REPORT ON DATA DRIVEN RESEARCH

Alvaro Balderrama

October 3rd, 2024

IBM Developer

SKILLS NETWORK

OUTLINE

- Executive Summary

1. Introduction
2. Methodology
3. Results
   - 3.1. Data collection and data wrangling methodology
   - 3.2. EDA and interactive visual analytics methodology
   - 3.3. Predictive analysis methodology
   - 3.4. EDA with visualization results
   - 3.5. EDA with SQL results
   - 3.6. Interactive map with Folium results
   - 3.7. Plotly Dash dashboard results
   - 3.8. Predictive analysis (classification) results
4. Discussion
   - Findings & Implications
5. Conclusion

- Appendix

EXECUTIVE SUMMARY

This report was developed in October 2024, as part of the final task of the online course “Data Science and Machine Learning Capstone Project”, part of the Professional Certificate Program: IBM Python Data Science. It is the fifth course of a series which included four other courses: (i) Python Basics for Data Science; (ii) Analyzing Data with Python; (iii) Visualizing Data with Python; (iv) Machine Learning with Python: A Practical Introduction.

The Report includes a summary of the content related to data wrangling, EDA visual analytics, predictive analysis, SQL, interactive map with Folium, Plotly Dash dashboard, and predictive analysis (classification).

1. INTRODUCTION

- Data science is an interdisciplinary field that combines scientific methods, algorithms, and systems to extract knowledge and insights from structured and unstructured data. It involves a blend of skills from areas such as statistics, computer science, and domain expertise, with the goal of uncovering meaningful patterns, trends, and insights that can inform decisions or drive predictions.
- The Professional Certificate program IBM Python Data Science covers the essential aspects needed to kickstart a career that involves data science to some extent.
- The completion of the tasks required presented next, is the final project of a series of courses that started in December 2023 and concluded in October 2024.

2. METHODOLOGY

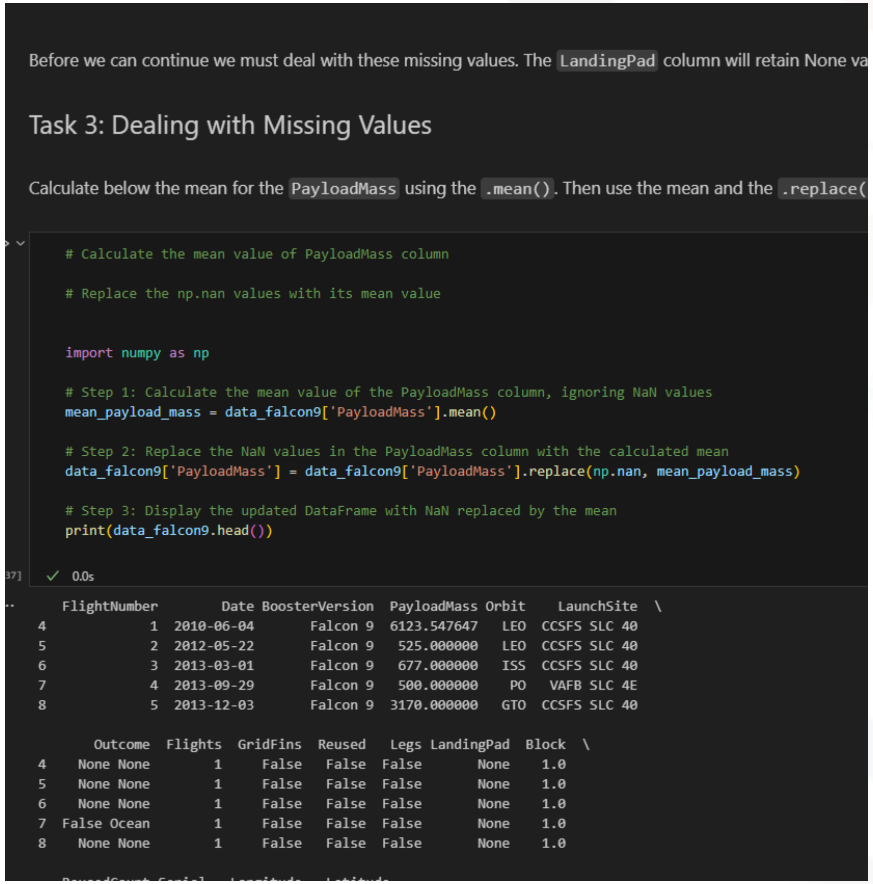
The process of the courses of this certificate program were documented by taking screenshots during the videos, and compiled in Power Point. For this presentation, some of those slides are presented to summarize specific topics.

A GitHub repository was created for this assignment:
https://github.com/alvarobalderrama/MLCapstoneProject/tree/main

3. RESULTS

3.1. Data collection and data wrangling methodology
3.2. EDA and interactive visual analytics methodology
3.3. Predictive analysis methodology
3.4. EDA with visualization results
3.5. EDA with SQL results
3.6. Interactive map with Folium results
3.7. Plotly Dash dashboard results
3.8. Predictive analysis (classification) results

3.1. Data collection and data wrangling methodology



Before we can continue we must deal with these missing values. The `LandingPad` column will retain None va

