IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Dmitrii Khrameshin
02/08/2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection
    - Using Resources APIs
    - Using Web Scrapping
  - Data Wrangling
    - Dealing with empty values
    - Dummy-features encoding
  - Data Analysis
    - Exploratory Analysis
    - Predictive (ML + Statistical) Analysis
  - Visualization
    - Folium interactive maps
    - Plotly dashboard
- Summary of all results
  - Prediction of launch success/failure
  - Screenshots of Results and dashboard on slides

Introduction

- Project background and context

  Falcon 9 rockets of Space X company significantly lowered the cost of satellites launches into a space. Business advantage is more than twice. The price has been lowered from $165 mil. to $62 mil. Most part of price reduction based on reuse of 1st stage of spacecraft. Thus it depends on success of landing of said 1st stage. The rocket company Space Y wants to compete with Space X on this field. It need to determine their future launch price based on predicted success rate of planned launches. The goal of this project is to predict if the 1st stage would land successfully, using collected data of Space X historical launches, with the help of Statistical analysis and Machine Learning algorithms.

- Problems you want to find answers
  - Explore which data related to space launches could be collected.
  - Determine features that deal most part in the success rate of returning of the 1st stage
  - Make an interactive tool to predict possibility of success based on most impacting factors
  - Present findings

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected with python scripts from Space X web site (public API) and Web Scrapping a page with historical data from Wikipedia.
- Perform data wrangling
  - Deal with missing data; Hot-encoding text features to numerical values; Normalization.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Grid search on Regression, Support Vector machines, Decision Tree Classifier, and K-nearest neighbors models to determine best model for future predictions.

Data Collection

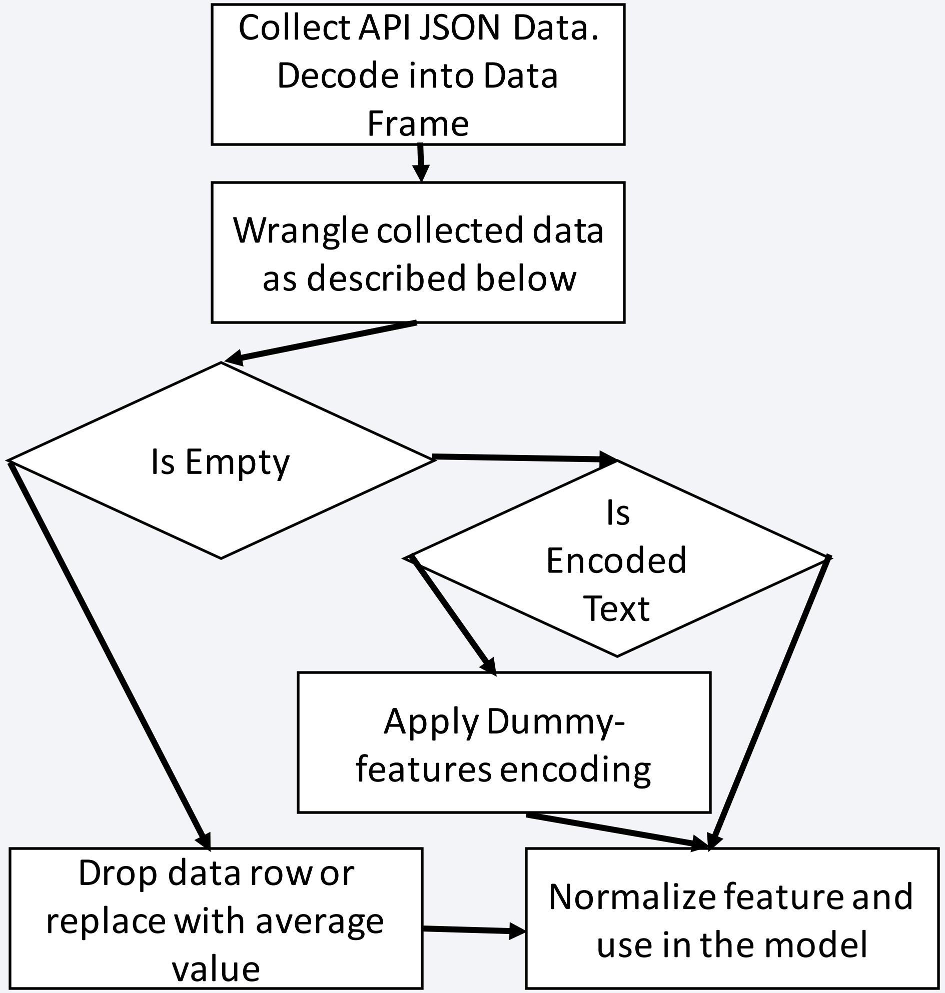
- Data was collecting using Space X site public REST API and Web Scrapping of Wikipedia page.
- Get method of the 'requests' library was used to retrieve JSON Data of the Space X web site.
- Scrapping of Falcon 9 launches data was performed with 'BeautifulSoup' library on Wikipedia page data.
- Collected data was converted into 'DataFrame' object, filtered and wrangled to standardize and normalize values.
- Prepared data was later used in Data Analysis for Machine Learning and building prediction model.

Data Collection – SpaceX API

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

- Notebook for data collection from API:

https://github.com/Dmvkh/testrepo/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

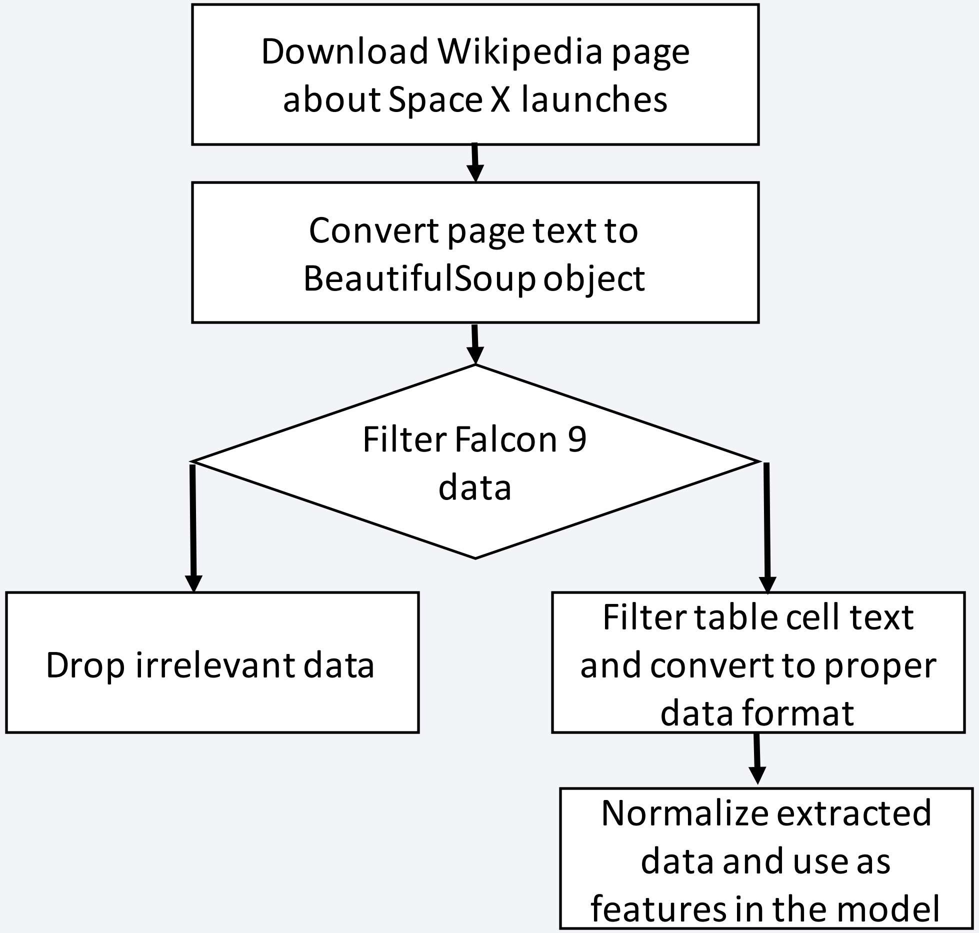

Collect API JSON Data. Decode into Data Frame
Wrangle collected data as described below
Is Empty
Is Encoded Text
Apply Dummy-features encoding
Drop data row or replace with average value
Normalize feature and use in the model

Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts

- Notebook for data collection from Web Scrapping:

- https://github.com/Dmvkh/testrepo/blob/main/jupyter-labs-webscraping.ipynb



Download Wikipedia page about Space X launches
Convert page text to BeautifulSoup object
Filter Falcon 9 data
Drop irrelevant data
Filter table cell text and convert to proper data format
Normalize extracted data and use as features in the model

Data Wrangling

- Collected Tata was transformed into Data Frames.
- Empty data was either completely dropped from resulting set, or replaced with average values using mean() function.
- Text encoded data was transformed into numeric variables using Hot dummy encoder.
- Resulting set was normalized to reduce data noise impact on model resilience.
- Collected features were grouped by launch site and destination orbit.
- Those were later used in the Visualization for Exploratory data analysis.

Link for Data Wrangling notebook:

https://github.com/Dmvkh/testrepo/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- Exploratory Data Analysis was performed of prepared data. The purpose of this analysis was to reveal relationship between such features as 'Lauch Site', 'Flight Number', 'Destination Orbit', 'Rocket Mass' and determine success rate for each factor combination.
- Link for performed EDA with data visualization:

https://github.com/Dmvkh/testrepo/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- Collected Data was loaded int SQLite Data Base file in the table named 'SPACEXTABLE'.
- Using means of SQL Language several queries were performed to reveal this information:
  - Names of launch sites
  - Average payload mass for Falcon 9
  - Dates of landing with successful outcome
  - Relations between boosters, payload mass and success rate
  - Max. Payload mass for each booster version
  - Date related information of successful launches (returns of 1st stage)

Link to the notebook:

https://github.com/Dmvkh/testrepo/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Map every Lauch Site was marked with its name on correspondent location.
- For each Lauch Site was added a group of markers (circle ring) indicating launches performed on this site and colored in regards with was the launch successful or not.
- Clustered markers allow to zoom in and see those launches in more convenient way.
- Distances between launch site and nearest railroad, coast line, town and highway were calculated.
- Also connecting lines were added between the Launch Site and some points of interest.

This allowed to conclude that launch sites are prone to be located near the shores, and has connections with rail roads.

Also said sites are built at rather distanced areas from big cities.

Link to the notebook with folium map:

https://github.com/Dmvkh/testrepo/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- Active Plotly dashboard was built for the relevand data about Space X rocket launches.
- This shows charts of successful/unsuccessful launches regarding each site or summary for all of them. Also scatter plot, showing type of space craft, its launch No and varying payload mass slider were added for said sites.
- Those charts/plots allow to see relationships between bonded parameters and success rate of each mission for each launch site.

