IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Matt cust
09 July 2024

Outline

Executive Summary

- Summary of methodologies
  - All methodologies used can be found within the scripts stored in this git hub project: https://github.com/Matt-Cust/IBM-DS0720EN-Capstone-project
- Summary of all results

I know, a bad joke but I couldn't help it. I have to apologise, I am required to write reports often in y current job and have decided to have some fun with this one.

Section	Marks	Possible
Graded Quiz: Web Scraping	3	4
Graded Quiz: Data Wrangling	3	4
Graded Quiz: Exploratory Analysis using SQL	5	5
Graded Quiz: Complete the EDA with Visualization	3	3
Graded Quiz: Interactive Visual Analytics and Dashboard	10	10
Graded Quiz: Predictive Analysis	3	3
Peer Review: Submit your Work and Review your Peers	TBD	40

Executive Summary

- Summary of methodologies
  - All methodologies used can be found within the scripts stored in this git hub project: https://github.com/Matt-Cust/IBM-DS0720EN-Capstone-project
- Summary of all results
  - In summary several ML methods were attempted with varying parameters, these were all compared and the highest accuracy was somewhere in the 75-85% success at predicting safe landing of the stage 1 booster, not a bad level of accuracy but not enough for me to bet on.
  - I also think that location played a part something to do with payload mass, and overall they became better at safe landings as time went on.

Introduction

- Project background and context
  - In an effort to demonstrate and test upon all DS skills acquired throughout the course a final capstone project was performed. In this project data from SpaceX was provided, it was then cleaned via python and analysed with the primary goal of determining if it would be possible to correctly predict the safe landing of stage 1 boosters based upon other launch information.
- Problems you want to find answers
  - In this project I wanted to determine if I would be able to recall and employ my learnings from over the past few months regarding data science in python.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - .get from requests was used to get the data from the spaceX APIs - https://api.spacexdata.com/v4/rockets/ / https://api.spacexdata.com/v4/cores/ although in reality I just used the existing script to read_csv from the provided URL
  - There was also a section on web scraping where a series of table were extracted from Wikipedia where Beautiful soup was used to gather the information, and pre-fab functions used to coalesce the data into usable formats.
- Perform data wrangling
  - Missing data was identified and either the rows removed or missing data replaced with he column average, and abhorrent data type columns were shifted to a preferred data type.
- Perform exploratory data analysis (EDA) using visualization and SQL

Methodology

Executive Summary

- Perform interactive visual analytics using Folium and Plotly Dash.
- Perform predictive analysis using classification models
  - 4 different modeling methods were tested, and parameters within each model were assessed, the optimal parameters from each model were selected and then the models were compared using several metrics to determine which refined model performed the best.

Data Collection (1)

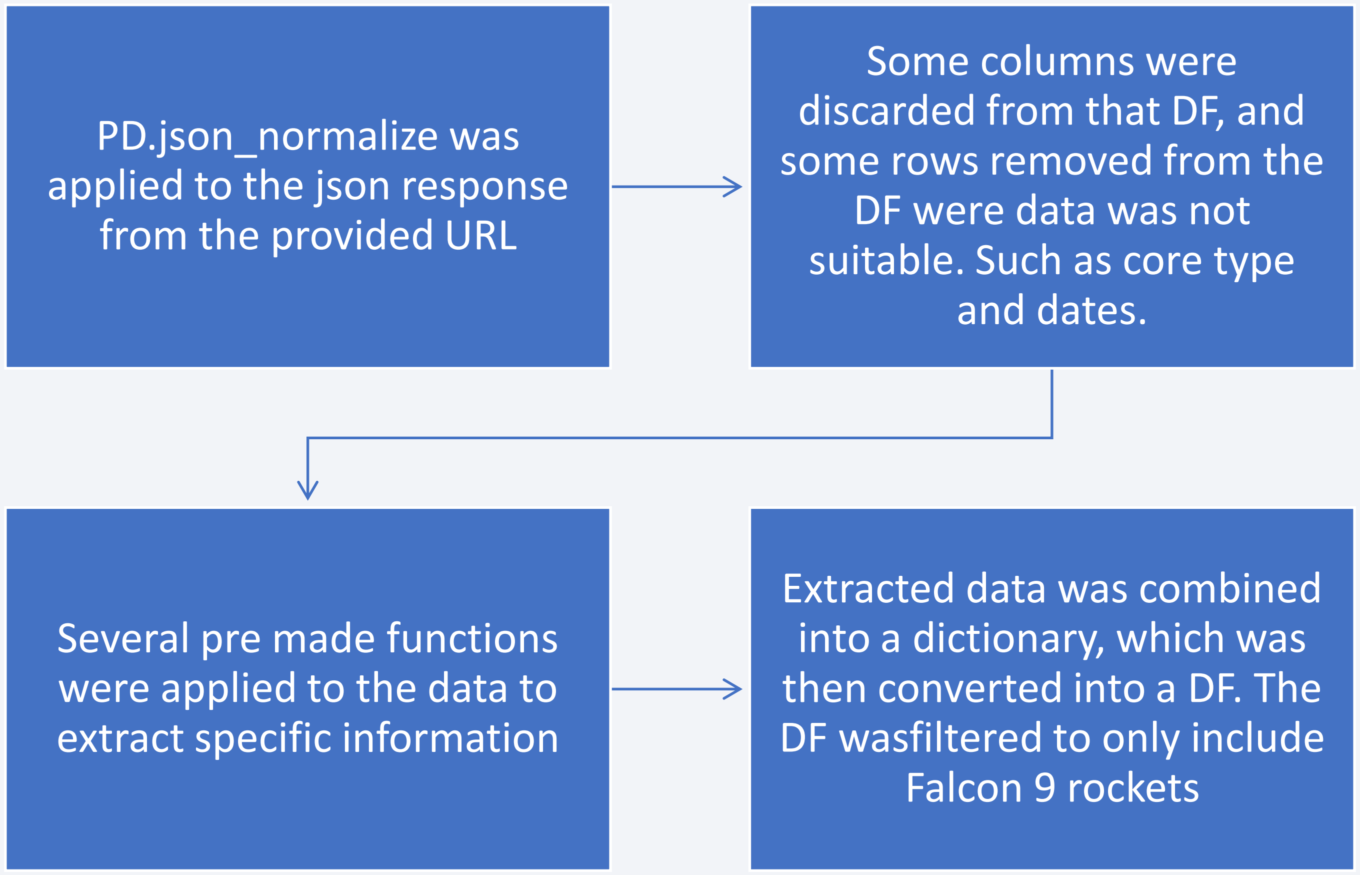
- See following slides, I sort of got ahead of myself and started populating this one and deleted the pre existing text.

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

- Add the GitHub URL of the completed SpaceX API calls notebook (must include completed code cell and outcome cell), as an external reference and peer-review purpose

- https://github.com/Matt-Cust/IBM-DS0720EN-Capstone-project/blob/main/L1-%20jupyter-labs-spacex-data-collection-api%20(1).ipynb



PD.json_normalize was applied to the json response from the provided URL

Some columns were discarded from that DF, and some rows removed from the DF were data was not suitable. Such as core type and dates.

Several pre made functions were applied to the data to extract specific information

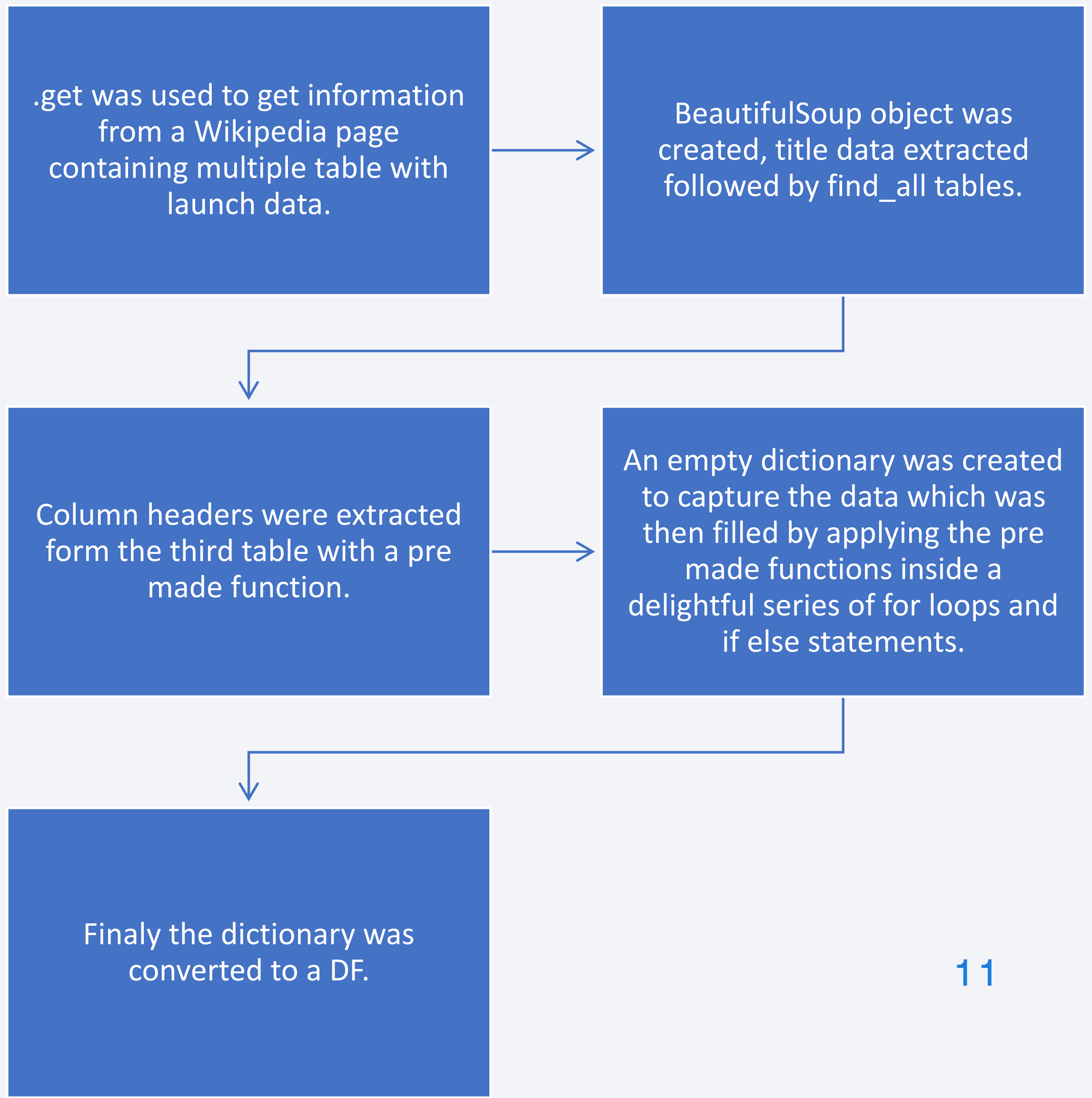
Extracted data was combined into a dictionary, which was then converted into a DF. The DF wasfiltered to only include Falcon 9 rockets

Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts

- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose

- https://github.com/Matt-Cust/IBM-DS0720EN-Capstone-project/blob/main/L2-%20jupyter-labs-webscraping%20(1).ipynb



.get was used to get information from a Wikipedia page containing multiple table with launch data.

BeautifulSoup object was created, title data extracted followed by find_all tables.

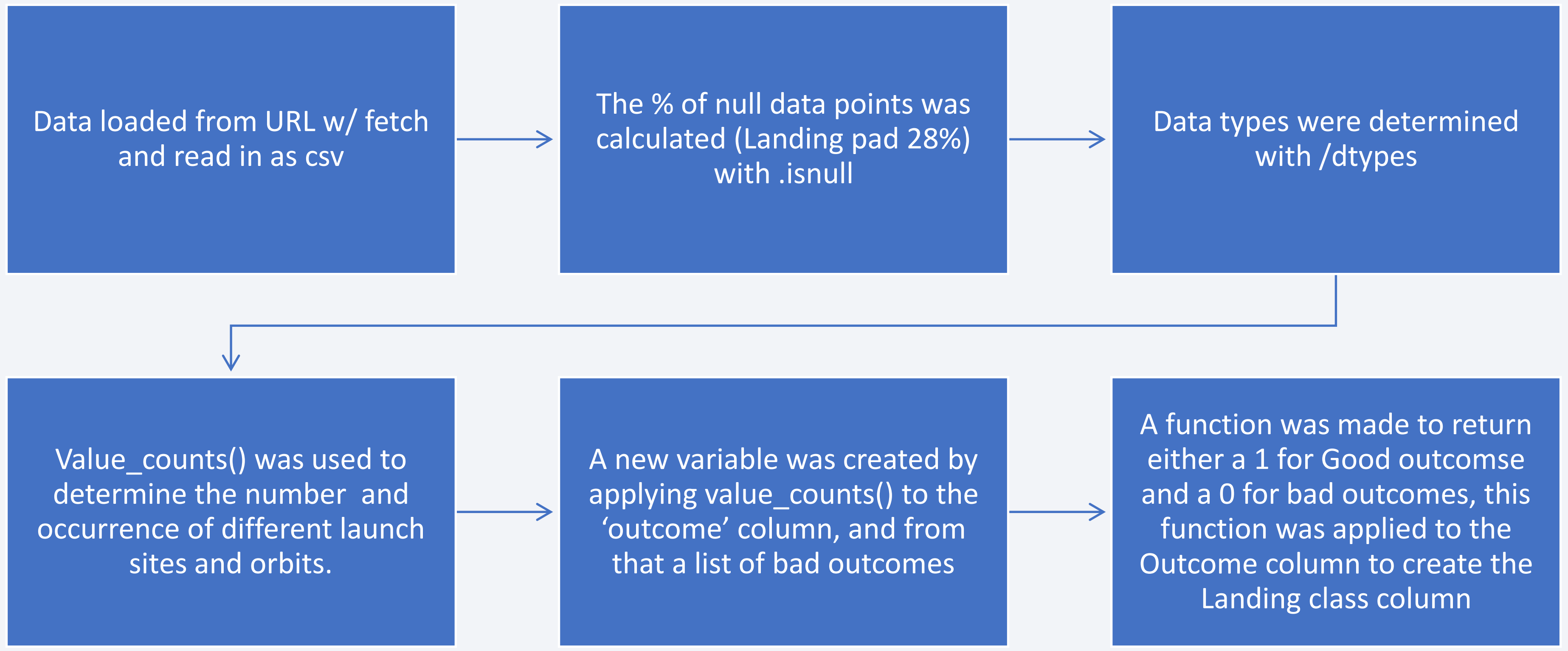
Column headers were extracted form the third table with a pre made function.

An empty dictionary was created to capture the data which was then filled by applying the pre made functions inside a delightful series of for loops and if else statements.

Finaly the dictionary was converted to a DF.

Data Wrangling

- https://github.com/Matt-Cust/IBM-DS0720EN-Capstone-project/blob/main/L3-%20labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb



Data loaded from URL w/ fetch and read in as csv

The % of null data points was calculated (Landing pad 28%) with .isnull

Data types were determined with /dtypes

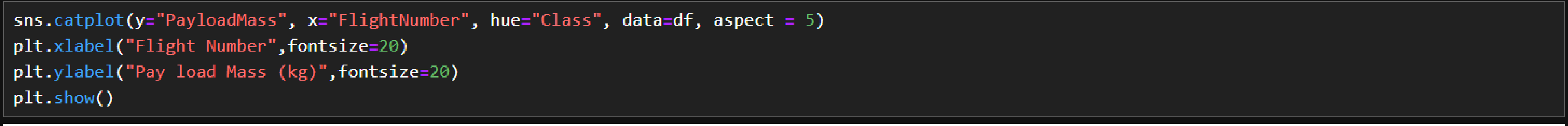
Value_counts() was used to determine the number and occurrence of different launch sites and orbits.

