Data Science Capstone project

Daniel Arroyo

01-09-2021

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Summary of methodologies

Many python libraries were used through different stages helping us understand and prepare de data, analyze and visualize from many angles with plots and SQL queries among other things.

- Summary of all results

We can quite guess if SpaceX is going to land successfully the first stage of the rocket launch thanks to many machine learning models made in this project,

Introduction

- Project background and context

The commercial space age is here, companies are making space travel affordable for everyone. Many people are into this market but SpaceX shines between all of them because of their relatively inexpensive rocket launches (Falcon 9 = 62 millions) much of the savings is because SpaceX can reuse first stage.

Sometimes the first stage does not land. Sometimes it will crash. Other times, Space X will sacrifice the first stage due to the mission parameters like payload, orbit, and customer.

As a data scientist working for the new rocket company SpaceY that would like to compete with SpaceX. We will have to gather information about SpaceX and create our own dashboards.

- Problems you want to find answers

We will need to determine the price of each launch and if SpaceX will reuse the first launching stage by training a machine learning model and using other public information.

Methodology

- Data collection methodology:
  - Describe how data were collected
- Perform data wrangling
  - Describe how data were processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Methodology

Data collection

- Describe how data sets were collected.

  In order to collect data through SpaceX API. First, we settled a series of helper functions that will help us use the API to extract information using identification numbers in the launch data. Then we started requesting rocket launch data from SpaceX API, decode the response content as JSON and turn it to a Pandas dataframe. We will now use the API again to get information about the launches using the IDs given for each launch. The data obtained from specified columns is stored in lists and those list are combined into a dictionary that will be used to create the Pandas dataframe. Finally, we will remove the Falcon 1 launches keeping only the Falcon 9 launches which are the aim of the analysis.

- You need to present your data collection process use key phrases and flowcharts

  Key phrases and flowcharts are shown in the next slideshows.

Data collection – SpaceX API

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

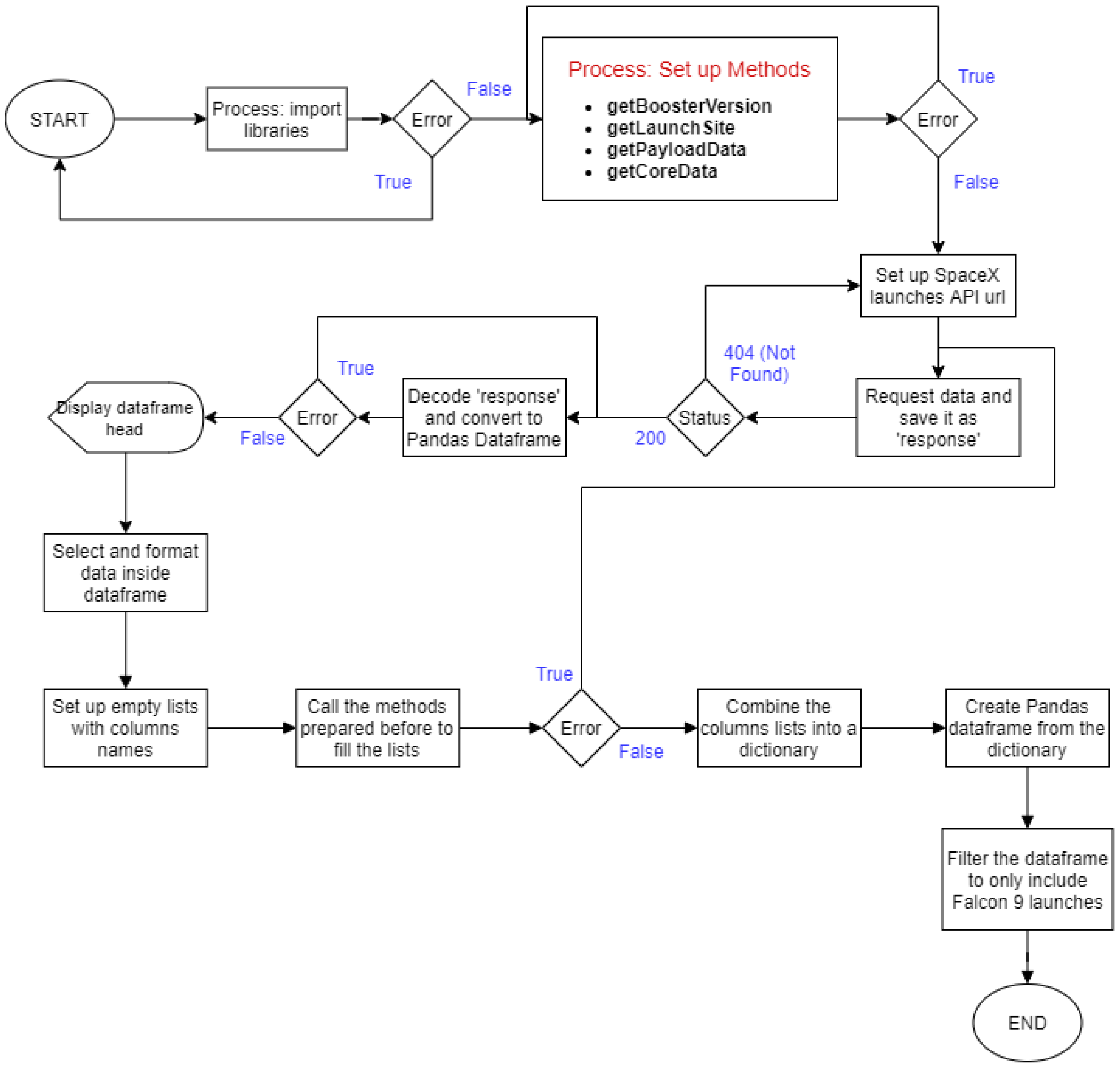
Its important to set up all SpaceX API endpoints right in most of the cells like.

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

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/DaniArroyo/Data-Science-Capstone/blob/main/Capstone_SpaceX_DataCollection.ipynb

Added a flowchart of SpaceX API calls here



START

Process: import libraries

Error

False

True

Process: Set up Methods

- getBoosterVersion
- getLaunchSite
- getPayloadData
- getCoreData

Error

True

False

Set up SpaceX launches API url

Request data and save it as 'response'

Status

404 (Not Found)

200

Decode 'response' and convert to Pandas Dataframe

Error

True

False

Display dataframe head

Select and format data inside dataframe

Set up empty lists with columns names

Call the methods prepared before to fill the lists

Error

True

False

Combine the columns lists into a dictionary

Create Pandas dataframe from the dictionary

Filter the dataframe to only include Falcon 9 launches

END

Data collection – Web scraping

Add a flowchart of web scraping here

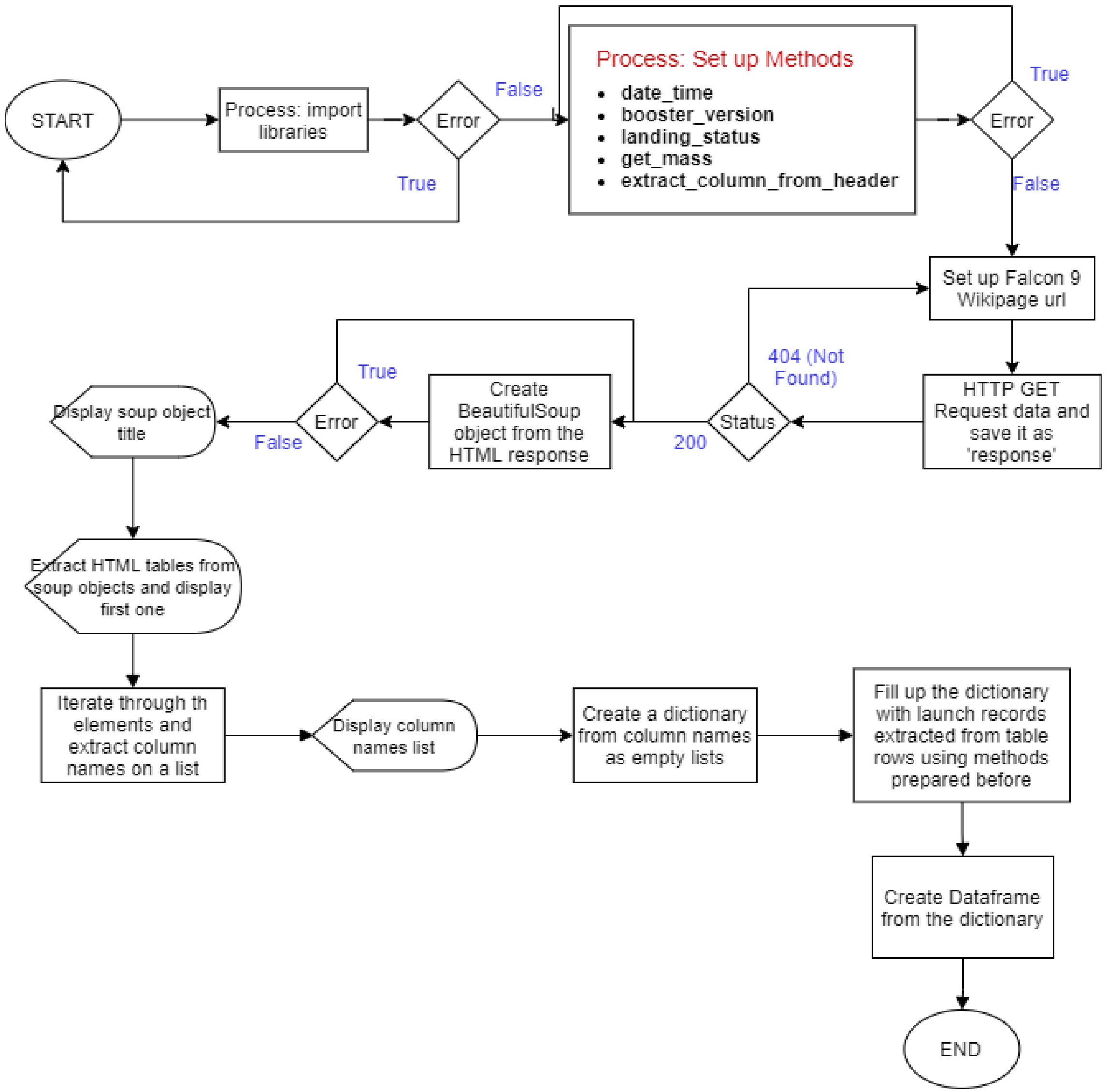
Present your web scraping process use key phrases and flowcharts

Its important to set up correctly the Wikipage URL and have some HTML structure knowledge to manipulate SOUP object and response data.

```
static_url="https://en.wikipedia.org/w
/index.php?title=List_of_Falcon_9_a
nd_Falcon_Heavy_launches&oldid=1027
686922"
```

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

https://github.com/DaniArroyo/Data-Science-Capstone/blob/main/Web%20Scraping%20Lab.ipynb



START
Process: import libraries
Error
False
True
Process: Set up Methods
date_time
booster_version
landing_status
get_mass
extract_column_from_header
Error
True
False
Set up Falcon 9 Wikipage url
HTTP GET Request data and save it as 'response'
Status
404 (Not Found)
200
Create BeautifulSoup object from the HTML response
Error
True
False
Display soup object title
Extract HTML tables from soup objects and display first one
Iterate through th elements and extract column names on a list
Display column names list
Create a dictionary from column names as empty lists
Fill up the dictionary with launch records extracted from table rows using methods prepared before
Create Dataframe from the dictionary
END

Data wrangling

- Describe how data were processed

  First, identify null values in each column and identify their type: numerical or categorical. To know more about the data, we use value_counts() method in some columns to determine the count of each outcome.