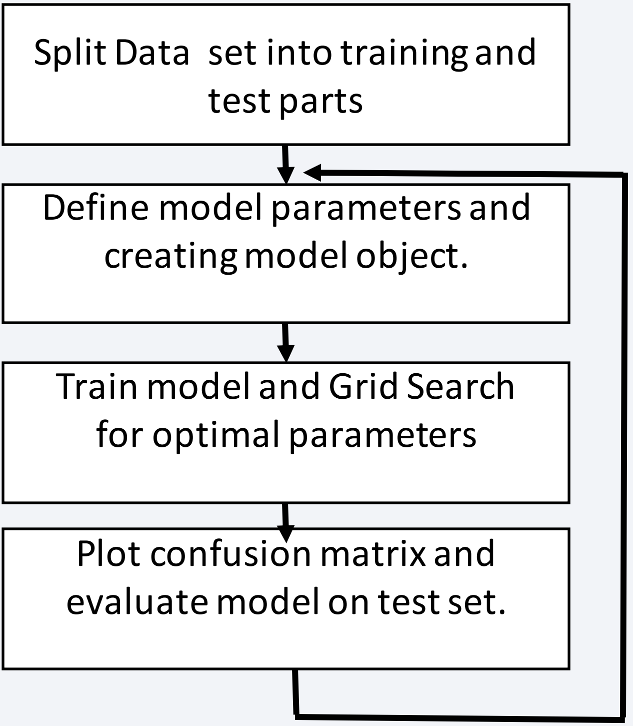
Links to dashboard python file and saved dashboard:

https://github.com/Dmvkh/testrepo/blob/main/Dash.pdf

https://github.com/Dmvkh/testrepo/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Collected data was processed using python libraries, like pands and numpy. Library sklearn was used to build machine learning models, for Grid Search optimum values and for testing of those algorithms:
  - Logistic Regression
  - Support Vector Machine
  - Decision Tree
  - K-nearest Neighbors
- Seaborn library was used to plot confusion matrices for each model.
- Grid Search estimator scored predictions on test part of dataset for each model.



Split Data set into training and test parts

Define model parameters and creating model object.

Train model and Grid Search for optimal parameters

Plot confusion matrix and evaluate model on test set.

Link to related notebook:

https://github.com/Dmvkh/testrepo/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

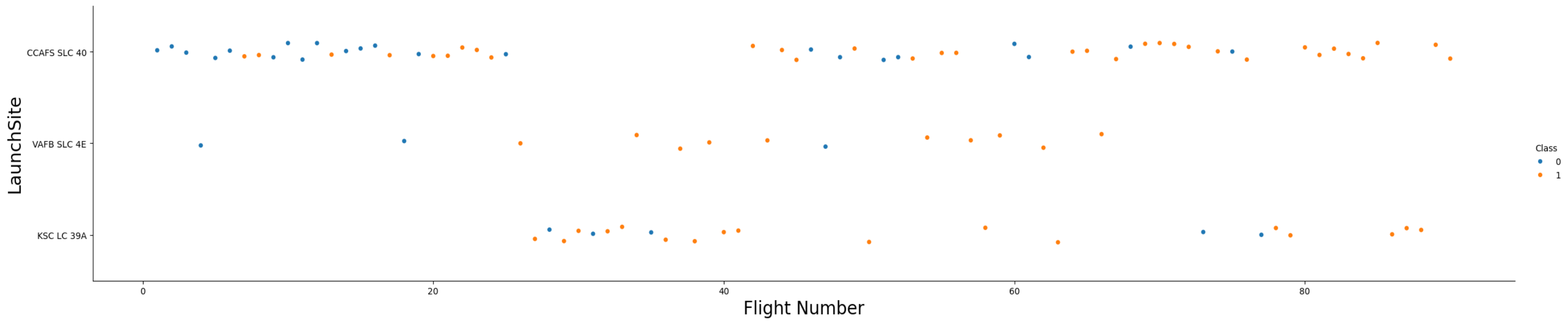
- Exploratory data analysis showed that success rate depends from Launch Site, Payload Mass, Destination Orbit and No of consecutive launch performed for this rocket type.
- Interactive analytics allowed to see detailed launch success rates for each site.
- Predictive analysis results allowed to make prediction for future launches with high confidence.

Section 2

Insights drawn from EDA

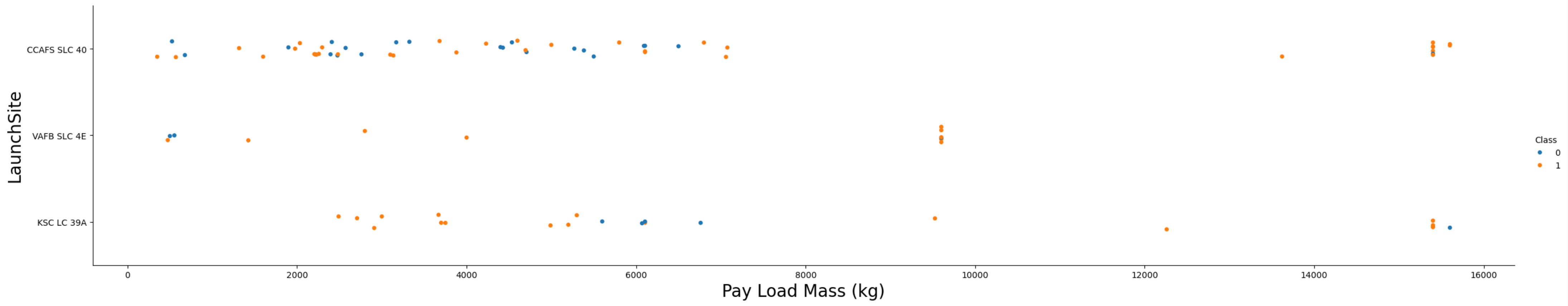
Flight Number vs. Launch Site

- We can see from this plot, that the more consecutive launches each site perform, the higher a success rate is.



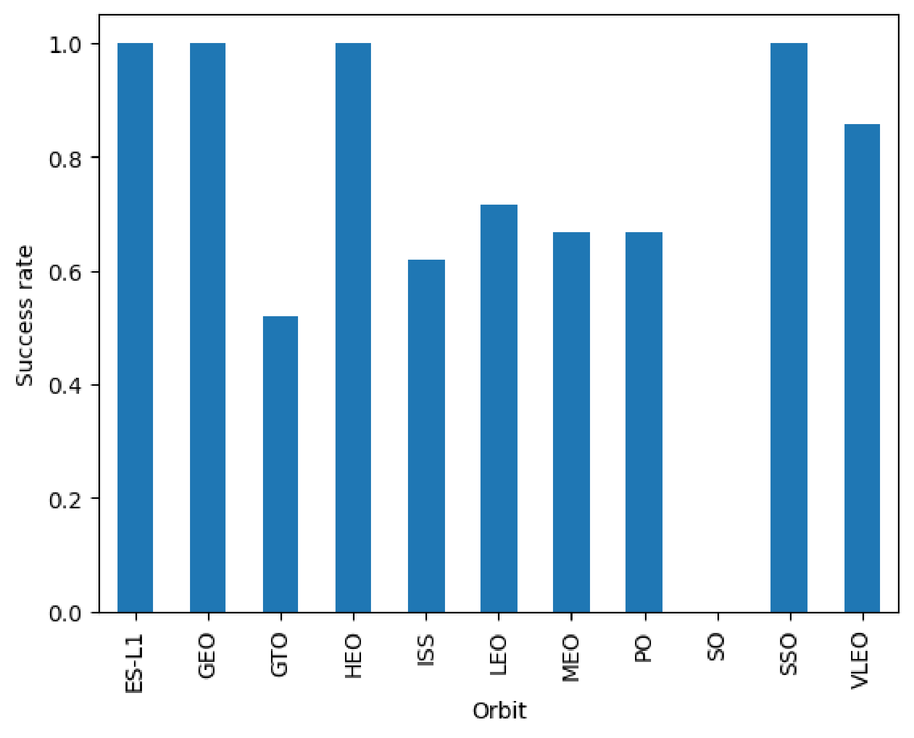
Payload vs. Launch Site

- This plot shows us that launch success has positive correlation with payload mass for most of the cases.



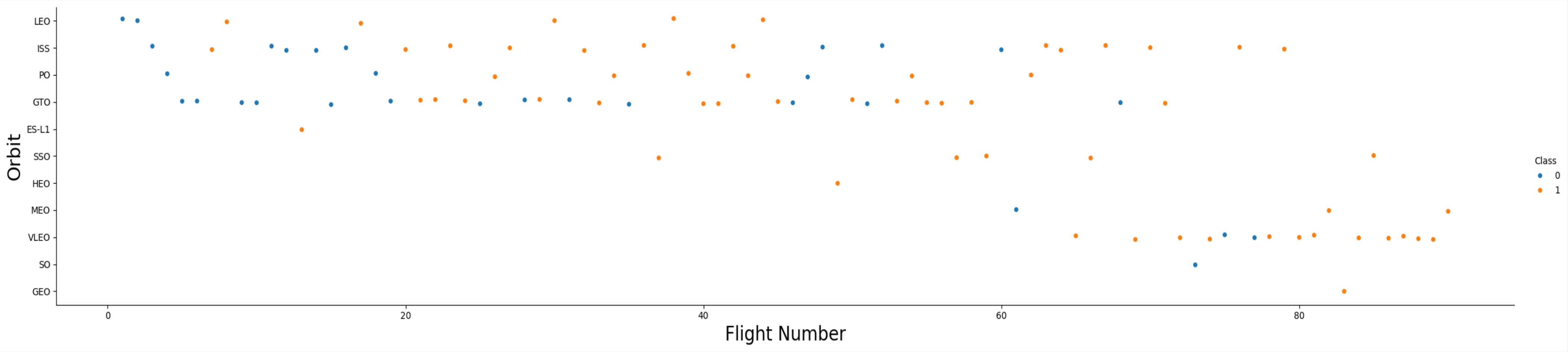
Success Rate vs. Orbit Type

- This bar chart demonstrates that success rate is highly dependent from destination orbit. GTO is most dangerous one, while VLEO, SSO, HEO, GEO and ES-L1 are most stable.



