'text', and 'label'.

**3.Data Preprocessing**

**Code:**



**Explanation:**

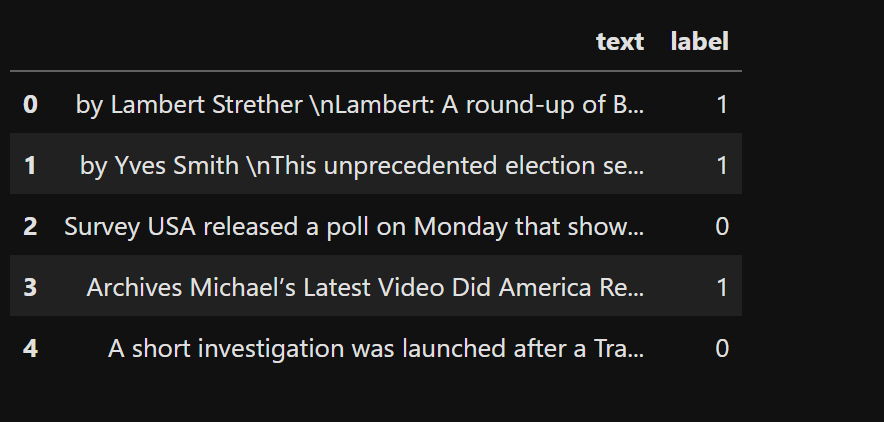
df.drop(['id', 'title', 'author'], axis=1): Removes columns that are not essential for detecting fake news (i.e., metadata like ID, title, and author).

The remaining columns are:

'text': the article content.

'label': the target (1 = Real, 0 = Fake).