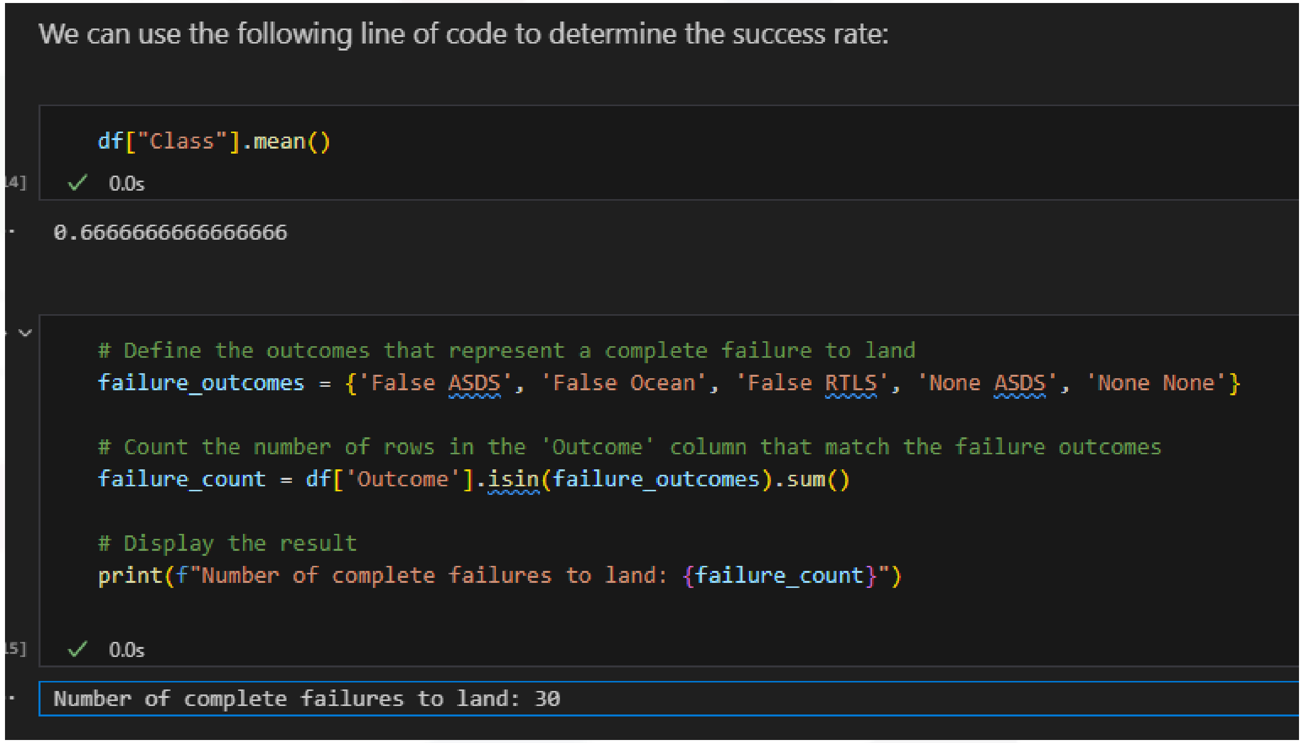
Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace(`

```python
# Calculate the mean value of PayloadMass column

# Replace the np.nan values with its mean value


import numpy as np

# Step 1: Calculate the mean value of the PayloadMass column, ignoring NaN values
mean_payload_mass = data_falcon9['PayloadMass'].mean()

# Step 2: Replace the NaN values in the PayloadMass column with the calculated mean
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, mean_payload_mass)

# Step 3: Display the updated DataFrame with NaN replaced by the mean
print(data_falcon9.head())
```

✓ 0.0s

```
   FlightNumber        Date BoosterVersion  PayloadMass Orbit    LaunchSite  \
4             1  2010-06-04       Falcon 9  6123.547647   LEO  CCSFS SLC 40
5             2  2012-05-22       Falcon 9   525.000000   LEO  CCSFS SLC 40
6             3  2013-03-01       Falcon 9   677.000000   ISS  CCSFS SLC 40
7             4  2013-09-29       Falcon 9   500.000000    PO   VAFB SLC 4E
8             5  2013-12-03       Falcon 9  3170.000000   GTO  CCSFS SLC 40

       Outcome  Flights  GridFins  Reused   Legs LandingPad  Block  \
4    None None        1     False   False  False       None    1.0
5    None None        1     False   False  False       None    1.0
6    None None        1     False   False  False       None    1.0
7  False Ocean        1     False   False  False       None    1.0
8    None None        1     False   False  False       None    1.0

   ReusedCount Serial   Longitude   Latitude
```



We can use the following line of code to determine the success rate:

```python
df["Class"].mean()
```

✓ 0.0s

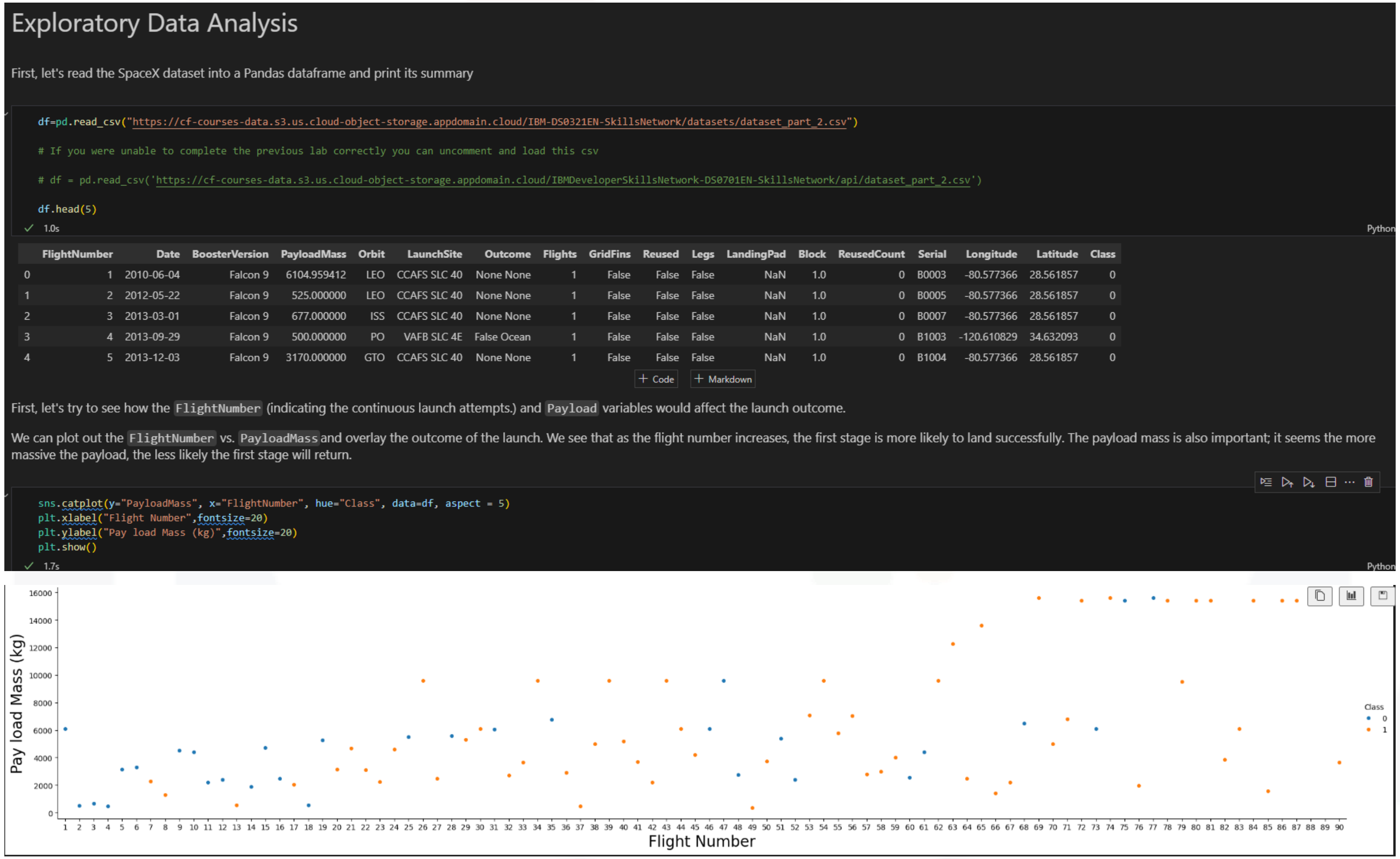
```
0.6666666666666666
```

```python
# Define the outcomes that represent a complete failure to land
failure_outcomes = {'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}

# Count the number of rows in the 'Outcome' column that match the failure outcomes
failure_count = df['Outcome'].isin(failure_outcomes).sum()

# Display the result
print(f"Number of complete failures to land: {failure_count}")
```

✓ 0.0s

```
Number of complete failures to land: 30
```

3.2. EDA and interactive visual analytics methodology



Exploratory Data Analysis

First, let's read the SpaceX dataset into a Pandas dataframe and print its summary

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv")

# If you were unable to complete the previous lab correctly you can uncomment and load this csv

# df = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/api/dataset_part_2.csv')

df.head(5)
```

✓ 1.0s

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

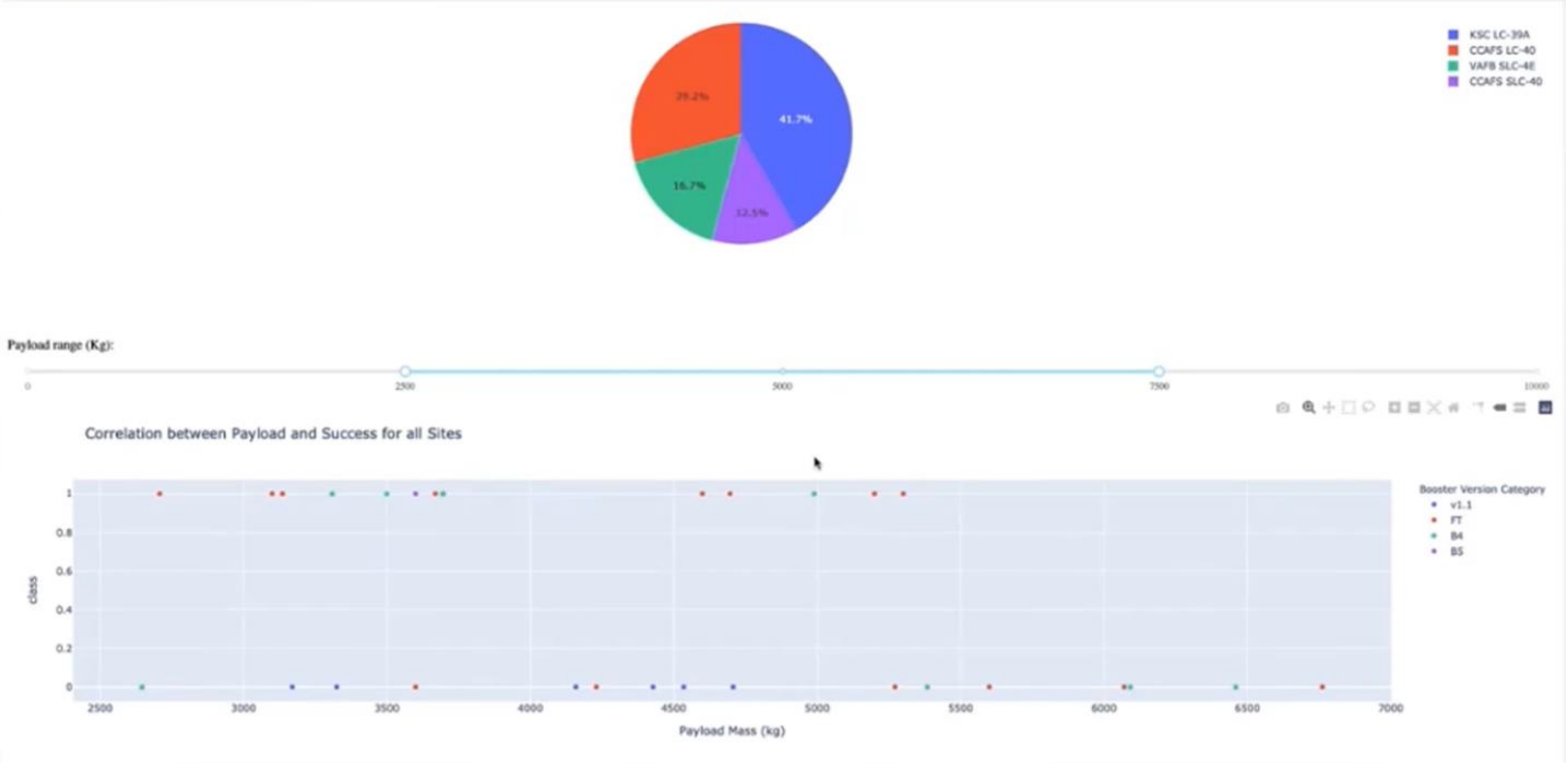
First, let's try to see how the `FlightNumber` (indicating the continuous launch attempts.) and `Payload` variables would affect the launch outcome.

We can plot out the `FlightNumber` vs. `PayloadMass` and overlay the outcome of the launch. We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.

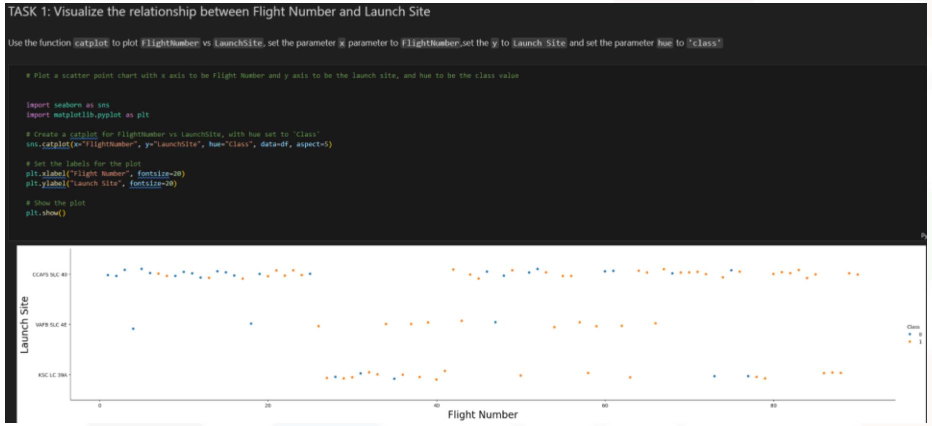
```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```

✓ 1.7s

3.3. Predictive analysis methodology



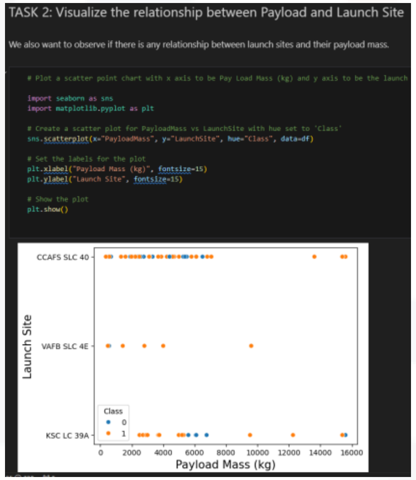
3.4. EDA with visualization results



TASK 1: Visualize the relationship between Flight Number and Launch Site

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

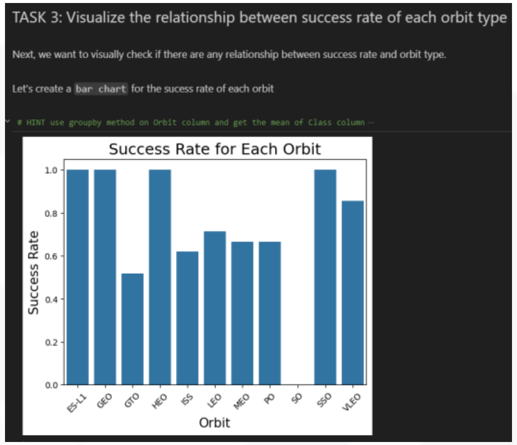
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value


import seaborn as sns
import matplotlib.pyplot as plt

# Create a catplot for FlightNumber vs LaunchSite, with hue set to 'Class'
sns.catplot(x="FlightNumber", y="LaunchSite", hue="Class", data=df, aspect=5)

# Set the labels for the plot
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)

# Show the plot
plt.show()
```



TASK 2: Visualize the relationship between Payload and Launch Site

We also want to observe if there is any relationship between launch sites and their payload mass.

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch

import seaborn as sns
import matplotlib.pyplot as plt

# Create a scatter plot for PayloadMass vs LaunchSite with hue set to 'Class'
sns.scatterplot(x="PayloadMass", y="LaunchSite", hue="Class", data=df)

# Set the labels for the plot
plt.xlabel("Payload Mass (kg)", fontsize=15)
plt.ylabel("Launch Site", fontsize=15)

# Show the plot
plt.show()
```

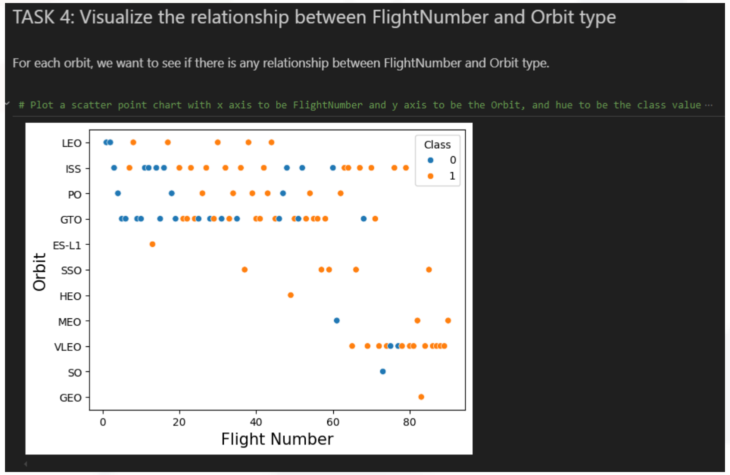


TASK 3: Visualize the relationship between success rate of each orbit type

Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a `bar chart` for the sucess rate of each orbit

```
# HINT use groupby method on Orbit column and get the mean of Class column ···
```

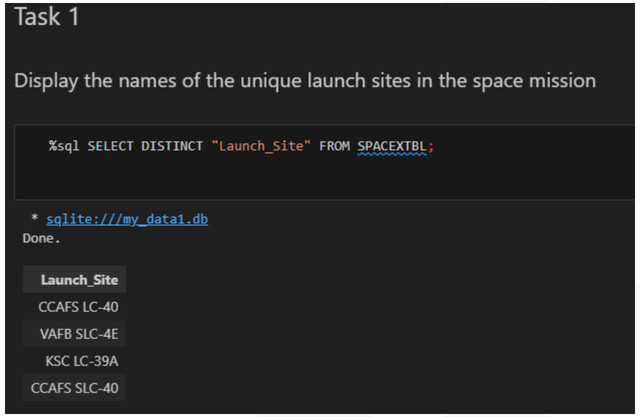


TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value ···
```

3.5. EDA with SQL results



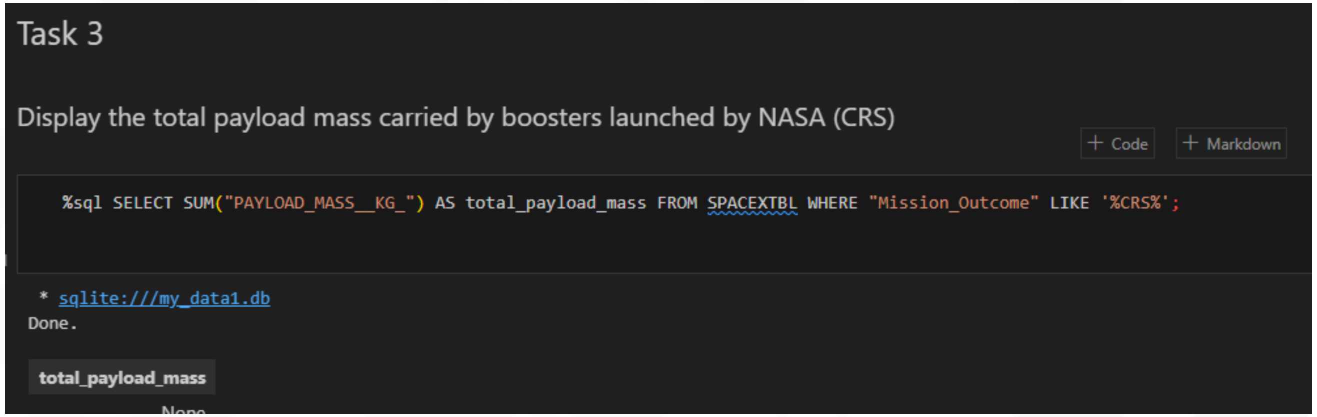
Task 1

Display the names of the unique launch sites in the space mission

```
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTBL;
```

```
 * sqlite:///my_data1.db
Done.
```

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40



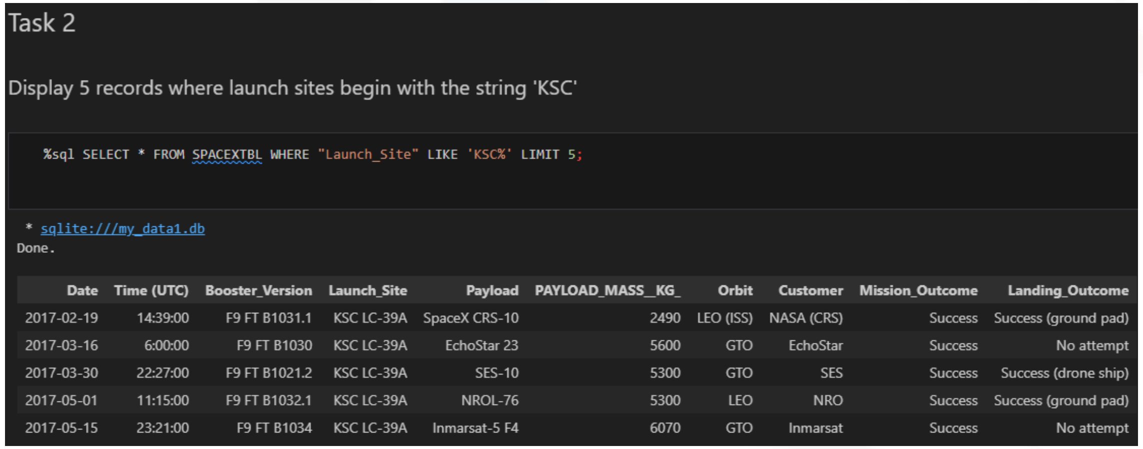
Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

```
%sql SELECT SUM("PAYLOAD_MASS__KG_") AS total_payload_mass FROM SPACEXTBL WHERE "Mission_Outcome" LIKE '%CRS%';
```

```
 * sqlite:///my_data1.db
Done.
```

total_payload_mass
None



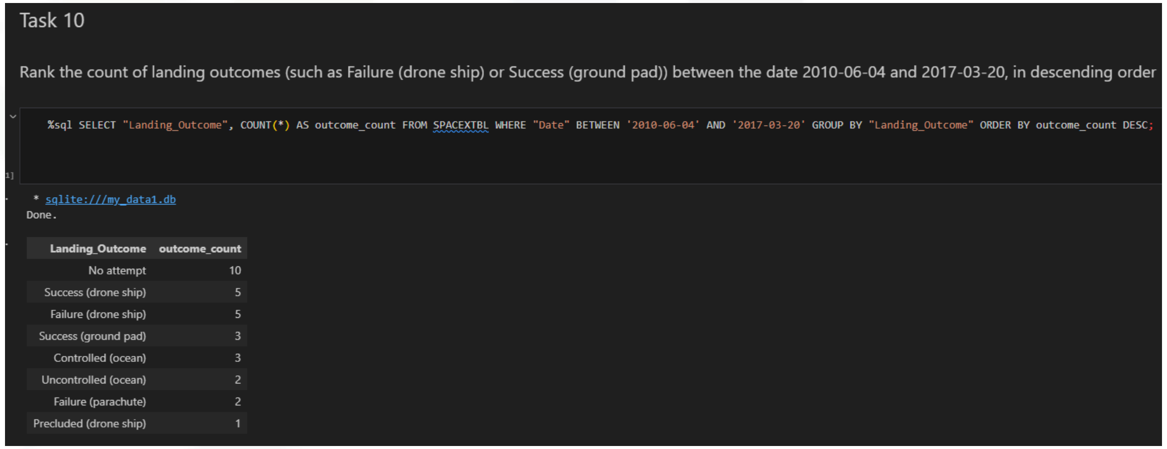
Task 2

Display 5 records where launch sites begin with the string 'KSC'

```
%sql SELECT * FROM SPACEXTBL WHERE "Launch_Site" LIKE 'KSC%' LIMIT 5;
```

```
 * sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-03-16	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt



Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
%sql SELECT "Landing_Outcome", COUNT(*) AS outcome_count FROM SPACEXTBL WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY "Landing_Outcome" ORDER BY outcome_count DESC;
```

```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	outcome_count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

3.6. Interactive map with Folium results

Analyze Launch Site Geo Data with Folium

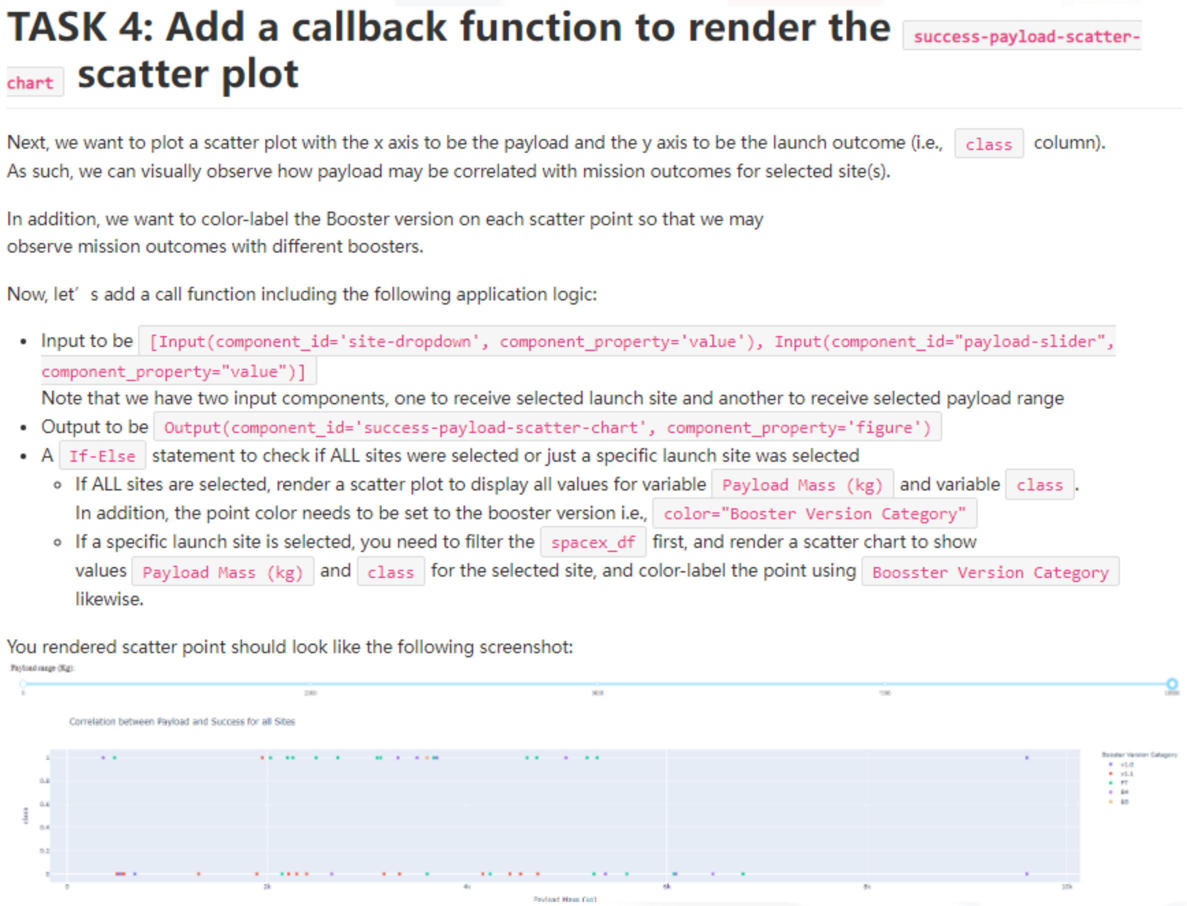
- Mark the locations and proximities of launch sites

- Discover patterns via exploring the map

- Explain how to choose an optimal launch site locations

7
AFS
SLC-
40
26
AFS
LC-
40

3.7. Plotly Dash dashboard results



TASK 4: Add a callback function to render the `success-payload-scatter-chart` scatter plot

Next, we want to plot a scatter plot with the x axis to be the payload and the y axis to be the launch outcome (i.e., `class` column). As such, we can visually observe how payload may be correlated with mission outcomes for selected site(s).

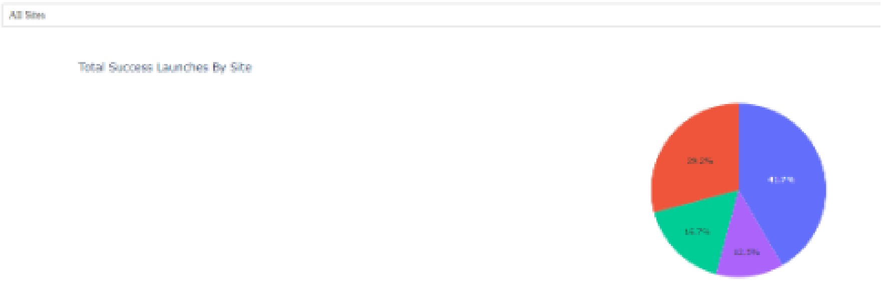
In addition, we want to color-label the Booster version on each scatter point so that we may observe mission outcomes with different boosters.

Now, let' s add a call function including the following application logic:

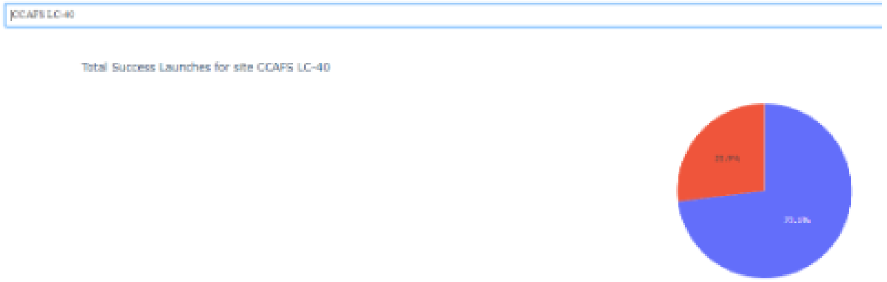
- Input to be `[Input(component_id='site-dropdown', component_property='value'), Input(component_id="payload-slider", component_property="value")]`
  Note that we have two input components, one to receive selected launch site and another to receive selected payload range
- Output to be `Output(component_id='success-payload-scatter-chart', component_property='figure')`
- A `If-Else` statement to check if ALL sites were selected or just a specific launch site was selected
  - If ALL sites are selected, render a scatter plot to display all values for variable `Payload Mass (kg)` and variable `class`.
    In addition, the point color needs to be set to the booster version i.e., `color="Booster Version Category"`
  - If a specific launch site is selected, you need to filter the `spacex_df` first, and render a scatter chart to show values `Payload Mass (kg)` and `class` for the selected site, and color-label the point using `Boosster Version Category` likewise.

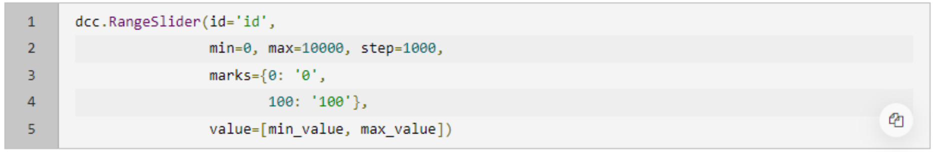
You rendered scatter point should look like the following screenshot:

- Pie chart for all sites are selected



- Pie chart for is selected



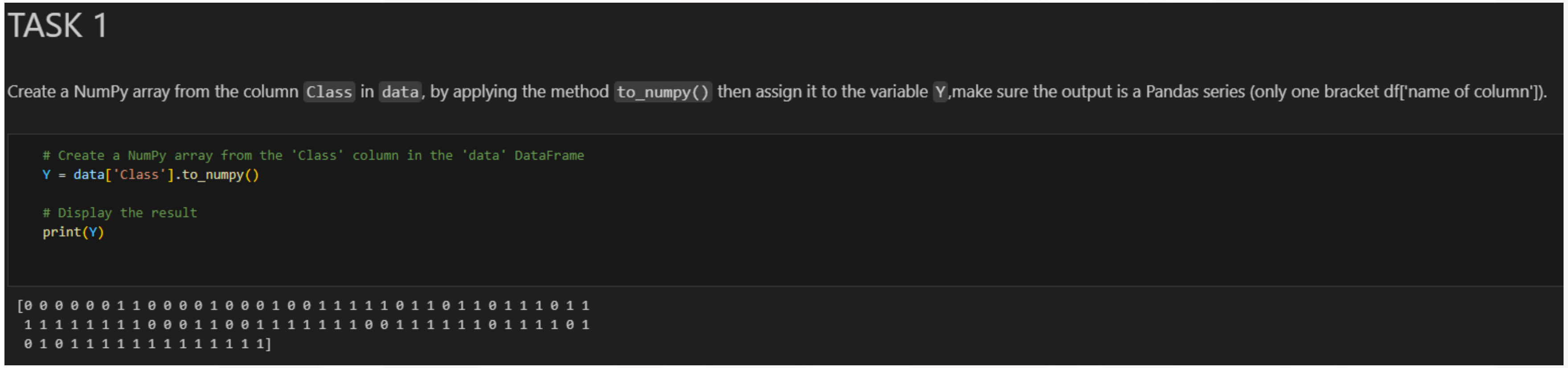


```
dcc.RangeSlider(id='id',
                min=0, max=10000, step=1000,
                marks={0: '0',
                       100: '100'},
                value=[min_value, max_value])
```

IBM Developer

SKILLS NETWORK

3.8. Predictive analysis (classification) results

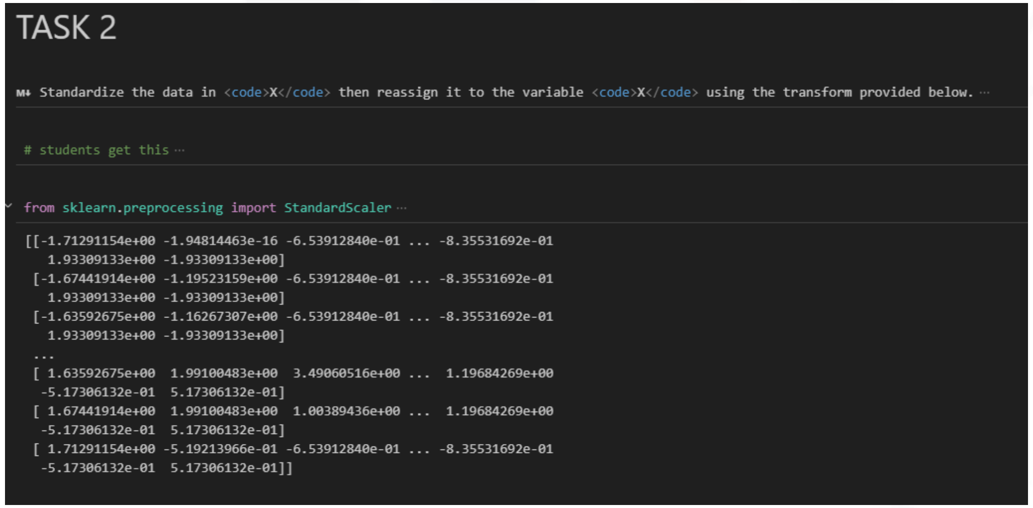


TASK 1

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` then assign it to the variable `Y`,make sure the output is a Pandas series (only one bracket df['name of column']).

```
# Create a NumPy array from the 'Class' column in the 'data' DataFrame
Y = data['Class'].to_numpy()

# Display the result
print(Y)
```

```
[0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1
 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 0 1
 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1]
```



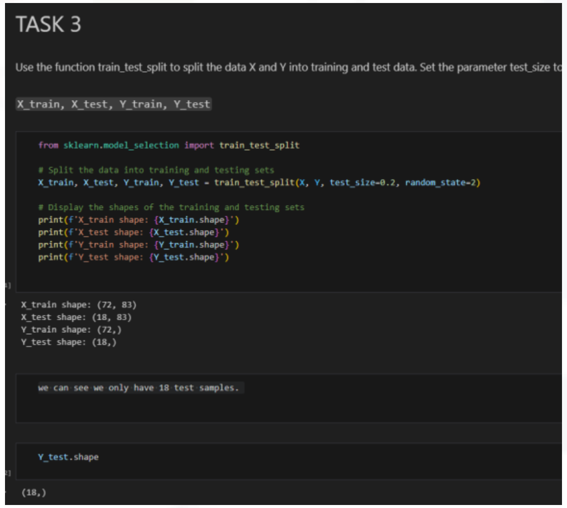
TASK 2

```
M↓ Standardize the data in <code>X</code> then reassign it to the variable <code>X</code> using the transform provided below. ···
```

```
# students get this ···
```

```
from sklearn.preprocessing import StandardScaler ···
```

```
[[-1.71291154e+00 -1.94814463e-16 -6.53912840e-01 ... -8.35531692e-01
   1.93309133e+00 -1.93309133e+00]
 [-1.67441914e+00 -1.19523159e+00 -6.53912840e-01 ... -8.35531692e-01
   1.93309133e+00 -1.93309133e+00]
 [-1.63592675e+00 -1.16267307e+00 -6.53912840e-01 ... -8.35531692e-01
   1.93309133e+00 -1.93309133e+00]
 ...
 [ 1.63592675e+00  1.99100483e+00  3.49060516e+00 ...  1.19684269e+00
  -5.17306132e-01  5.17306132e-01]
 [ 1.67441914e+00  1.99100483e+00  1.00389436e+00 ...  1.19684269e+00
  -5.17306132e-01  5.17306132e-01]
 [ 1.71291154e+00 -5.19213966e-01 -6.53912840e-01 ... -8.35531692e-01
  -5.17306132e-01  5.17306132e-01]]
```



TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to

`X_train, X_test, Y_train, Y_test`

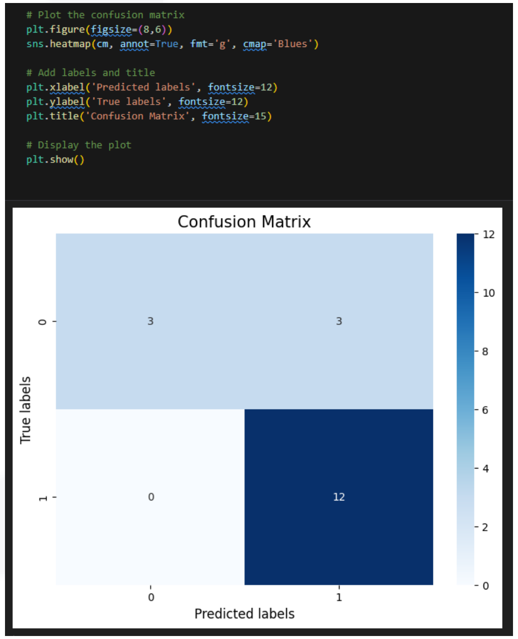
```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)

# Display the shapes of the training and testing sets
print(f'X_train shape: {X_train.shape}')
print(f'X_test shape: {X_test.shape}')
print(f'Y_train shape: {Y_train.shape}')
print(f'Y_test shape: {Y_test.shape}')
```

```
X_train shape: (72, 83)
X_test shape: (18, 83)
Y_train shape: (72,)
Y_test shape: (18,)
```

```
we can see we only have 18 test samples.
```

```
Y_test.shape
```

```
(18,)
```



```
# Plot the confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='g', cmap='Blues')

# Add labels and title
plt.xlabel('Predicted labels', fontsize=12)
plt.ylabel('True labels', fontsize=12)
plt.title('Confusion Matrix', fontsize=15)

# Display the plot
plt.show()
```

4. DISCUSSION

- The work developed during the Certificate course was presented in the Results section
- The topics covered were data wrangling, EDA, predictive analysis, visualizations, SQL, interactive maps, dashboarding, and classification.

5. CONCLUSION

- Report developed in October 2024 for the final task of the "Data Science and Machine Learning Capstone Project" in the IBM Python Data Science Professional Certificate Program. It is the fifth course after Python Basics, Analyzing Data, Visualizing Data, and Machine Learning with Python.
- Covers data wrangling, EDA, predictive analysis, SQL, Folium maps, Plotly Dash, and classification.
- This report is documented with screenshots from the exercises.
- A GitHub repository was created for the assignment.