  Now we iterate through landing_outcomes() keys to create a set with the bad outcomes, then we iterate again to add 1 if the landing was successful or 0 if the outcome is found inside the set we created before.

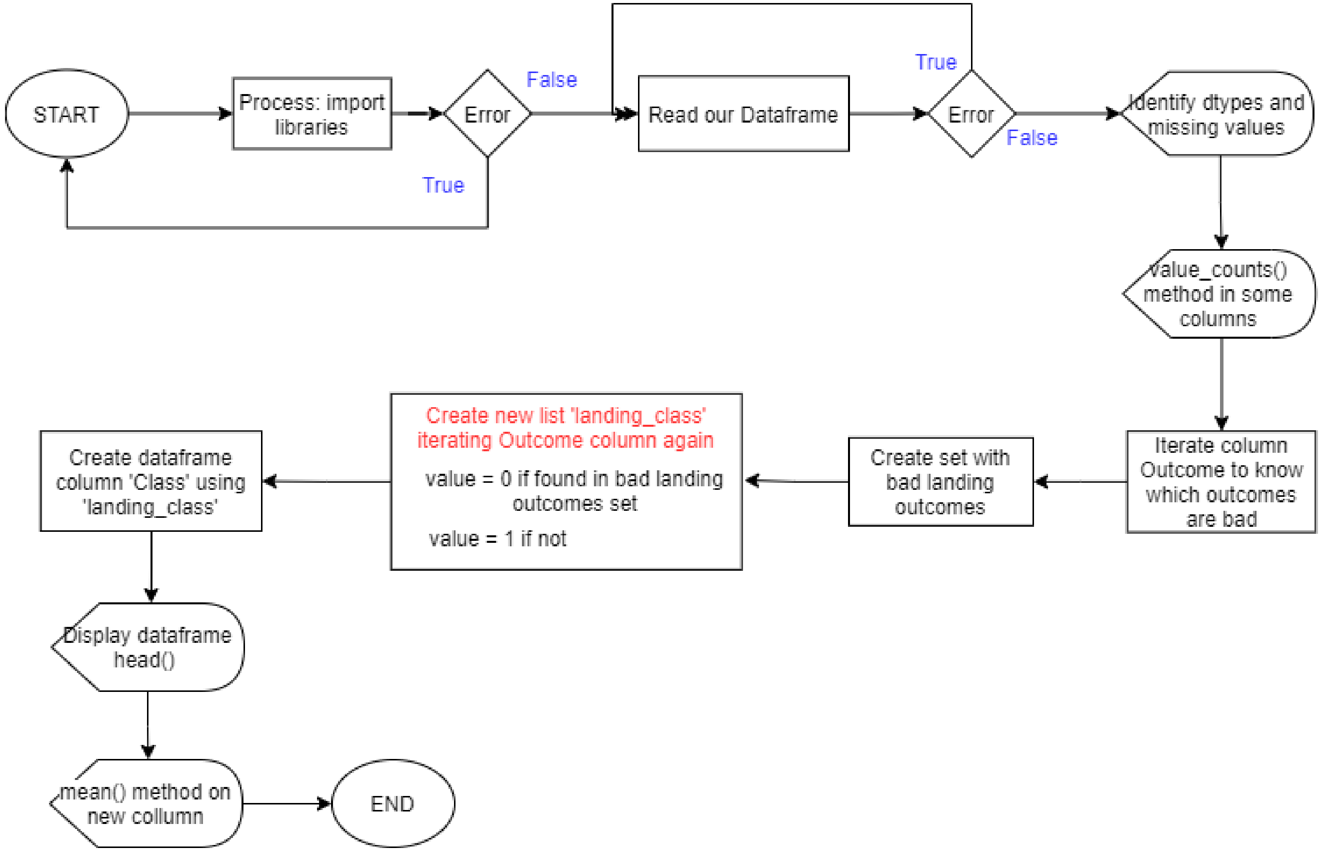
  Lastly with the last set we made we create a column in the dataframe called class and using method mean() we can get the success rate which is 0.6666.

- Add the GitHub URL of your completed data wrangling related notebooks, as an external reference and peer-review purpose

  https://github.com/DaniArroyo/Data-Science-Capstone/blob/main/Data%20Wrangling.ipynb

Data wrangling

- You need to present your data wrangling process using key phrases and flowcharts


START
Process: import libraries
Error
False
True
Read our Dataframe
True
Error
False
Identify dtypes and missing values
value_counts() method in some columns
Iterate column Outcome to know which outcomes are bad
Create set with bad landing outcomes
Create new list 'landing_class' iterating Outcome column again
value = 0 if found in bad landing outcomes set
value = 1 if not
Create dataframe column 'Class' using 'landing_class'
Display dataframe head()
mean() method on new collumn
END

EDA with data visualization

- Summarize what charts were plotted and why used those charts

  Most used charts are scatter plots which are the best to observe and show relationships between two numeric variables using parameters like hue that produce points with different colors.

  There's also used a bar chart to visualize the relationship between success rate of each orbit type.

  Lastly a linear chart to see how the success rate increase through years.

- Add the GitHub URL of your completed EDA with data visualization notebook, as an external reference and peer-review purpose

  https://github.com/DaniArroyo/Data-Science-Capstone/blob/main/EDA%20with%20Visualization%20lab.ipynb

EDA with SQL

- Summarize performed SQL queries using bullet points

  All queries are SELECT with 1 or 2 required CONDITIONS, most statements have methods like MIN(), MAX(), AVG() and most used one is COUNT().

  Also, in some of them there's used SUBQUERIES commonly known as a query inside another query,

  Finally, last query contains statements GROUP BY AND ORDER BY to sort and group data about the landing outcomes.

- Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose

  https://github.com/DaniArroyo/Data-Science-Capstone/blob/main/EDA%20With%20SQL.ipynb

Build an interactive map with Folium

- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map
- Explain why you added those objects

  First object used are markers that help us visualize locations like launch sites and their proximities by pinning them on the map, on the other hand coordinates are just plain numbers that can not give intuitive insights about places.

  Circles used have the same purpose as markers in this folium maps but with less customization.

  As we have many launch records with the exact same coordinate, we used marker clusters to simplify from having many markers in the same coordinate.

  Last object used are Polylines to draw a line between launch sites and their proximities like Cabrillo Highway and VAFB SLC-4E launch site.

- Add the GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose

  https://github.com/DaniArroyo/Data-Science-Capstone/blob/main/Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- Explain why you added those plots and interactions
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

https://github.com/DaniArroyo/Data-Science-Capstone/blob/main/Interactive%20Dashboard.py

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

First, we choose the columns that will be our ‘y’ and ‘x’ and split their values as train and test data. Next step is to create a GridSearchCV object for different models (4 in our case), each one with their own parameters, then fit the object to find the best parameters for each model and their best score.

Finally, we can calculate their accuracy on test data and plot a confusión matrix to compare all models performance.

- Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose

https://github.com/DaniArroyo/Data-Science-Capstone/blob/main/Machine%20Learning%20Prediction%20lab.ipynb

Results

- Exploratory data analysis results

Through EDA we found out strong relationships between Payload mass and some of the orbits like GTO(negative) or Polar LEO ISS(positive), or between orbit and success rate.

Also, we can observe that the success rate since 2013 kept increasing till 2020.

- Interactive analytics demo in screenshots

We created interactive folium maps to understand important facts about the success rate of SpaceX rocket launch in each launch site. Also, we build an interactive analytics dashboard.

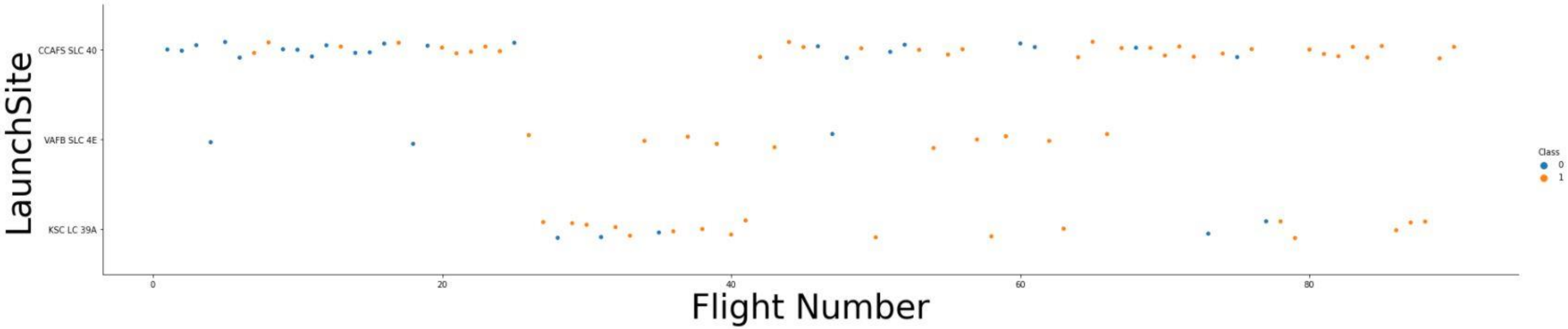
- Predictive analysis results

Most of the machine learning models that we built have great accuracy and have a high chance of predicting whether SpaceX is not going to land the first stage successfully or not.