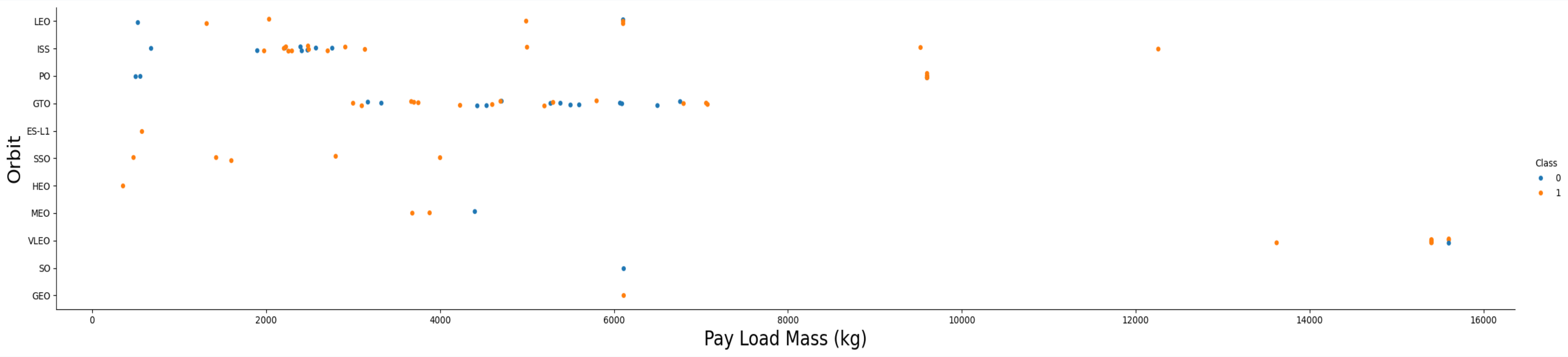
Flight Number vs. Orbit Type

- This plot shows us that success rate for most orbits (except, probably, GTO) also depends from number of performed launches to said orbit and is increasing with each flight.



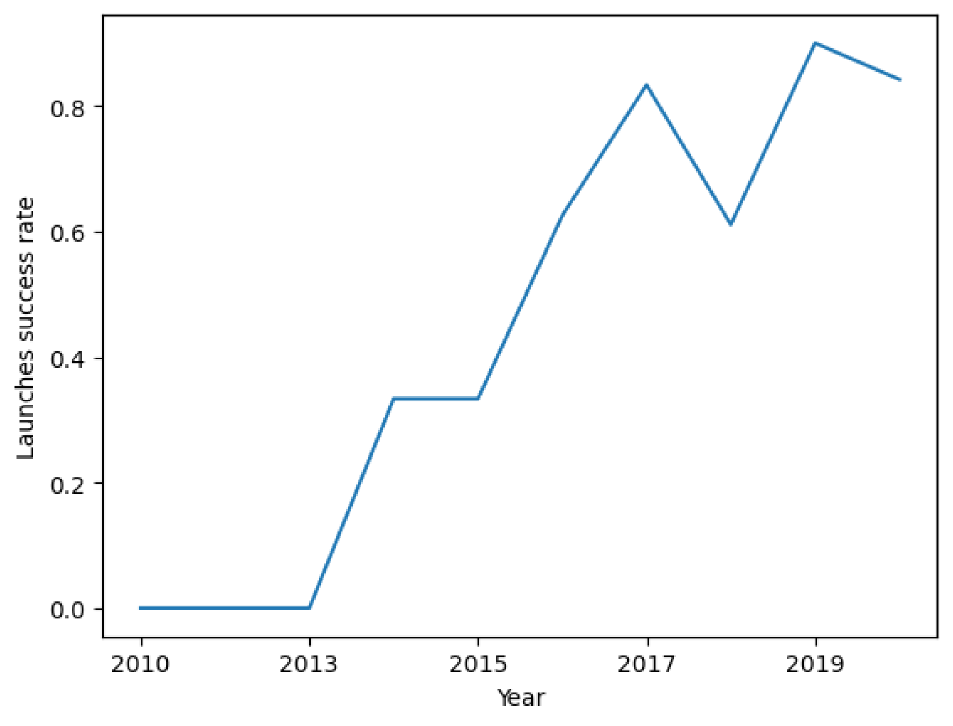
Payload vs. Orbit Type

- This plot shows us that increase of payload mass increases success rate for each orbit, except GTO where such relationship is not obvious.



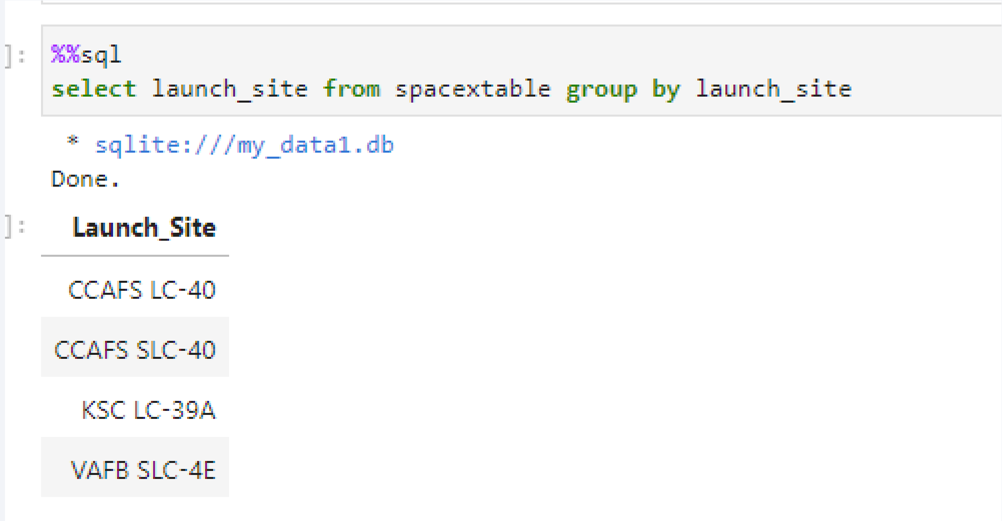
Launch Success Yearly Trend

- On this chart we can see that launches gradually increased successful outcome of missons since 2013, save slight decrease in 2018 and 2020



All Launch Site Names

- Grouping select query for desired table column allowed to receive distinct names of launch sites.



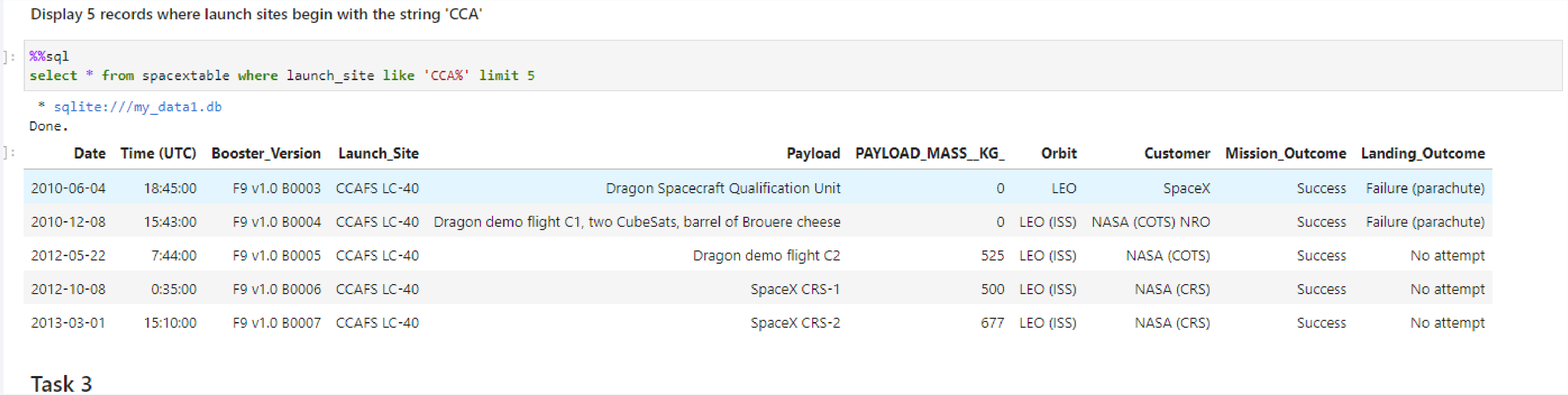
```
%%sql
select launch_site from spacextable group by launch_site
```

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

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

Launch Site Names Begin with 'CCA'

- After filtering select query, output was shortened using 'limit' keyword



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

```
]: %%sql
   select * from spacextable where launch_site like 'CCA%' limit 5
```

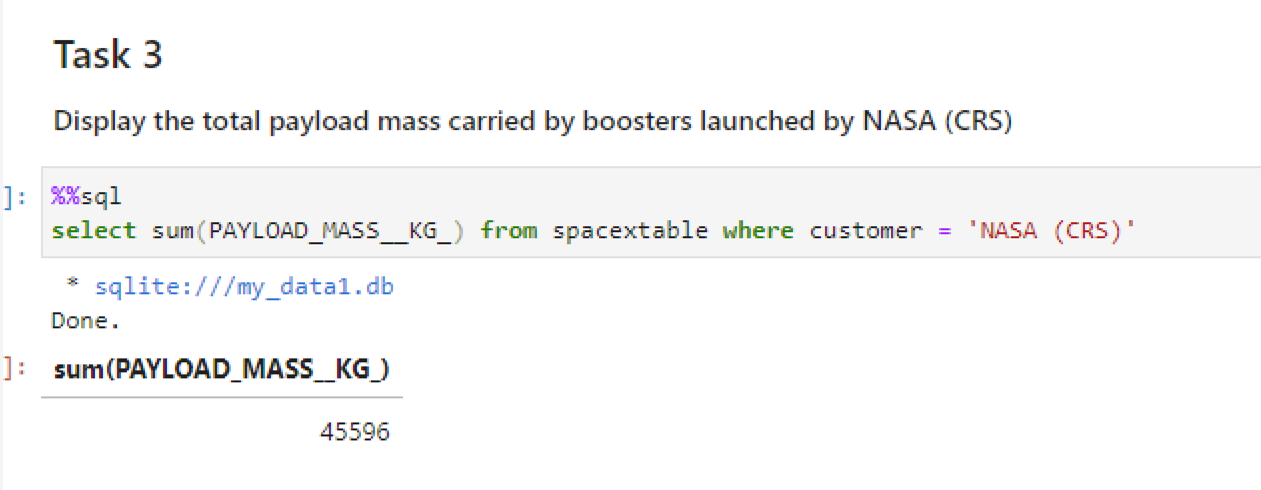
```
 * sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Task 3

Total Payload Mass

- Totl mass for NASA (CRS) customer was calculated using 'sum' function of SQL



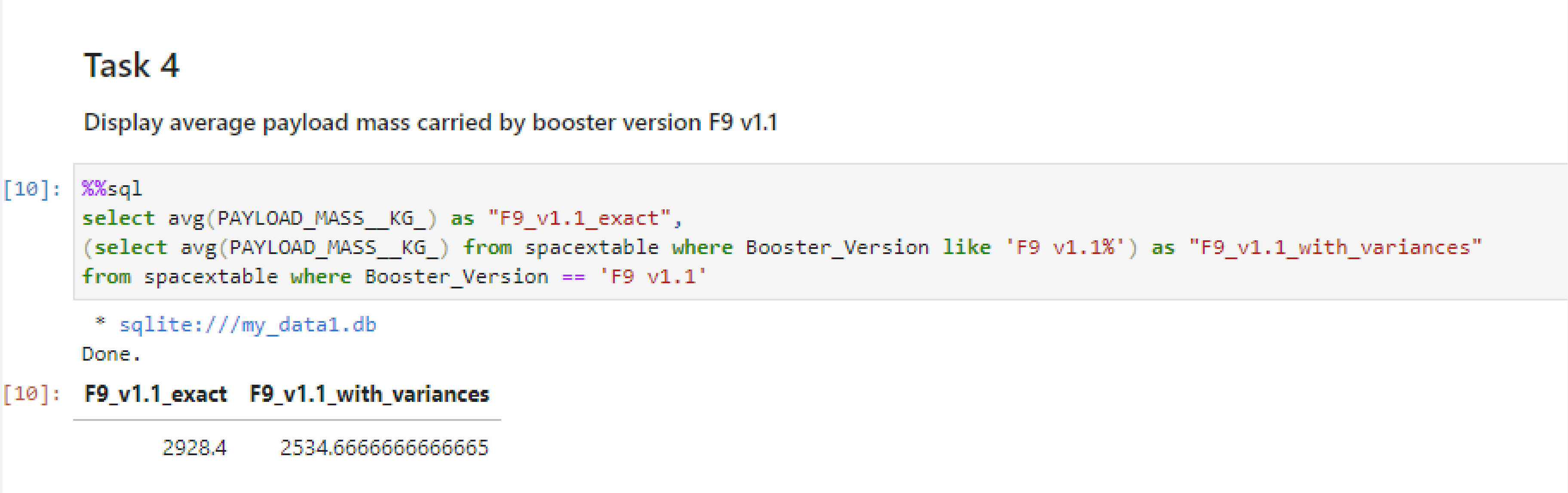
Task 3

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