A new variable was created by applying value_counts() to the 'outcome' column, and from that a list of bad outcomes

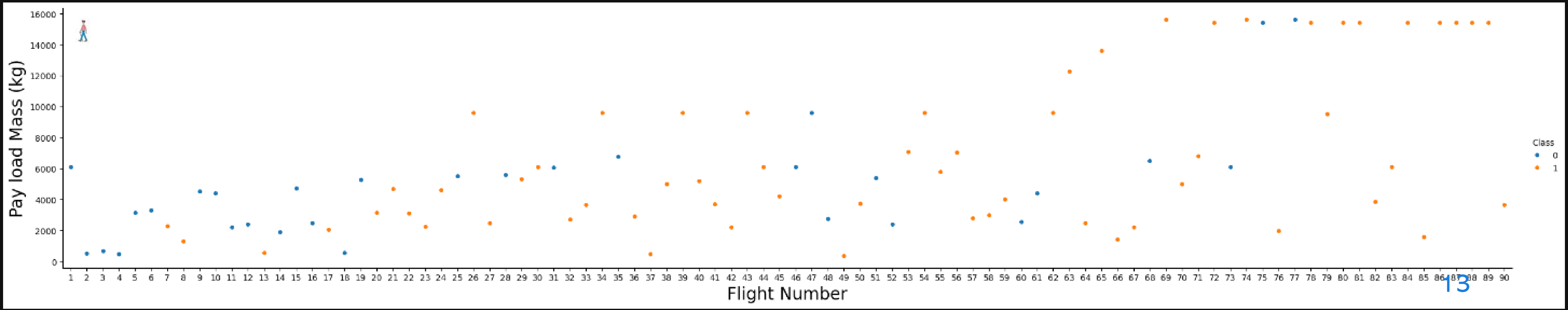
A function was made to return either a 1 for Good outcomse and a 0 for bad outcomes, this function was applied to the Outcome column to create the Landing class column

EDA with Data Visualization 1

- https://github.com/Matt-Cust/IBM-DS0720EN-Capstone-project/blob/main/L5-%20jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb
- Summarize what charts were plotted and why you used those charts
- Graphs from Jupyter labs in night mode

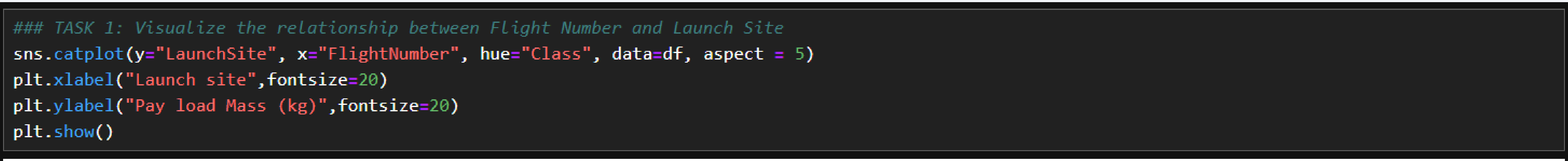


```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```

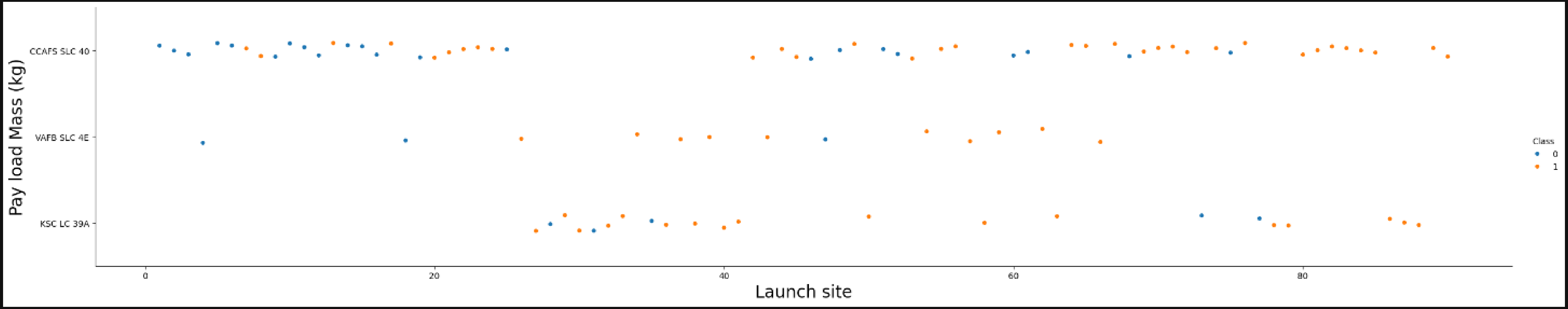


We see that different launch sites have different success rates. `CCAFS LC-40`, has a success rate of 60 %, while `KSC LC-39A` and `VAFB SLC 4E` has a success rate of 77%.

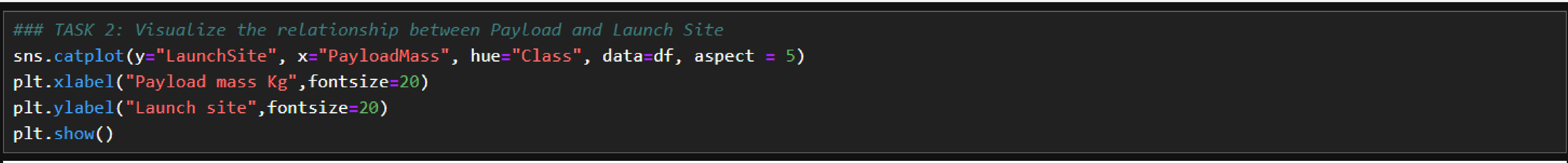
EDA with Data Visualization 2



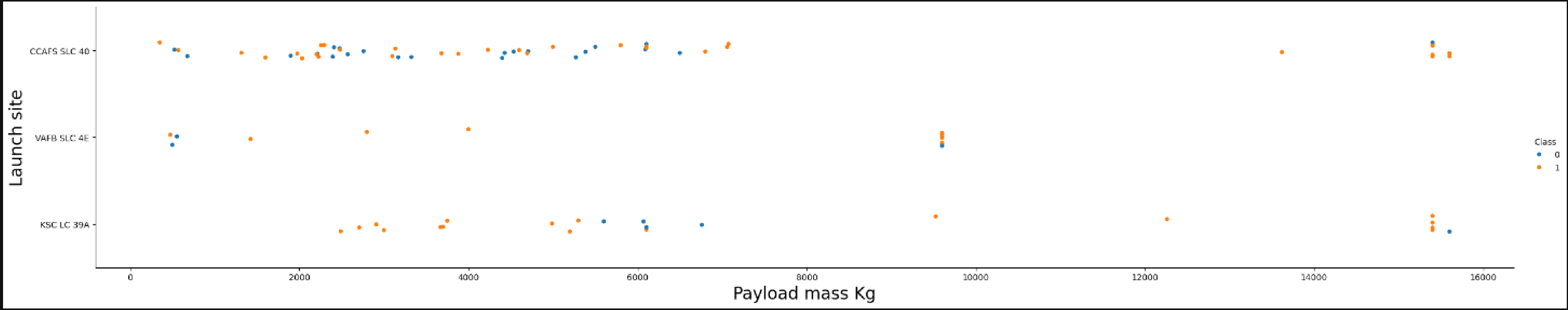
```
### TASK 1: Visualize the relationship between Flight Number and Launch Site
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Launch site",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



EDA with Data Visualization 3



```
### TASK 2: Visualize the relationship between Payload and Launch Site
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload mass Kg",fontsize=20)
plt.ylabel("Launch site",fontsize=20)
plt.show()
```



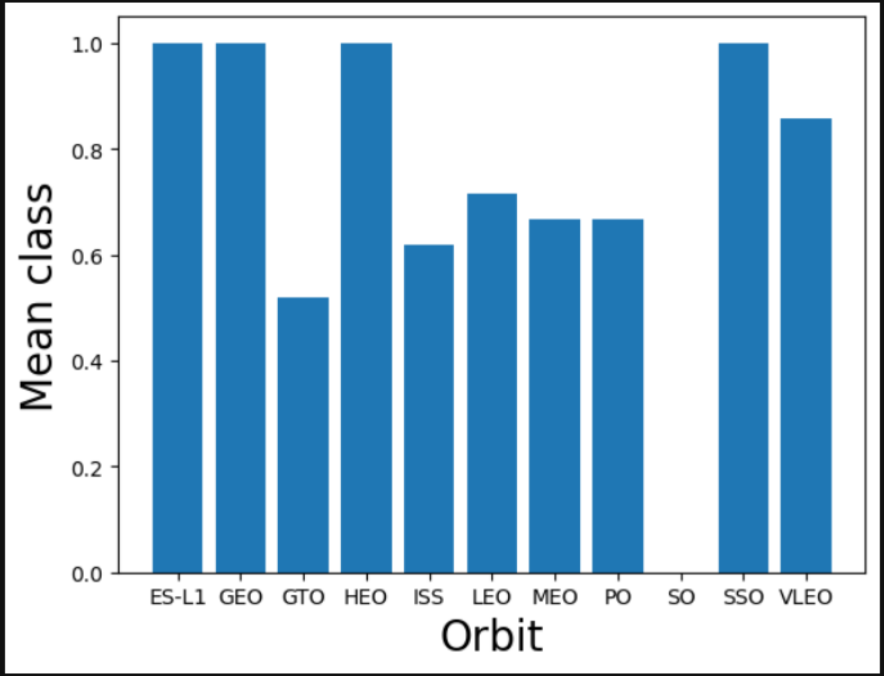
We also want to observe if there is any relationship between launch sites and their payload mass.

EDA with Data Visualization 4

```
### TASK  3: Visualize the relationship between success rate of each orbit type
tmp = df.groupby(['Orbit'])['Class'].mean().reset_index()
tmp
```

	Orbit	Class
0	ES-L1	1.000000
1	GEO	1.000000
2	GTO	0.518519
3	HEO	1.000000
4	ISS	0.619048
5	LEO	0.714286
6	MEO	0.666667
7	PO	0.666667
8	SO	0.000000
9	SSO	1.000000
10	VLEO	0.857143

```
plt.bar(tmp['Orbit'], tmp['Class'])
plt.xlabel("Orbit",fontsize=20)
plt.ylabel("Mean class",fontsize=20)
plt.show()
```



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

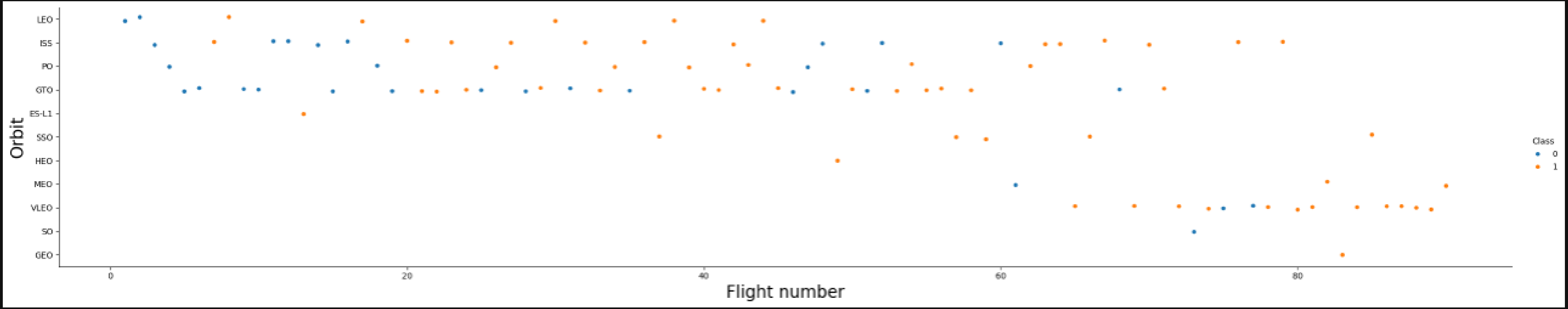
Let's create a `bar chart` for the sucess rate of each orbit

EDA with Data Visualization 5

```
### TASK  4: Visualize the relationship between FlightNumber and Orbit type
```

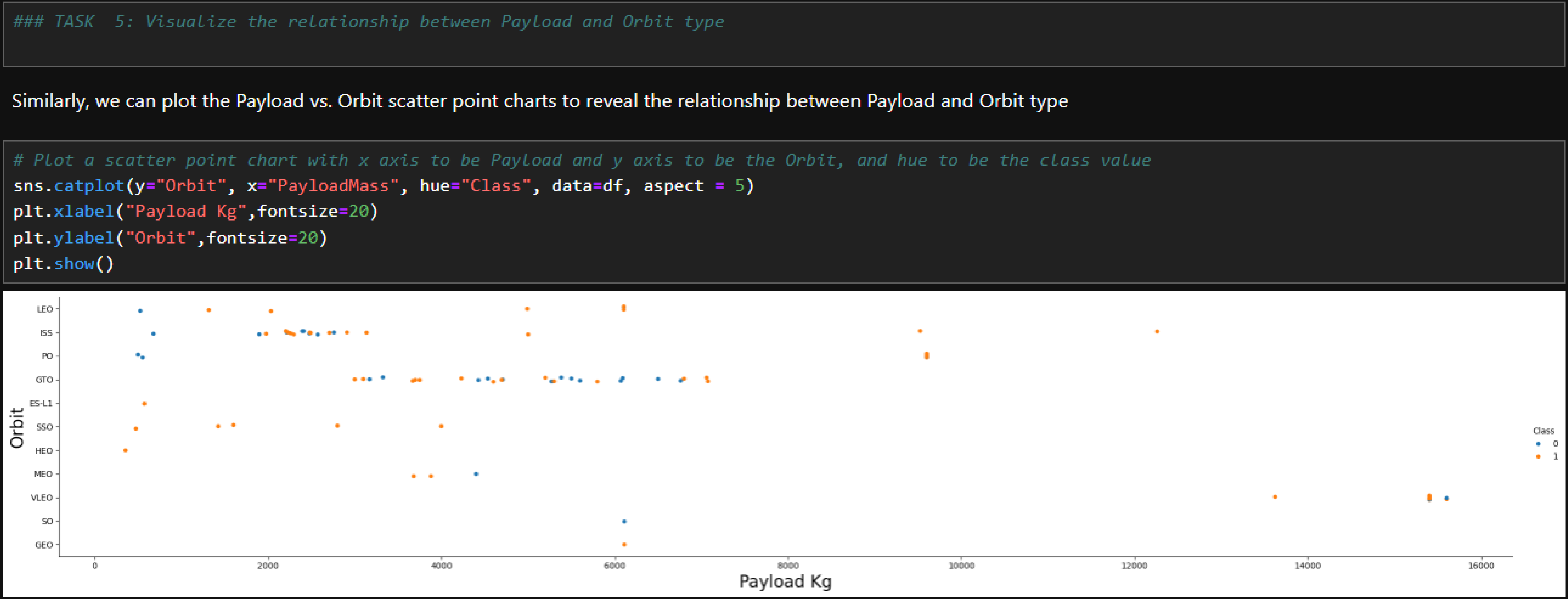
For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

```
# 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
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when

EDA with Data Visualization 6



```
### TASK  5: Visualize the relationship between Payload and Orbit type
```

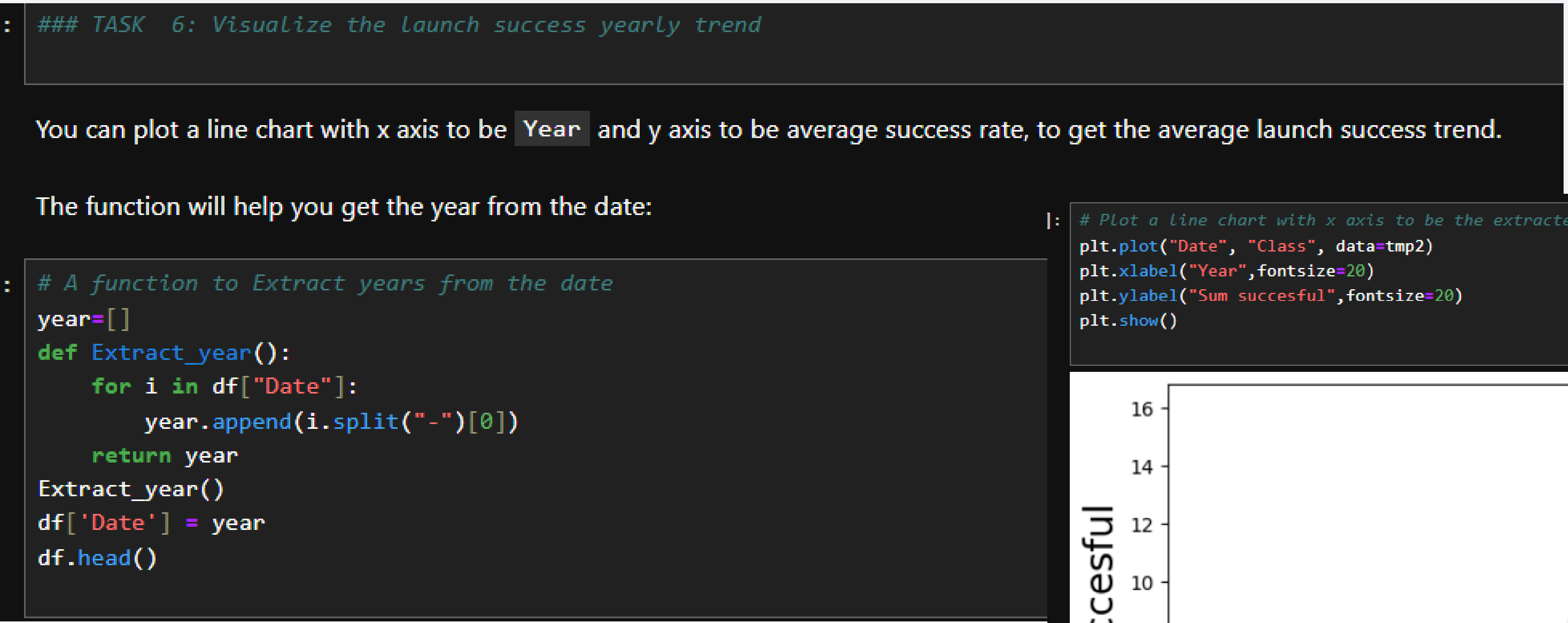
Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload Kg",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

EDA with Data Visualization 7

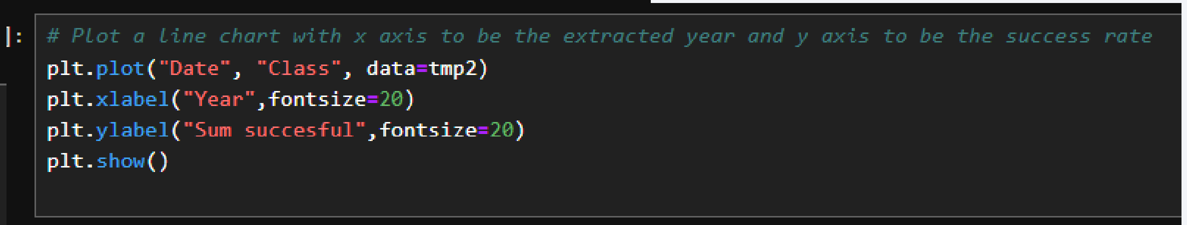