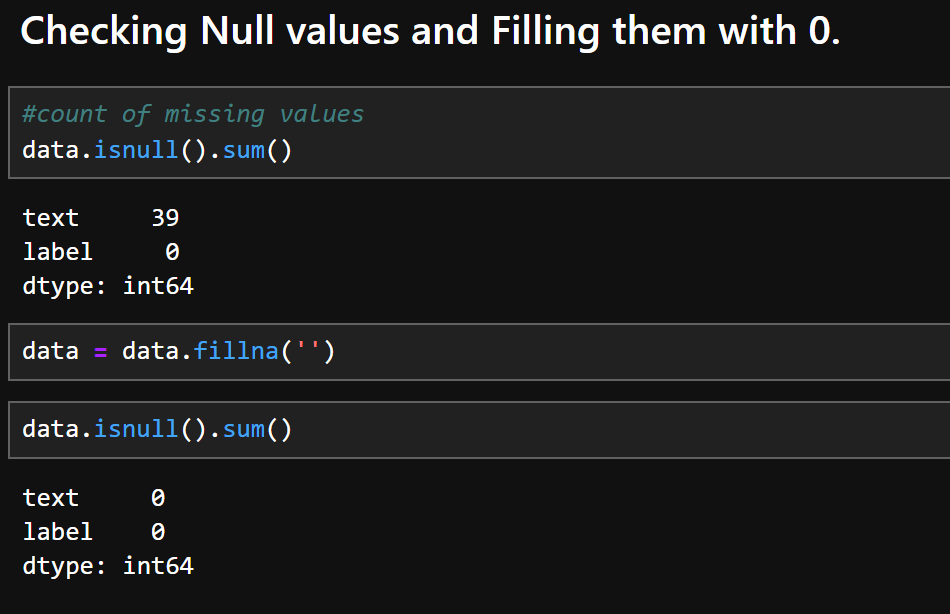
**Output:**



**standardization, Data visualization**

**5.Checking Null Values and Filling Them**

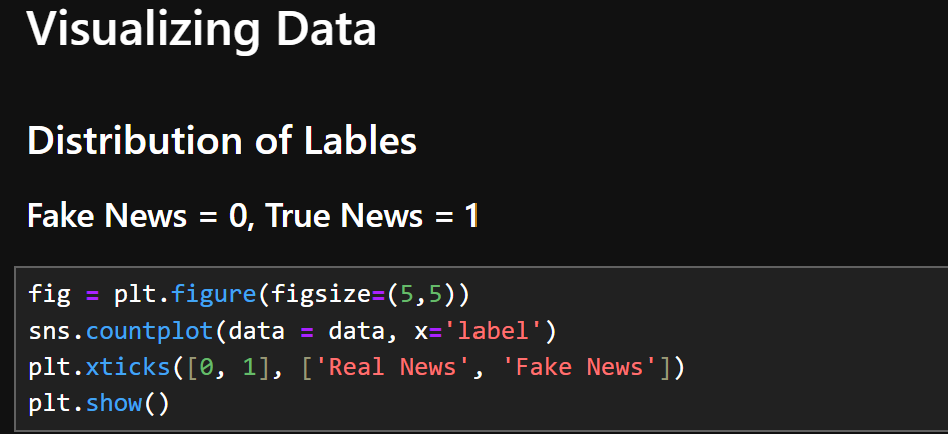
**Code:**



**Explanation:**  
There are 39 missing entries in the 'text' column, which is critical for our fake news detection model.

**Visualizing Data**

**Code:**



### Explanation:

fig = plt.figure(figsize=(5,5)): Creates a square figure for the plot.

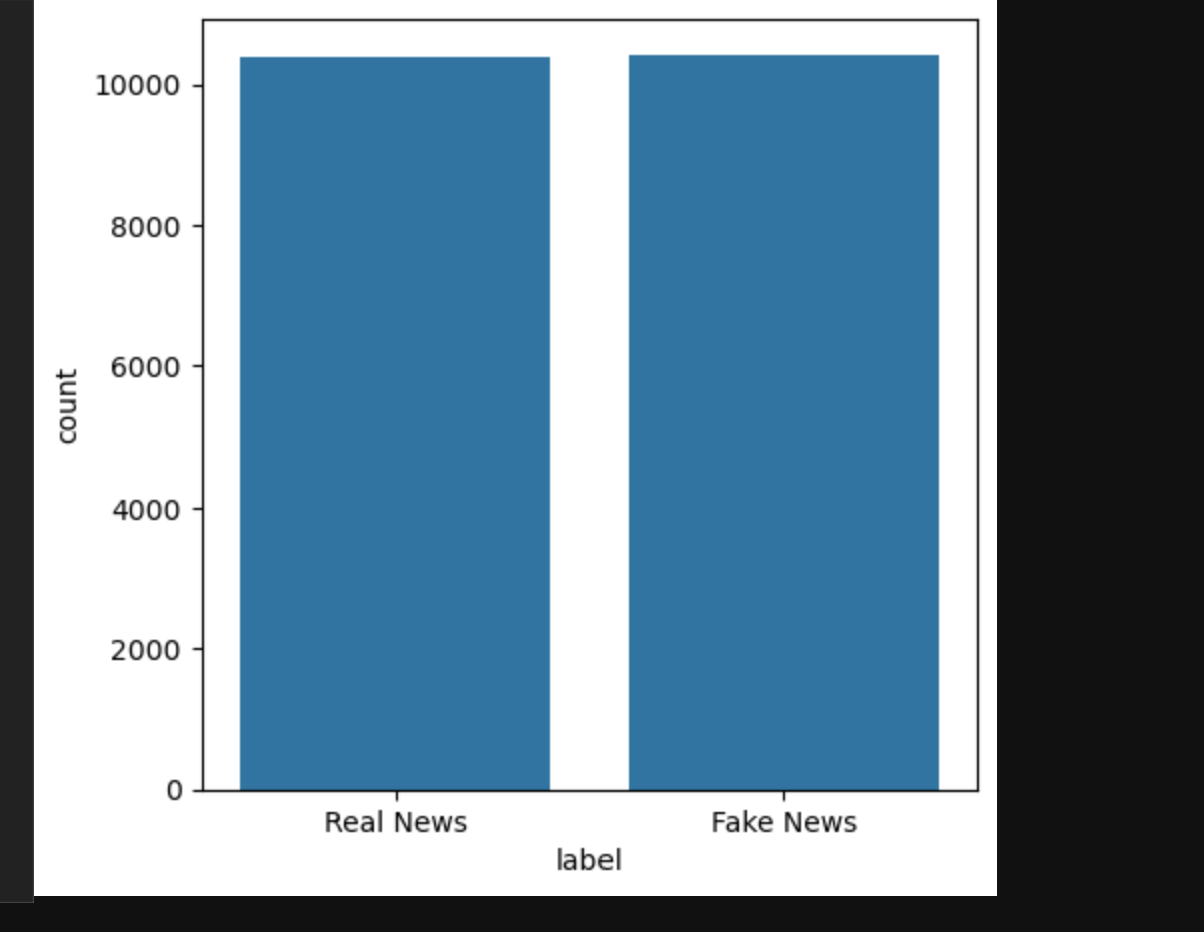
sns.countplot(...): Plots the frequency count of each class label in the dataset.

x='label': Specifies the x-axis should show the labels (0 for fake, 1 for real).

plt.xticks(...): Renames tick labels to be more readable: 'Real News' and 'Fake News'.

