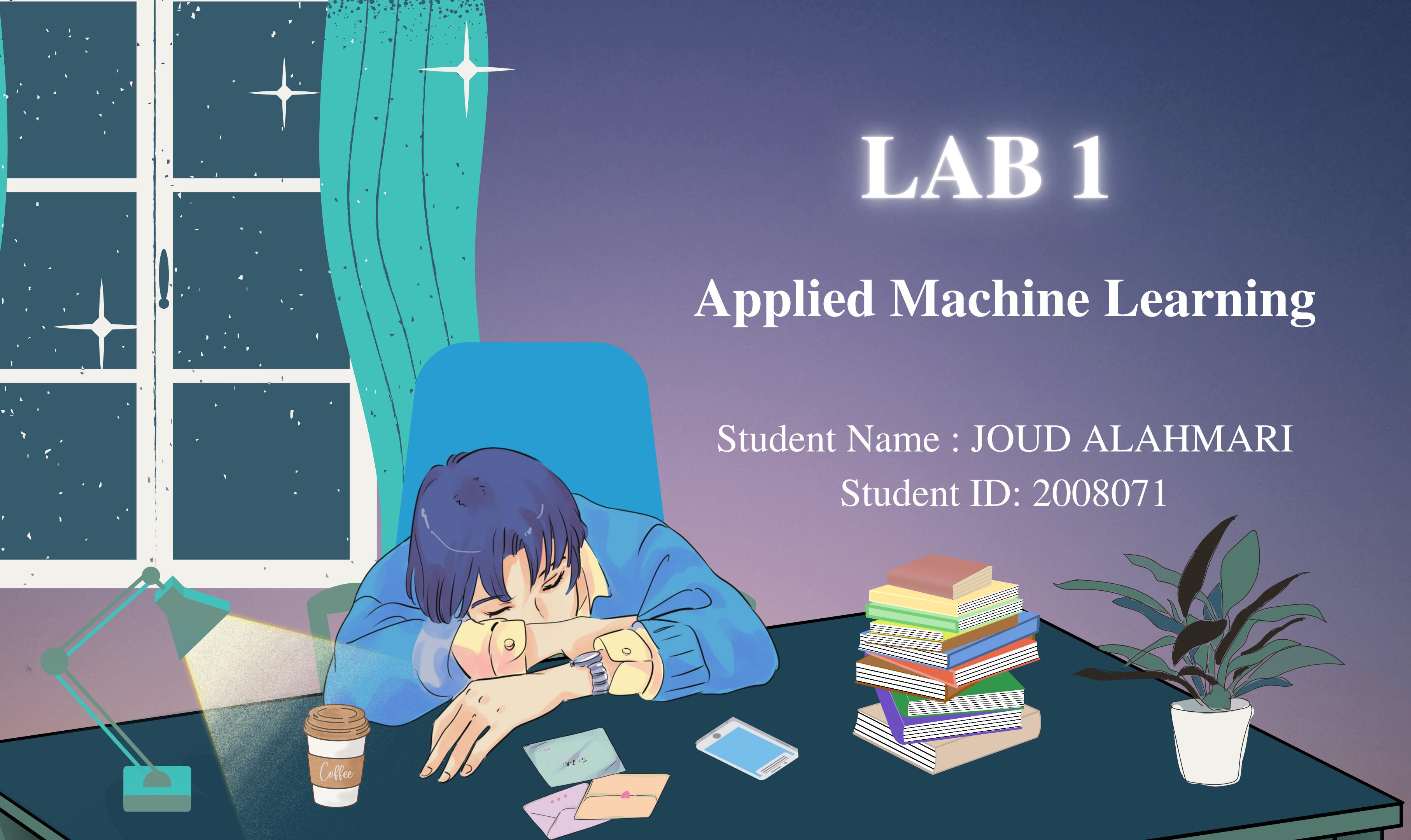


LAB 1

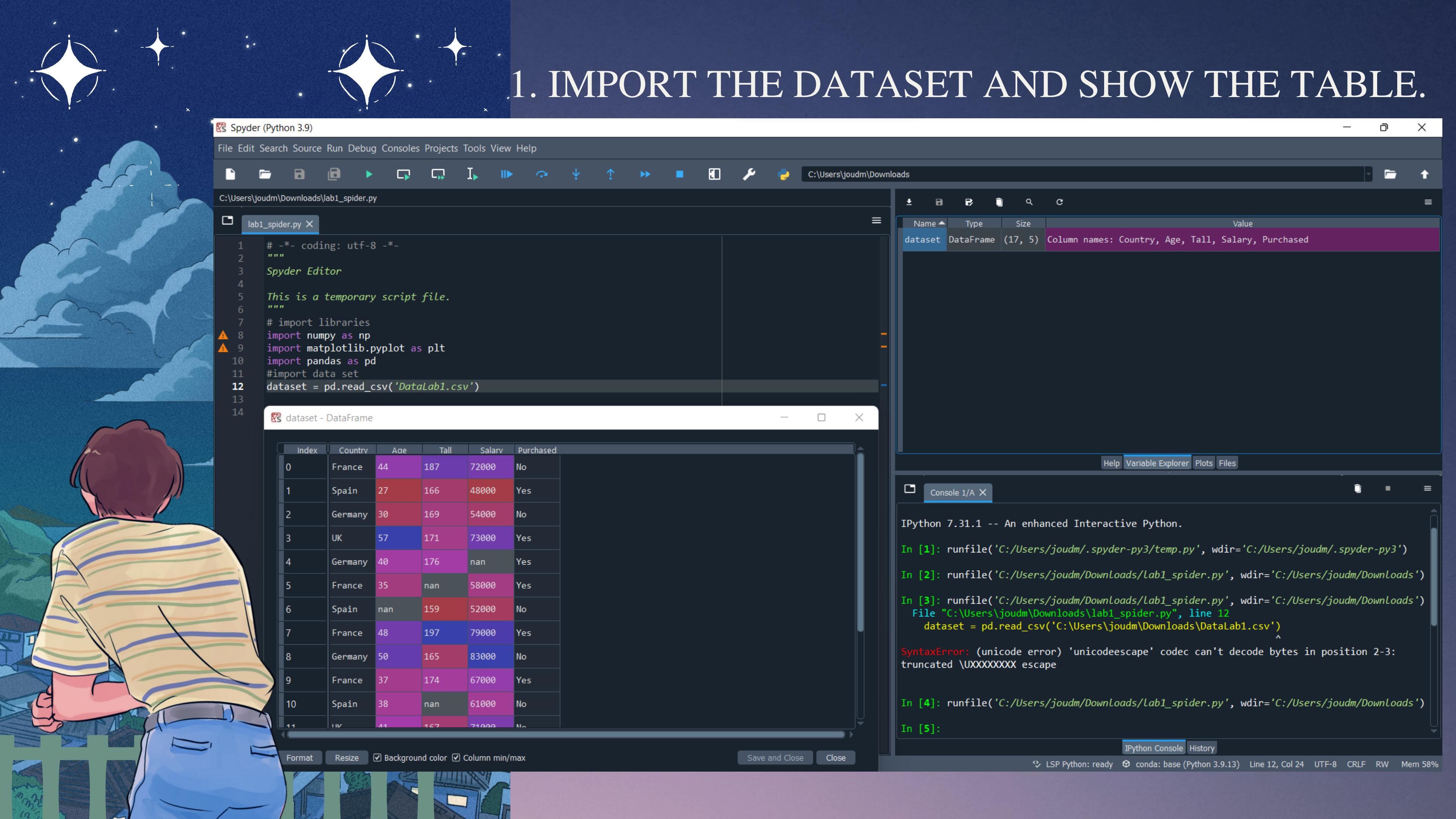
Applied Machine Learning

Student Name : JOUD ALAHMARI

Student ID: 2008071



1. IMPORT THE DATASET AND SHOW THE TABLE.



The screenshot shows a Python development environment using Spyder (Python 3.9). The main window displays a script file named `lab1_spider.py` which imports libraries and reads a CSV dataset. A preview window shows the first 12 rows of the dataset, and the Variable Explorer shows the imported `dataset` as a DataFrame with 17 rows and 5 columns. The IPython Console shows the execution of the script, where it fails to run due to a syntax error related to encoding.