```
]: %%sql
   select sum(PAYLOAD_MASS__KG_) from spacextable where customer = 'NASA (CRS)'

    * sqlite:///my_data1.db
   Done.

]: sum(PAYLOAD_MASS__KG_)
                     45596
```

Average Payload Mass by F9 v1.1

- Average payload mass was calculated for booster version F9 v1.1 (with and without variances)



Task 4

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

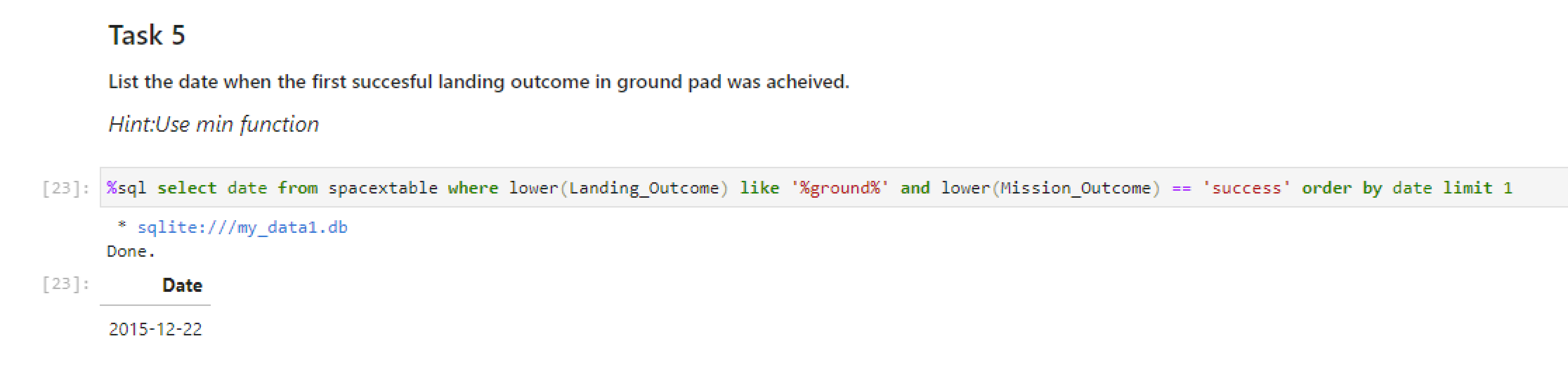
```
[10]: %%sql
      select avg(PAYLOAD_MASS__KG_) as "F9_v1.1_exact",
      (select avg(PAYLOAD_MASS__KG_) from spacextable where Booster_Version like 'F9 v1.1%') as "F9_v1.1_with_variances"
      from spacextable where Booster_Version == 'F9 v1.1'

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

[10]:

F9_v1.1_exact	F9_v1.1_with_variances
2928.4	2534.6666666666665

First Successful Ground Landing Date

- To find 1st date of successful landing was used filter by columns below with sorting by date



Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
[23]: %sql select date from spacextable where lower(Landing_Outcome) like '%ground%' and lower(Mission_Outcome) == 'success' order by date limit 1
```

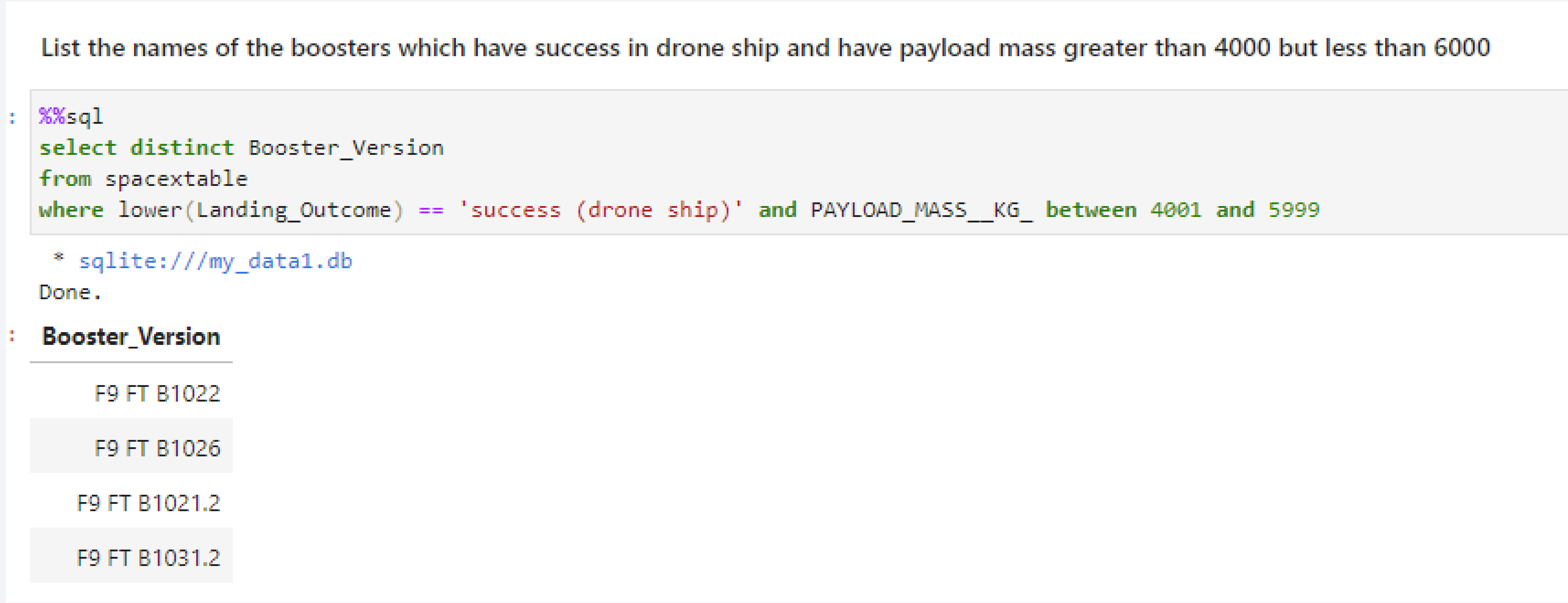
```
 * sqlite:///my_data1.db
Done.
```

[23]:

Date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- To list successful drone ships filtering was applied to corresponding columns with keyword 'distinct'



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

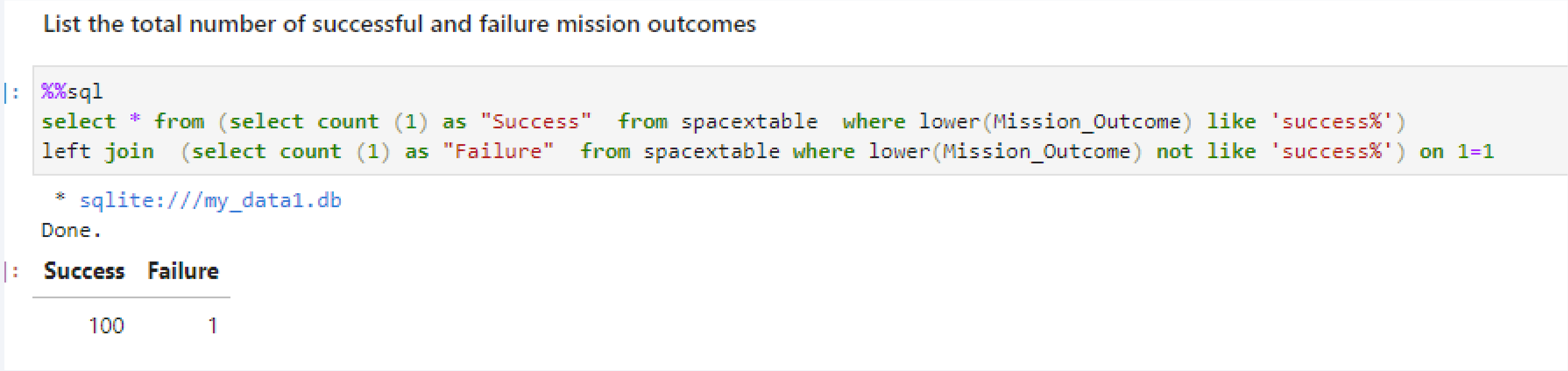
```
%%sql
select distinct Booster_Version
from spacextable
where lower(Landing_Outcome) == 'success (drone ship)' and PAYLOAD_MASS__KG_ between 4001 and 5999
```

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

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

Total Number of Successful and Failure Mission Outcomes

- To count successful and failed mission outcomes two corresponded subqueries were joined to get result as single row



List the total number of successful and failure mission outcomes

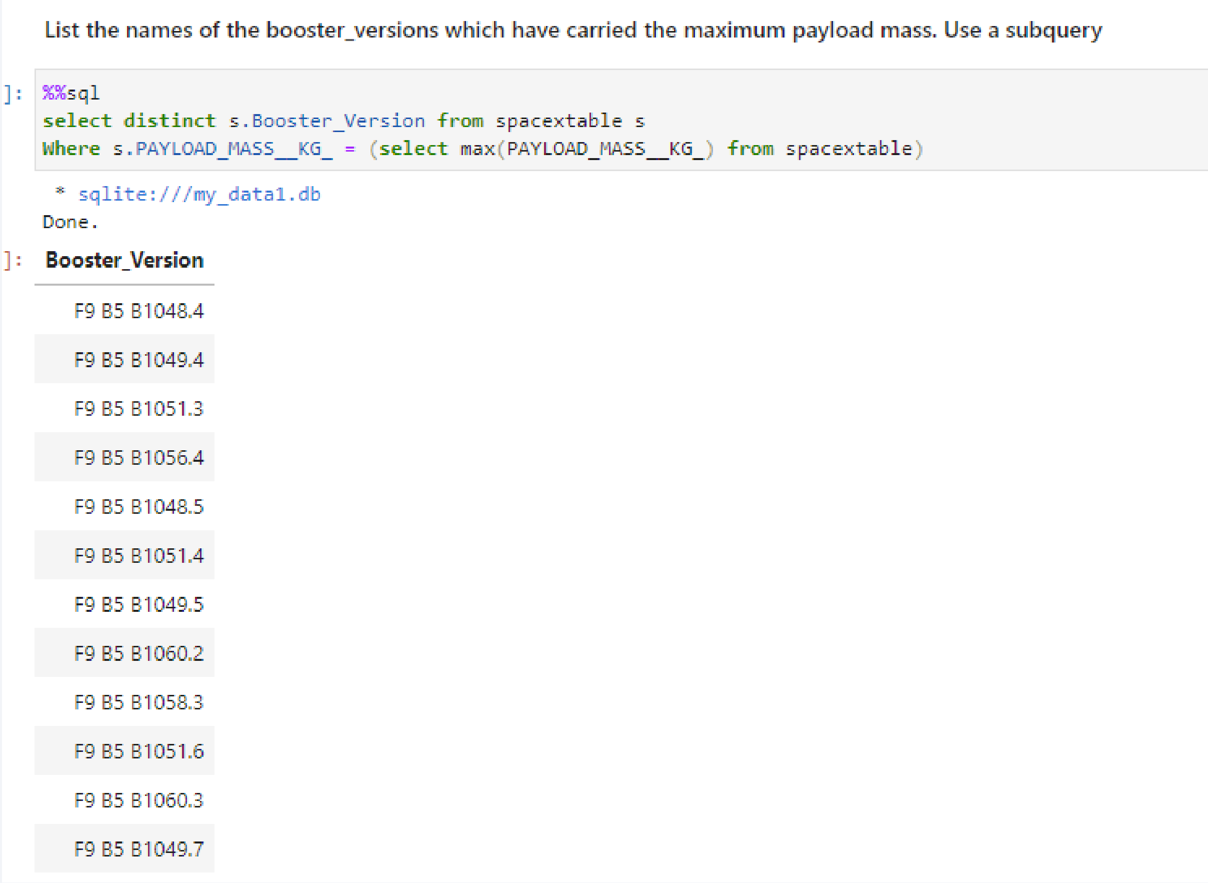
```
%%sql
select * from (select count (1) as "Success"  from spacextable  where lower(Mission_Outcome) like 'success%')
left join  (select count (1) as "Failure"  from spacextable where lower(Mission_Outcome) not like 'success%') on 1=1
```

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

Success	Failure
100	1

Boosters Carried Maximum Payload

- In this task subquery was used to list distinct boosters with maximum payload mass



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

```
]: %%sql
select distinct s.Booster_Version from spacextable s
Where s.PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacextable)
```

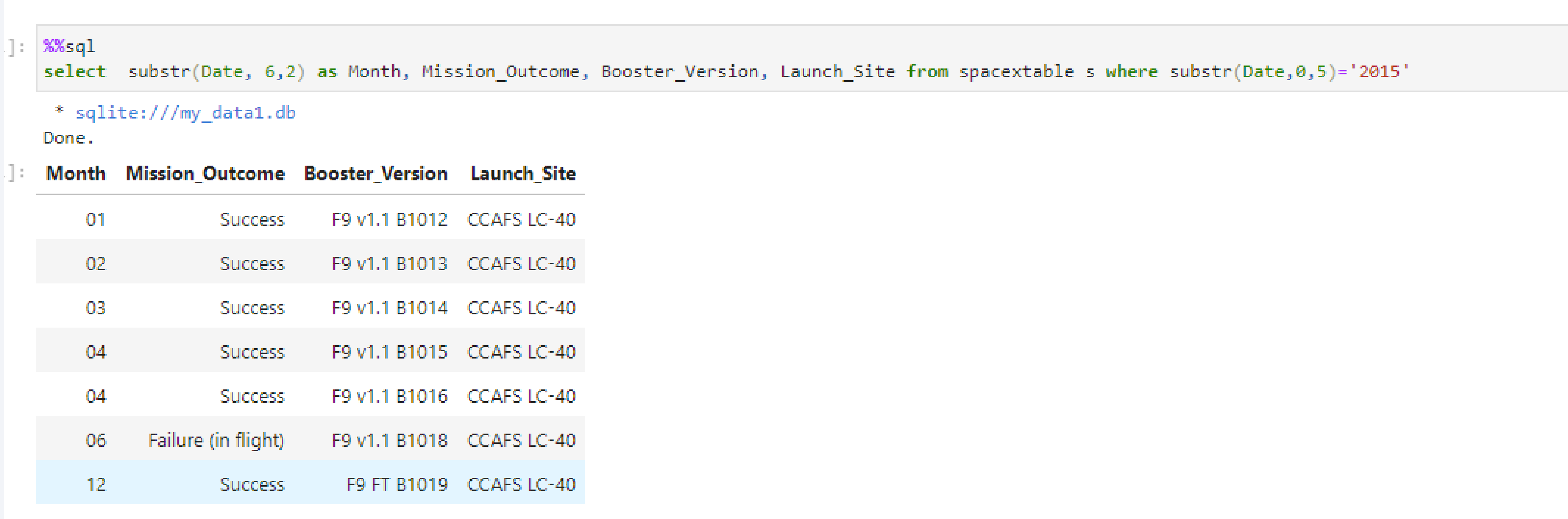
```
 * sqlite:///my_data1.db
Done.
```

]:

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

- This query returns table containing Month, Mission Outcome, Booster version and Launch site name for year 2015



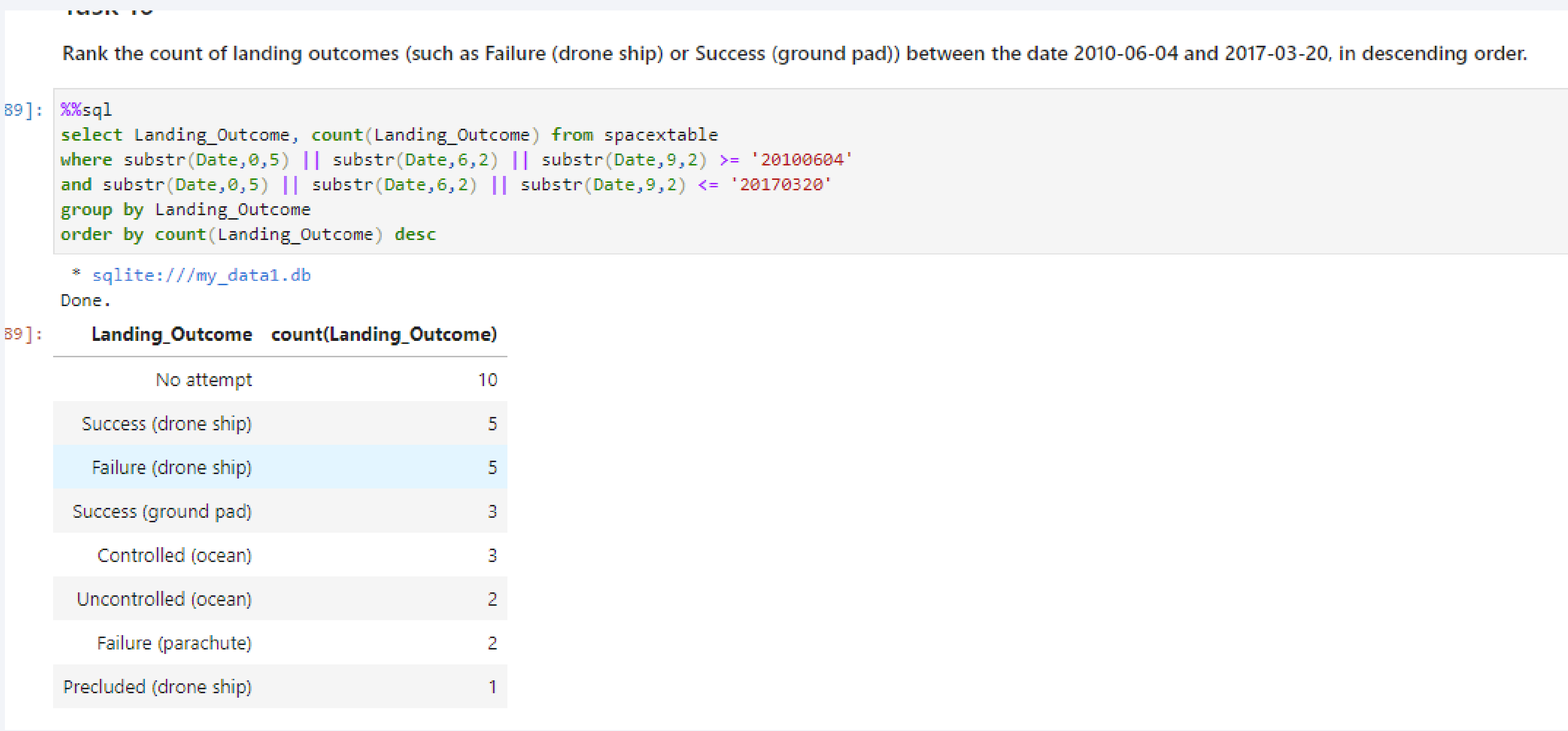
```
%%sql
select  substr(Date, 6,2) as Month, Mission_Outcome, Booster_Version, Launch_Site from spacextable s where substr(Date,0,5)='2015'
```

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

Month	Mission_Outcome	Booster_Version	Launch_Site
01	Success	F9 v1.1 B1012	CCAFS LC-40
02	Success	F9 v1.1 B1013	CCAFS LC-40
03	Success	F9 v1.1 B1014	CCAFS LC-40
04	Success	F9 v1.1 B1015	CCAFS LC-40
04	Success	F9 v1.1 B1016	CCAFS LC-40
06	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40
12	Success	F9 FT B1019	CCAFS LC-40

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

- To rank table with landing outcomes, grouping and ordering within set diapason (using concatenated strings) was applied



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.

89]:

```
%%sql
select Landing_Outcome, count(Landing_Outcome) from spacextable
where substr(Date,0,5) || substr(Date,6,2) || substr(Date,9,2) >= '20100604'
and substr(Date,0,5) || substr(Date,6,2) || substr(Date,9,2) <= '20170320'
group by Landing_Outcome
order by count(Landing_Outcome) desc
```

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

89]:

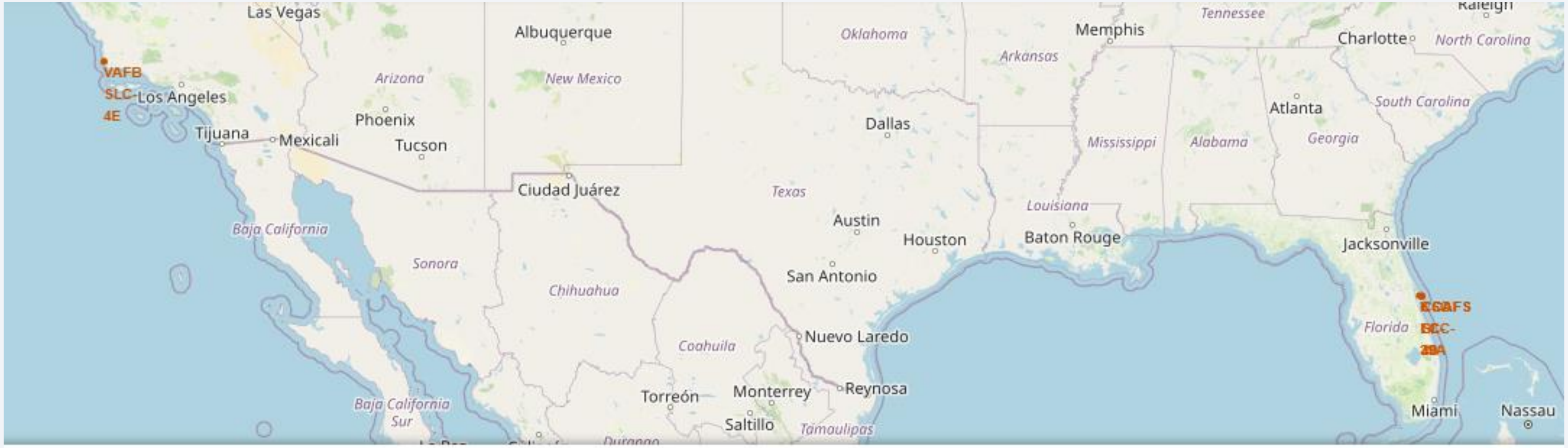
Landing_Outcome	count(Landing_Outcome)
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

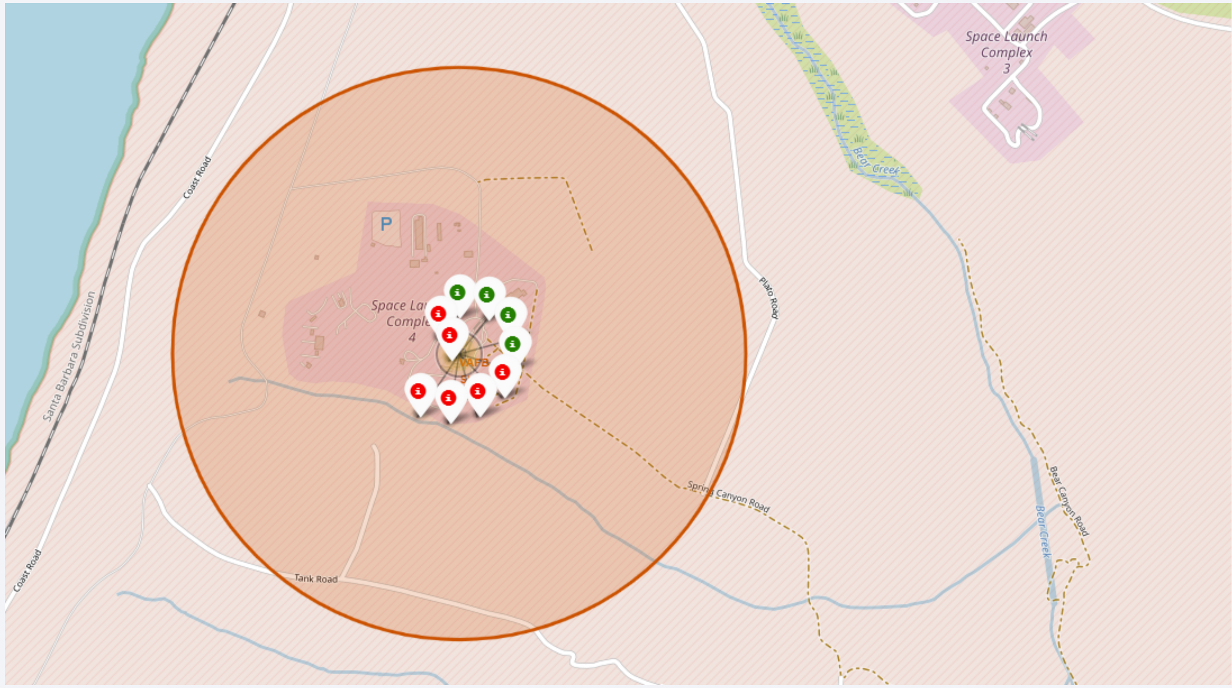
Lunch sites on the global map

- On the screen below we can see, that Space X launch sites are located on the East and West coasts of USA. Three in Florida and one in California.



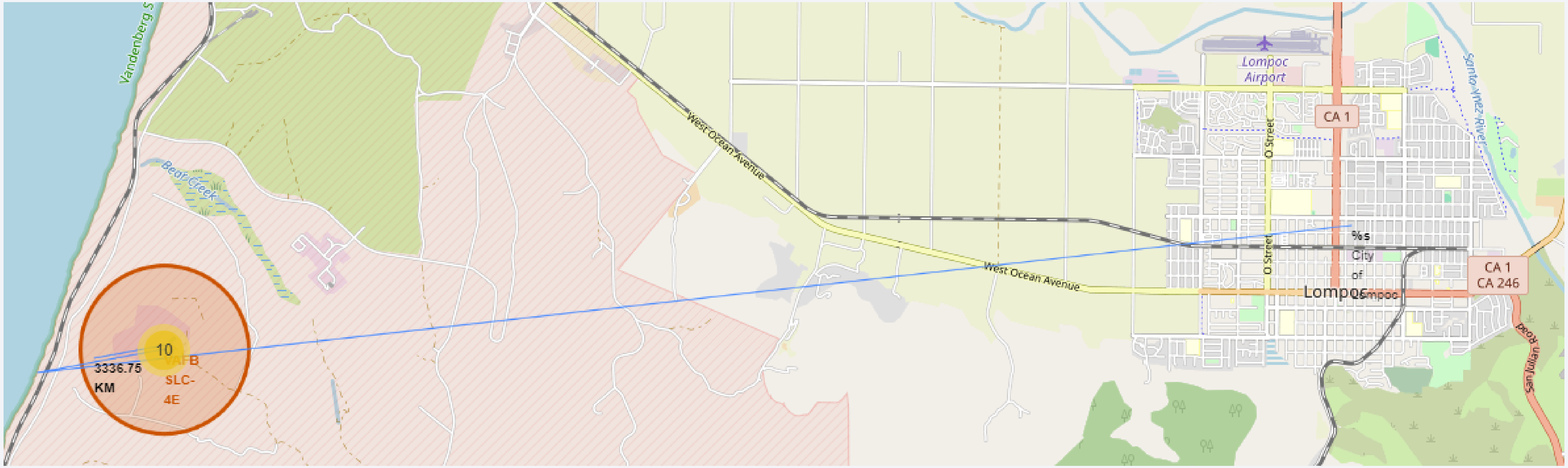
VAFB SLC-4E Launch Outcomes

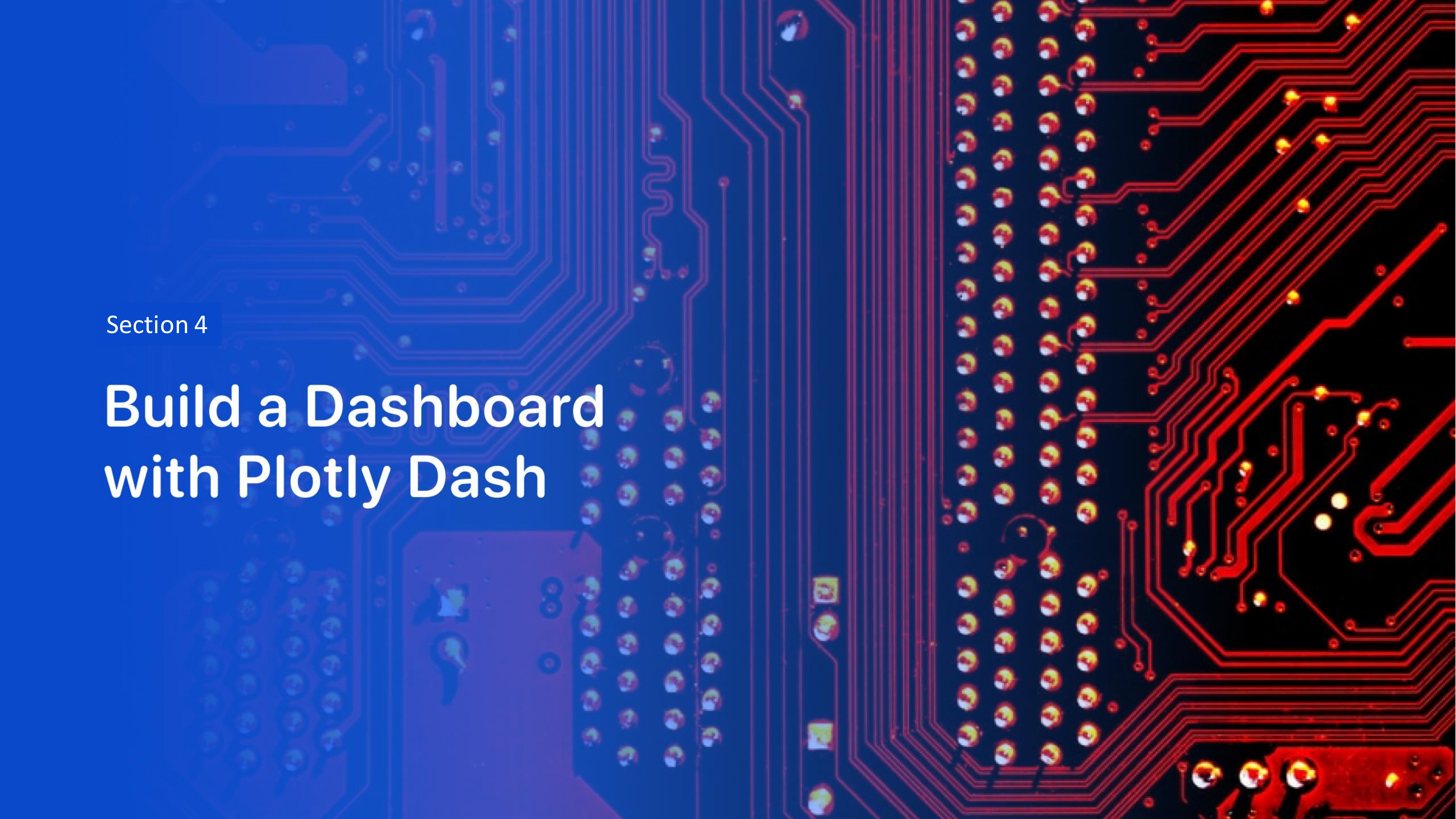
- This map shows launch outcomes performed at the VAFB SLC-4E Launch site.
- Successful launches are colored in green, failed – in red.



<Folium Map Screenshot 3>

- This screenshot shows California Launch Site's proximity to coast line, railroad and nearest town 'Lompoc'.



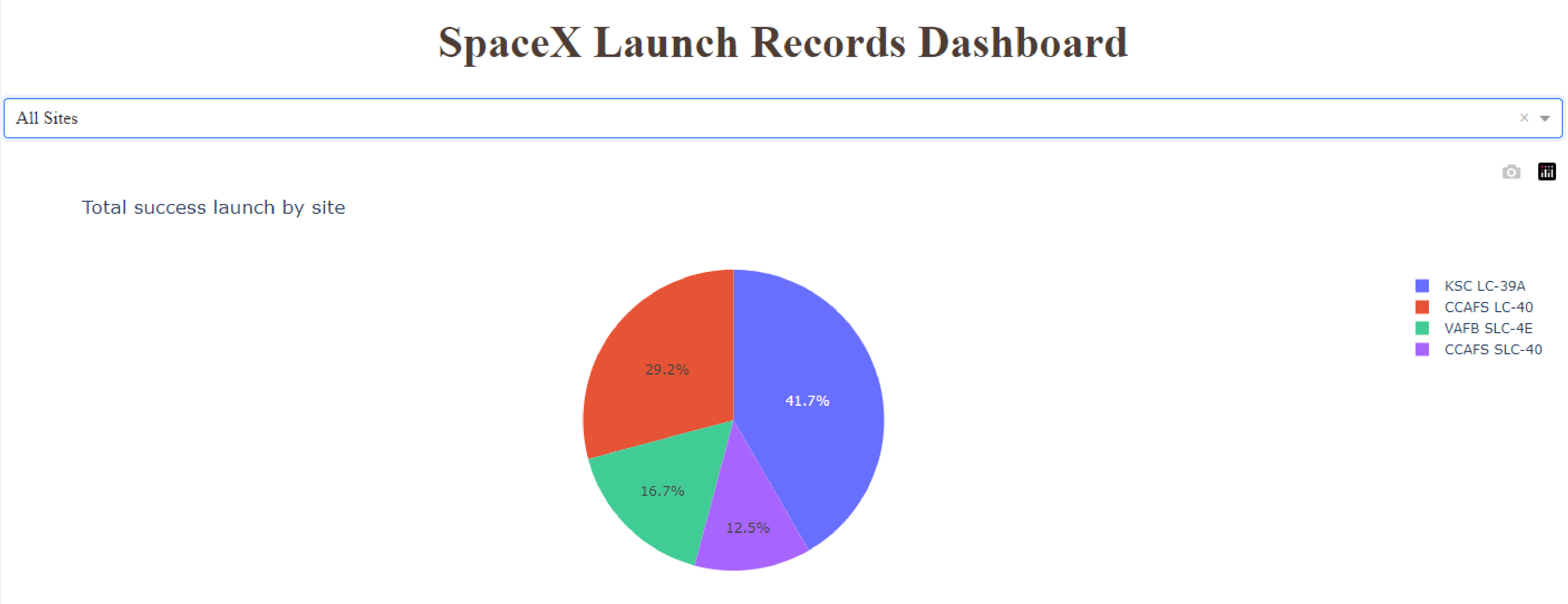


Section 4

Build a Dashboard with Plotly Dash

Launch success rates for All Sites pie chart

- This screenshot shows success rates to each of Space X launch sites.

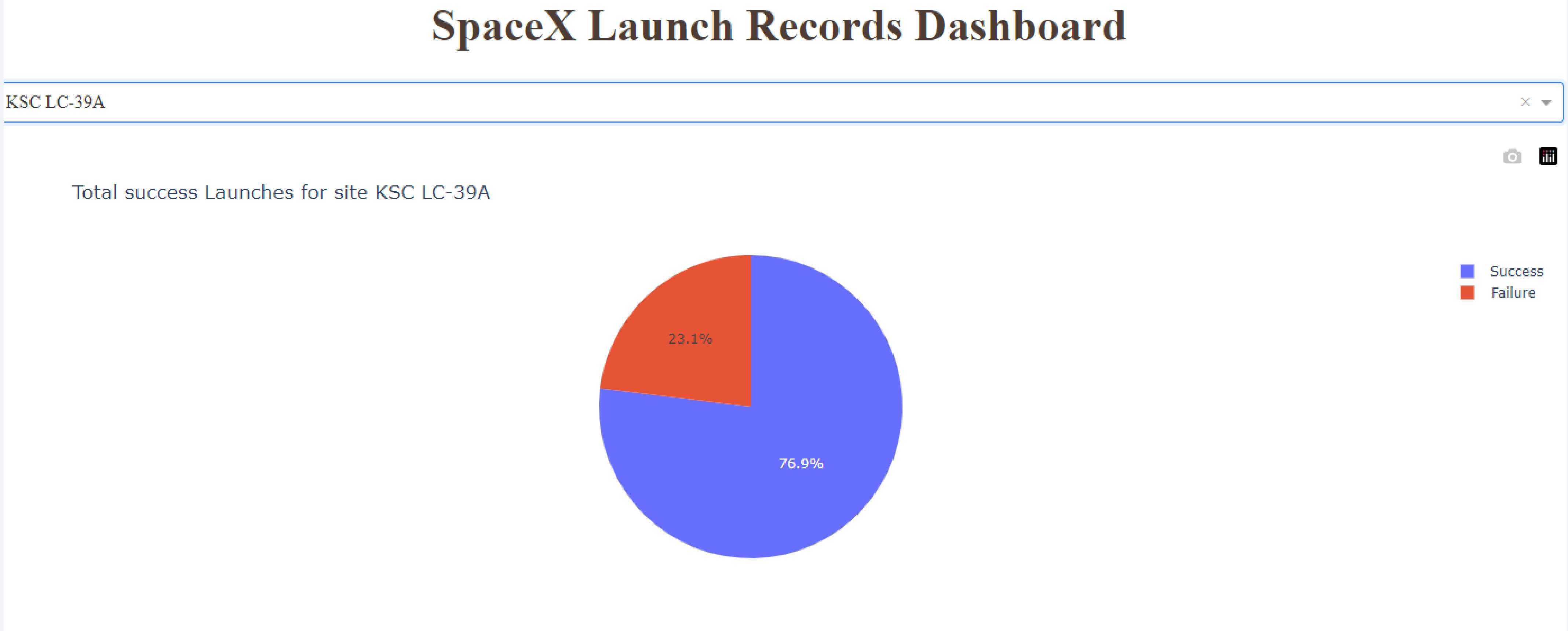


SpaceX Launch Records Dashboard

All Sites

Most successful Launch Site (by ratio)

- Site KSC LC-39A has the most successful ratio of launches



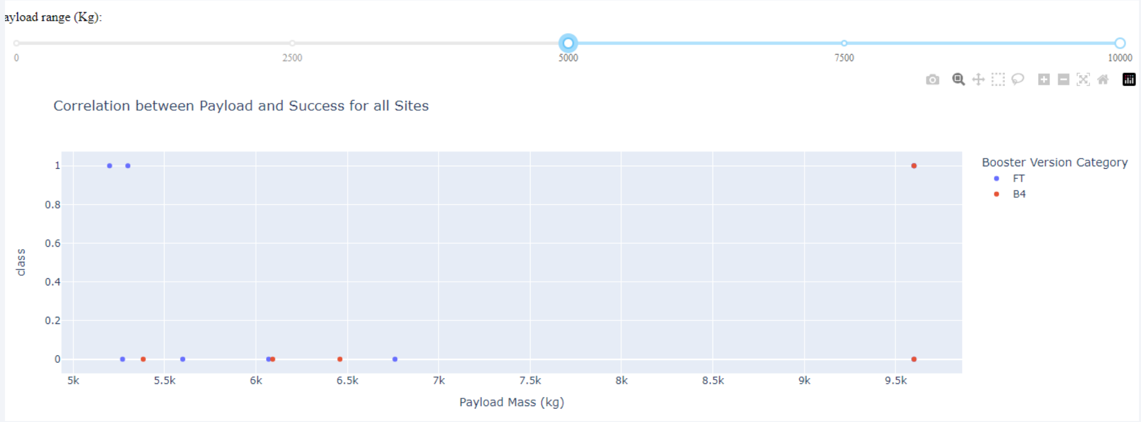
SpaceX Launch Records Dashboard

KSC LC-39A

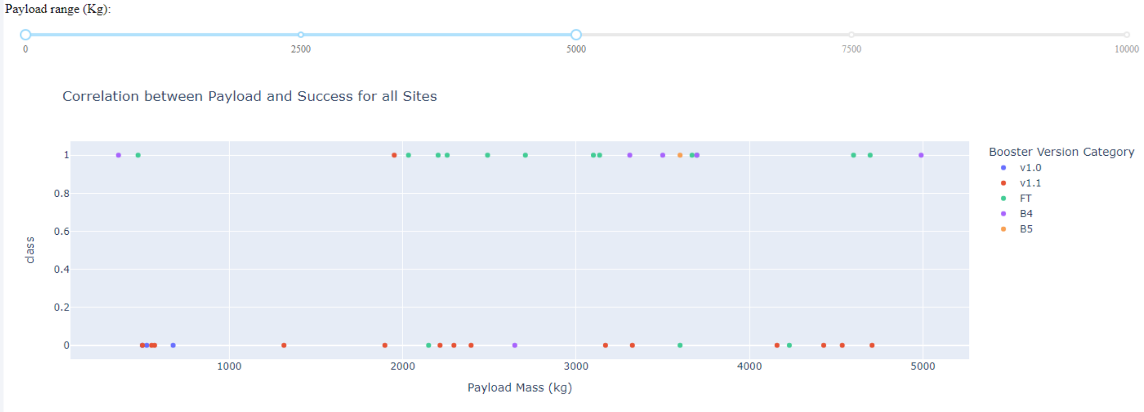
Total success Launches for site KSC LC-39A

Payload mass and launch success for all Sites

- This screenshot shows success rate for launches with high payload mass



- This screenshot shows success rate for launches with low payload mass



Section 5

Predictive Analysis (Classification)

Classification Accuracy

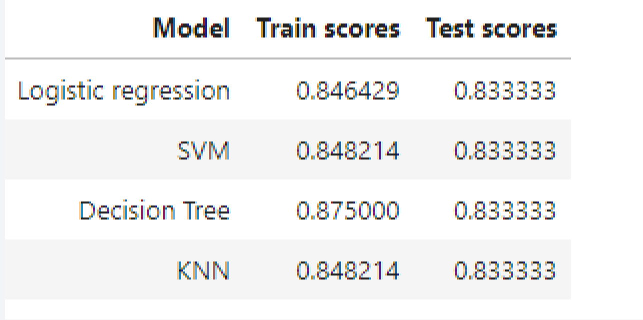
- Though Decision Tree model shown best accuracy on train data, all models shown same score results on test set.

- As all models show same accuracy,

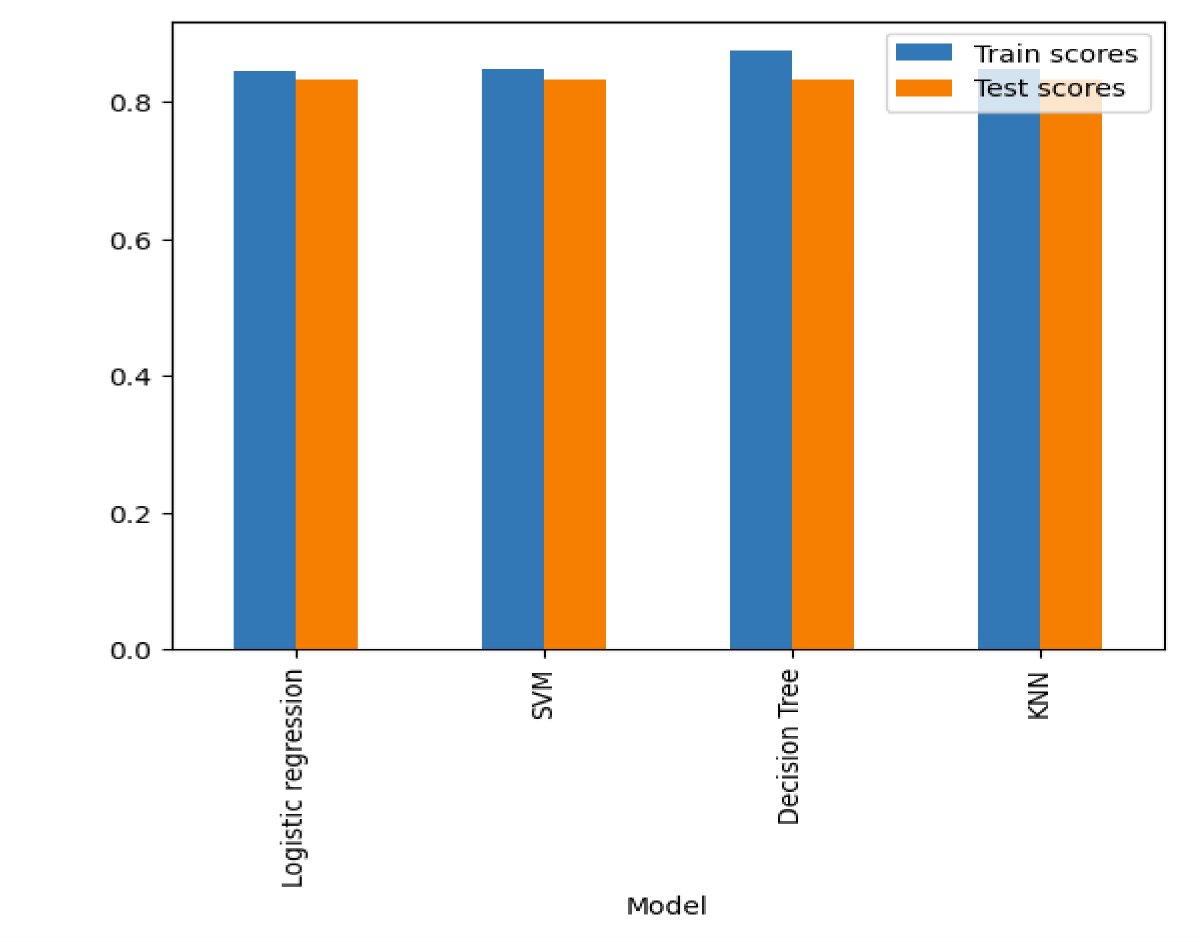
decision tree was selected, since

it has best scores by sum on test

and train data.

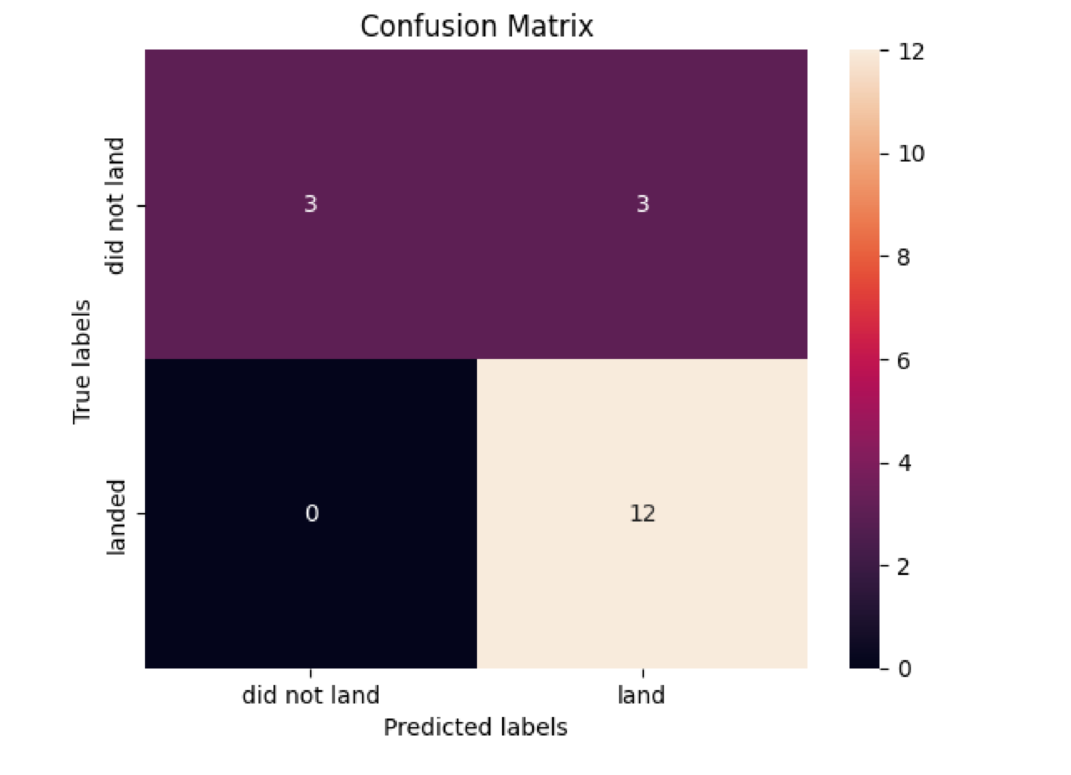


Model	Train scores	Test scores
Logistic regression	0.846429	0.833333
SVM	0.848214	0.833333
Decision Tree	0.875000	0.833333
KNN	0.848214	0.833333



Confusion Matrix

- Confusion matrix of Decision Tree model. This matrix shows that model has some problems with false-positive results, but is very resilient in regards of false-negative predictions (zero false-negatives on test data)

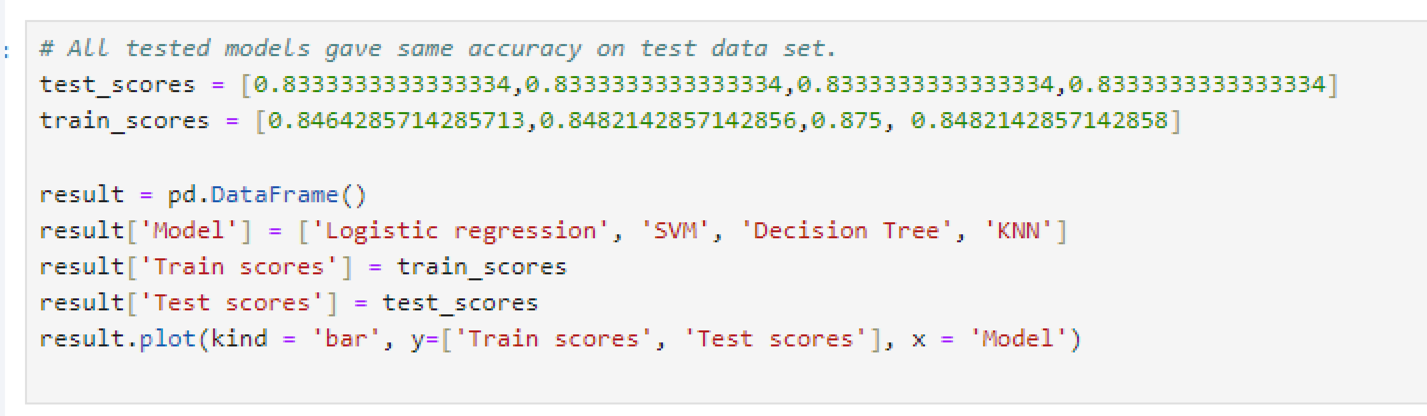


Conclusions

- Destination orbit has high impact on mission outcome.
- Outcome is also correlated with Payload mass and Launch Site.
- Consecutive launches of same mission increases chances of success.
- GEO orbit is most prone to be failed.
- All machine learning algorithms shown close results on train data and similar results on test data. By sum of scores on a whole dataset decision tree algorithm was chosen as best one.

Appendix

- Python code of plotting precise Machine Learning models scores in Notebook.



```
# All tested models gave same accuracy on test data set.
test_scores = [0.8333333333333334,0.8333333333333334,0.8333333333333334,0.8333333333333334]
train_scores = [0.8464285714285713,0.8482142857142856,0.875, 0.8482142857142858]

result = pd.DataFrame()
result['Model'] = ['Logistic regression', 'SVM', 'Decision Tree', 'KNN']
result['Train scores'] = train_scores
result['Test scores'] = test_scores
result.plot(kind = 'bar', y=['Train scores', 'Test scores'], x = 'Model')
```

Thank you!