```
### TASK  6: Visualize the launch success yearly trend
```

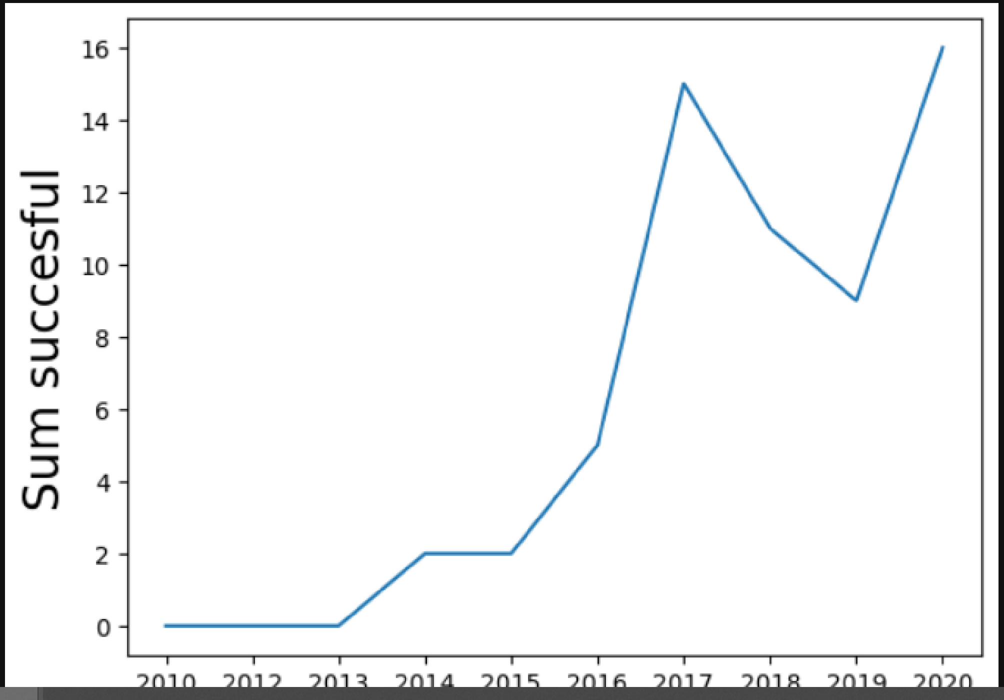
You can plot a line chart with x axis to be `Year` and y axis to be average success rate, to get the average launch success trend.

The function will help you get the year from the date:

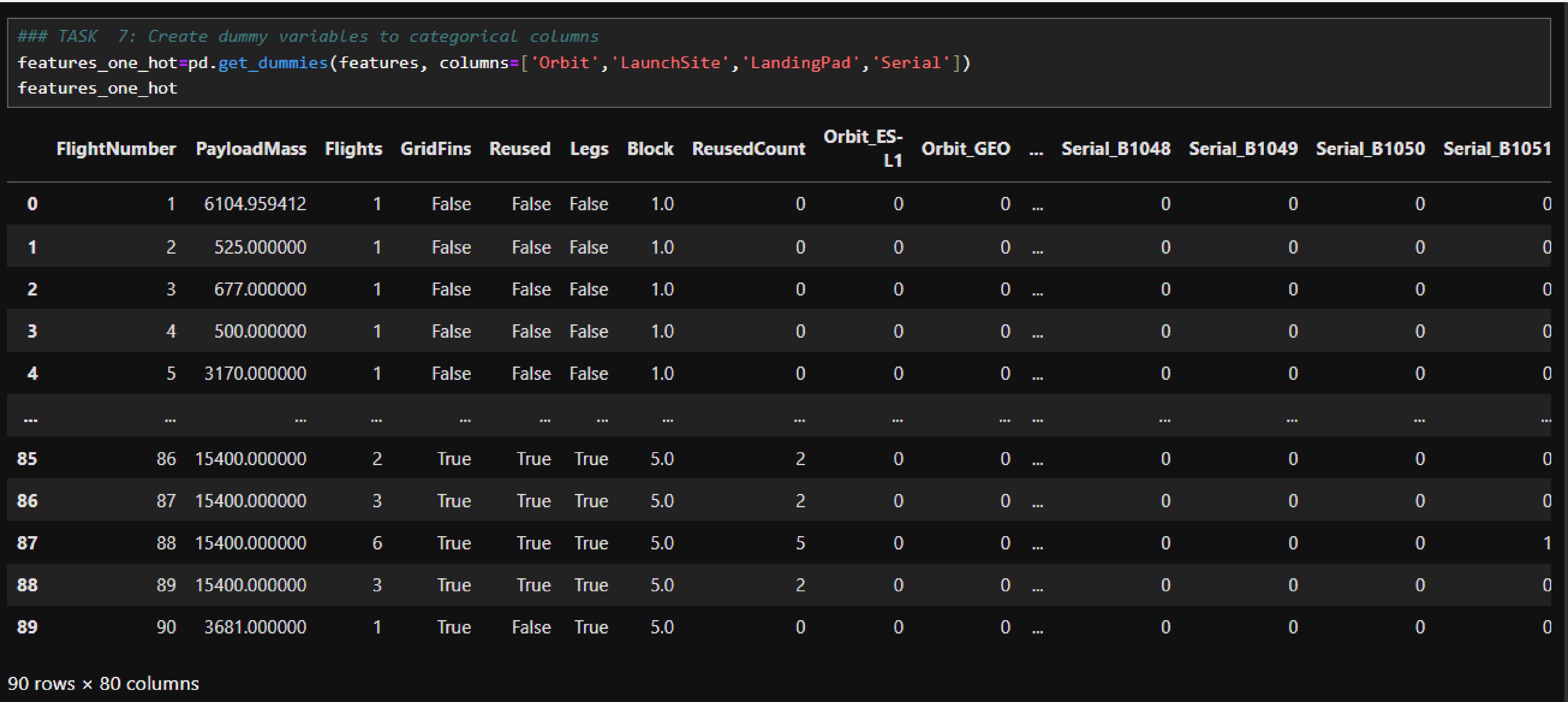
```
# A function to Extract years from the date
year=[]
def Extract_year():
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
Extract_year()
df['Date'] = year
df.head()
```



```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
plt.plot("Date", "Class", data=tmp2)
plt.xlabel("Year",fontsize=20)
plt.ylabel("Sum succesful",fontsize=20)
plt.show()
```



EDA with Data Visualization 8

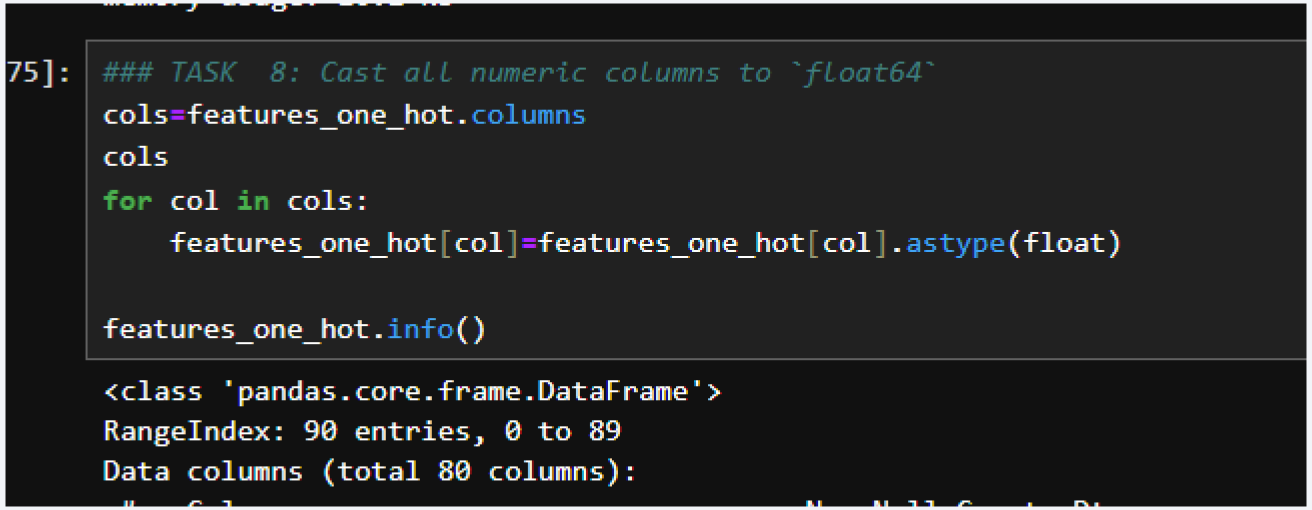


```
### TASK  7: Create dummy variables to categorical columns
features_one_hot=pd.get_dummies(features, columns=['Orbit','LaunchSite','LandingPad','Serial'])
features_one_hot
```

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial_B1049	Serial_B1050	Serial_B1051
0	1	6104.959412	1	False	False	False	1.0	0	0	0	...	0	0	0	0
1	2	525.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0
2	3	677.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0
3	4	500.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0
4	5	3170.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
85	86	15400.000000	2	True	True	True	5.0	2	0	0	...	0	0	0	0
86	87	15400.000000	3	True	True	True	5.0	2	0	0	...	0	0	0	0
87	88	15400.000000	6	True	True	True	5.0	5	0	0	...	0	0	0	1
88	89	15400.000000	3	True	True	True	5.0	2	0	0	...	0	0	0	0
89	90	3681.000000	1	True	False	True	5.0	0	0	0	...	0	0	0	0

90 rows × 80 columns

EDA with Data Visualization 9



```
75]: ### TASK  8: Cast all numeric columns to `float64`
     cols=features_one_hot.columns
     cols
     for col in cols:
         features_one_hot[col]=features_one_hot[col].astype(float)