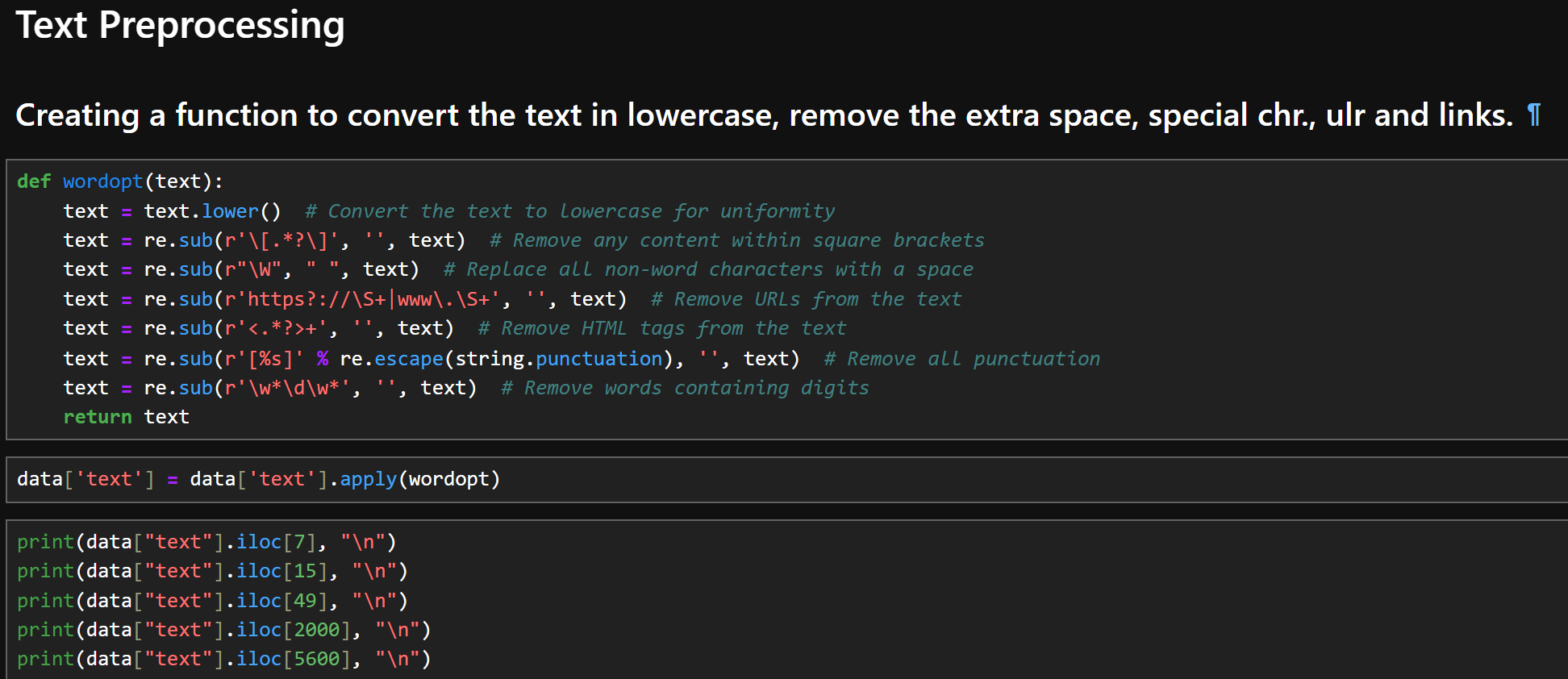
plt.show(): Displays the plot.