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.

# import libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# import data set
dataset = pd.read_csv('DataLab1.csv')
```

Index	Country	Age	Tall	Salary	Purchased
0	France	44	187	72000	No
1	Spain	27	166	48000	Yes
2	Germany	30	169	54000	No
3	UK	57	171	73000	Yes
4	Germany	40	176	nan	Yes
5	France	35	nan	58000	Yes
6	Spain	nan	159	52000	No
7	France	48	197	79000	Yes
8	Germany	50	165	83000	No
9	France	37	174	67000	Yes
10	Spain	38	nan	61000	No
11	UK	41	167	71000	No

Variable Explorer:

Name	Type	Size	Value
dataset	DataFrame	(17, 5)	Column names: Country, Age, Tall, Salary, Purchased

IPython Console:

```
IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/joudm/.spyder-py3/temp.py', wdir='C:/Users/joudm/.spyder-py3')
In [2]: runfile('C:/Users/joudm/Downloads/lab1_spider.py', wdir='C:/Users/joudm/Downloads')
In [3]: runfile('C:/Users/joudm/Downloads/lab1_spider.py', wdir='C:/Users/joudm/Downloads')
File "C:\Users\joudm\Downloads\lab1_spider.py", line 12
    dataset = pd.read_csv('C:\Users\joudm\Downloads\DataLab1.csv')
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3:
truncated \UXXXXXXXXX escape

In [4]: runfile('C:/Users/joudm/Downloads/lab1_spider.py', wdir='C:/Users/joudm/Downloads')
In [5]:
```

2. SEPARATE THE INDEPENDENT AND DEPENDENT VARIABLES.

The screenshot shows the Spyder IDE interface with several windows open:

- Code Editor:** The script file `lab1_spider.py` is displayed. It imports numpy, matplotlib.pyplot, and pandas. It reads a CSV file named `DataLab1.csv` into a DataFrame. Then, it extracts the independent variables (`X`) and the dependent variable (`y`). There are two warning icons (yellow triangles) in the code editor.
- Variable Explorer:** A table showing the variables and their properties:

Name	Type	Size	Value
dataset	DataFrame	(17, 5)	Column names: Country, Age, Tall, Salary, Purchased
X	Array of object	(17, 4)	ndarray object of numpy module
y	Array of float64	(17,)	[72000. 48000. 54000. ... nan 64000. 83000.]
- Console 1/A:** The console window shows the following session:

```
In [1]: runfile('C:/Users/joudm/.spyder-py3/temp.py', wdir='C:/Users/joudm/.spyder-py3')
In [2]: runfile('C:/Users/joudm/Downloads/lab1_spider.py', wdir='C:/Users/joudm/Downloads')
In [3]: runfile('C:/Users/joudm/Downloads/lab1_spider.py', wdir='C:/Users/joudm/Downloads')
      File "C:\Users\joudm\Downloads\lab1_spider.py", line 12
          dataset = pd.read_csv('C:\Users\joudm\Downloads\DataLab1.csv')
                                         ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3:
truncated \UXXXXXXXXX escape