     features_one_hot.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 90 entries, 0 to 89
     Data columns (total 80 columns):
```

EDA with SQL

- https://github.com/Matt-Cust/IBM-DS0720EN-Capstone-project/blob/main/L4-%20jupyter-labs-eda-sql-edx_sqllite%20(1).ipynb
- Task 1 - Display the names of the unique launch sites in the space mission
  - %sql select min(payload_mass__kg_) from SPACEXTBL;
  - %sql SELECT * from SPACEXTABLE LIMIT 20;
  - %sql SELECT avg(PAYLOAD_MASS__KG_) as Average_Payload_Mass from SPACEXTABLE;
  - %sql select count("Mission_Outcome") as MISSION_OUTCOME_COUNT,Launch_Site from SPACEXTBL group by "Launch_Site";
  - %sql SELECT distinct(LAUNCH_SITE) FROM SPACEXTABLE;
- Task 2 - Display 5 records where launch sites begin with the string 'KSC'
  - %sql SELECT * FROM SPACEXTABLE WHERE LAUNCH_SITE LIKE 'KSC%' LIMIT 5;

EDA with SQL

- Task 3 - Display the total payload mass carried by boosters launched by NASA (CRS)
  - %sql SELECT sum(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE "Customer" IS 'NASA (CRS)’;
- Task 4- Display average payload mass carried by booster version F9 v1.1
  - %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE "Booster_Version" IS 'F9 v1.1’;
- Task 5 - List the date where the succesful landing outcome in drone ship was achieved
  - %sql SELECT MIN("Date") FROM SPACEXTABLE WHERE "Landing_Outcome" IS 'Success (drone ship)’;
- Task 6 - List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
  - %sql SELECT ("Booster_Version") FROM SPACEXTABLE WHERE "Landing_Outcome" IS 'Success (ground pad)' AND "PAYLOAD_MASS__KG_" BETWEEN 4000 and 6000;

EDA with SQL

- Task 7 - List the total number of successful and failure mission outcomes
  - %sql SELECT "MISSION_OUTCOME", COUNT(*) as "total_number" FROM SPACEXTABLE GROUP BY "MISSION_OUTCOME";
- Task 8 – List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
  - %sql SELECT "BOOSTER_VERSION" FROM SPACEXTABLE WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL);
- Task 9 - List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
  - %sql SELECT substr("Date",6,2) as month, "Date","Booster_Version", "Launch_Site   ", "Landing_Outcome" FROM SPACEXTABLE where "Landing_Outcome" = 'Success (ground pad)' and substr("Date",0,5)='2017';
- 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("count_outcomes") FROM SPACEXTABLE WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY "Landing _Outcome" ORDER BY "count_outcomes" DESC;

Build an Interactive Map with Folium

- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map
  - Folium.Circle
  - Folium.Marker
  - MarkerCluster
  - Icons
  - Folium.PolyLine
- Explain why you added those objects
  - These were added because I was told to. But I think the point was to show the locations of the launch sites and counts of launches, and the count of Class's from each site, and then to show location of nearby useful amenities.
- Add the GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose
  - https://github.com/Matt-Cust/IBM-DS0720EN-Capstone-project/blob/main/L6-%20lab_jupyter_launch_site_location.jupyterlite.ipynb

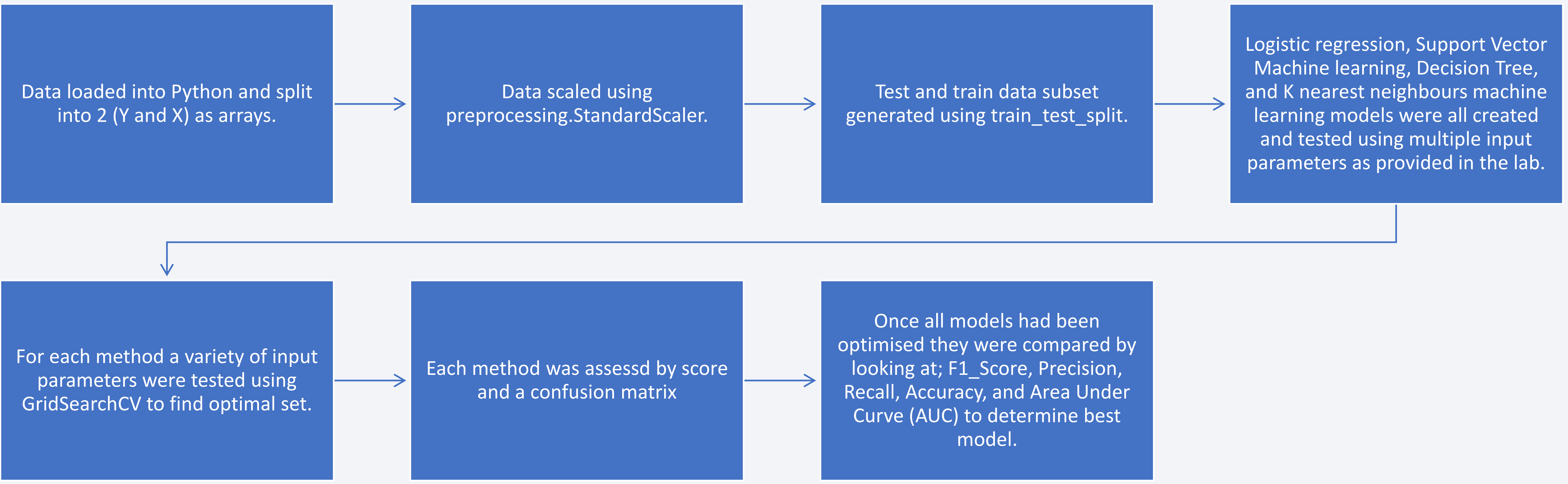
Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
  - A drop down menu to select launch site
  - A pie chart dependent upon the drop down showing Success count for each lunch site or total success/failure for the selected launch site.
  - A slider to select payload range
  - A scatter plot dependent upon the slider showing success or failure for different booster versions.
- Explain why you added those plots and interactions
  - Again because I was told to. However the answer you're hopefully looking for is to see which launch sites had the highest success Vs Failure rates, and to investigate how booster version and payload mass impacted success rate.
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose
  - https://github.com/Matt-Cust/IBM-DS0720EN-Capstone-project/blob/main/L7-%20SpaceXDash.ipynb

Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved, and found the best performing classification model
- You need present your model development process using key phrases and flowchart
  - Flow chart on the next slide.
- Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose
  - https://github.com/Matt-Cust/IBM-DS0720EN-Capstone-project/blob/main/L8-%20SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Predictive Analysis (Classification)



Data loaded into Python and split into 2 (Y and X) as arrays.

Data scaled using preprocessing.StandardScaler.

Test and train data subset generated using train_test_split.

Logistic regression, Support Vector Machine learning, Decision Tree, and K nearest neighbours machine learning models were all created and tested using multiple input parameters as provided in the lab.

For each method a variety of input parameters were tested using GridSearchCV to find optimal set.

Each method was assessd by score and a confusion matrix

Once all models had been optimised they were compared by looking at; F1_Score, Precision, Recall, Accuracy, and Area Under Curve (AUC) to determine best model.

Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

I'm assuming this is some sort of intro page not requiring population as then next few slides seem to wan the results as listed above.

Section 2

Insights drawn from EDA

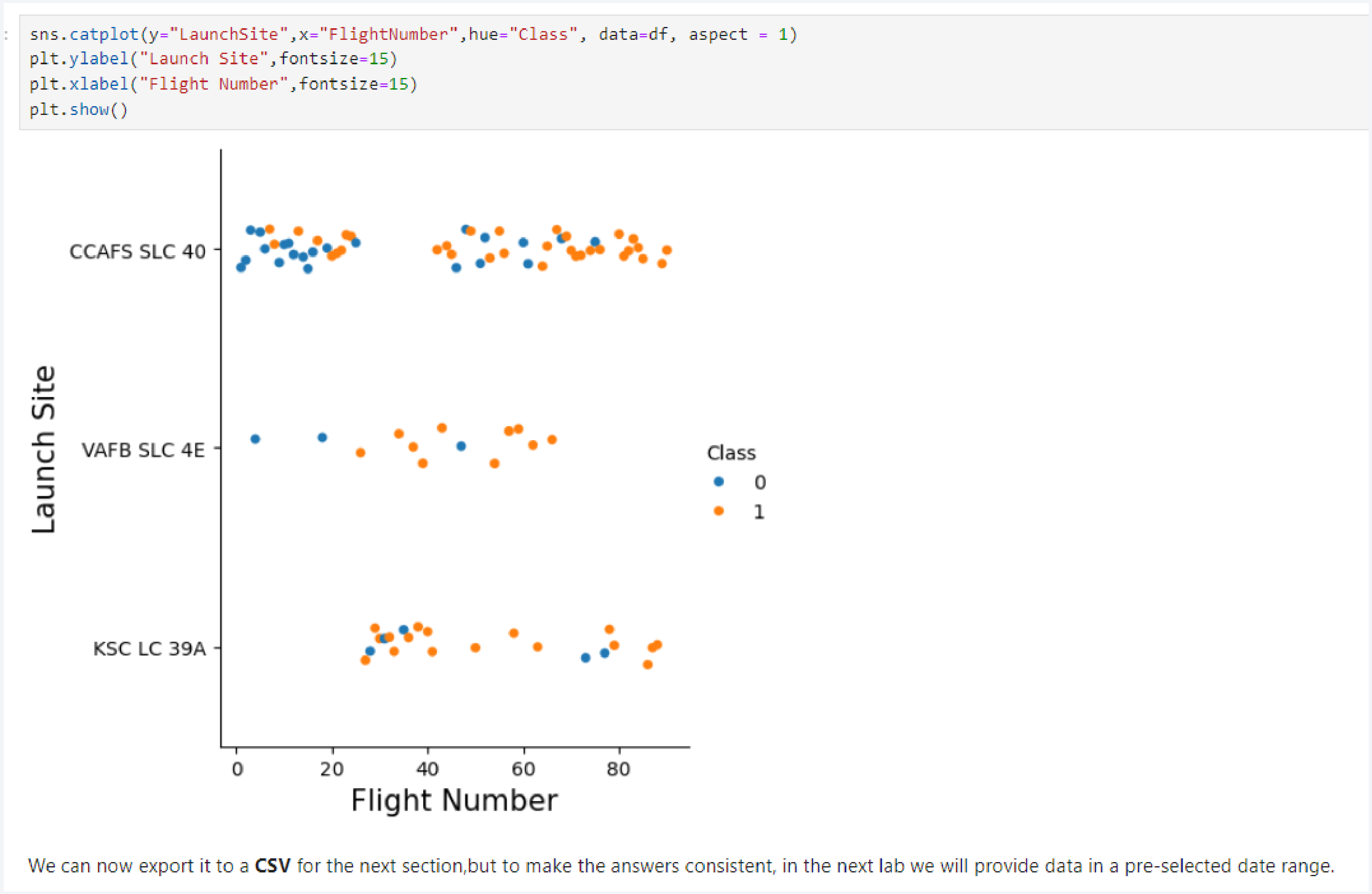
Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site

- Show the screenshot of the scatter plot with explanations

- Y axis is jittered launch site, and X axis is launch number. Blue point is class 0 (failed lading), and Orange point is Class 1 (Successful landing)

- In general higher success rate with higher flight number



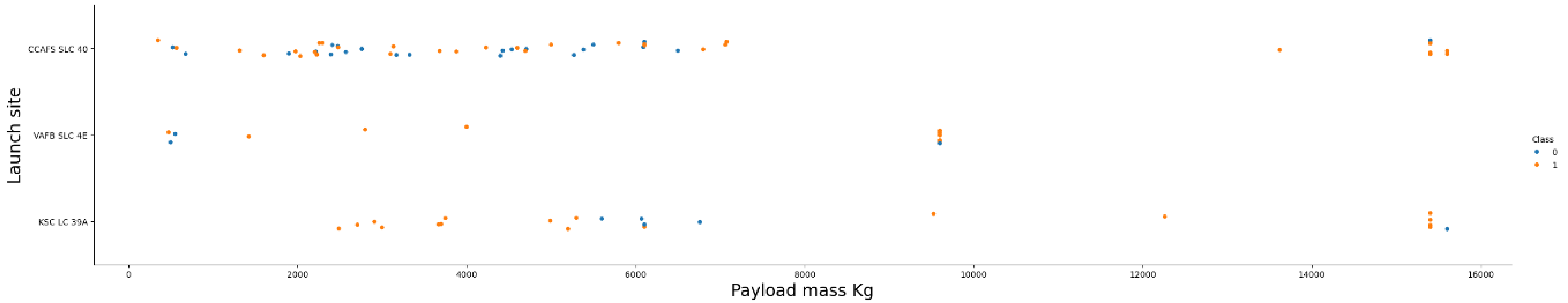
```
sns.catplot(y="LaunchSite",x="FlightNumber",hue="Class", data=df, aspect = 1)
plt.ylabel("Launch Site",fontsize=15)
plt.xlabel("Flight Number",fontsize=15)
plt.show()
```

We can now export it to a **CSV** for the next section,but to make the answers consistent, in the next lab we will provide data in a pre-selected date range.

Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site
- Show the screenshot of the scatter plot with explanations
- Y is jittered launch site, x is Payload mass, colour is Class, higher average success rate at higher mass, and no heavy launches from VAFB SLC 4E.

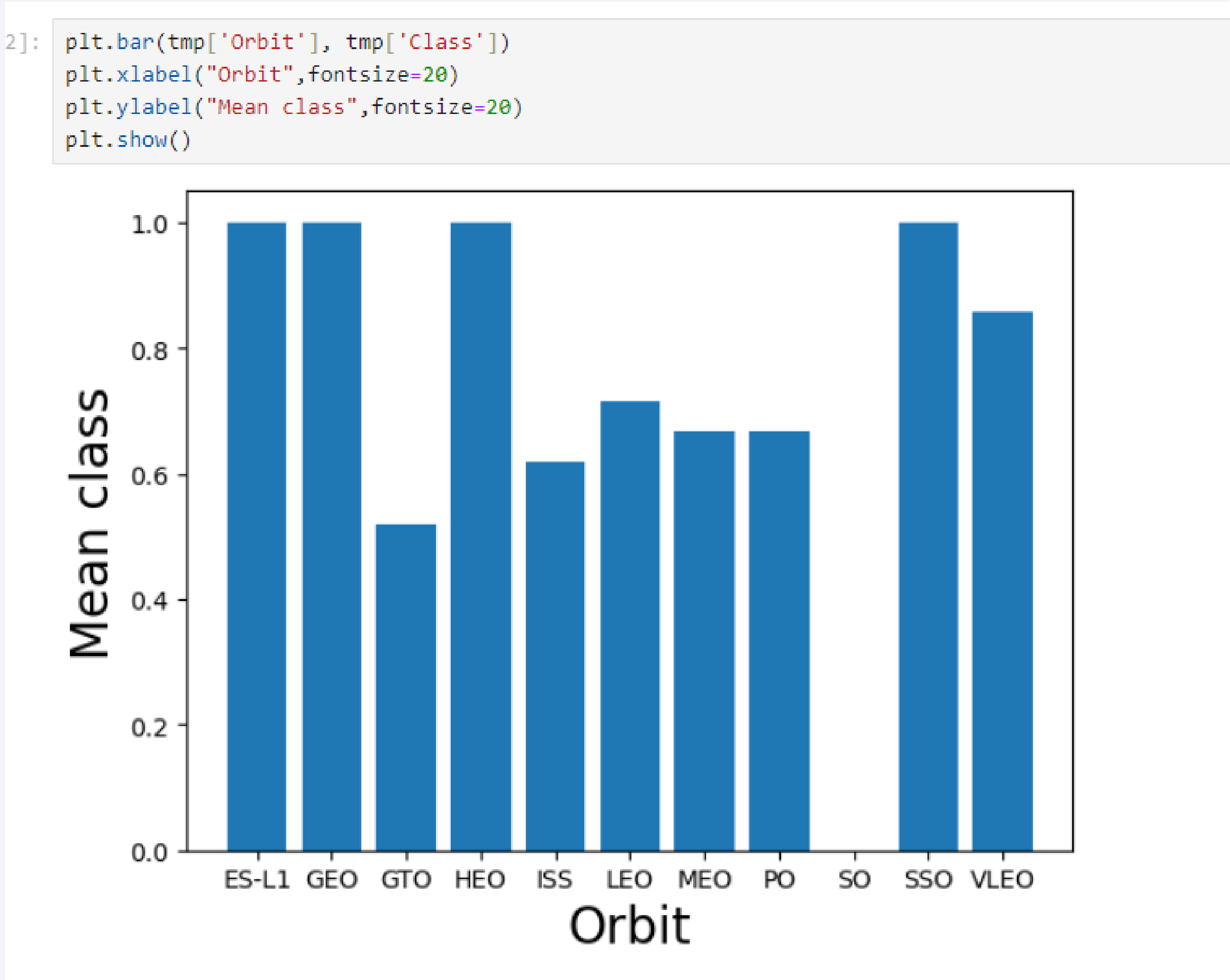
```
[8]: ### TASK 2: Visualize the relationship between Payload and Launch Site
     sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
     plt.xlabel("Payload mass Kg",fontsize=20)
     plt.ylabel("Launch site",fontsize=20)
     plt.show()
```



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

Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type
- Show the screenshot of the scatter plot with explanations
- X axis orbit type, Y axis averaged Class for each orbit type. ES-L1, GEO, HEO, SSO all 100% Calss 1 SO either no success or not attempted.

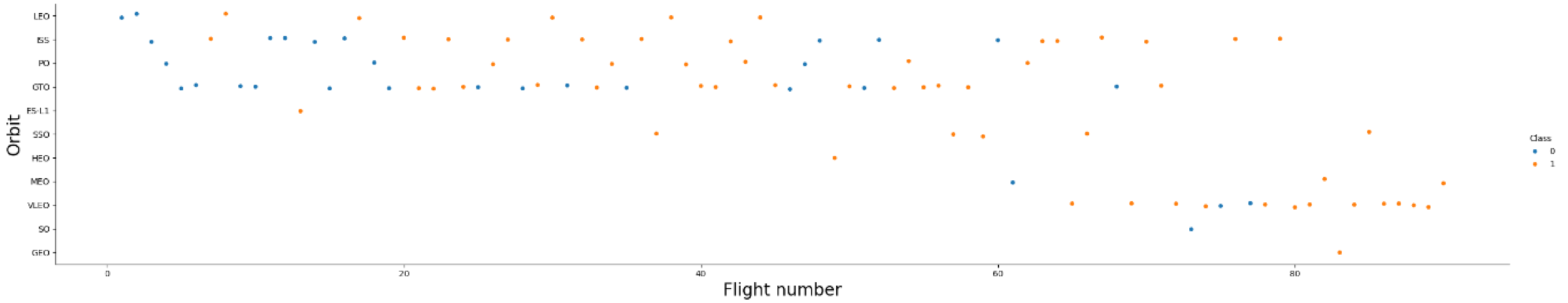


```
2]: plt.bar(tmp['Orbit'], tmp['Class'])
    plt.xlabel("Orbit",fontsize=20)
    plt.ylabel("Mean class",fontsize=20)
    plt.show()
```

Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type
- Show the screenshot of the scatter plot with explanations
- X is flight number, y is orbit type, colour is class

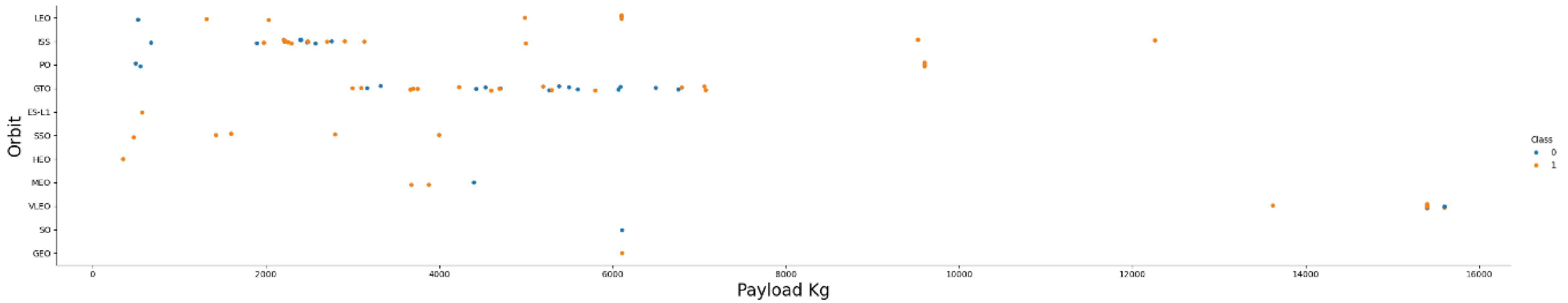
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type
- Show the screenshot of the scatter plot with explanations
- X payload mass Y Orbit type, colour Class. With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.

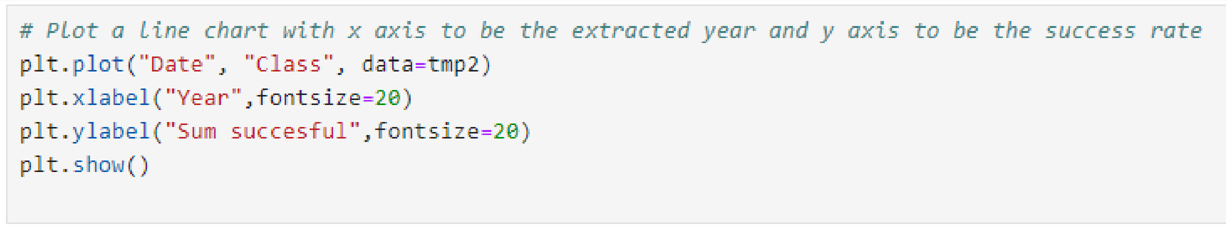
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload Kg",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



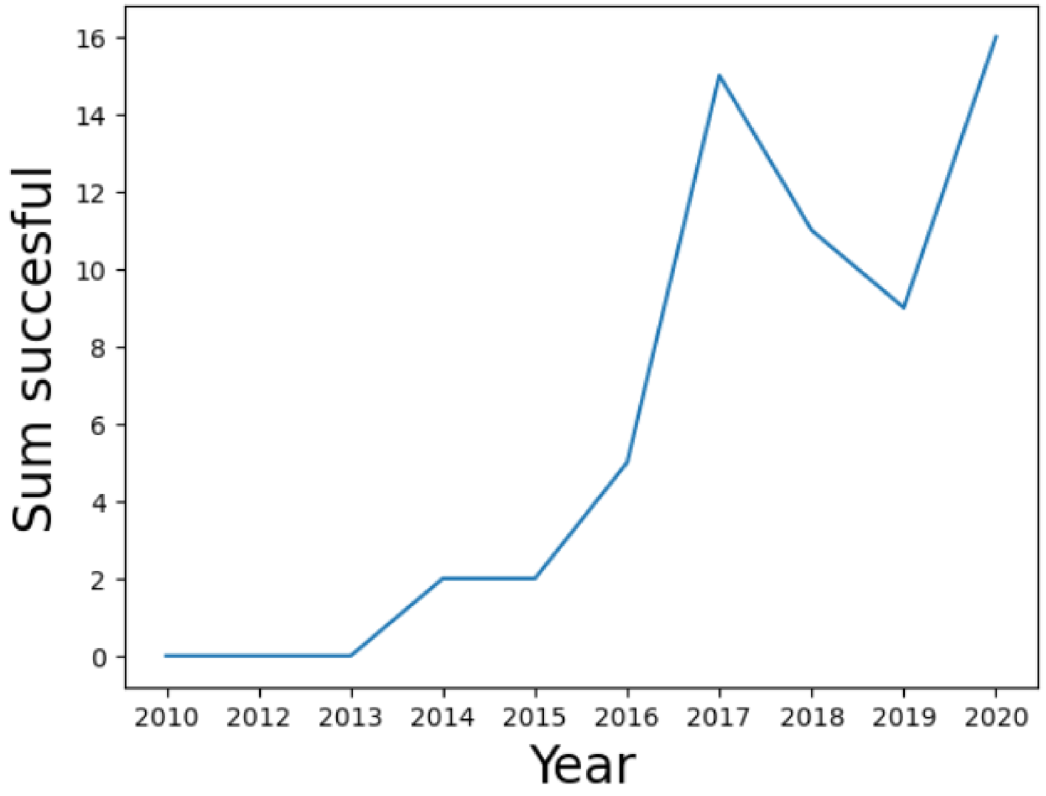
With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

Launch Success Yearly Trend

- Show a line chart of yearly average success rate
- Show the screenshot of the scatter plot with explanations
- X Years, Y Sum of successful Landings.
- Total success count increases with time, apart from a reduction in 2018/2019



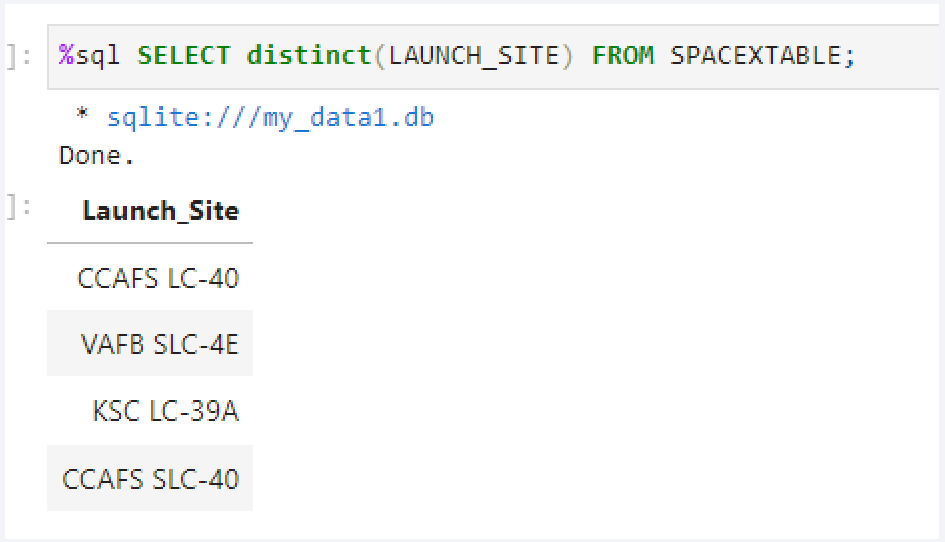
```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
plt.plot("Date", "Class", data=tmp2)
plt.xlabel("Year",fontsize=20)
plt.ylabel("Sum succesful",fontsize=20)
plt.show()
```



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

- Find the names of the unique launch sites
- Present your query result with a short explanation here
- Use distinct to get each unique value from LAUNCH_SITE variable in SPACEXTABLE.



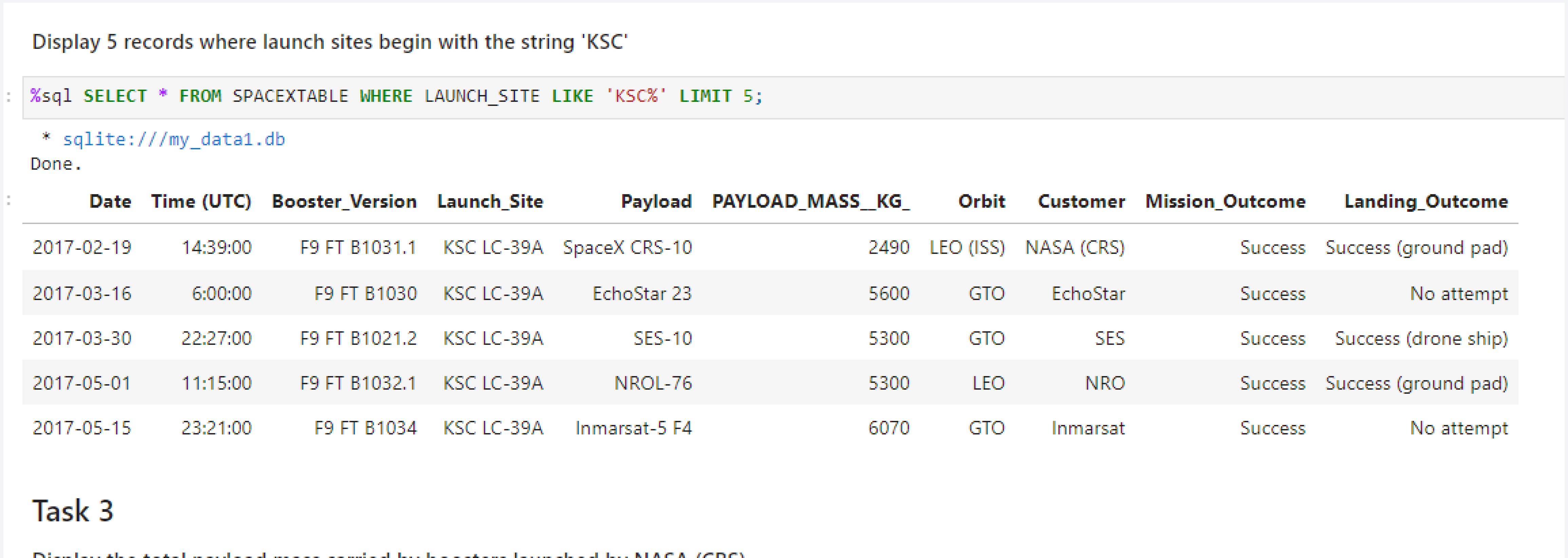
```
]: %sql SELECT distinct(LAUNCH_SITE) FROM SPACEXTABLE;