**Output:**



**Text Preprocessing**

**Code:**



### Explanation:

This function performs multiple text-cleaning steps:

**Converts text to lowercase** so that "The" and "the" are treated the same.

**Removes square-bracket content**, e.g., references like [1]

**Replaces special characters** and symbols with a space for better tokenization.

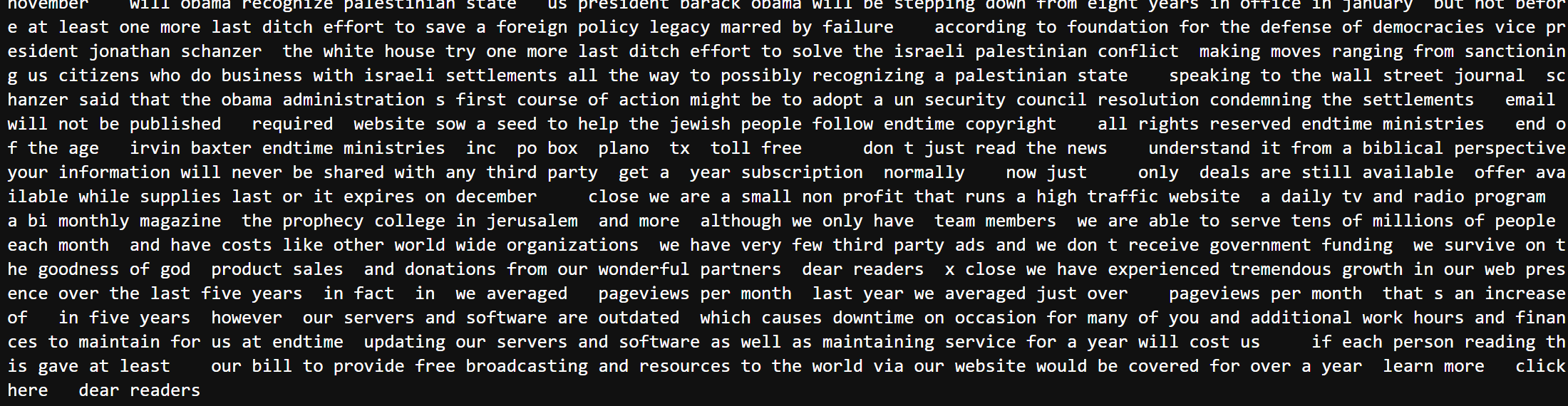
**Strips URLs** to avoid irrelevant data.

**Removes HTML tags** which are unnecessary for analysis.

**Eliminates punctuation** that might clutter the model’s understanding.

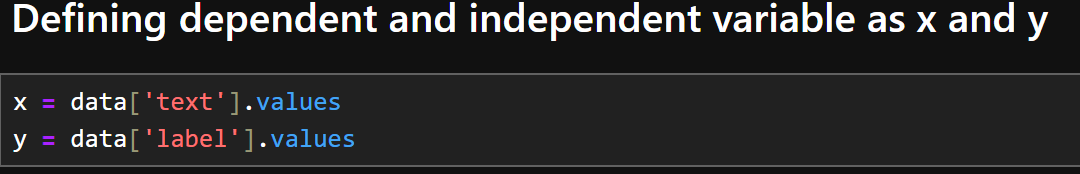
**Deletes digit-containing words**, assuming they hold less semantic meaning for fake news detection.

**Output**



**8.Defining Dependent and Independent Variable**

**Code:**



### Explanation:

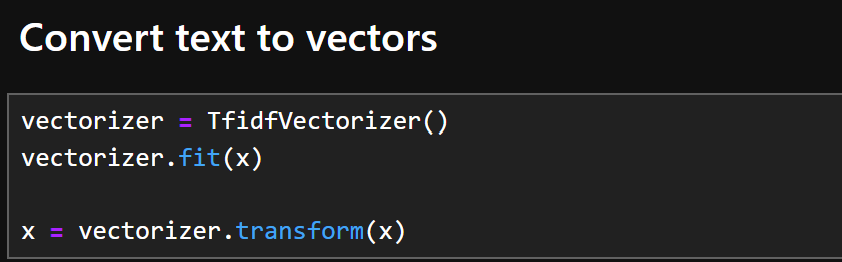
In this step:

x represents the **independent variable**, which is the **text** column containing the cleaned news articles.

y represents the **dependent variable**, which is the **label** column indicating whether the news is **fake (0)** or **real (1)**.