EDA with Visualization

Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site
- Show the screenshot of the scatter plot with explanations

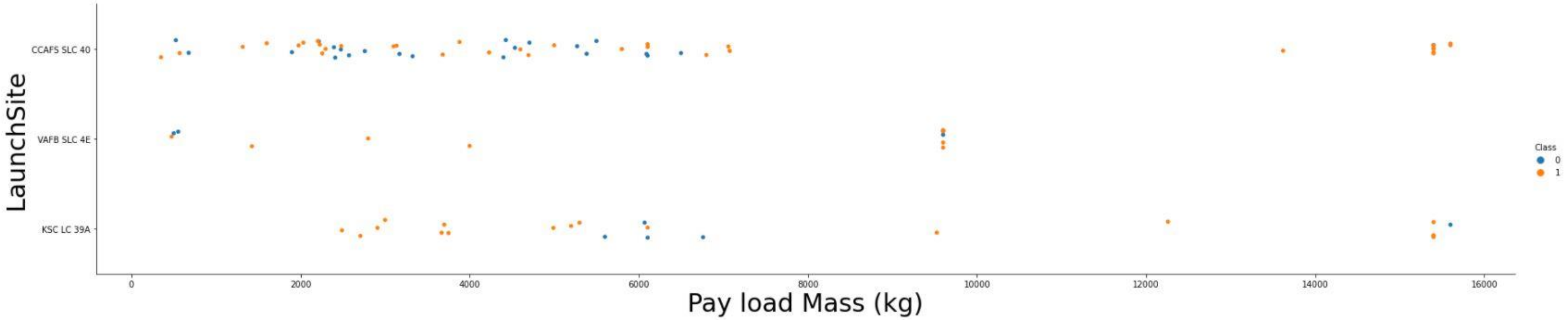


In this scatter plot we can see that CCAFS SLC 40 had intervals between 4 and 3 unsuccessful landings before start getting successful ones having a 60% successful landing rate in 55 landings followed up by VAFB SLC 4E having a 77% in 13 landings and lastly by KSC LC 39A having 77.2% in 22 landings. Let's keep in mind that it was the first launchsite used in this analysis with most landings than the other 2.

Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site

- Show the screenshot of the scatter plot with explanations



In this scatter plot we can see that CCAFS SLC 4O Has a higher success rate on a bigger payload Mass range than others 2 launch sites which have way less success rate on specific payload Mass ranges like KSC LC 39A launch site in range between 2000Kg and 5000Kg.

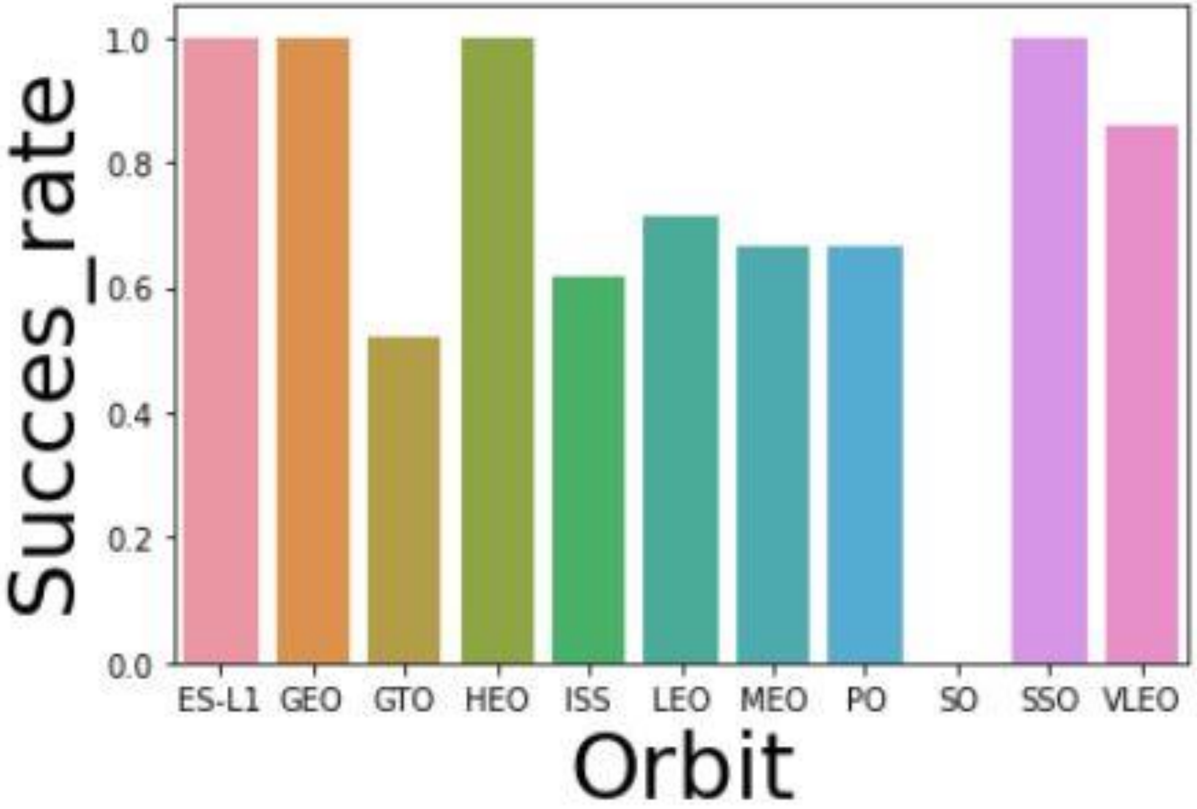
Success rate vs. Orbit type

Show a barchart for the success rate of each orbit type

Show the screenshot of the scatter plot with explanations

ES-L1, GEO, HEO and SSO are the orbits that have the highest success rate followed by VLEO orbit. GTO, ISS, LEO, MEO and PO orbits can be found at 50% - 70% success rate range.

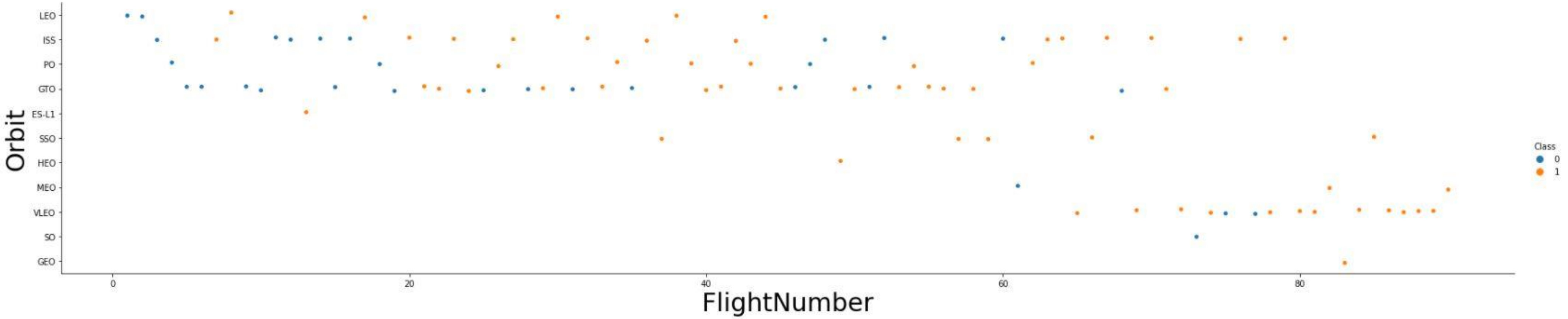
To conclude S0 orbit has the lowest success rate with only 1 unsuccessful landing leading to a 0% success rate.



Flight Number vs. Orbit type

- Show a scatter point of Flight number vs. Orbit type

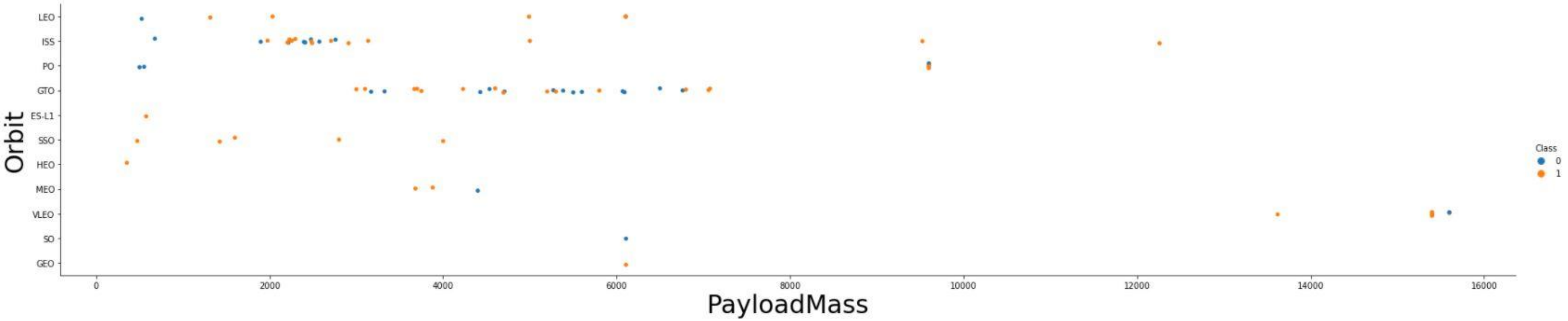
- Show the screenshot of the scatter plot with explanations



In orbits like LEO, SSO and VLEO the success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO and ISS orbit.

Payload vs. Orbit type

- Show a scatter point of payload vs. orbit type
- Show the screenshot of the scatter plot with explanations

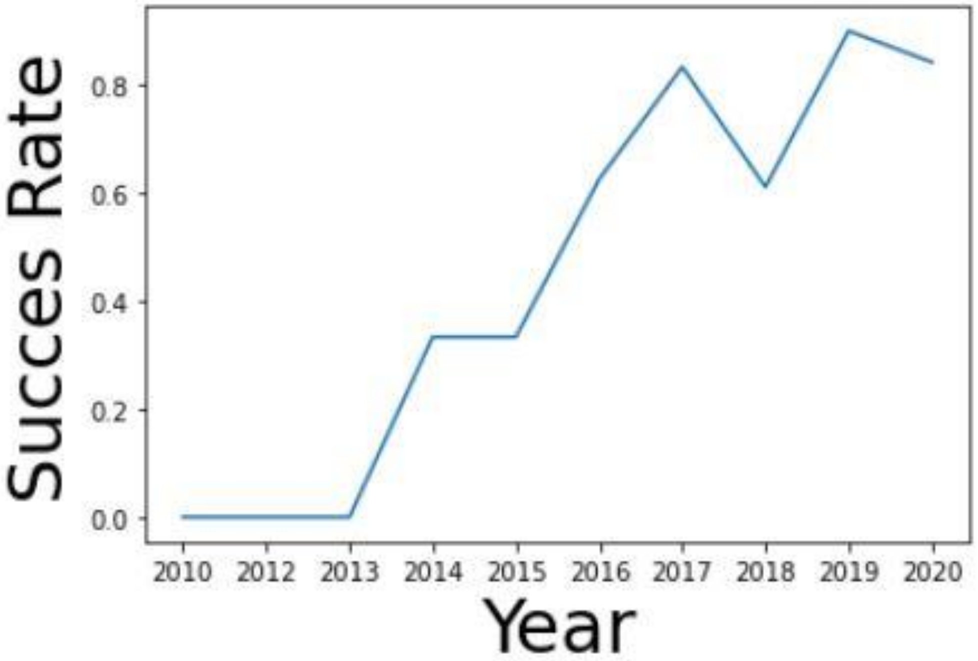


Here we see that heavy payloads have a negative influence on GTO and MEO orbits, and a positive influence on ISS and LEO orbits. We can also see that on ISS orbit there's a Payload Mass range where rates of success are lower as it happens on PO orbit with light payloads.

Launch success yearly trend

- Show a line chart of yearly average success rate
- Show the screenshot of the linear plot with explanations

As seen in this linear plot succes rate started increasing in 2013 and kept like that till 2020 with little drops in 2018 and 2020.



EDA with SQL

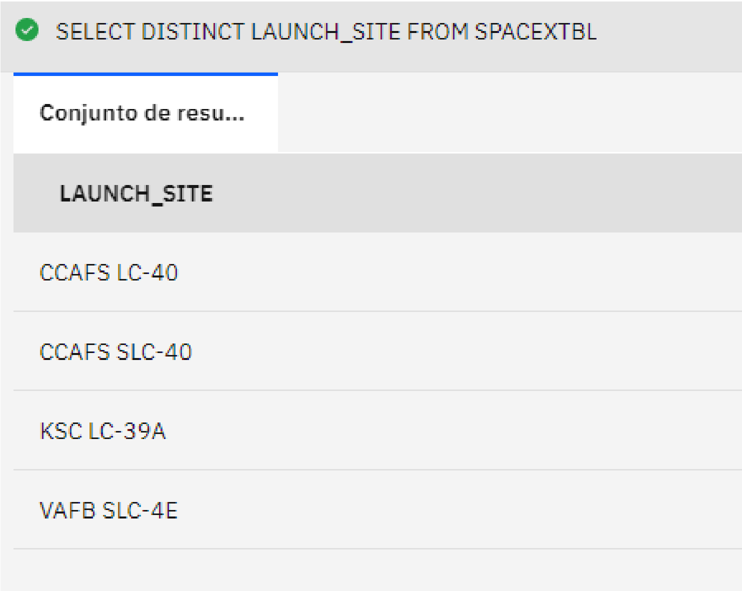
All launch site names

- Find the names of the unique launch sites
- Present your query result with a short explanation here

```
SELECT DISTINCT LAUNCH_SITE
FROM SPACEXTBL
```

Here we use DISTINCT statement to return only distinct (different) values.

Only 4 Launch sites are returned.



```
SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL
```

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

Launch site names begin with `CCA`

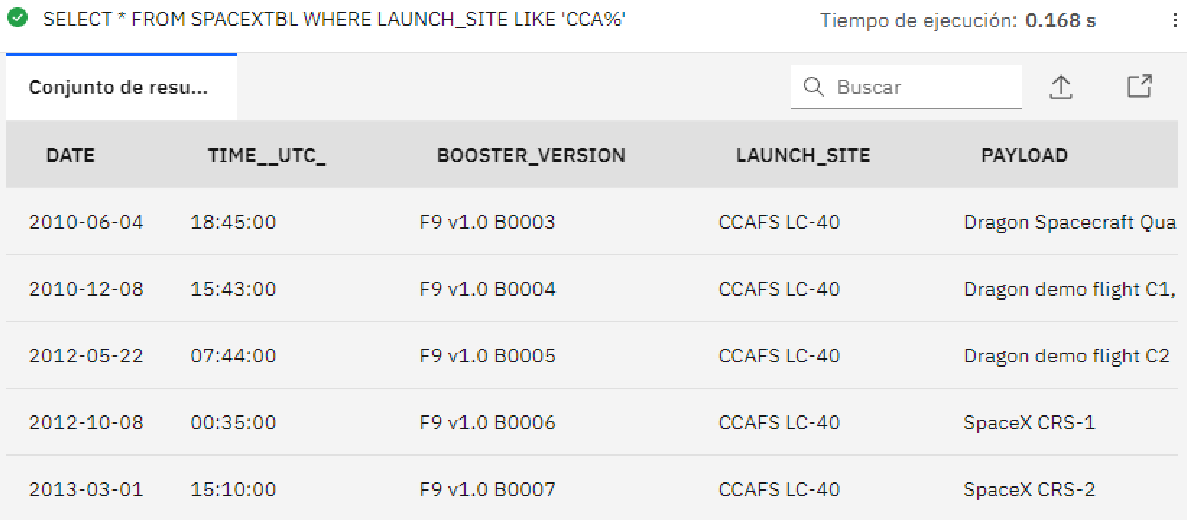
- Find all launch sites begin with `CCA`
- Present your query result with a short explanation here

SELECT * FROM SPACEXTBL

WHERE LAUNCH_SITE

LIKE 'CCA%'

In this query we use LIKE statement with '%' at the end of the value which we want to use as condition



SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%'

Tiempo de ejecución: **0.168 s**

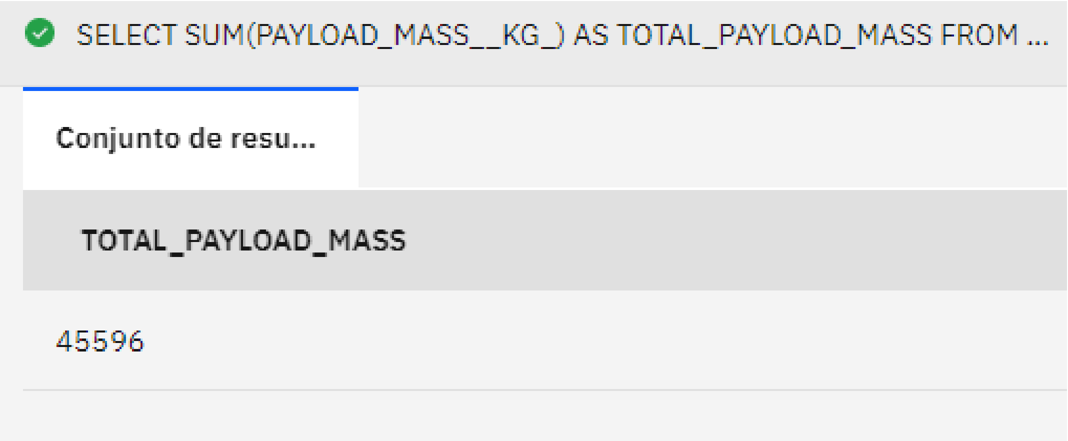
DATE	TIME__UTC_	BOOSTER_VERSION	LAUNCH_SITE	PAYLOAD
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qua
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1,
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2

Total payload mass

- Calculate the total payload carried by boosters from NASA
- Present your query result with a short explanation here

```
SELECT SUM(PAYLOAD_MASS__KG_)
AS TOTAL_PAYLOAD_MASS

FROM SPACEXTBL

WHERE CUSTOMER = 'NASA (CRS)'
```

In this query we use SUM() method to add up each number that meets the condition required, NASA boosters in this case.



SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS FROM ...

Conjunto de resu...

TOTAL_PAYLOAD_MASS
45596

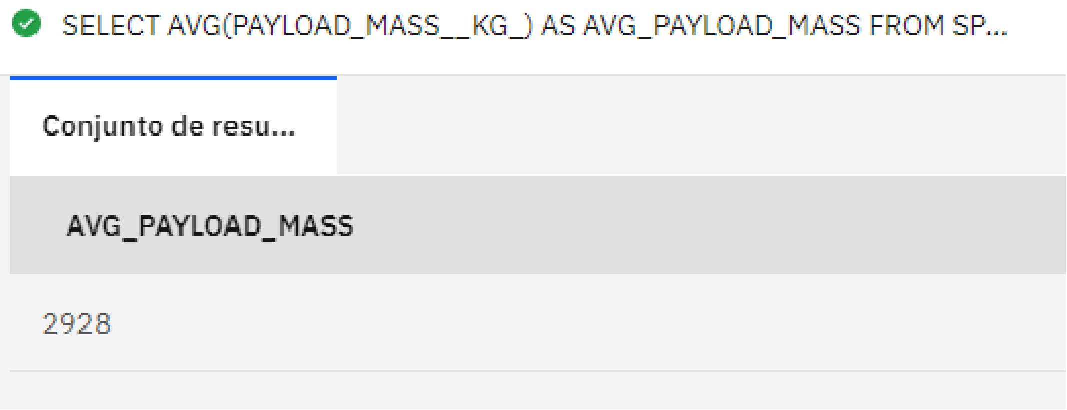
Total payload mass carried by NASA boosters is 45596Kg.

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

```
SELECT AVG(PAYLOAD_MASS__KG_)
AS AVG_PAYLOAD_MASS

FROM SPACEXTBL

WHERE BOOSTER_VERSION = 'F9 v1.1'
```

In this query we use AVG() method to get mean payload mass that meets the condition of being carried by a specified booster version.



SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_PAYLOAD_MASS FROM SP...

Conjunto de resu...

AVG_PAYLOAD_MASS
2928

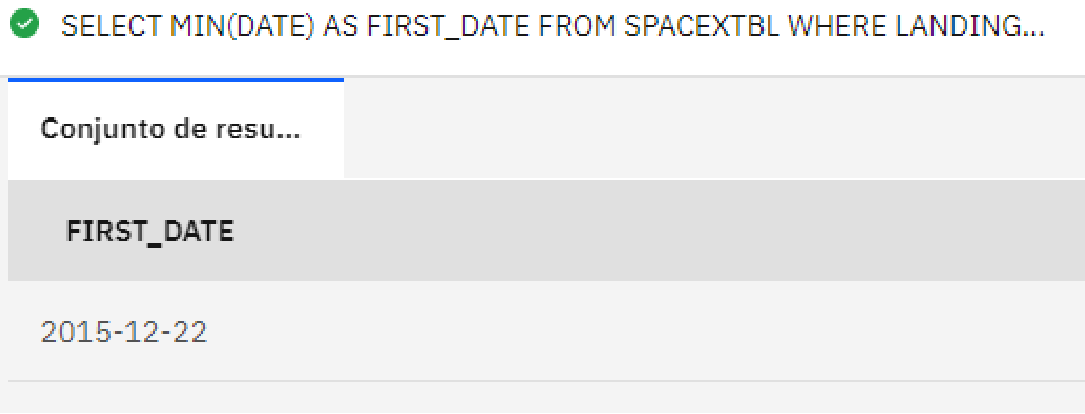
Average payload mass carried by F9 v1.1 booster version is 2928Kg.

First successful ground landing date

- Find the date when the first successful landing outcome in ground pad
- Present your query result with a short explanation here

```
SELECT MIN(DATE) AS FIRST_DATE
FROM SPACEXTBL
WHERE LANDING__OUTCOME =
'Success (ground pad)'
```

In this query we use MIN() method to get the first date of first successful landing outcome in ground pad.



SELECT MIN(DATE) AS FIRST_DATE FROM SPACEXTBL WHERE LANDING...

Conjunto de resu...

```
FIRST_DATE
2015-12-22
```

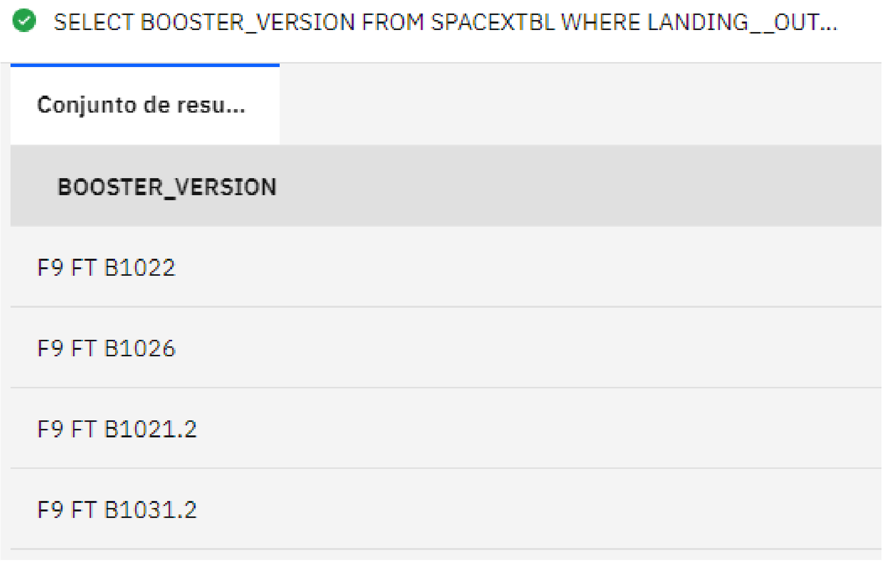
First successful landing outcome in ground pad was 2015-12-22

Successful drone ship landing with payload between 4000 and 6000

- List the names of boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Present your query result with a short explanation here

```
SELECT BOOSTER_VERSION
FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Success (drone ship)'
AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000
```

In this query we use 2 conditions to return needed booster versions, in this case only 4 booster versions will be returned.



SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING__OUT...

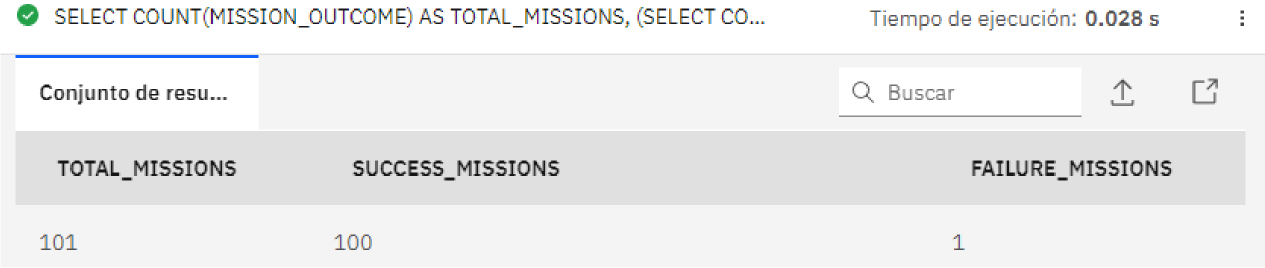
BOOSTER_VERSION
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

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

```
SELECT COUNT(MISSION_OUTCOME) AS TOTAL_MISSIONS,

(SELECT COUNT(MISSION_OUTCOME) AS SUCCESS_MISSIONS FROM SPACEXTBL WHERE
MISSION_OUTCOME LIKE 'Success%'),

(SELECT COUNT(MISSION_OUTCOME) AS FAILURE_MISSIONS FROM SPACEXTBL WHERE
MISSION_OUTCOME LIKE 'Failure%')

FROM SPACEXTBL
```

In this query we use COUNT() method and subqueries to count every mission outcomes and total of all missions combined.



SELECT COUNT(MISSION_OUTCOME) AS TOTAL_MISSIONS, (SELECT CO...

Tiempo de ejecución: **0.028 s**

Conjunto de resu...

TOTAL_MISSIONS	SUCCESS_MISSIONS	FAILURE_MISSIONS
101	100	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

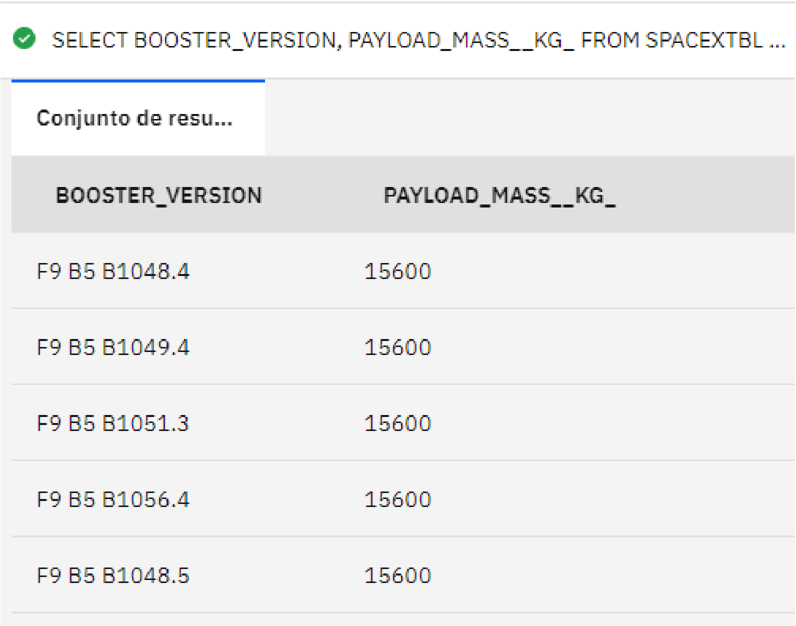
SELECT BOOSTER_VERSION, PAYLOAD_MASS__KG_

FROM SPACEXTBL

WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)

In this query we use a subquery inside the condition querying for the max value of payload mass.

As a result, 12 different booster versions are shown to us.



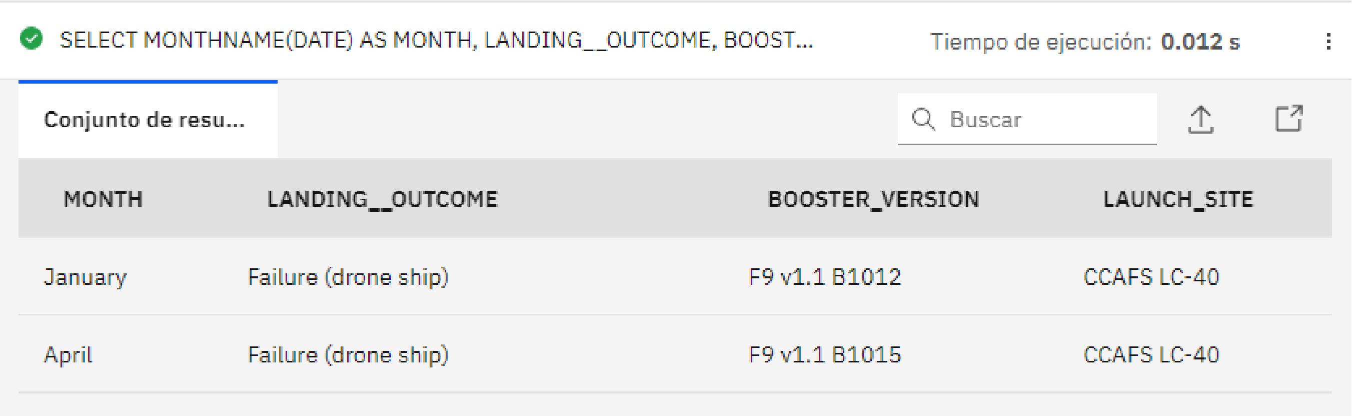
SELECT BOOSTER_VERSION, PAYLOAD_MASS__KG_ FROM SPACEXTBL ...

BOOSTER_VERSION	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600

2015 launch records

- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015
- Present your query result with a short explanation here

```
SELECT MONTHNAME(DATE) AS MONTH, LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL

WHERE LANDING__OUTCOME = 'Failure (drone ship)'

AND DATE LIKE '2015%'
```



SELECT MONTHNAME(DATE) AS MONTH, LANDING__OUTCOME, BOOST...

Tiempo de ejecución: **0.012 s**

MONTH	LANDING__OUTCOME	BOOSTER_VERSION	LAUNCH_SITE
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

In this query we use a MONTHNAME() method to extract month name from dates, and a couple of conditions required,

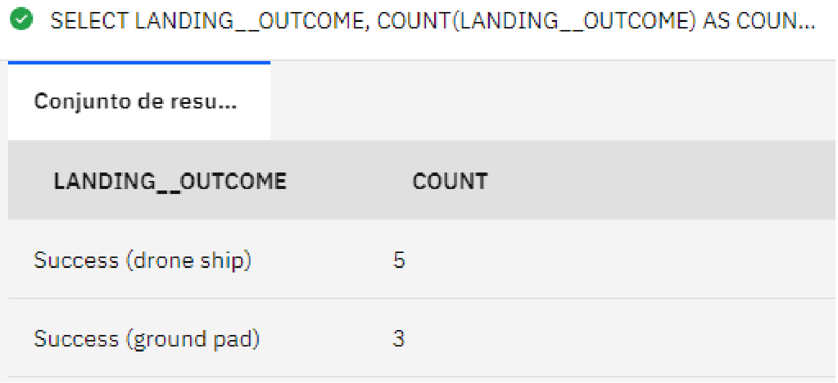
As a result, only 2 records are shown, first from January and second from April.

Rank success count between 2010-06-04 and 2017-03-20

- Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.
- Present your query result with a short explanation here

```
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS COUNT

FROM SPACEXTBL

WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' AND LANDING__OUTCOME LIKE 'Succes%'

GROUP BY LANDING__OUTCOME

ORDER BY COUNT(LANDING__OUTCOME) DESC
```

In this query we use COUNT() to count every time a landing outcome meets the required conditions, then we group by those conditions and order them in descending order.



```
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS COUN...
```

Conjunto de resu...

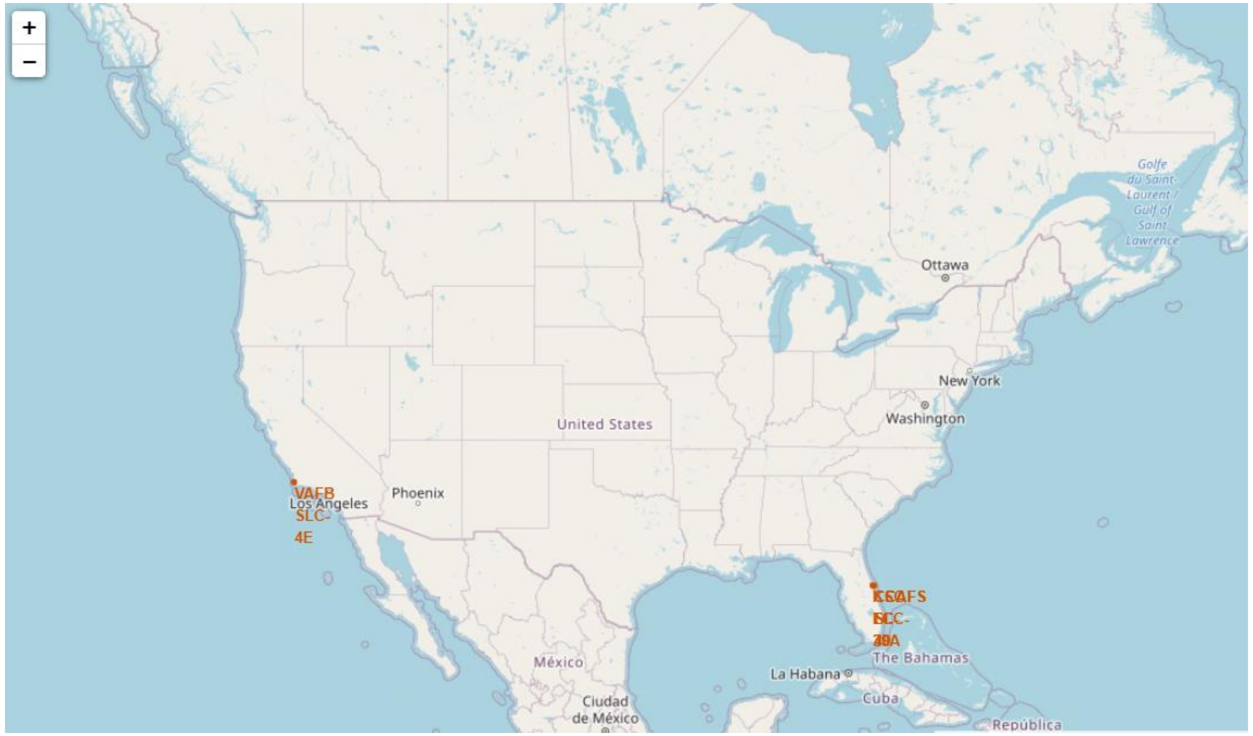
LANDING__OUTCOME	COUNT
Success (drone ship)	5
Success (ground pad)	3

As a result, we only see 2 rows, each one with its own counter.

Interactive map with Folium

Launch sites folium map

- Replace <Folium map screenshot 1> title with an appropriate title
- Show the screenshot of all launch sites’ location markers on a global map
- Explain the important elements and findings on the screenshot



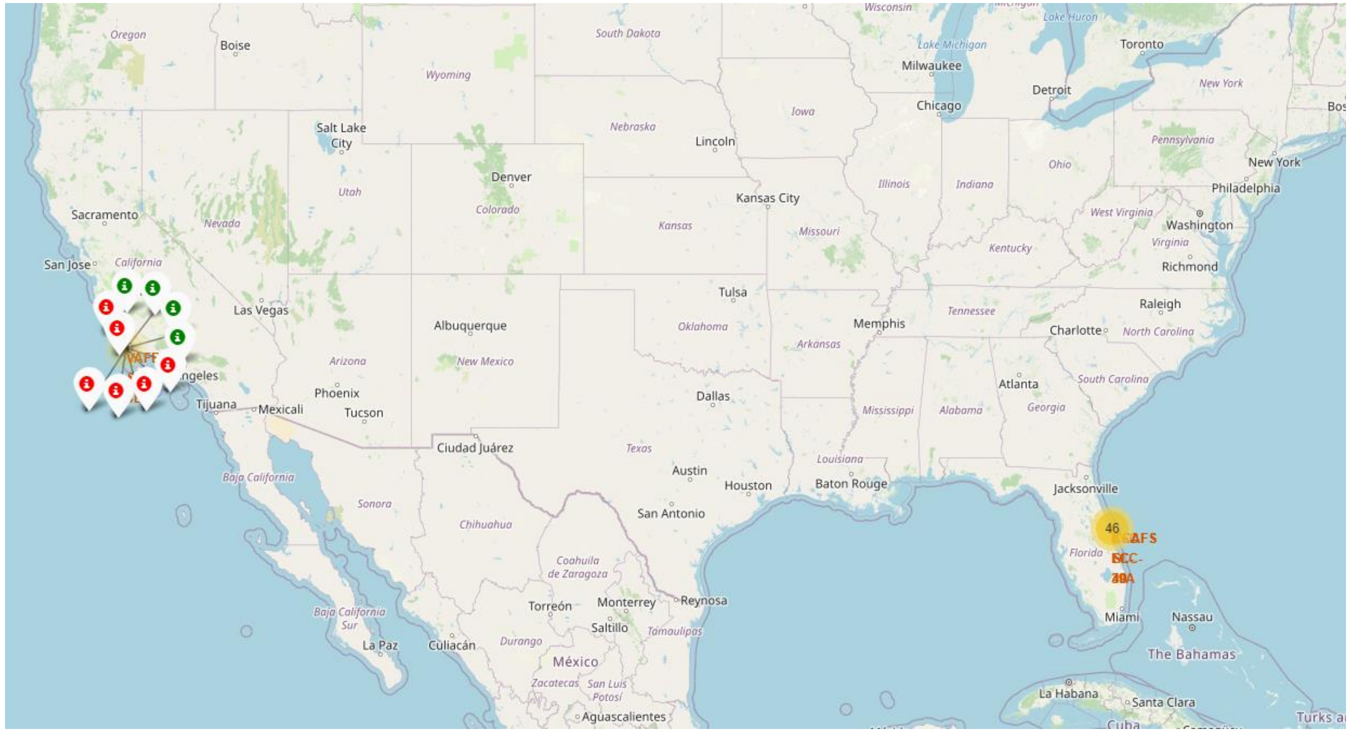
All US launch sites are the closest they can to they Equator line inside US territory, so their spacecrafts are going to be faster if launched from those launch sites than from anywhere in US.

As we can see in the map all these launch sites are close to the coast because most rockets travel eastward, so if anything goes wrong during their ascent, the debris will essentially fall into an ocean's waters, far away from densely populated areas.

Also Launching a rocket from the east coast gives an additional boost to the rocket, due to the rotational speed of Earth.

Launch records folium map

- Replace <Folium map screenshot 2> title with an appropriate title
- Show the screenshot of color-labeled launch records on the map
- Explain the important elements and findings on the screenshot

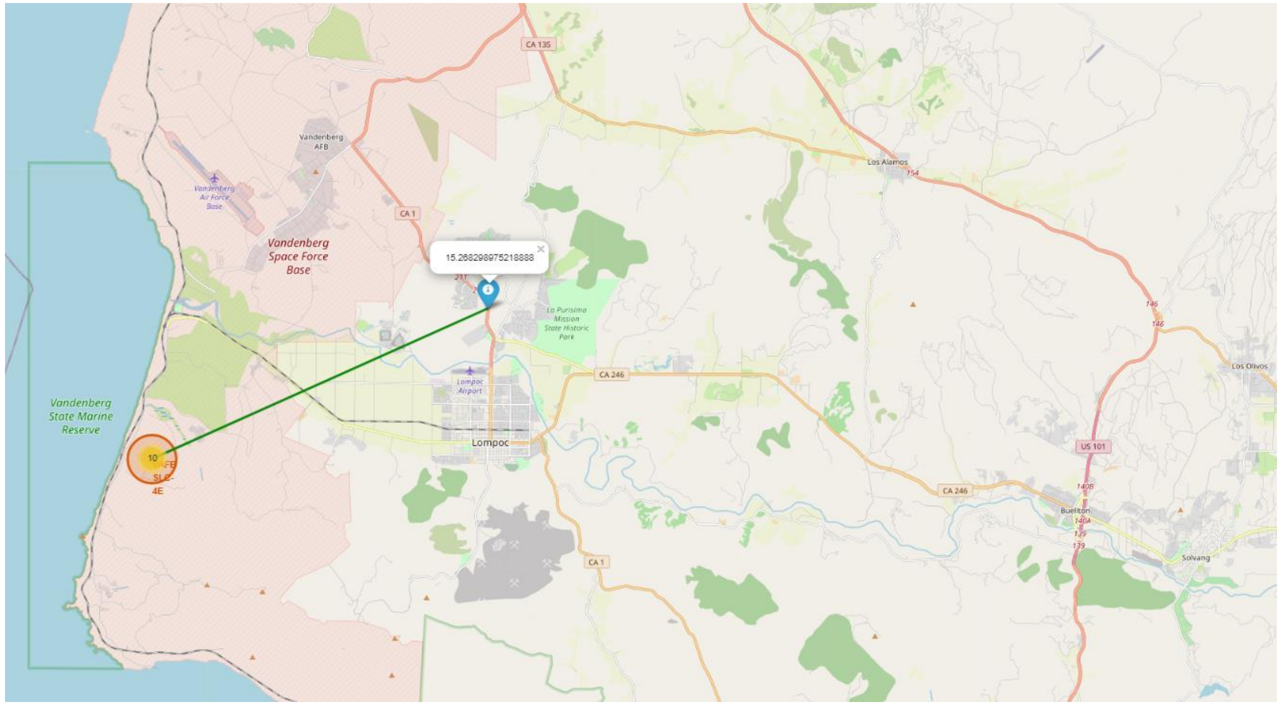


Now we can see on map by clicking on each launch site its launch records such as how many launches does a launch site has, how many of those launches were successful and where is a higher success rate.

For example, VAFB SLC-4E launch site has a total of 10 launches, 4 of them were successful, leading to a success rate of 40%.

VAFB SLC-4E Launch site and proximities

- Replace <Folium map screenshot 3> title with an appropriate title
- Show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed
- Explain the important elements and findings on the screenshot



15.268298975218888
Vandenberg AFB
Vandenberg Air Force Base
Vandenberg Space Force Base
Vandenberg State Marine Reserve
La Purisima Mission State Historic Park
Lompoc Airport
Lompoc
Los Alamos
Los Olivos
Buellton
Solvang
CA 135
CA 1
CA 246
US 101
10

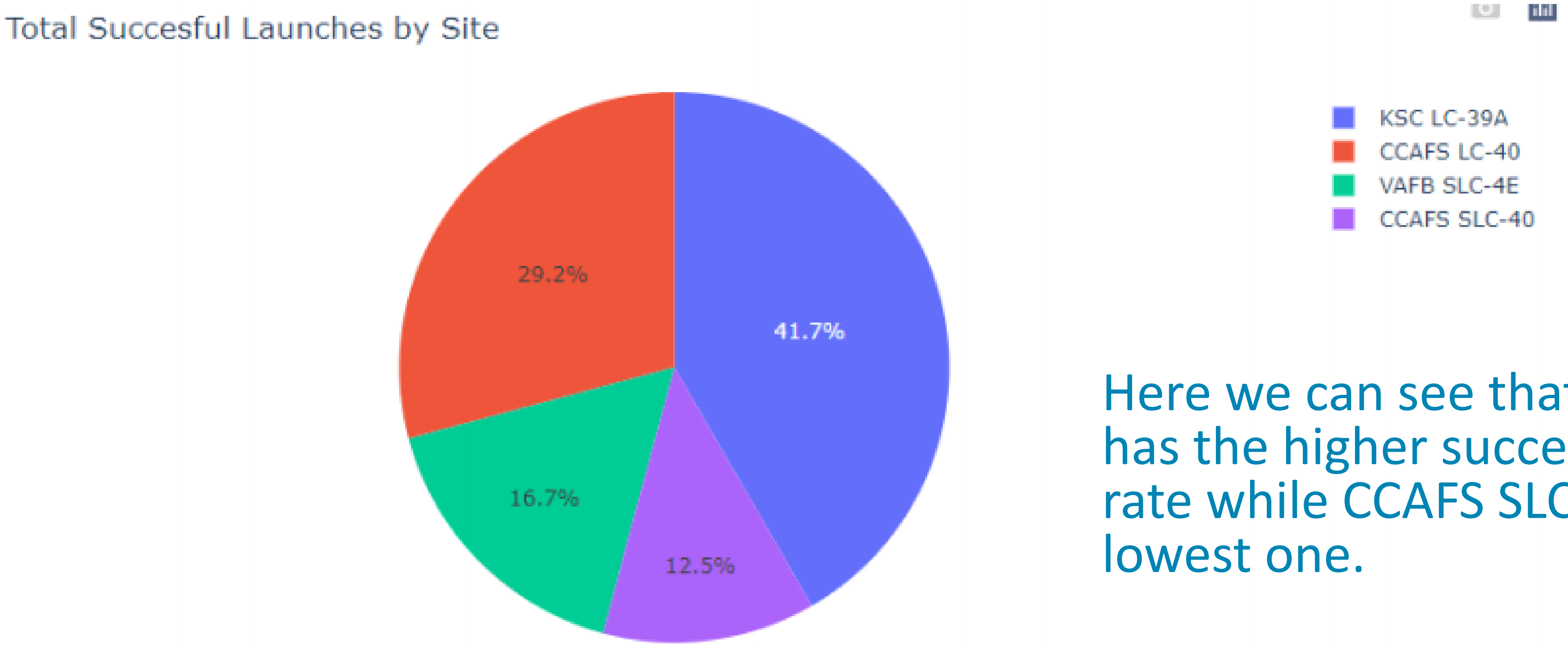
On this map we can see Cabrillo Highway as a blue info marker with a popup showing the distance between the highway and VAFB SLC-4E launch site which is 15.2682km.

There's also a green Polyline between the highway and VAFB SLC-4E launch site.

Build a Dashboard with Plotly Dash

Successful Launches by Launch Site

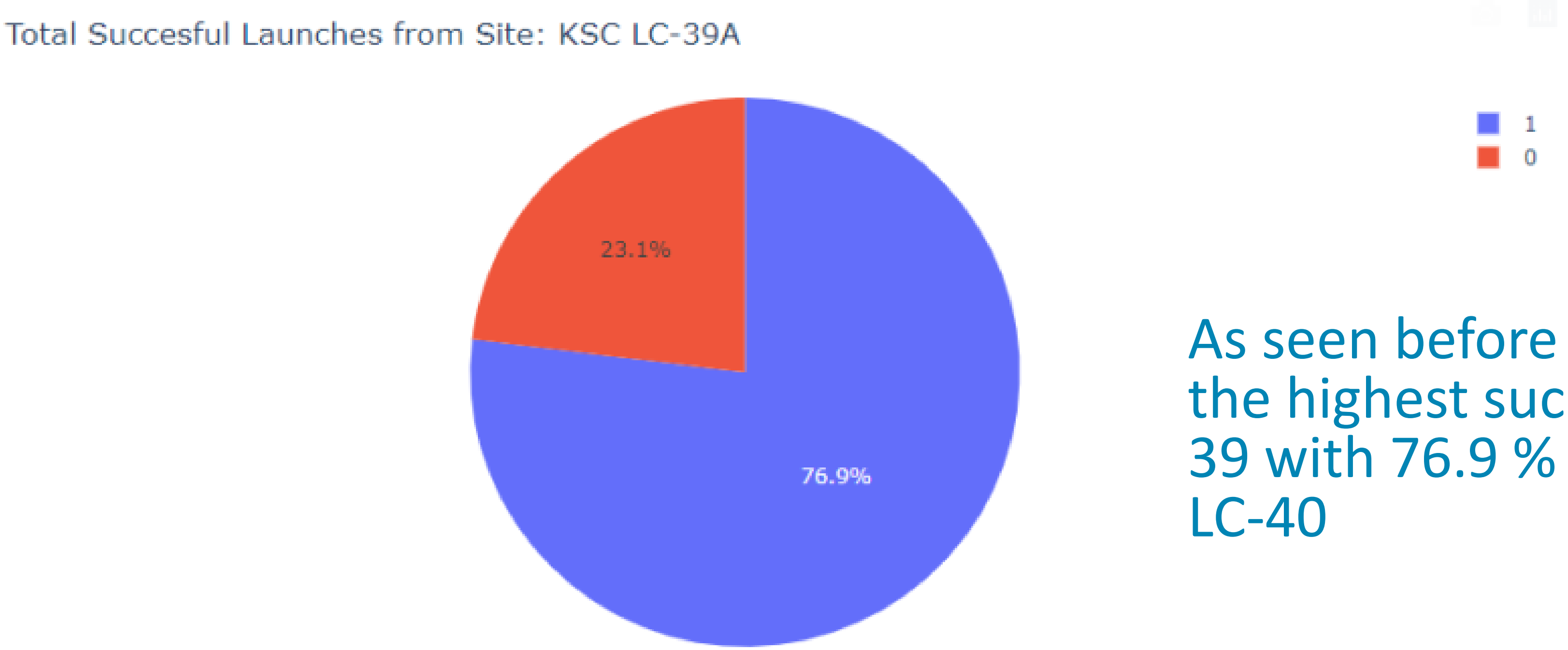
- Replace <Dashboard screenshot 1> title with an appropriate title
- Show the screenshot of launch success count for all sites, in a piechart
- Explain the important elements and findings on the screenshot



Here we can see that KSC LC-39A has the higher successful launches rate while CCAFS SLC-40 has the lowest one.

Launch Site with Highest Success Rate

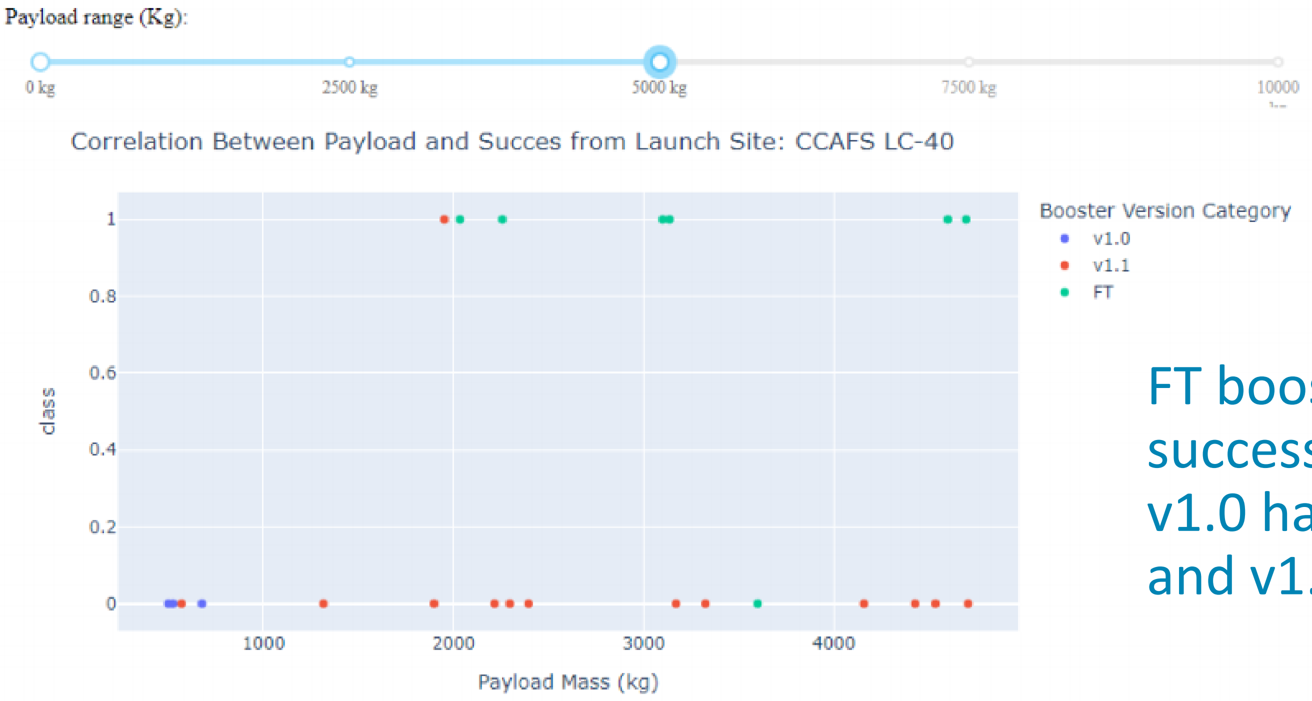
- Replace <Dashboard screenshot 2> title with an appropriate title
- Show the screenshot of the piechart for the launch site with highest launch success ratio
- Explain the important elements and findings on the screenshot



As seen before the launch site with the highest success rate is KSC LC-39 with 76.9 % followed by CCAFS LC-40

Boosters Succes Rates with Different Payloads

- 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



FT booster version has the highest success rate in this payload range while v1.0 has not a single successful launch and v1.1 only 1 out of 13

Predictive analysis (Classification)

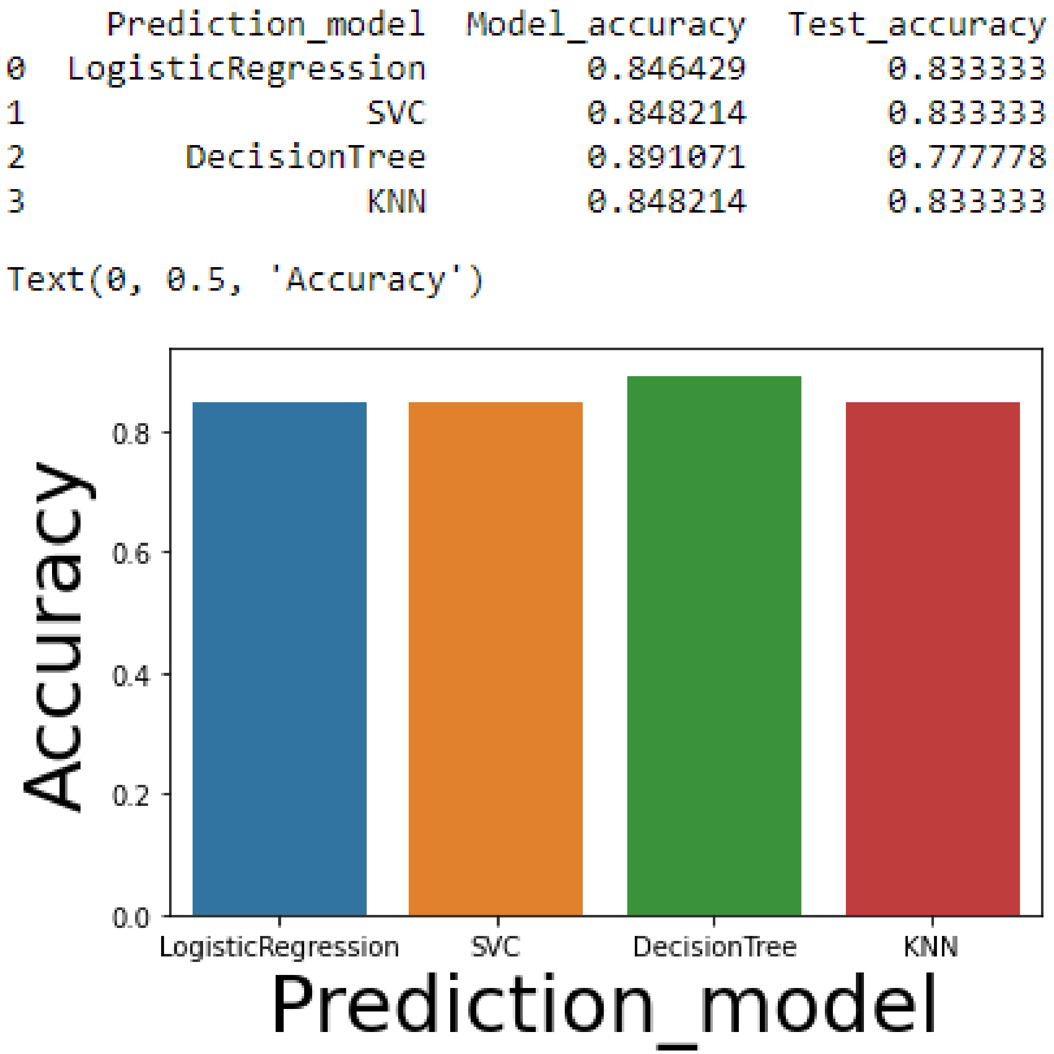
Classification Accuracy

- Visualize all the built model accuracy for all built models, in a barchart
- Find which model has the highest classification accuracy

As we can see in this barchart the model with a higher accuracy on training data is DecisionTree.

Also, it looks that DecisionTree model works worse with unseen data.

Lastly, we can say that other 3 models LogisticRegression, SVC (Support Vector Classifier) and KNN (K-Nearest Neighbor) have the highest classification accuracy. Mathematical operations are show on the next slideshow



```
      Prediction_model  Model_accuracy  Test_accuracy
0  LogisticRegression        0.846429       0.833333
1                 SVC        0.848214       0.833333
2        DecisionTree        0.891071       0.777778
3                 KNN        0.848214       0.833333

Text(0, 0.5, 'Accuracy')
```

Confusion Matrix

Show the confusion matrix of the best performing model with explanation

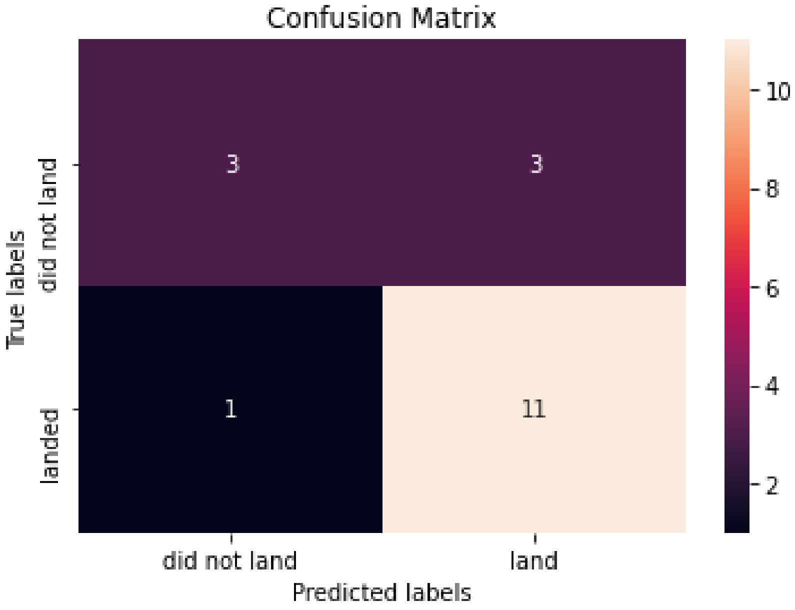
First image shows the confusion matrix of the model with highest training accuracy (DecisionTree).

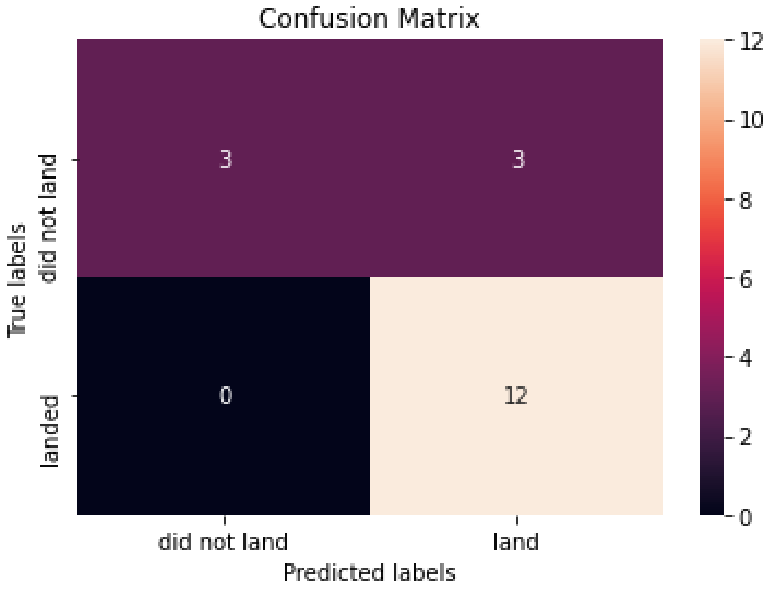
Second image shows the confusion matrix of the rest of the models LogisticRegression, SVC (Support Vector Classifier) and KNN (K-Nearest Neighbor).

$$(TP+TN)/(TP+TN+FP+FN).$$

14/18 = 0.7777 DecisionTree

15/18 = 0.83333 Rest of models





CONCLUSION

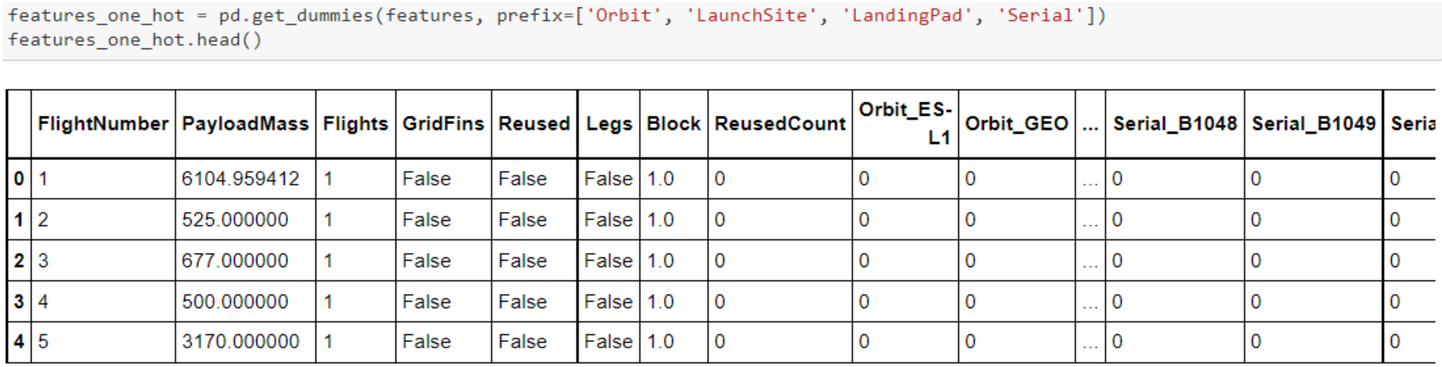
- Now we understand many of the reasons why SpaceX has the highest success rate saving that much money during the process.
- SpaceY will be able to reasonably accurately predict if future SpaceX launches will land the first stage of the rocket and implement it on future projects.
- Machine learning classification models have similar accuracies even though they were high.

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

Some of the features engineering used in the EDA with Visualization lab where some important data manipulation is made like the creation of dummy variables to categorical columns

applying OneHotEncoder or the casting of numeric columns to float64 which more suitable dtype in the case.



```
features_one_hot = pd.get_dummies(features, prefix=['Orbit', 'LaunchSite', 'LandingPad', 'Serial'])
features_one_hot.head()
```

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial_B1049	Seria
0	1	6104.959412	1	False	False	False	1.0	0	0	0	...	0	0	0
1	2	525.000000	1	False	False	False	1.0	0	0	0	...	0	0	0
2	3	677.000000	1	False	False	False	1.0	0	0	0	...	0	0	0
3	4	500.000000	1	False	False	False	1.0	0	0	0	...	0	0	0
4	5	3170.000000	1	False	False	False	1.0	0	0	0	...	0	0	0