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

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

Launch Site Names Begin with 'KSC'

- Find 5 records where launch sites' names start with `KSC`
- Present your query result with a short explanation here
- Use WHERE and LIKE to find entries where Launsite contains "KSC", use LIMIT to limit the number of results to 5.



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

```
%sql SELECT * FROM SPACEXTABLE WHERE LAUNCH_SITE LIKE 'KSC%' LIMIT 5;
```

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

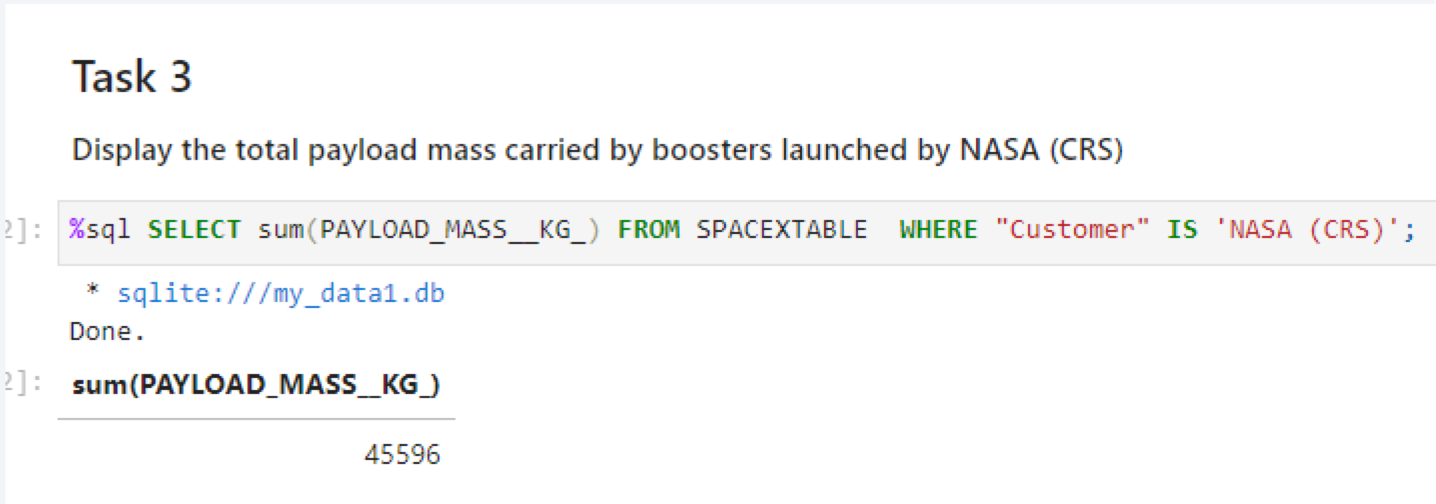
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-03-16	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	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 3

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

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Present your query result with a short explanation here
- Use WHERE and IS to limit Customer to be NASA and call SUM on PAYLOADMASS



Task 3

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

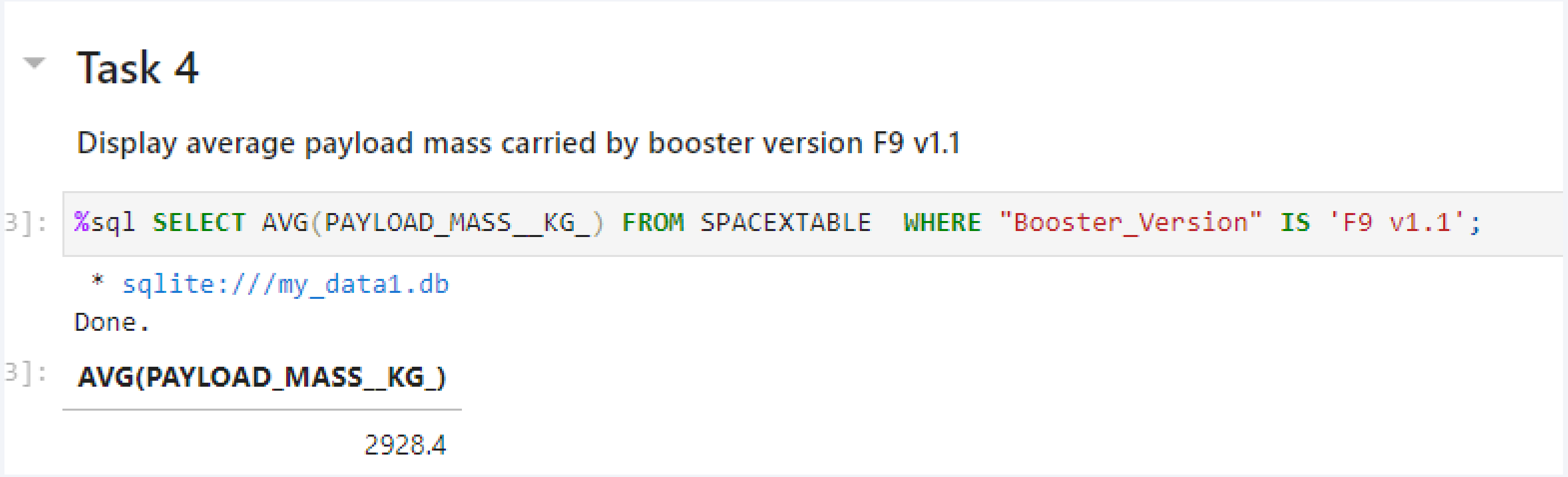
```
2]: %sql SELECT sum(PAYLOAD_MASS__KG_) FROM SPACEXTABLE  WHERE "Customer" IS 'NASA (CRS)';
```

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

```
2]: sum(PAYLOAD_MASS__KG_)
                    45596
```

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Present your query result with a short explanation here
- As before se WHERE and IS to select specific booster version and call AVG on PAYLOADMASS.



Task 4

Display average payload mass carried by booster version F9 v1.1

```
3]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE  WHERE "Booster_Version" IS 'F9 v1.1';
```

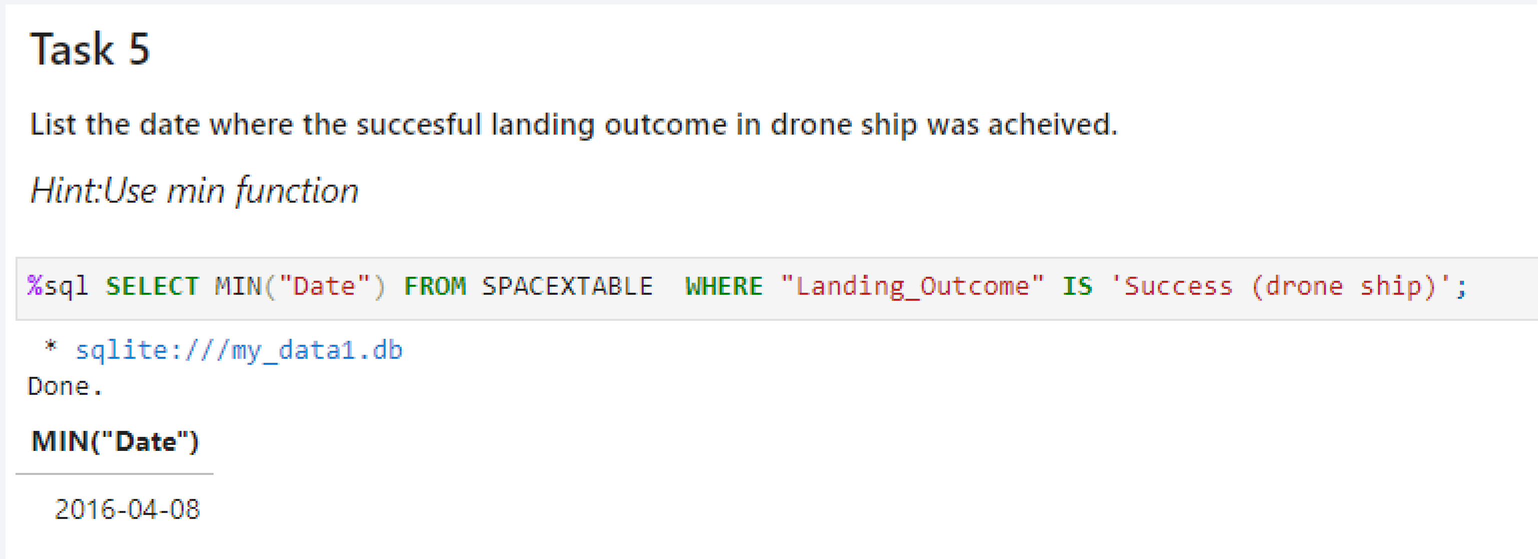
```
 * sqlite:///my_data1.db
Done.
```

3]:

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on drone ship.Present your query result with a short explanation here
- Again WHERE and IS this time on Success., the call MIN on DATE to get first event.



Task 5

List the date where the succesful landing outcome in drone ship was acheived.

Hint:Use min function

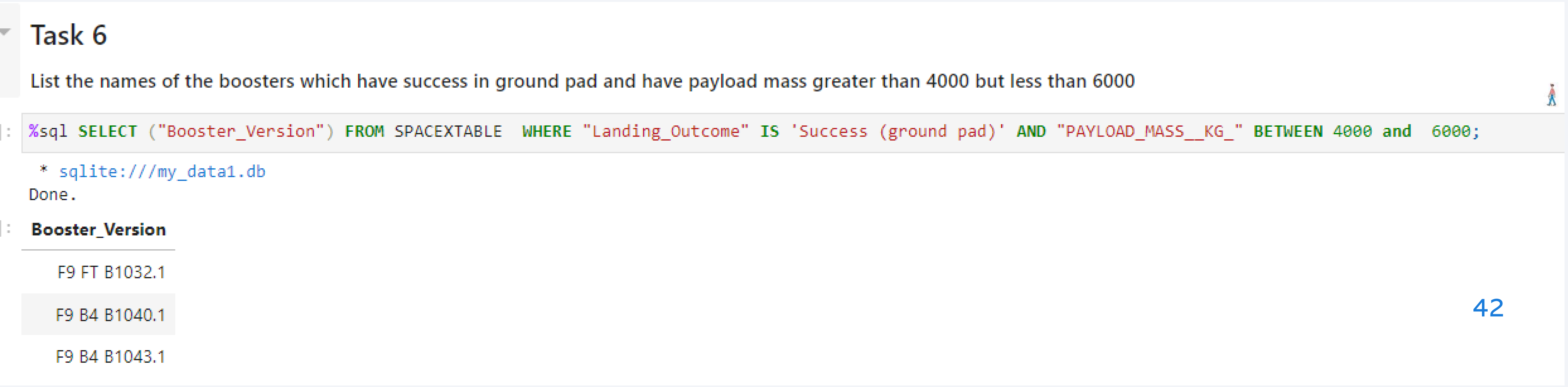
```
%sql SELECT MIN("Date") FROM SPACEXTABLE  WHERE "Landing_Outcome" IS 'Success (drone ship)';
```

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

MIN("Date")
2016-04-08

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000
- Present your query result with a short explanation here
- WHERE and IS again, but this time coupled with AND to add a second criteria and BETWEEN to set payload mass range.



Task 6

List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000

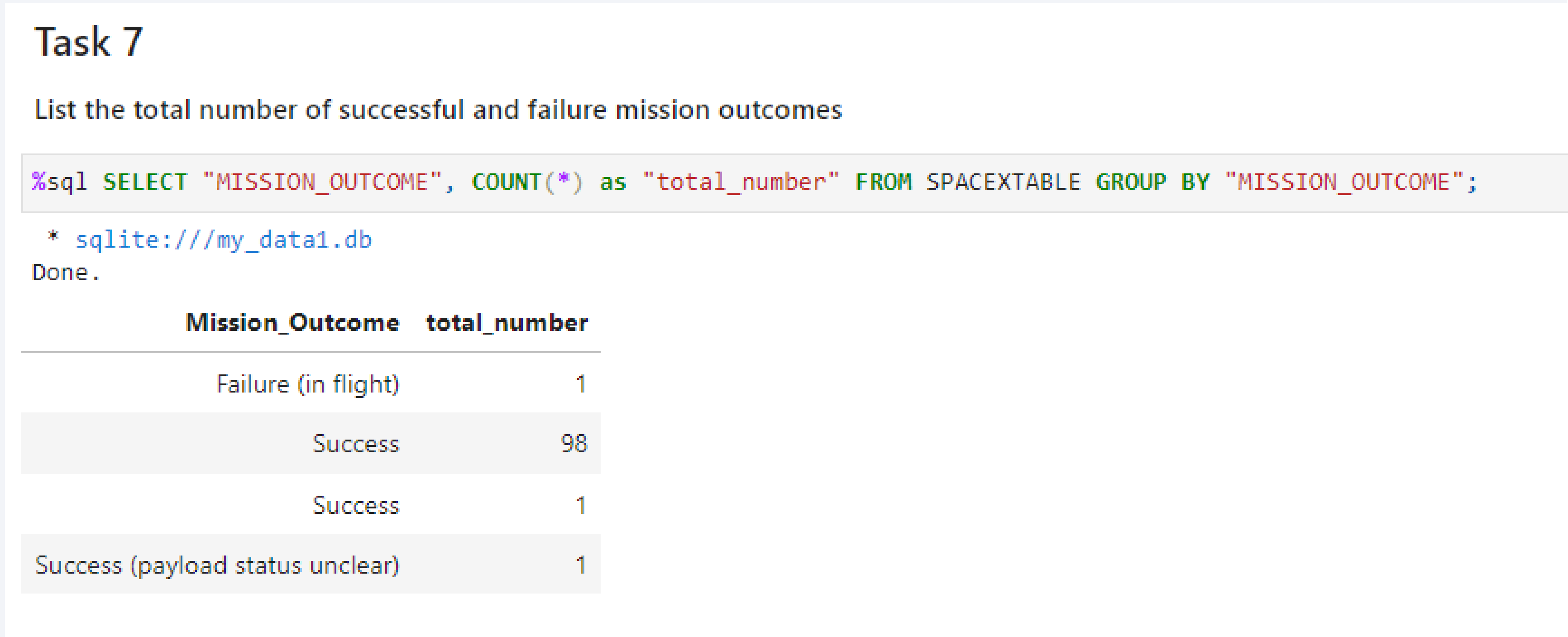
```
%sql SELECT ("Booster_Version") FROM SPACEXTABLE  WHERE "Landing_Outcome" IS 'Success (ground pad)' AND "PAYLOAD_MASS__KG_" BETWEEN 4000 and  6000;
```

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

Booster_Version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Present your query result with a short explanation here
- Use GROUPBY on MISSION OUTCOME and apply COUNT.