**9.Converting Text to Vectors using TF-IDF**

**Code:**

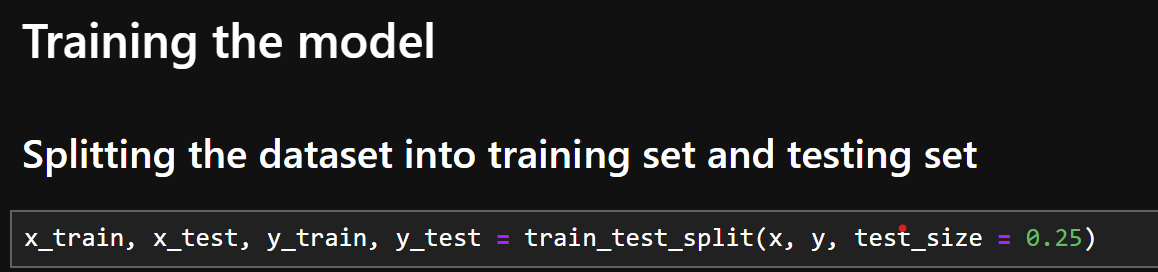


### Explanation:

This step transforms the raw text data into numerical format using **TF-IDF (Term Frequency-Inverse Document Frequency)** vectorization

**10.Training Model:**

**Code:**



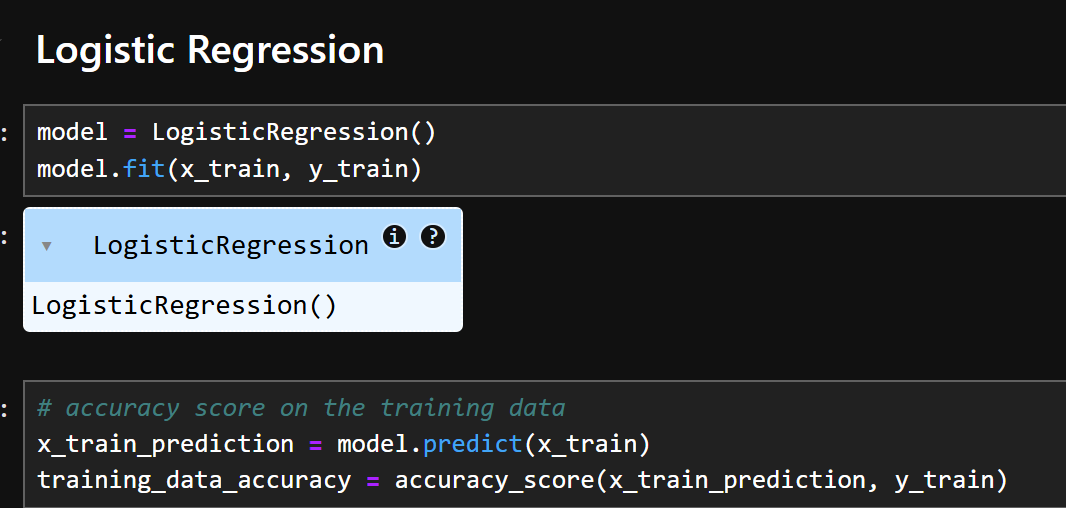
**Explanation:**

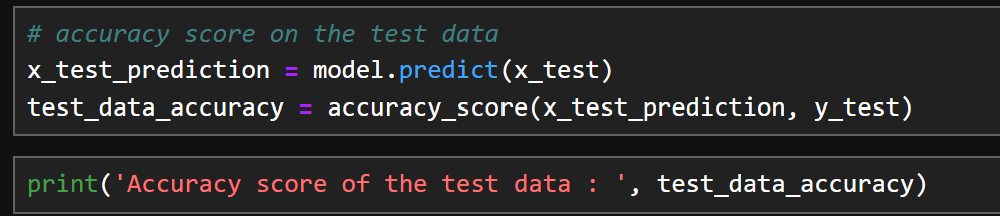
Training model on x-test,x\_train and y\_test,y\_train sets