In [4]: runfile('C:/Users/joudm/Downloads/lab1_spider.py', wdir='C:/Users/joudm/Downloads')
In [5]: runfile('C:/Users/joudm/Downloads/lab1_spider.py', wdir='C:/Users/joudm/Downloads')
In [6]:
```
- Plots:** Two plots are shown side-by-side. The left plot is titled "y - NumPy object array" and displays a vertical stack of colored bars (purple, red, blue, green, yellow, orange, pink). The right plot is titled "X - NumPy object array (read only)" and displays a table of data:

	0	1	2	3
0	72000	44.0	187.0	72000.0
1	48000	27.0	166.0	48000.0
2	54000	30.0	169.0	54000.0
3	73000	57.0	171.0	73000.0
4	nan	40.0	176.0	nan
5	58000	35.0	nan	58000.0
6	52000	nan	159.0	52000.0
7	79000	48.0	197.0	79000.0
8	83000	50.0	165.0	83000.0
9	France	37.0	174.0	67000.0
10	Spain	38.0	nan	61000.0

3. CALCULATE THE MEDIAN FOR EACH FEATURE OR COLUMN THAT CONTAINS A MISSING VALUE AND REPLACE THE RESULT FOR THE MISSING VALUE.

The screenshot shows a Jupyter Notebook interface with several panes:

- Code Editor (lab1_spider.py):** Displays Python code for data processing. It includes separating independent and dependent variables, extracting selected rows and columns from a dataset, calculating the median for missing values in specific columns, and applying the SimpleImputer to the data.
- Variable Explorer:** Shows the current state of variables in the notebook's workspace. Variables include `ct`, `dataset`, `imputer`, `X`, and `y`.
- Plots:** Not visible in the screenshot.
- Files:** Not visible in the screenshot.
- Console 1/A:** Displays the history of code runs. Runs 4 through 9 show the command `runfile('C:/Users/joudm/Downloads/lab1_spider.py', wdir='C:/Users/joudm/Downloads')`. Run 10 shows the command `In [10]: runfile('C:/Users/joudm/Downloads/lab1_spider.py', wdir='C:/Users/joudm/Downloads')`.
- Data Preview (X - NumPy object array (read only)):** A modal window showing a table of data with columns labeled 0, 1, 2, and 3. The data consists of 12 rows of country names and their corresponding values.

	0	1	2	3
0	France	44.0	187.0	72000.0
1	Spain	27.0	166.0	48000.0
2	Germany	30.0	169.0	54000.0
3	UK	57.0	171.0	73000.0
4	Germany	40.0	176.0	71000.0
5	France	35.0	170.0	58000.0
6	Spain	39.0	159.0	52000.0
7	France	48.0	197.0	79000.0
8	Germany	50.0	165.0	83000.0
9	France	37.0	174.0	67000.0
10	Spain	38.0	170.0	61000.0
11	UK	41.0	167.0	71000.0

4.ENCODING THE CATEGORICAL DATA

The screenshot shows a Jupyter Notebook interface with two main panes. The left pane displays a Python script named `lab1_spider.py` containing code for data preprocessing and encoding. The right pane shows the resulting variables and their values.

Code in `lab1_spider.py`:

```
15 #extract the independent variables -function of the Pandas library- .
16 #extract selected rows and columns from the dataset.
17 X= dataset.iloc[:, :-1].values
18 y= dataset.iloc[:, 3].values
19
20 #3. Calculate the median for each feature or column that contains a missing value and
21 from sklearn.impute import SimpleImputer
22 imputer = SimpleImputer(missing_values= np.nan, strategy='median')
23 imputer = imputer.fit(X[:, 1:4])
24 X[:, 1:4] = imputer.transform(X[:, 1:4])
25
26
27 # 4.Encoding the categorical data
28 from sklearn.preprocessing import LabelEncoder, OneHotEncoder
29 from sklearn.compose import ColumnTransformer
30
31 ct = ColumnTransformer([('encoder', OneHotEncoder(), [0])], remainder ='passthrough')
32 X = np.array(ct.fit_transform(X), dtype= float)
33 y = LabelEncoder().fit_transform(y)
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
```

Variables and their values:

Name	Type	Size	Value
ct	compose._column_transformer.ColumnTransformer	1	ColumnTransformer object of sklearn.compose._column_transformer module
dataset	DataFrame	(17, 5)	Column names: Country, Age, Tall, Salary, Purchased
imputer	impute._base.SimpleImputer	1	SimpleImputer object of sklearn.impute._base module
X	Array of float64	(17, 7)	[[1.00e+00 0.00e+00 0.00e+00 ... 4.40e+01 1.87e+02 7.20e+04] [0.00e+0 ...]]
y	Array of int64	(17,)	[8 0 2 ... 13 5 12]

Output Data:

A NumPy array named `X` is displayed in a separate window. The array has 17 rows and 7 columns. The columns represent the transformed features: Country (0), Age (1), Tall (2), Salary (3), Purchased (4), and two additional numerical columns (5 and 6).

	0	1	2	3	4	5	6
0	1	0	0	0	44	187	72000
1	0	0	1	0	27	166	48000
2	0	1	0	0	30	169	54000
3	0	0	0	1	57	171	73000
4	0	1	0	0	40	176	71000
5	1	0	0	0	35	170	58000
6	0	0	1	0	39	159	52000
7	1	0	0	0	48	197	79000
8	0	1	0	0	50	165	83000
9	1	0	0	0	37	174	67000
10	0	0	1	0	38	170	61000

5. SPLITTING THE DATASET INTO THE TRAINING SET AND TEST SET AND ASSIGN 30% FOR TEST SET. (RANDOM_STATE = 0).

The screenshot shows a Jupyter Notebook interface with the following elements:

- Code Cell (lab1_spider.py):** Contains Python code for data preprocessing and splitting. The relevant part is:

```
32 X = np.array(ct.fit_transform(X), dtype= float)
33 y = LabelEncoder().fit_transform(y)
34
35 #5. Splitting the dataset into the Training set and Test Set and assign 30% for test set. (random_state = 0).
36 from sklearn.model_selection import train_test_split
37 X_train,X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=0)
```
- Data Preview Window (X_train):** Shows a 12x7 NumPy array. The columns are labeled 0 through 6. The last column contains values like 54000, 71000, 61000, etc.
- Data Preview Window (X_test):** Shows a 6x7 NumPy array. The last column contains values like 48000, 52000, 83000, etc.
- Variable Explorer:** A sidebar table showing the variables used in the code. It includes:

Name	Type	Size	Value
ct	compose._column_transformer.ColumnTransformer	1	ColumnTransformer object of sklear...
dataset	DataFrame	(17, 5)	Column names: Country, Age, Tall, Salary, Purchased
imputer	impute._base.SimpleImputer	1	SimpleImputer object of sklearn.impute._base module
X	Array of float64	(17, 7)	[[1.00e+00 0.00e+00 0.00e+00 ... 4...
X_test	Array of float64	(6, 7)	[[0.00e+00 0.00e+00 1.00e+00 ... 2...
X_train	Array of float64	(11, 7)	[[0.00e+00 1.00e+00 0.00e+00 ... 3...
y	Array of int64	(17,)	[8 0 2 ... 13 5 12]
y_test	Array of int64	(6,)	[0 1 12 6 10 13]
y_train	Array of int64	(11,)	[2 13 4 ... 3 5 8]
- Output Cell (In [12]):** Displays the command used to split the data: `In [12]:`
- Bottom Navigation:** Includes buttons for "Format", "Save and Close", "Close", and tabs for "IPython Console" and "History".

C:\Users\joudm\Downloads\lab1_spider.py

lab1_spider.py X

```
32     X = np.array(ct.fit_transform(X), dtype= float)
33     y = LabelEncoder().fit_transform(y)
34
35     #5. Splitting the dataset into the Training set and Test Set and assign 30% for test set. (random_state = 0).
36     from sklearn.model_selection import train_test_split
37     X_train,X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=0)
38
39
40
41
42
```

y_test - NumPy object array

	0
0	0
1	1
2	12
3	6
4	10
5	13

y_train - NumPy object array

	0
0	2
1	13
2	4
3	11
4	12
5	7
6	9
7	8
8	3
9	5
10	8

Variable Explorer

Name	Type	Size	Value
ct	compose._column_transformer.ColumnTransformer	1	ColumnTransformer object of sklear...
dataset	DataFrame	(17, 5)	Column names: Country, Age, Tall, Salary, Purchased
imputer	impute._base.SimpleImputer	1	SimpleImputer object of sklearn.impute._base module
X	Array of float64	(17, 7)	[[1.00e+00 0.00e+00 0.00e+00 ... 4...
X_test	Array of float64	(6, 7)	[[0.00e+00 0.00e+00 1.00e+00 ... 2...
X_train	Array of float64	(11, 7)	[[0.00e+00 1.00e+00 0.00e+00 ... 3...
y	Array of int64	(17,)	[8 0 2 ... 13 5 12]
y_test	Array of int64	(6,)	[0 1 12 6 10 13]
y_train	Array of int64	(11,)	[2 13 4 ... 3 5 8]

Console 1/A X

```
Downloads')

In [7]: runfile('C:/Users/joudm/Downloads/lab1_spider.py', wdir='C:/Users/joudm/Downloads')

In [8]: runfile('C:/Users/joudm/Downloads/lab1_spider.py', wdir='C:/Users/joudm/Downloads')

In [9]: runfile('C:/Users/joudm/Downloads/lab1_spider.py', wdir='C:/Users/joudm/Downloads')

In [10]: runfile('C:/Users/joudm/Downloads/lab1_spider.py', wdir='C:/Users/joudm/Downloads')

In [11]: runfile('C:/Users/joudm/Downloads/lab1_spider.py', wdir='C:/Users/joudm/Downloads')

In [12]:
```

6. APPLY THE FEATURE SCALING

The screenshot shows a Jupyter Notebook interface with several panes:

- Code Editor (Top Left):** Displays the Python script `lab1_spider.py` containing code for feature scaling.
- Variable Explorer (Top Right):** Shows a table of variables with their types and values.
- Object Inspector (Bottom Left):** Shows the details of the `sc_X` object, which is a `StandardScaler` instance.
- Console (Bottom Right):** Displays the history of code runs and their outputs.

Code in `lab1_spider.py`:

```
32 X = np.array(ct.fit_transform(X), dtype= float)
33 y = LabelEncoder().fit_transform(y)
34
35 #5. Splitting the dataset into the Training set and Test Set and assign 30% for test set. (random_state = 0).
36 from sklearn.model_selection import train_test_split
37 X_train,X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=0)
38
39 #6.Feature Scaling
40 from sklearn.preprocessing import StandardScaler
41 sc_X = StandardScaler()
42 X_train = sc_X.fit_transform(X_train)
43 X_test = sc_X.fit_transform(X_test)
44
45
46 sc_X - Object
47
48
49
50 Name      Type      Size  Value
51 sc_X      preprocessing._data.StandardScaler  1 StandardScaler object of sklearn.preprocessing.
52 > _check_feature_names    method      1 method object
53 > _check_n_features     method      1 method object
54 > _get_param_names      method      1 method object
55 > _get_tags              method      1 method object
56 > _more_tags             method      1 method object
57 > _repr_html_inner       method      1 method object
58 > _repr_mimebundle_      method      1 method object
59 > _reset                 method      1 method object
60 > _validate_data         method      1 method object
61 > copy                  bool        1 True
62 > fit                   method      1 method object
63 > fit_transform           method      1 method object
64
65 Details
66 Documentation
67 1 Standardize features by removing the mean and scaling to unit variance.
68 Source code
69 2
70 3 The standard score of a sample `x` is calculated as:
71 4
72 5 z = (x - u) / s
73 6
```

Variable Explorer Data:

Name	Type	Size	Value
dataset	DataFrame	(17, 5)	Column names: Country, Age, Tall, Salary, Purchased
imputer	impute._base.SimpleImputer	1	SimpleImputer object of sklearn.impute._base module
sc_X	preprocessing._data.StandardScaler	1	StandardScaler object of sklearn...
X	Array of float64	(17, 7)	[[1.00e+00 0.00e+00 0.00e+00 ...]
X_test	Array of float64	(6, 7)	[-0.4472136 -1. 1.414...
X_train	Array of float64	(11, 7)	[-0.75592895 3.16227766 -0.471...
y	Array of int64	(17,)	[8 0 2 ... 13 5 12]
y_test	Array of int64	(6,)	[0 1 12 6 10 13]
y_train	Array of int64	(11,)	[2 13 4 ... 3 5 8]

Console History:

```
Downloads')
In [8]: runfile('C:/Users/joudm/Downloads/Lab1_spider.py', wdir='C:/Users/joudm/Downloads')
In [9]: runfile('C:/Users/joudm/Downloads/Lab1_spider.py', wdir='C:/Users/joudm/Downloads')
In [10]: runfile('C:/Users/joudm/Downloads/Lab1_spider.py', wdir='C:/Users/joudm/Downloads')
In [11]: runfile('C:/Users/joudm/Downloads/Lab1_spider.py', wdir='C:/Users/joudm/Downloads')
In [12]: runfile('C:/Users/joudm/Downloads/Lab1_spider.py', wdir='C:/Users/joudm/Downloads')
In [13]:
```

6. APPLY THE FEATURE SCALING

The screenshot shows a Jupyter Notebook environment with three main components:

- Code Cell:** A code editor window titled "lab1_spider.py X" containing Python code for feature scaling. The code includes importing libraries, fitting a ColumnTransformer, splitting the dataset, and applying StandardScaler to the training set.
- Variable Explorer:** A sidebar table showing the variables used in the code. It lists "dataset" as a DataFrame, "imputer" as a SimpleImputer object, "sc_X" as a StandardScaler object, "X" as an array of float64, "X_test" as an array of float64, "X_train" as an array of float64, and "y" as an array of int64.
- Output Cells:** Two output windows. The first, titled "X_test - NumPy object array", displays the scaled test data as a 6x7 grid. The second, titled "X_train - NumPy object array", displays the scaled training data as a 10x7 grid.

```
C:\Users\joudm\Downloads\lab1_spider.py
lab1_spider.py X
32 X = np.array(ct.fit_transform(X), dtype= float)
33 y = LabelEncoder().fit_transform(y)
34
35 #5. Splitting the dataset into the Training set and Test Set and assign 30% for test set. (random_state = 0).
36 from sklearn.model_selection import train_test_split
37 X_train,X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=0)
38
39 #6.Feature Scaling
40 from sklearn.preprocessing import StandardScaler
41 sc_X = StandardScaler()
42 X_train = sc_X.fit_transform(X_train)
43 X_test = sc_X.fit_transform(X_test)
44
45
```

X_test - NumPy object array

	0	1	2	3	4	5	6
0	-0.447214	-1	1.41421	0	-1.77164	-0.380221	-1.45291
1	-0.447214	-1	1.41421	0	-0.115041	-1.60863	-1.13591
2	-0.447214	1	-0.707107	0	1.4035	-0.555708	1.32083
3	2.23607	-1	-0.707107	0	-0.391141	1.02367	0.0528332
4	-0.447214	1	-0.707107	0	0.851306	0.146239	0.845331
5	-0.447214	1	-0.707107	0	0.0230083	1.37465	0.369832

X_train - NumPy object array

	0	1	2	3	4	5	6
0	-0.755929	3.16228	-0.471405	-0.755929	-1.29759	-0.698702	-1.77862
1	-0.755929	-0.316228	2.12132	-0.755929	-1.06738	-0.590453	0.249441
2	-0.755929	-0.316228	2.12132	-0.755929	-0.376721	-0.590453	-0.943536
3	1.32288	-0.316228	-0.471405	-0.755929	0.77437	2.33229	1.20382
4	1.32288	-0.316228	-0.471405	-0.755929	1.69524	-0.806952	1.68101
5	-0.755929	-0.316228	-0.471405	1.32288	-0.0313934	-0.915202	0.249441
6	-0.755929	-0.316228	-0.471405	1.32288	1.81035	-0.482203	0.488036
7	1.32288	-0.316228	-0.471405	-0.755929	0.313934	1.24979	0.368738
8	1.32288	-0.316228	-0.471405	-0.755929	-0.722048	-0.590453	-1.30143
9	-0.755929	-0.316228	-0.471405	1.32288	-0.376721	0.816793	-0.585643
10	-0.755929	-0.316228	-0.471405	1.32288	-0.722048	0.275545	0.368738