Task 7

List the total number of successful and failure mission outcomes

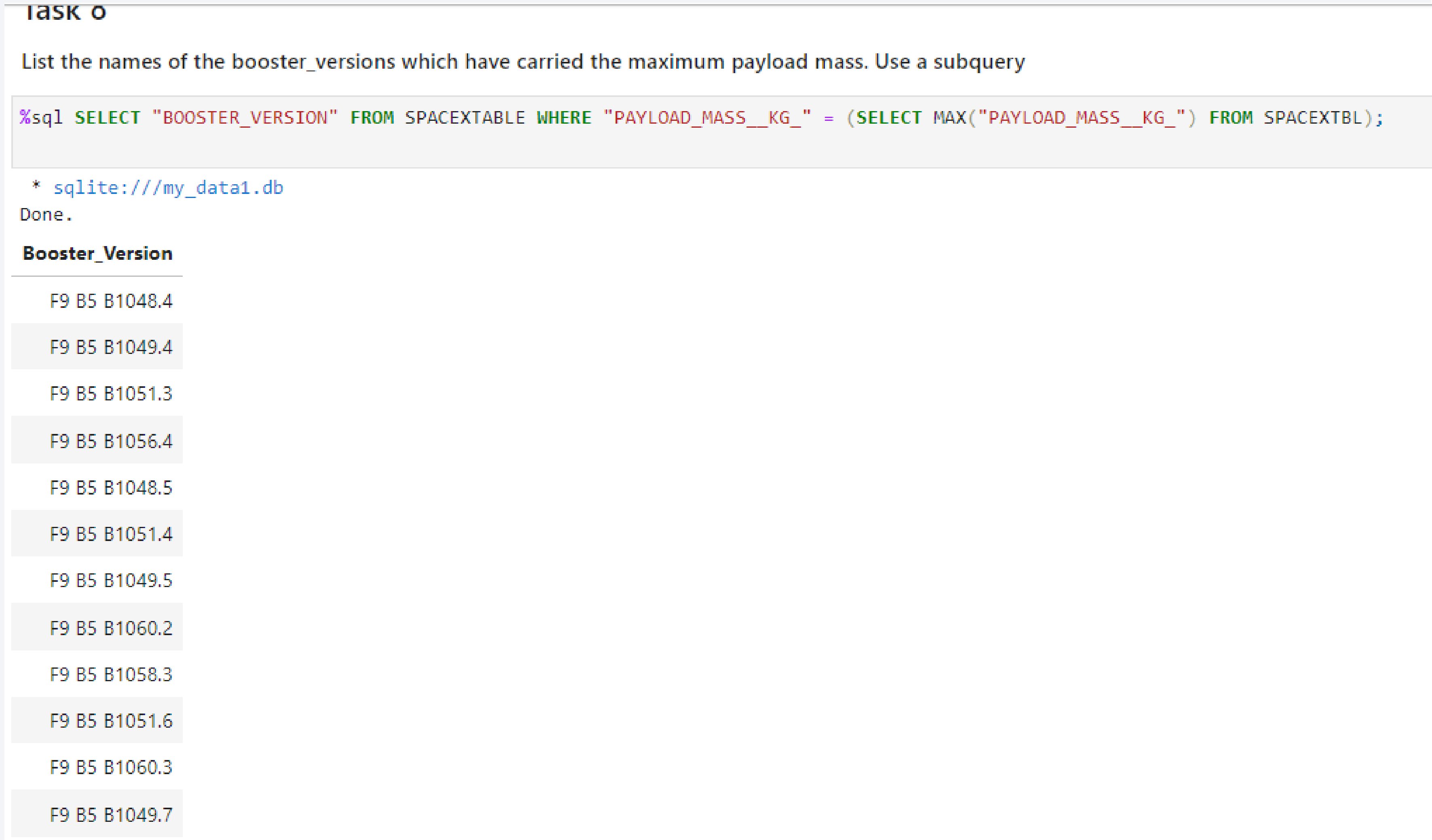
```
%sql SELECT "MISSION_OUTCOME", COUNT(*) as "total_number" FROM SPACEXTABLE GROUP BY "MISSION_OUTCOME";
```

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

Mission_Outcome	total_number
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Present your query result with a short explanation here
- Subquery to determine max payload mass, then apply into a WHERE call. Use this to SELECT BOOSTER VERSION.



Task 6

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

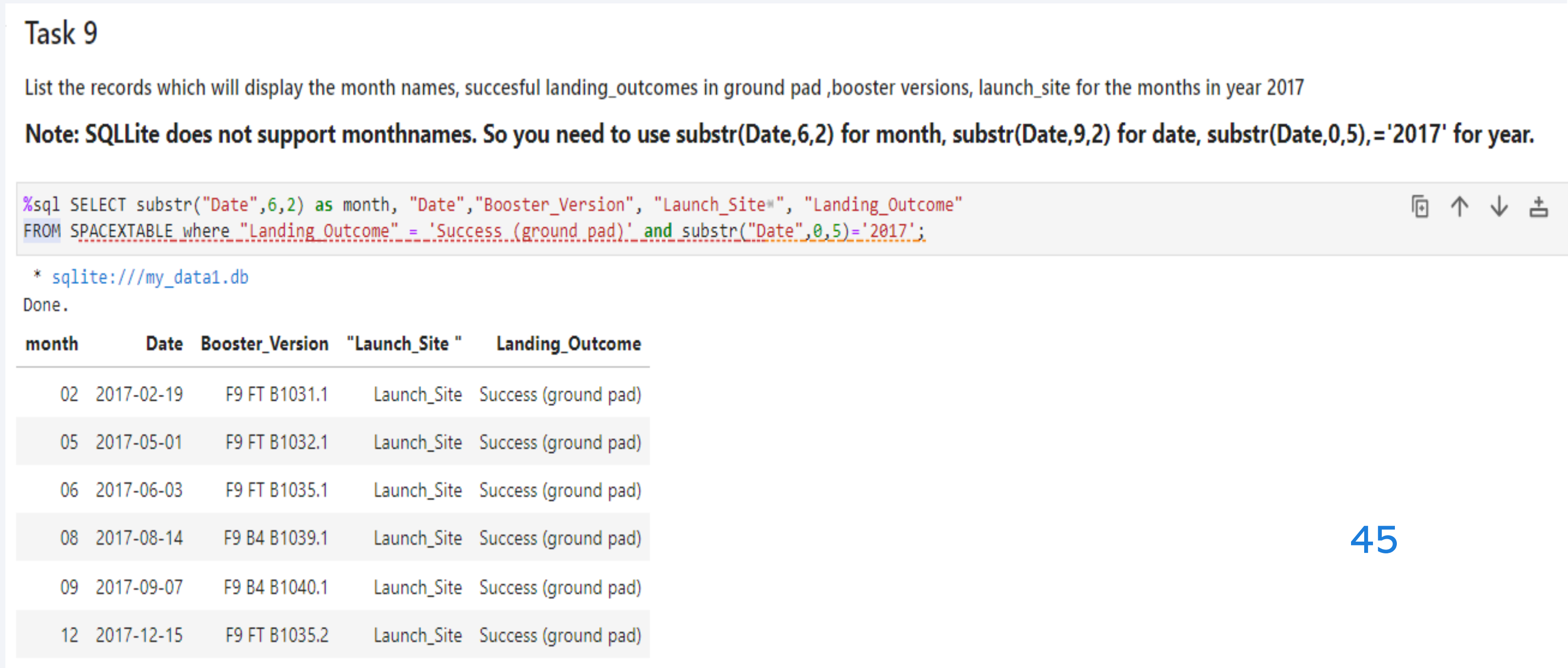
```
%sql SELECT "BOOSTER_VERSION" FROM SPACEXTABLE WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL);
```

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

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

- List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Present your query result with a short explanation here
- WHERE LANDING is a success and the first 4 chars in DATE ==2017 create a new column called month which are the 6th and 7th chars in date.
- SELECT to display:
    - DATE
    - GROUNDPAD
    - LAUNCH SITE
    - LANDING OUTCOME



Task 9

List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

Note: SQLLite does not support monthnames. So you need to use substr(Date,6,2) for month, substr(Date,9,2) for date, substr(Date,0,5),='2017' for year.

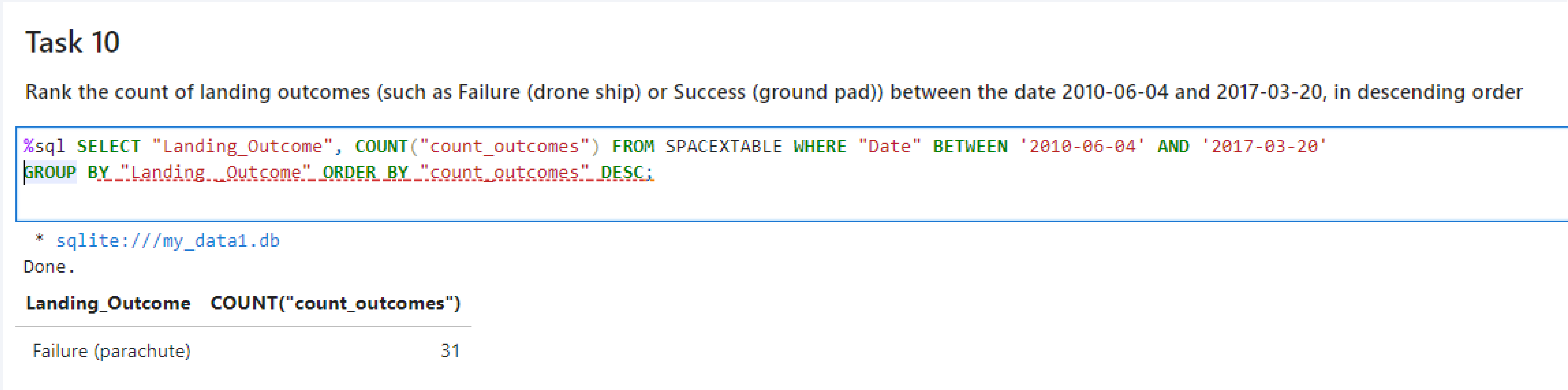
```
%sql SELECT substr("Date",6,2) as month, "Date","Booster_Version", "Launch_Site ", "Landing_Outcome"
FROM SPACEXTABLE where "Landing_Outcome" = 'Success (ground pad)' and substr("Date",0,5)='2017';
```

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

month	Date	Booster_Version	"Launch_Site "	Landing_Outcome
02	2017-02-19	F9 FT B1031.1	Launch_Site	Success (ground pad)
05	2017-05-01	F9 FT B1032.1	Launch_Site	Success (ground pad)
06	2017-06-03	F9 FT B1035.1	Launch_Site	Success (ground pad)
08	2017-08-14	F9 B4 B1039.1	Launch_Site	Success (ground pad)
09	2017-09-07	F9 B4 B1040.1	Launch_Site	Success (ground pad)
12	2017-12-15	F9 FT B1035.2	Launch_Site	Success (ground pad)

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- 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
- Present your query result with a short explanation here
- Use WHERE BETWEEN and AND to set criteria for selection, and then GROUPBY LANDINGOUTCOME, COUNT the outcomes and store in new column "Count_outcomes", which should be ordered in DESC (Descending) order.
- I couldn't really get this one to work.



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("count_outcomes") FROM SPACEXTABLE WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing _Outcome" ORDER BY "count_outcomes" DESC;
```

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

Landing_Outcome	COUNT("count_outcomes")
Failure (parachute)	31

Section 3

Launch Sites Proximities Analysis

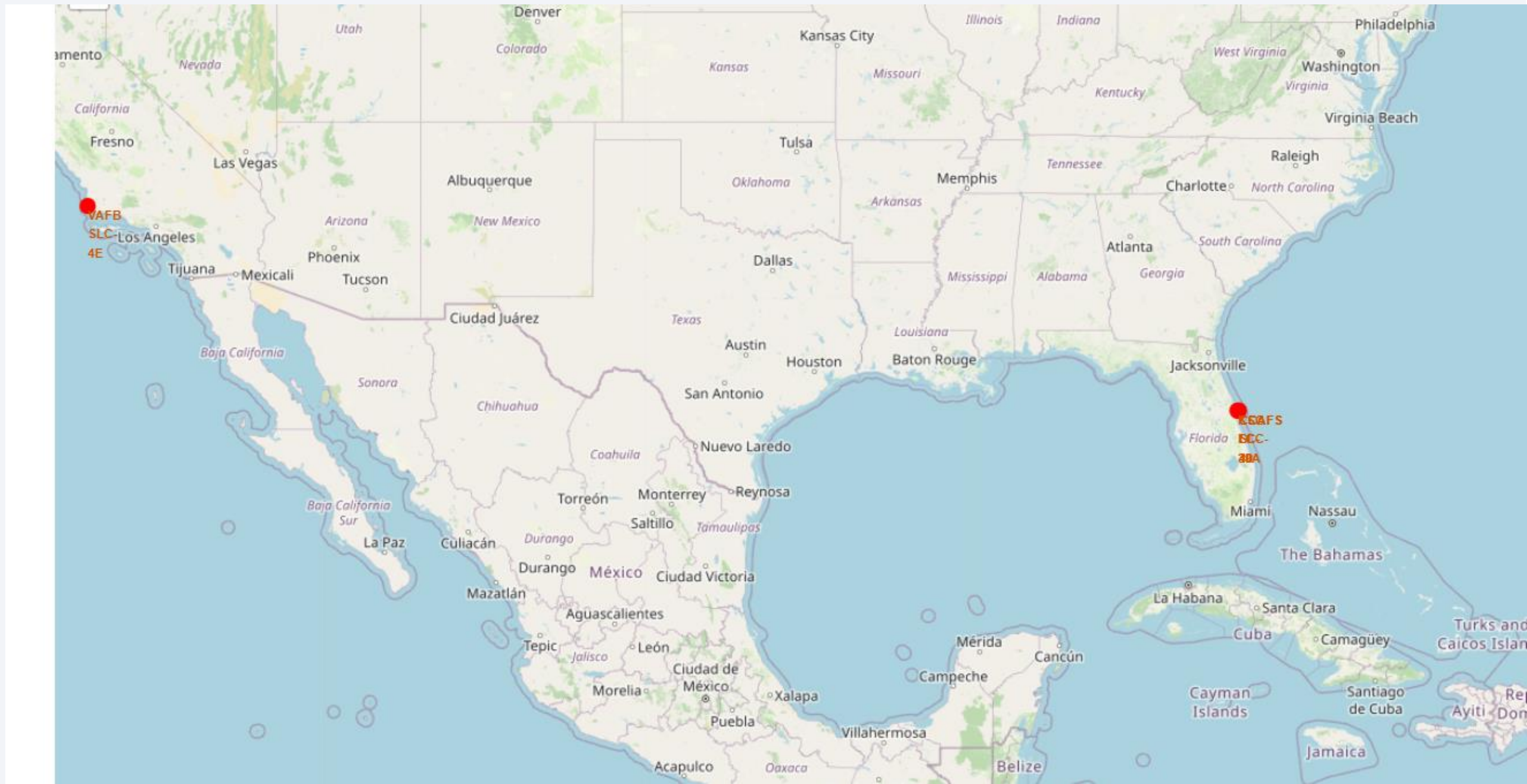
<Explore the generated folium map and make a proper screenshot to include all launch sites' location markers on a global map >

- Replace <Folium map screenshot 1> title with an appropriate title

- Explore the generated folium map and make a proper screenshot to include all launch sites' location markers on a global map

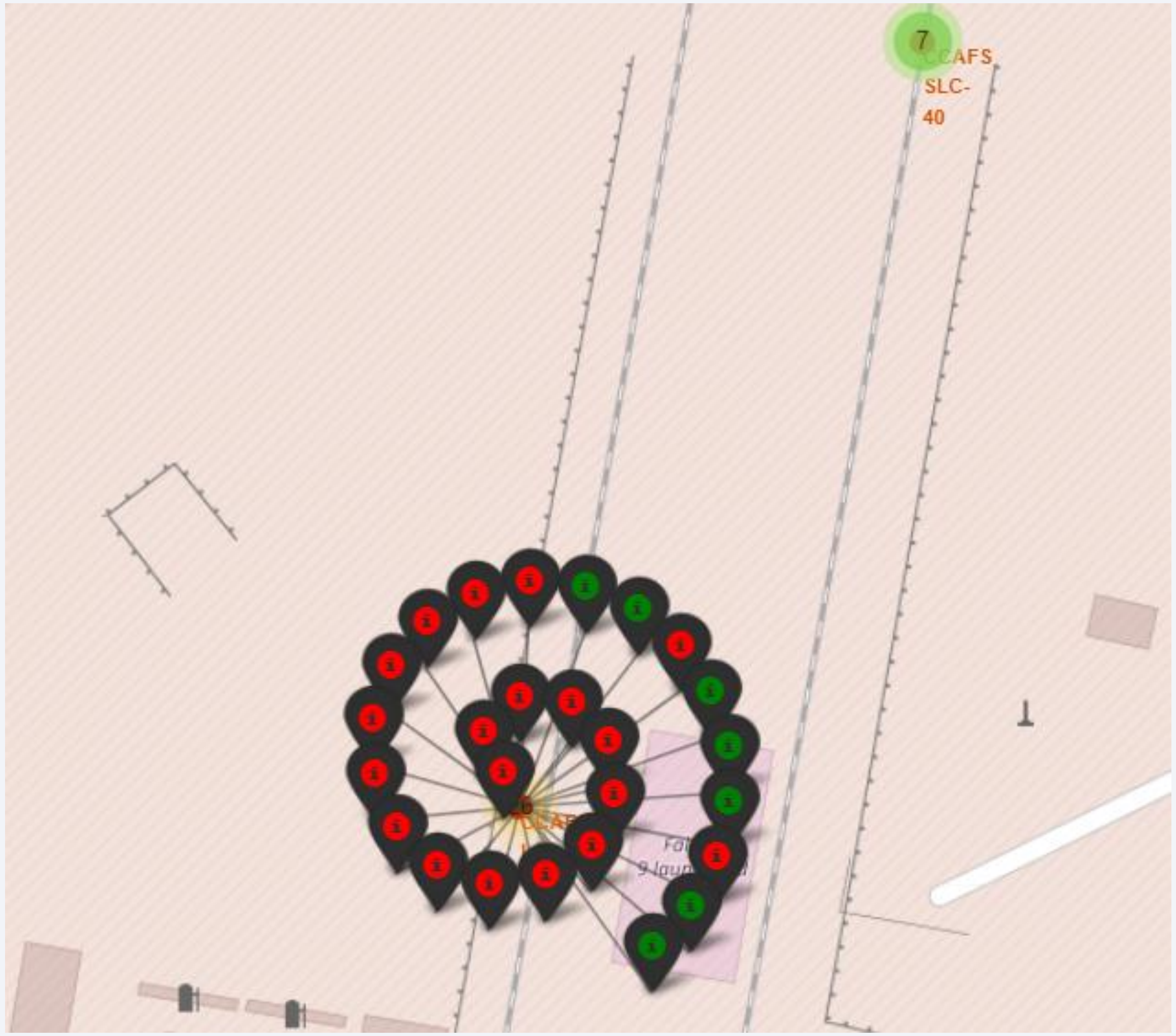
- Explain the important elements and findings on the screenshot

- I added the red dots and text to show the launch sites, the rest of the map doesn't matter.



Explore the folium map and make a proper screenshot to show the color-labeled launch outcomes on the map

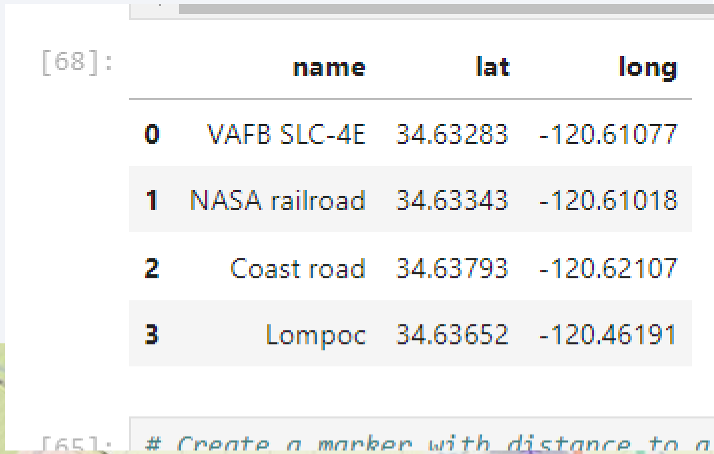
- Replace <Folium map screenshot 2> title with an appropriate title
- Explore the folium map and make a proper screenshot to show the color-labeled launch outcomes on the map
- Explain the important elements and findings on the screenshot
- Each icon represents an invidual launch the colour represent success or failure in booster return, the green 7 is another launch site.



Explore the generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed

- Sorry had to do multiple zooms to show all POIs. Left graph is apparently a NASA train track, middle is a road/highway, and right is to the nearest city.
- Sorry I couldn't get the distances to show on the graph, however I

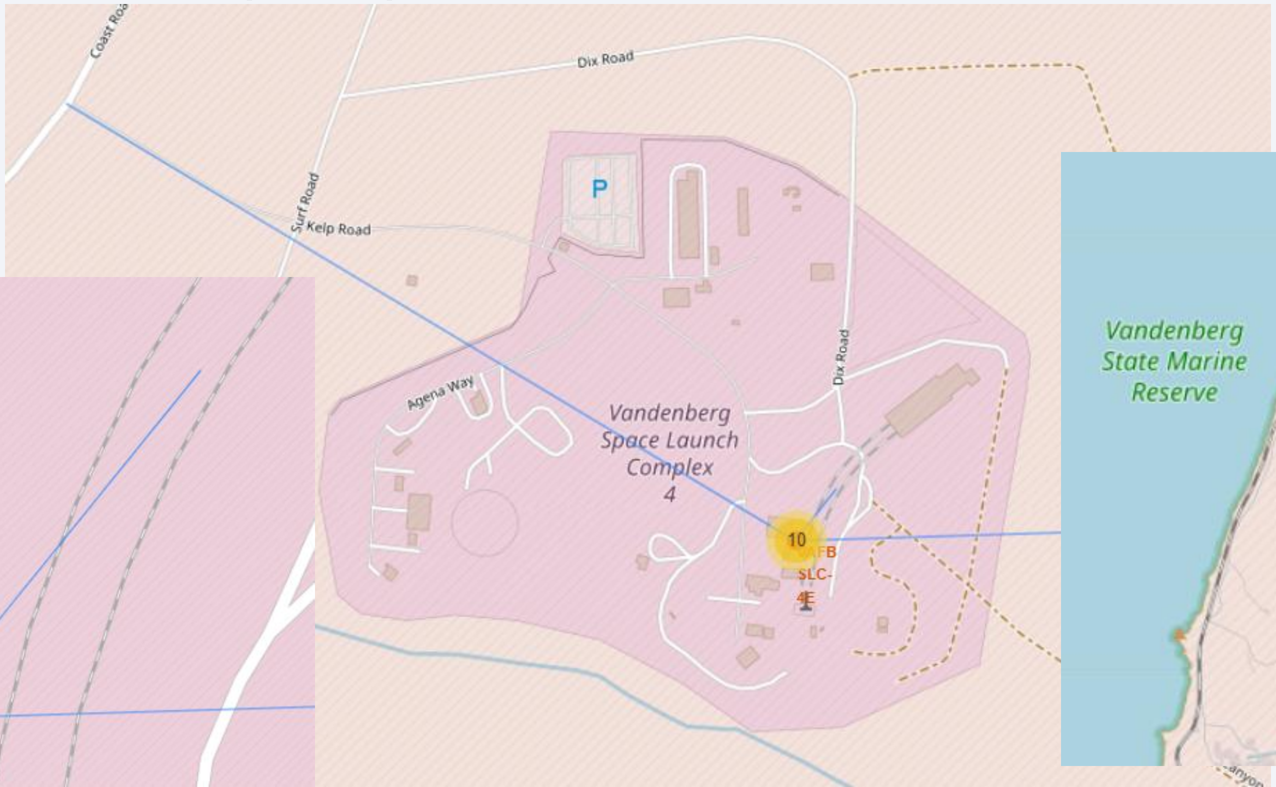
Did calculate them (Table)

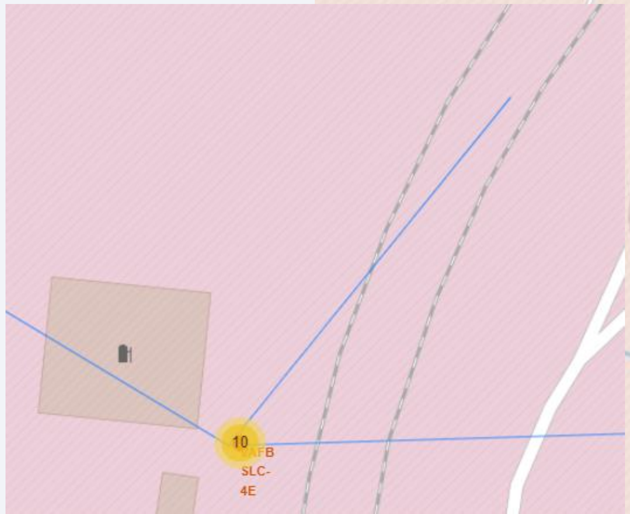


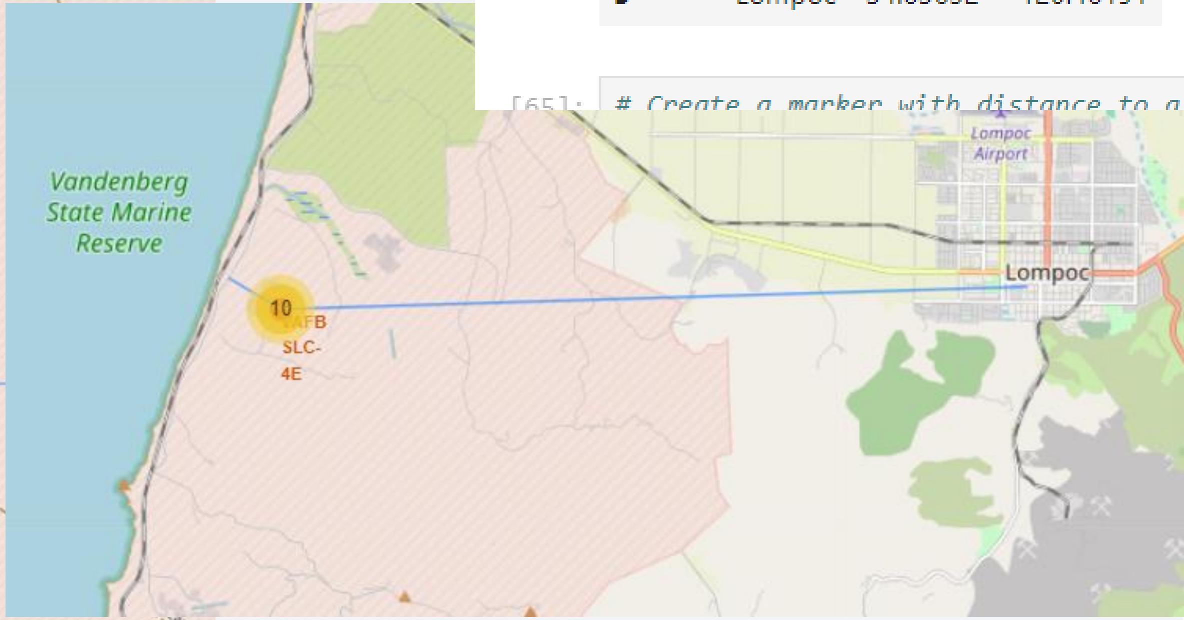
[68]:

	name	lat	long
0	VAFB SLC-4E	34.63283	-120.61077
1	NASA railroad	34.63343	-120.61018
2	Coast road	34.63793	-120.62107
3	Lompoc	34.63652	-120.46191

```
[65]: # Create a marker with distance to a
```







Section 4

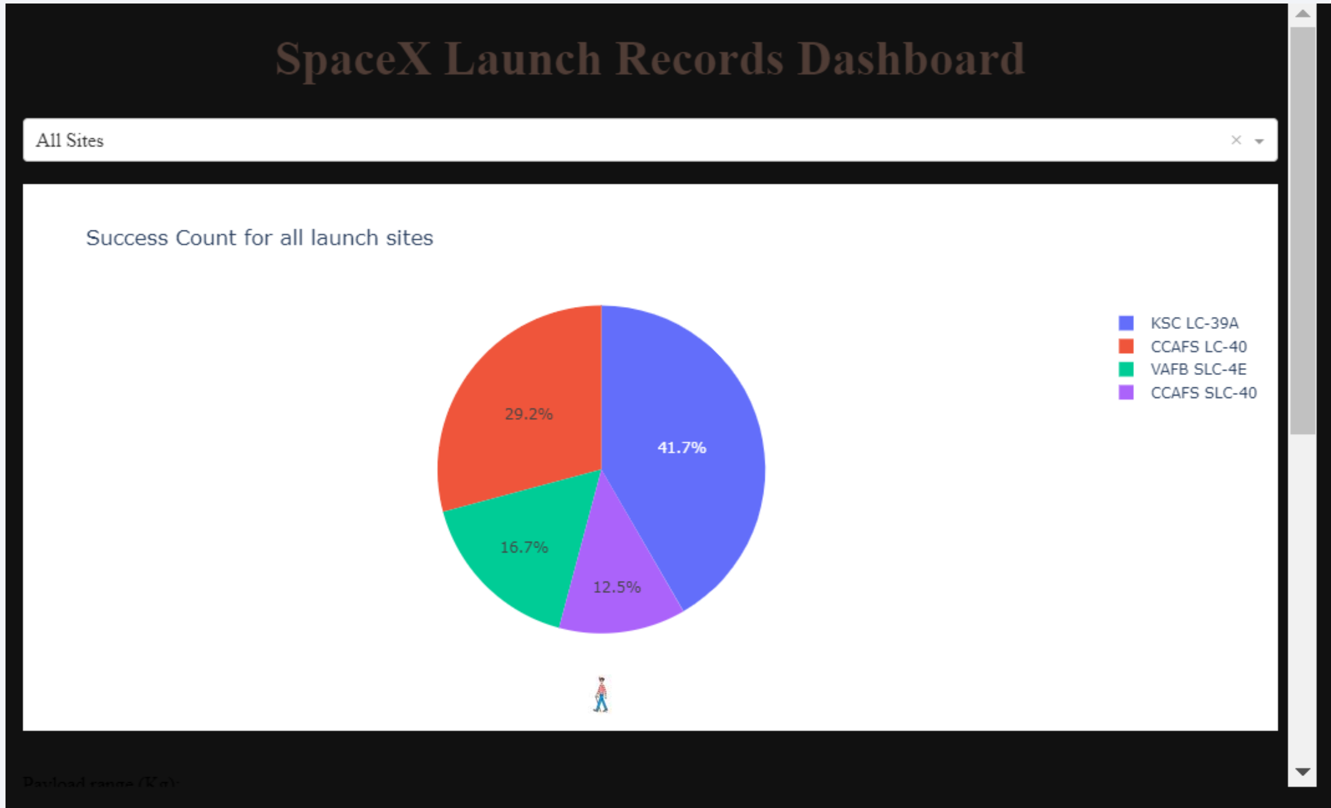
Build a Dashboard with Plotly Dash

< screenshot of launch success count for all sites, in a piechart >

- I couldn’t get this to work in the Jupyter labs IDE, so I ran it in Jupyter Notebooks
- This makes screenshots not as good unfortunately, and I prefer to use Dark/Night mode so background will be in black.
- Apologies for any inconvenience going forwards.

< screenshot of launch success count for all sites, in a piechart >

- Replace <Dashboard screenshot 1> title with an appropriate title
- Show the screenshot of launch success count for all sites, in a piechart
- Explain the important elements and findings on the screenshot
- Highest success count is KSC LC-39A with lowest being CCAFS SLC-40


SpaceX Launch Records Dashboard

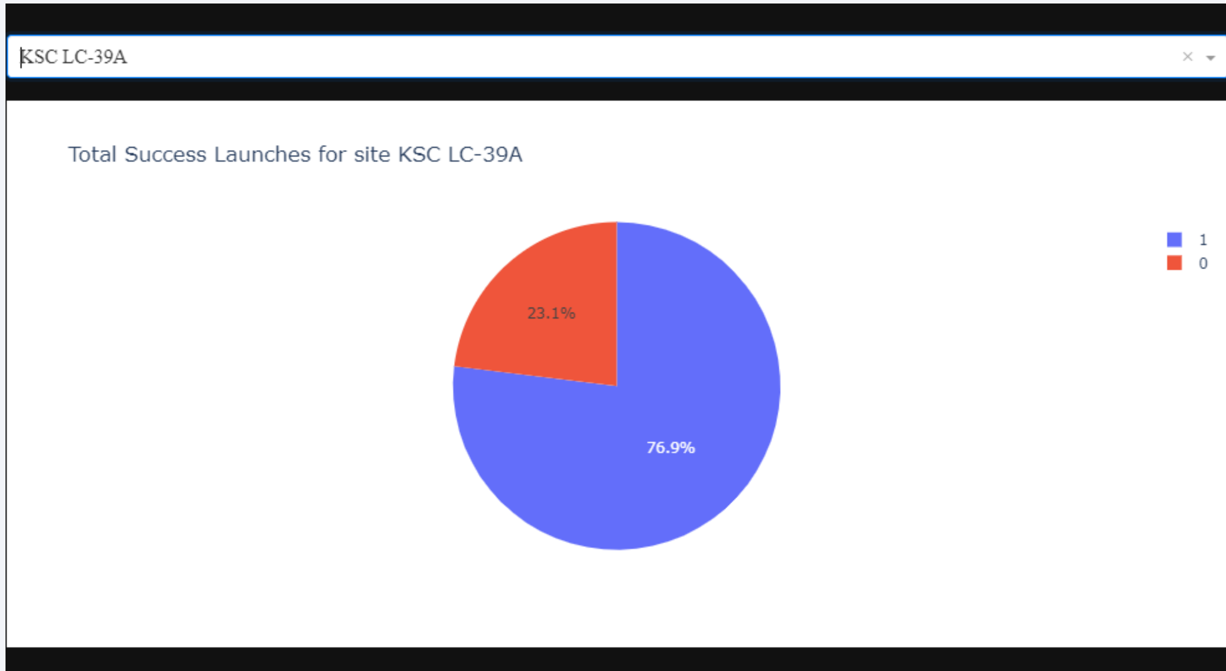
All Sites

Success Count for all launch sites

< screenshot of the piechart for the launch site with highest launch success ratio >

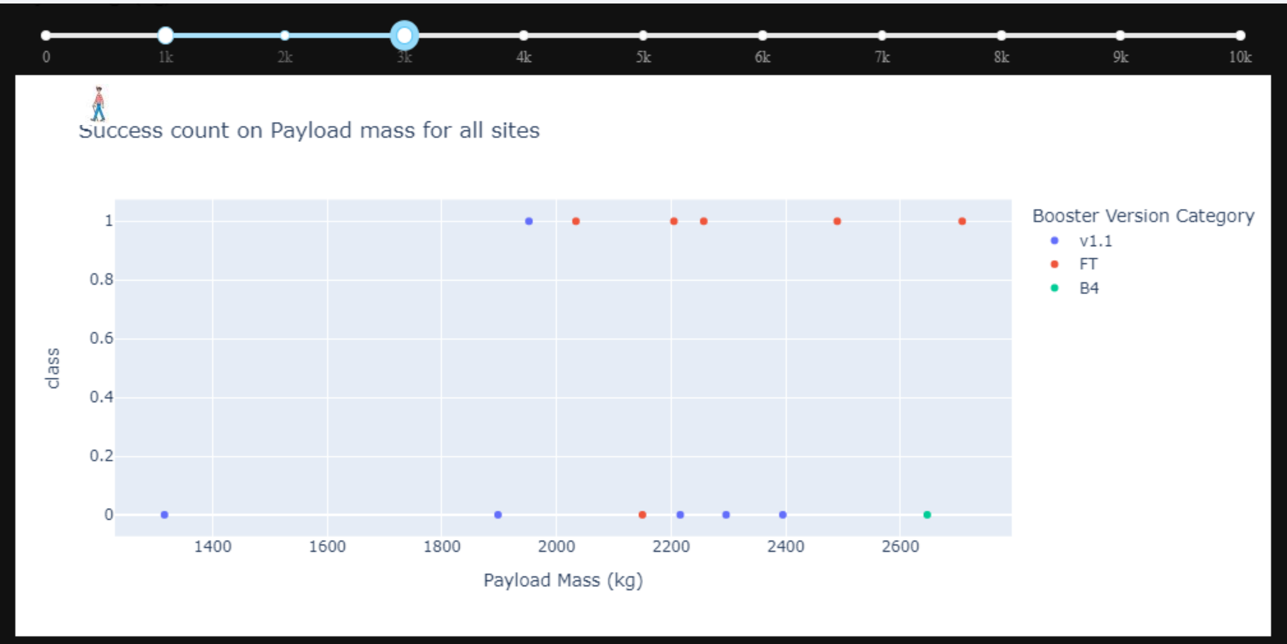
- Replace <Dashboard screenshot 2> title with an appropriate title
- Show the screenshot of the piechart for the launch site with highest launch success ratio
- Explain the important elements and findings on the screenshot
- KSC-LC39A has highest count and highest

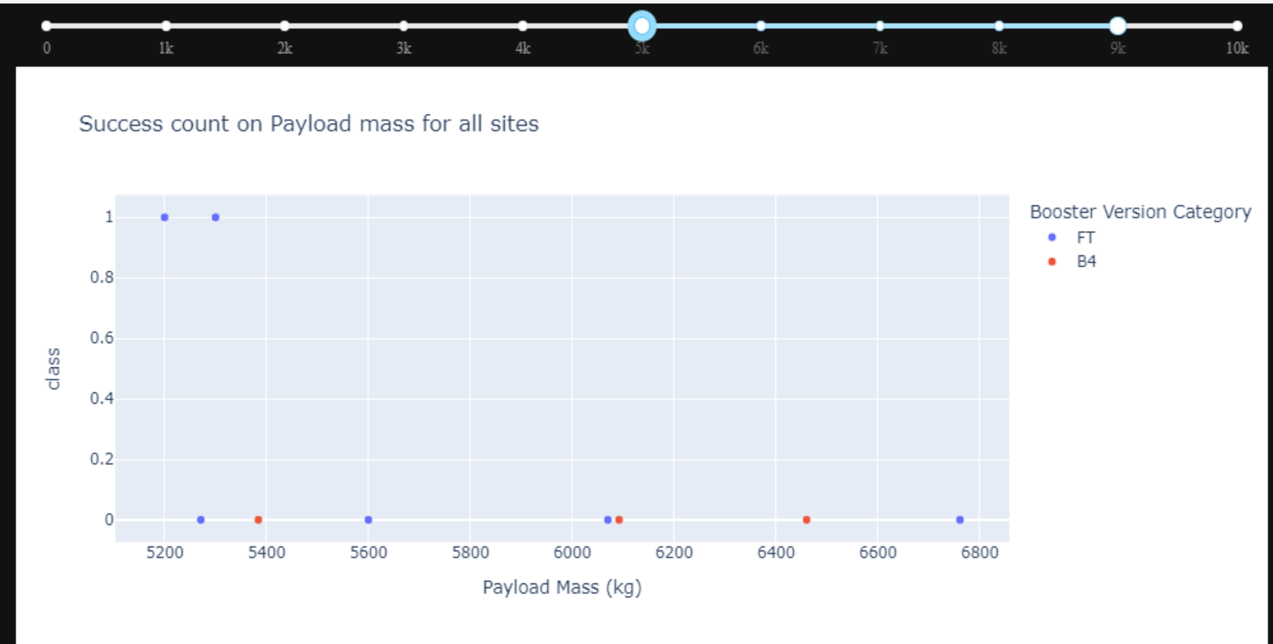
success rate at ~77%



< screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider >

- Replace <Dashboard screenshot 3> title with an appropriate title
- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider
- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.
- Left Range 1K-3K FT has highest success rate, Right Range 5K to 9K only FT has any successful landings.





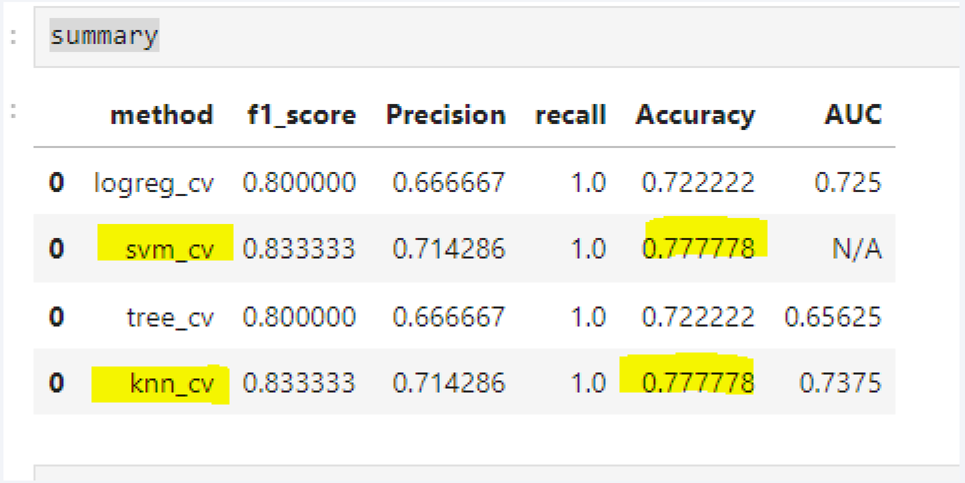
Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart

- Find which model has the highest classification accuracy
- SVM and KNN

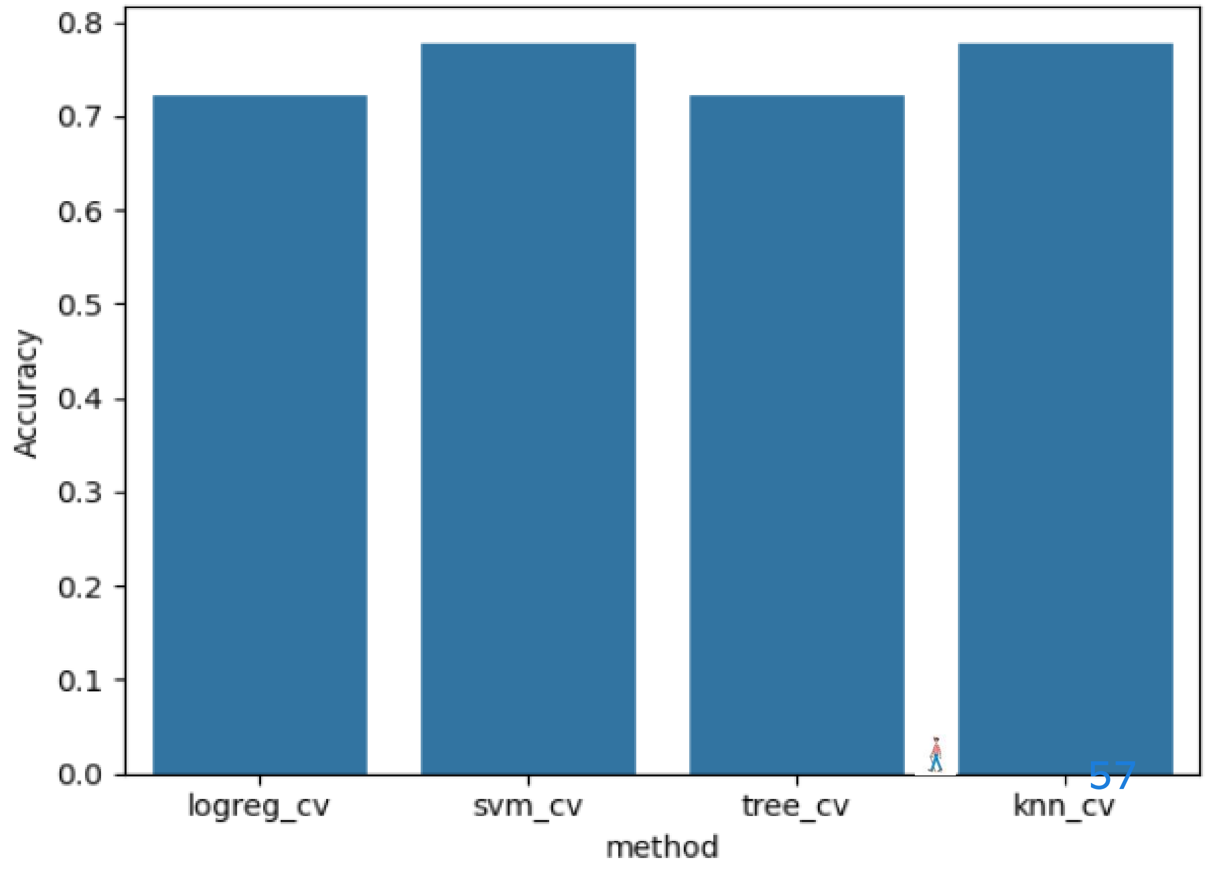


```
summary
```

	method	f1_score	Precision	recall	Accuracy	AUC
0	logreg_cv	0.800000	0.666667	1.0	0.722222	0.725
0	svm_cv	0.833333	0.714286	1.0	0.777778	N/A
0	tree_cv	0.800000	0.666667	1.0	0.722222	0.65625
0	knn_cv	0.833333	0.714286	1.0	0.777778	0.7375

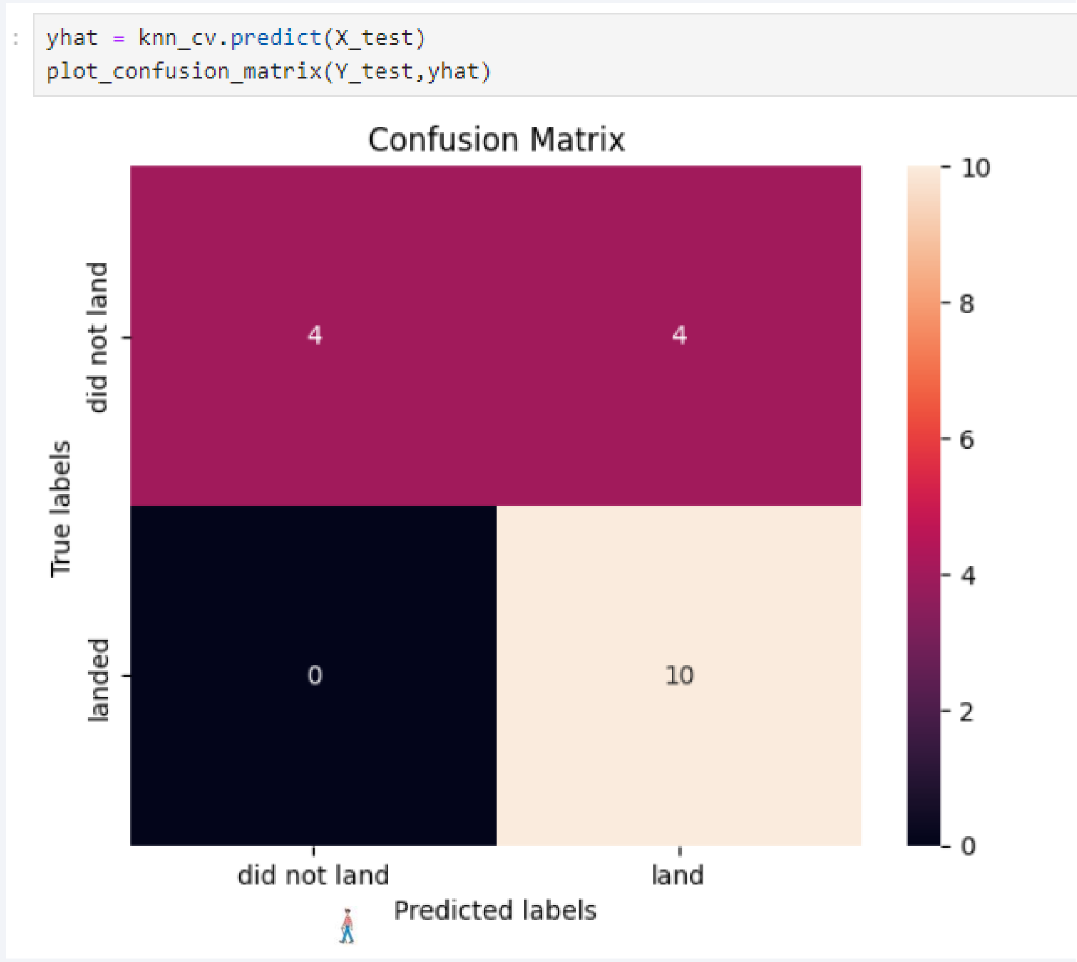
```
sns.barplot(summary, x="method", y="Accuracy")
```

```
<AxesSubplot:xlabel='method', ylabel='Accuracy'>
```



Confusion Matrix

- Show the confusion matrix of the best performing model with an explanation
- 0x FP and 10xTP
- 4x each of TN and FN
- Overall pretty much the same as the other confusion matrices the only reason I have declared this better than any other is because I like KNN.
- Seriously svm and decision tree had exactly the same matrix's, only logistic had a different one and only 1 value.
- But out of 18 tests only 4 were predicted incorrectly.



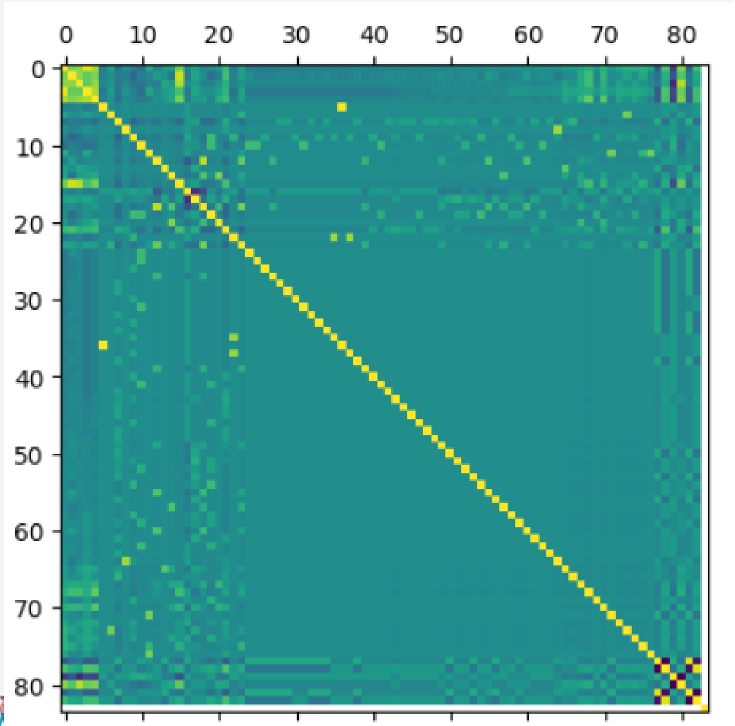
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

Conclusions

- Primary conclusion- It is possible to use ML techniques to make semi accurate predictions about the likelihood of successful landing of the stage 1 booster post launch
- Several factors including Launch site, launch number, booster version, payload mass, and intended orbit type can be employed as strong predictors.
- While overall it’s still better than flipping a coin it is not something that I Would put my money on.
- I bet you didn’t find all of my where’s Walleys (waldo for Americans)!

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project
- x= pd.DataFrame(X) // y= pd.DataFrame(Y, columns=['Class’]) // z=pd.concat([x,y])// z.corr()
- import matplotlib.pyplot as plt // plt.matshow(z.corr()) // plt.show()
- Handy for seeing which if any factor is a major contributor
- Note class is far right or bottom row/column



	Payload Mass	Flights	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	Orbit_GTO	Orbit_HEO	Orbit_ISS	...	Serial_B1059	Serial_B1060	Serial_B1062	GridFins_False	GridFins_True	Reused_False	Reused_True	Legs_False	Legs_True	Class	
FlightNumber																					
FlightNumber	1.00	0.60	0.65	0.93	0.74	-0.13	0.15	-0.26	0.01	-0.13	...	0.27	0.29	0.18	-0.44	0.44	-0.635844	0.635844	-0.373619	0.373619	NaN
Payload Mass	0.60	1.00	0.67	0.52	0.61	-0.13	0.00	-0.15	-0.13	-0.33	...	-0.01	0.22	-0.06	-0.26	0.26	-0.468393	0.468393	-0.192318	0.192318	NaN
Flights	0.65	0.67	1.00	0.52	0.68	-0.07	0.02	-0.21	-0.07	-0.25	...	0.13	0.03	-0.07	-0.19	0.19	-0.782631	0.782631	-0.157727	0.157727	NaN
Block	0.93	0.52	0.52	1.00	0.74	-0.17	0.10	-0.19	0.03	-0.12	...	0.20	0.18	0.10	-0.44	0.44	-0.548036	0.548036	-0.368994	0.368994	NaN
ReusedCount	0.74	0.61	0.68	0.74	1.00	-0.10	0.02	-0.27	-0.04	-0.12	...	0.17	0.04	-0.10	-0.41	0.41	-0.594109	0.594109	-0.375496	0.375496	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Reused_False	-0.64	-0.47	-0.78	-0.55	-0.59	0.09	-0.13	0.10	0.09	0.19	...	-0.15	-0.10	0.09	0.23	-0.23	1	-1	0.155552	-0.155552	NaN
Reused_True	0.64	0.47	0.78	0.55	0.59	-0.09	0.13	-0.10	-0.09	-0.19	...	0.15	0.10	-0.09	-0.23	0.23	-1	1	-0.155552	0.155552	NaN
Legs_False	-0.37	-0.19	-0.16	-0.37	-0.38	-0.05	-0.05	0.26	-0.05	-0.09	...	-0.11	-0.10	-0.05	0.90	-0.90	0.155552	-0.155552	1	-1	NaN
Legs_Tr	0.37	0.19	0.16	0.37	0.38	0.05	0.05	-0.26	0.05	0.09	...	0.11	0.10	0.05	-0.90	0.90	-0.15555	0.15555	-1	1	NaN

Thank you!