**Implementing Logistic Regression**

**11.Training and Evaluating the Logistic Regression Model**

**Code:**

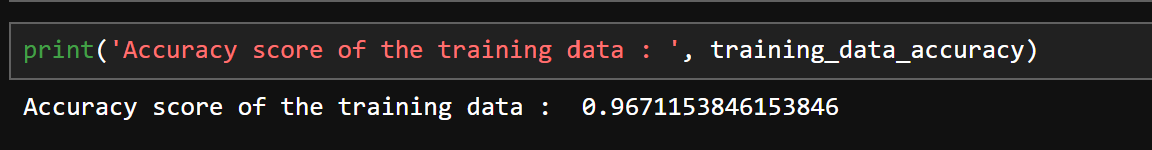
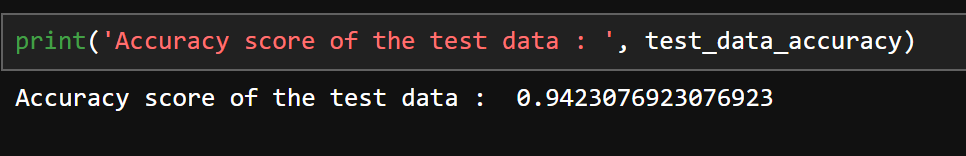




### Explanation:

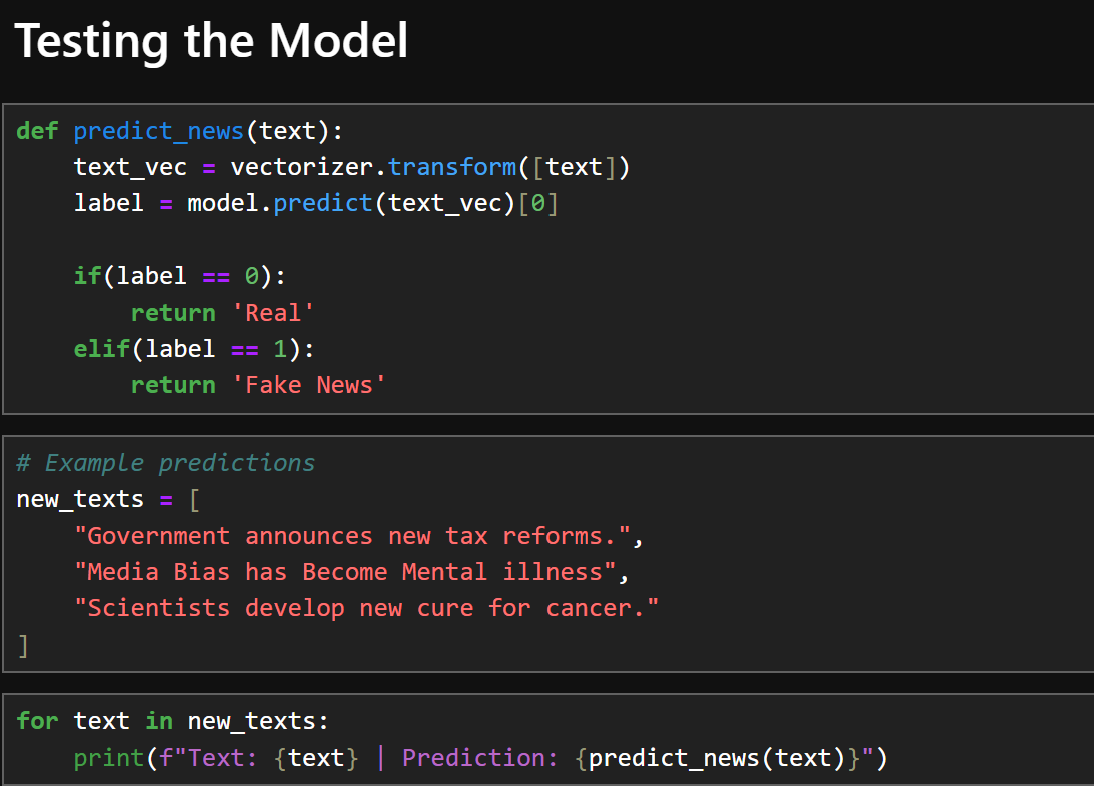
This block trains a **Logistic Regression** model to classify news as fake or real:

**Output:**

**12.Testing the Fake News Detection Model**

**Code:**



### Explanation:

This section creates a predict\_news() function to classify new/unseen news headlines or articles:

**Output**:

