IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

Outline

Executive Summary

The goal of this project was to develop a model to predict whether a Falcon 9 first stage will successfully land after launch. This way we can get better mission planning and cost estimation and reduce costs.

We had historical launch data and through exploratory data analysis we noticed launch success was highly correlated with features like mass of the payload, the orbit, launch site, the version of the booster being used, if parts were reused...

We applied several classification models and evaluated their performances. In particular, we evaluated logistic regression, support vector machines, decision trees and K-nearest neighbours. All four methods have similar high performance.

This high performance across all 4 methods prove launch success can be predicted accurately based our available data. Therefore, data science can be applied to decision making in this case to reduce rocket launches costs

Introduction

SpaceX disrupted the space launch market by changing cost structures and operational paradigms. At the forefront of these savings is Space X ability to reuse the first stage of Falcon 9 rockets.

Therefore, predicting if a first stage will land successfully after launch would become a very competitive advantage that will helps us reduce costs.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected from multiple open-access sources: the SpaceX REST API and Wikipedia. A dataset was compiled which provides information on launch dates, launch sites, payload characteristics, booster versions, reuse status, orbit types, and landing outcomes.
- Perform data wrangling
  - All rows with missing data were removed and one hot encoding was applied to non-numerical fields.
- 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

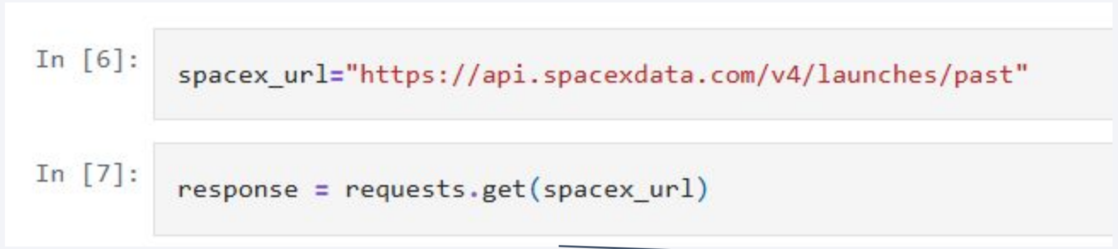
Data Collection

We used two different sources:

- Space X REST API: Historical information about Space X Falcon 9 launches including coordinates of launch, payload, orbit…
- Wikpedia: Web scraping to collect Falcon 9 historical launch records

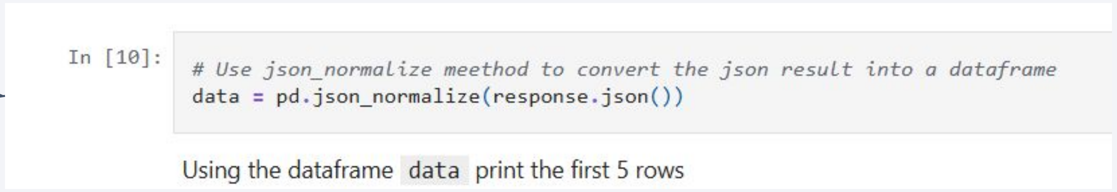
Data Collection – SpaceX API

1. **Request to API**



```
In [6]:  spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]:  response = requests.get(spacex_url)
```

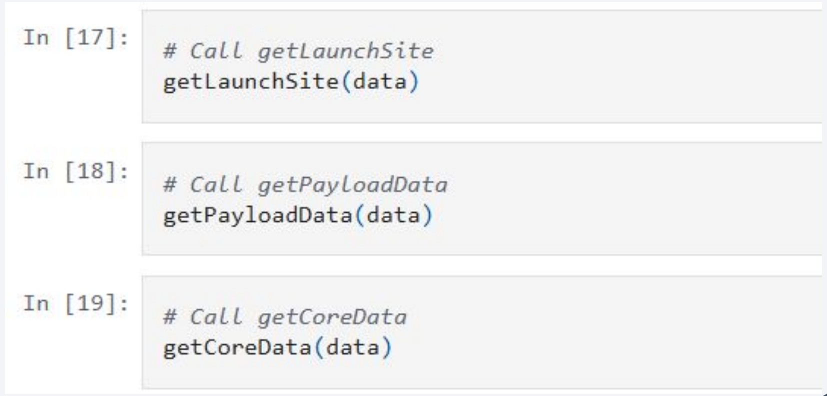
2. Conversion to JSON



```
In [10]:  # Use json_normalize meethod to convert the json result into a dataframe
          data = pd.json_normalize(response.json())
```

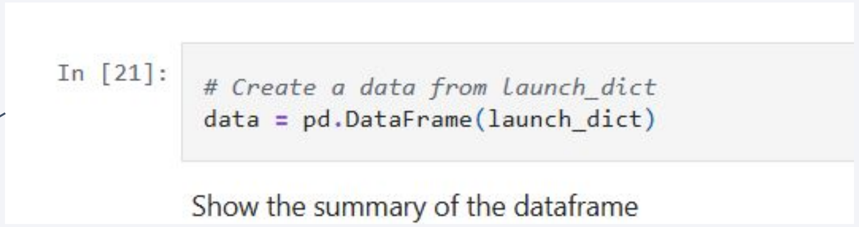
Using the dataframe `data` print the first 5 rows

3. Apply custom functions to format data



```
In [17]:  # Call getLaunchSite
          getLaunchSite(data)

In [18]:  # Call getPayloadData
          getPayloadData(data)

In [19]:  # Call getCoreData
          getCoreData(data)
```

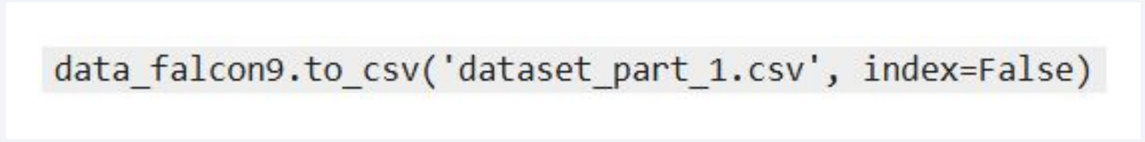
4. Build dataframe with new dataset



```
In [21]:  # Create a data from launch_dict
          data = pd.DataFrame(launch_dict)
```

Show the summary of the dataframe

5. Save dataset as text file

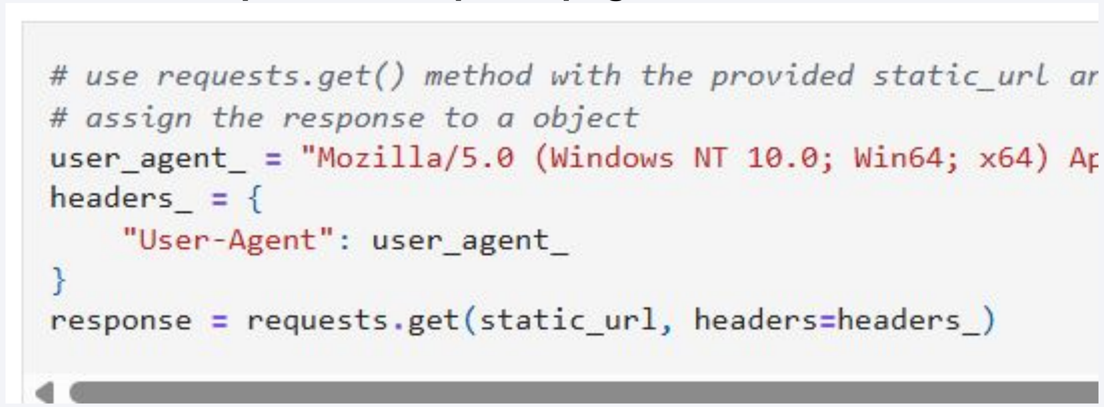


```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Source:
https://github.com/AntoData/IBM_capstone_project/blob/main/Module%201%20-%20Introduction/5.%20jupyter-labs-spacex-data-collection-api.ipynb

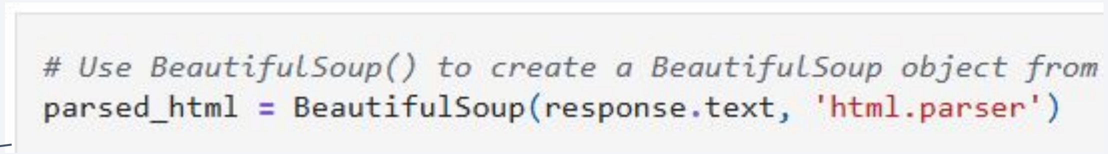
Data Collection - Scraping

1. **Get request to wikipedia page**



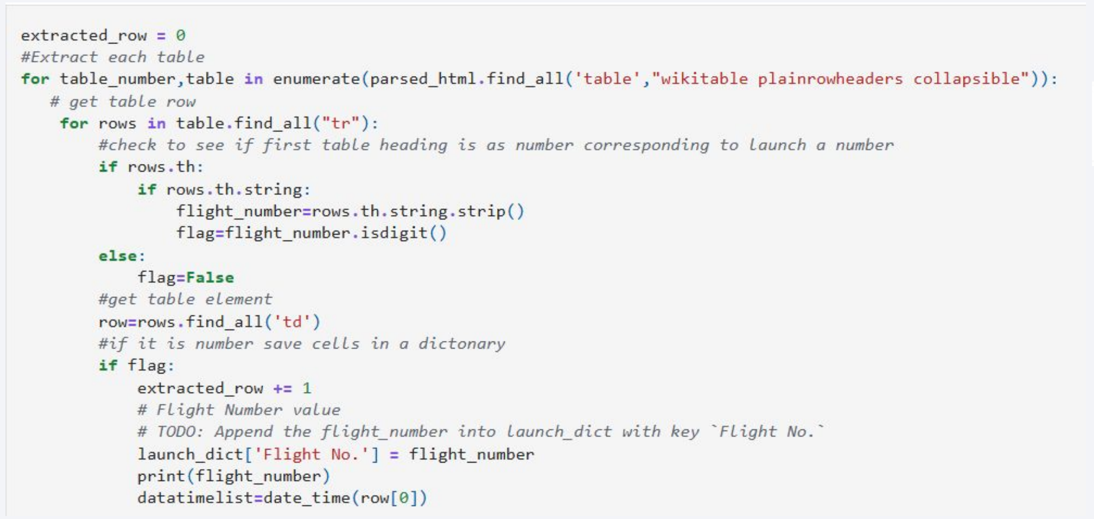
```
# use requests.get() method with the provided static_url ar
# assign the response to a object
user_agent_ = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Ap
headers_ = {
    "User-Agent": user_agent_
}
response = requests.get(static_url, headers=headers_)
```

2. Create BeautifulSoup HTML parser



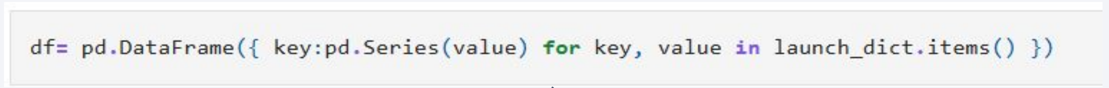
```
# Use BeautifulSoup() to create a BeautifulSoup object from
parsed_html = BeautifulSoup(response.text, 'html.parser')
```

3. Parse HTML into data structures



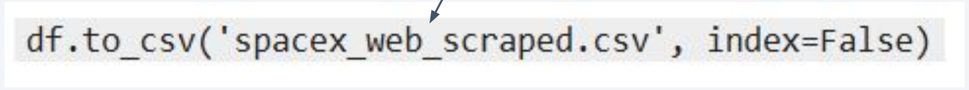
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(parsed_html.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'] = flight_number
            print(flight_number)
            datatimelist=date_time(row[0])
```

4. Create dataframe



```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

5. Save dataframe to text file

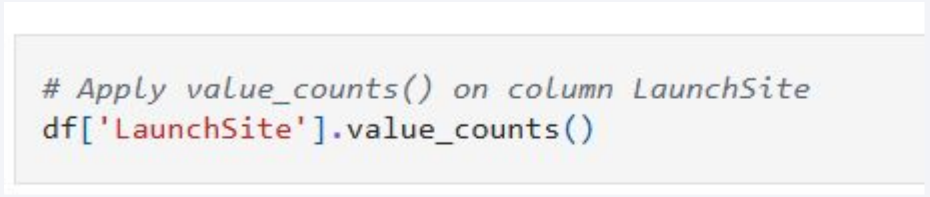


```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Source.
https://github.com/AntoData/IBM_capstone_project/blob/main/Module%201%20-%20Introduction/6.%20jupyter-labs-webscraping.ipynb

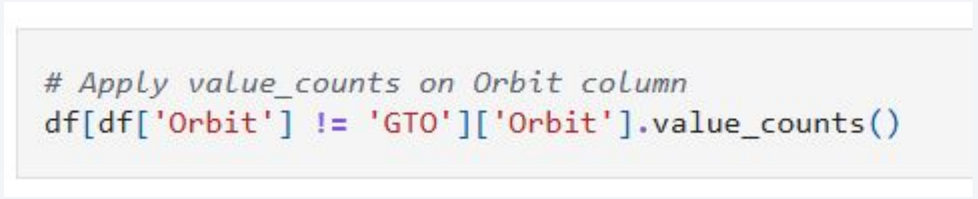
Data Wrangling

1. **Get number of instances per Launch Site**



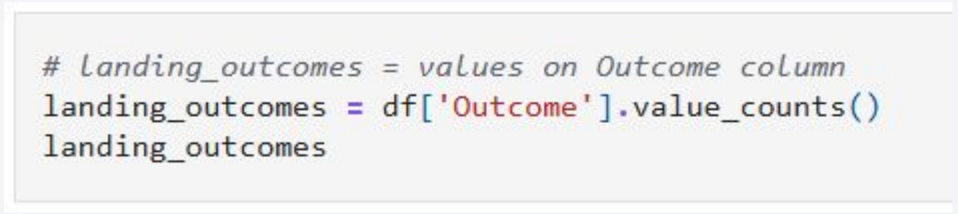
```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

2. Get number of launches whose orbit is not GTO



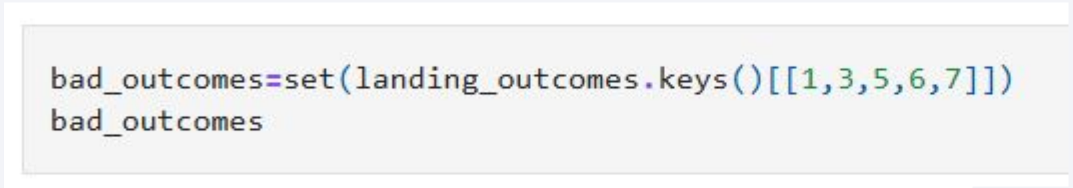
```
# Apply value_counts on Orbit column
df[df['Orbit'] != 'GTO']['Orbit'].value_counts()
```

3. Store landing outcomes from the dataset

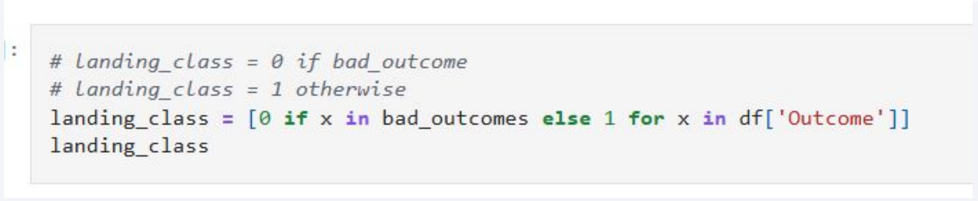


```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

4. Classify landings into successful/failure

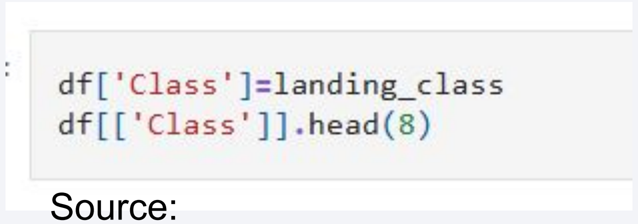


```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```



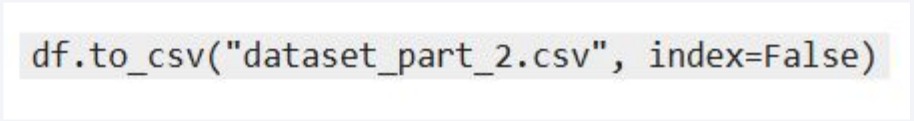
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = [0 if x in bad_outcomes else 1 for x in df['Outcome']]
landing_class
```

5. Create column in dataset for this flag



```
df['Class']=landing_class
df[['Class']].head(8)
```

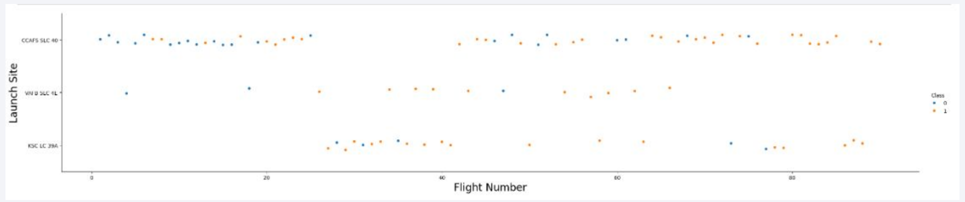
6. Save to text file



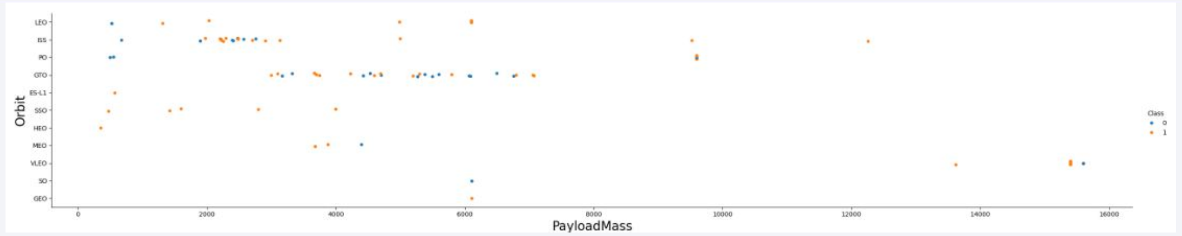
```
df.to_csv("dataset_part_2.csv", index=False)
```

Source:
https://github.com/AntoData/IBM_capstone_project/blob/main/Module%201%20-%20Introduction/8.%20labs-jupyter-spacex-Data%20wrangling.ipynb

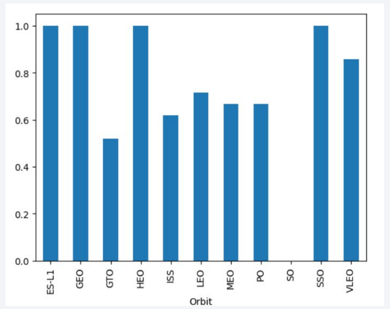
EDA with Data Visualization



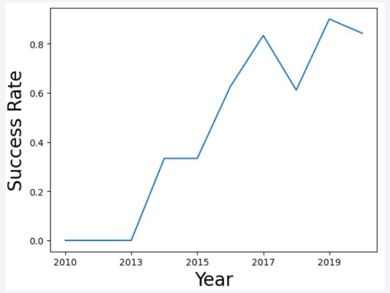
Flight numbers vs launch site: Flight numbers are almost perfectly synched so several close launches are done within the same launch site



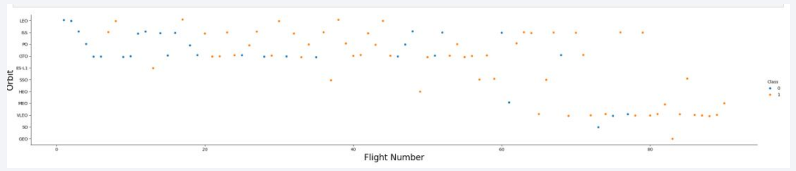
Payload Mass Vs. Launch Site: To realise the correlation between payload mass and launch site. For instance: For VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).



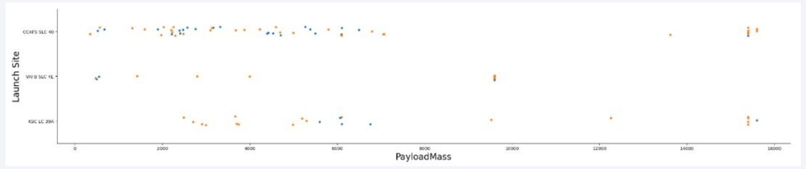
Success rate by orbit: To check if there is correlation between the orbit type and the success rate



Success rate evolution through the years: To check if there has been a positive evolution



FlightNumber vs Orbit type: You can observe that in the LEO orbit, success seems to be related to the number of flights



Payload Mass vs Orbit type: With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

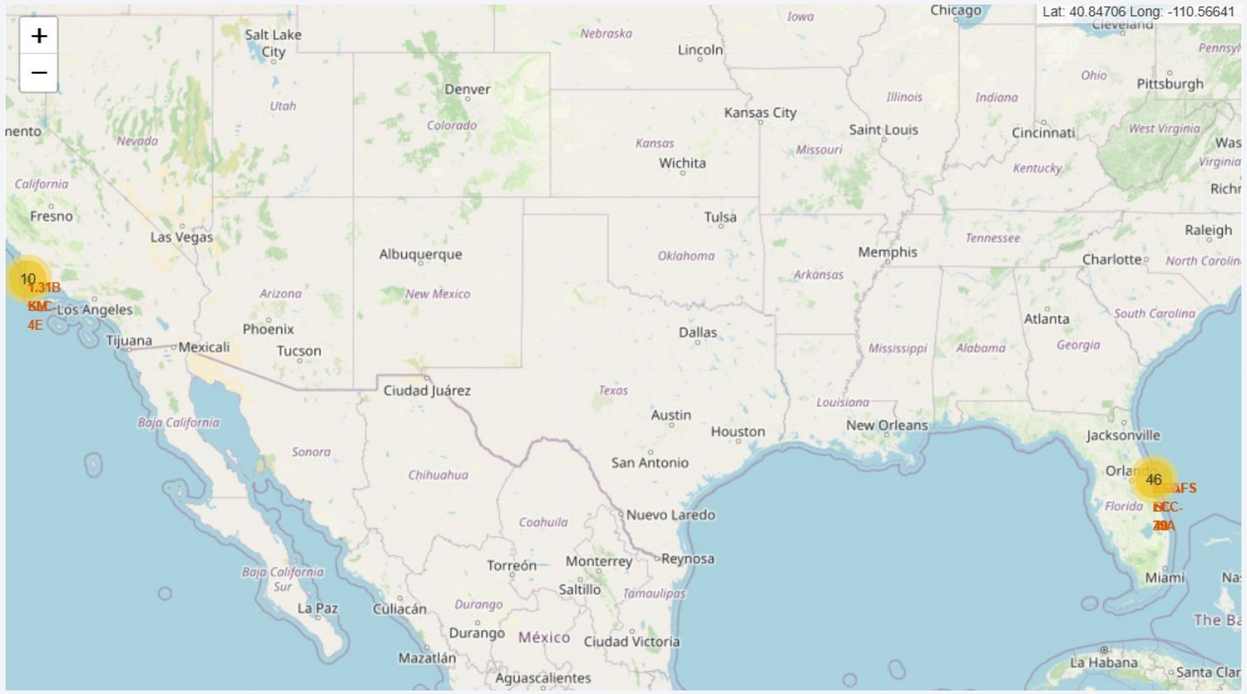
Source: https://github.com/AntoData/IBM_capstone_project/blob/main/Module%202%20-%20Exploratory%20Data%20Analysis%20(EDA)/3.%20edadataviz.ipynb

EDA with SQL

- Get names of the unique launch sites in the space mission
- Get 5 records where launch sites begin with the string 'CCA'
- Get the total payload mass carried by boosters launched by NASA (CRS)
- Get average payload mass carried by booster version F9 v1.1
- Get the date when the first successful landing outcome in ground pad was achieved.
- Get the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Get the total number of successful and failure mission outcomes
- Get all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.
- Get 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.
- 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.

Source: https://github.com/AntoData/IBM_capstone_project/blob/main/Module%202%20-%20Exploratory%20Data%20Analysis%20(EDA)/2.%20jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

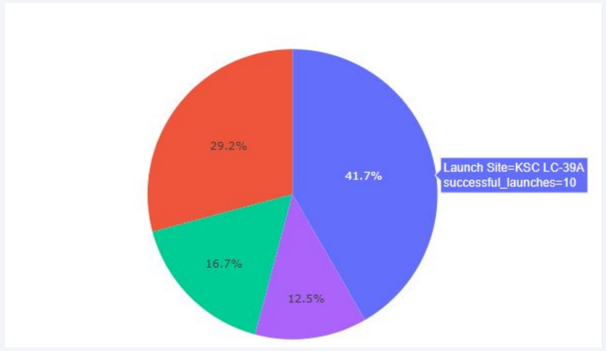


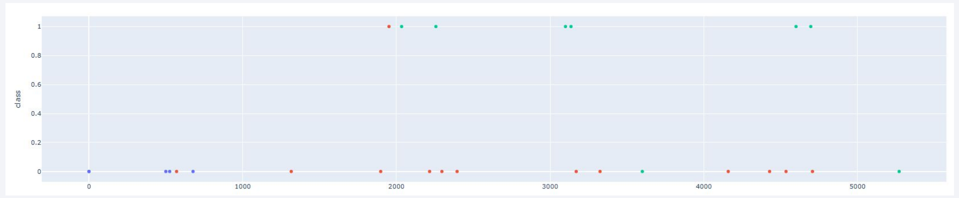
- We created a map with circles to highlight the different launch sites and marker clusters to display the different launch attempts in each site with different colour depending on their result. Finally, we drew a line from one of the sites to the closest train rail.
- The goal was to find an optimal location for our rocket launches

Source:
https://github.com/AntoData/IBM_capstone_project/blob/main/Module%203%20-%20Interactive%20Visual%20Analytics%20and%20Dashboard/2.%20lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

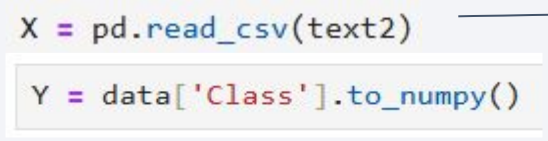
- Pie chart: Launch site vs success rate
  - If not particular launch site is selected: Percentage of successful launches in that site (vs other sites)
  - If a particular launch site is selected: Percentage of successful launches vs failures
  - Allows us to analyse the chances of success in each launch site
- Scatter plot: X axis payload, y axis launch outcome
  - Dropdown allows to filter by launch site
  - Slider allows to filter by payload mass
  - The point color is the booster version
  - Allows to explore the different outcomes depending on launch site, payload mass and booster version





Predictive Analysis (Classification)

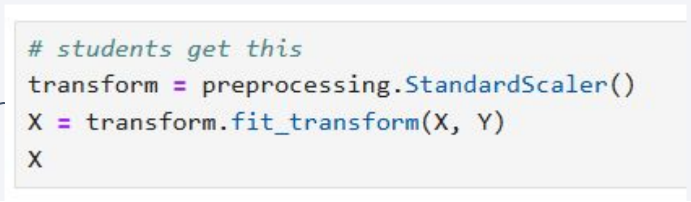
1, Define variables X and Y



```
X = pd.read_csv(text2)
```

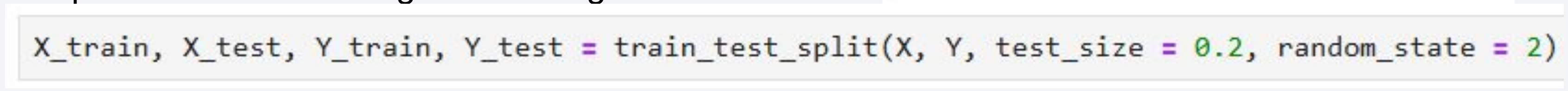
```
Y = data['Class'].to_numpy()
```

2. Apply Standard Scaler



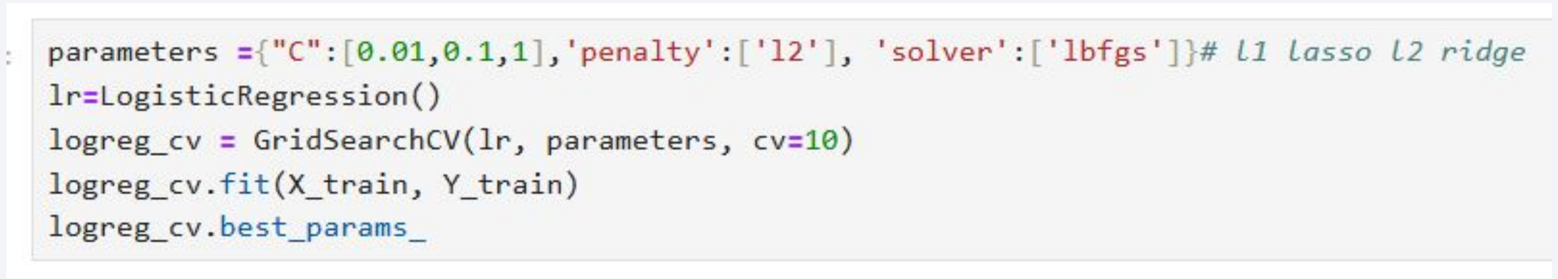
```
# students get this
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X, Y)
X
```

3. Split dataset into testing and training sets



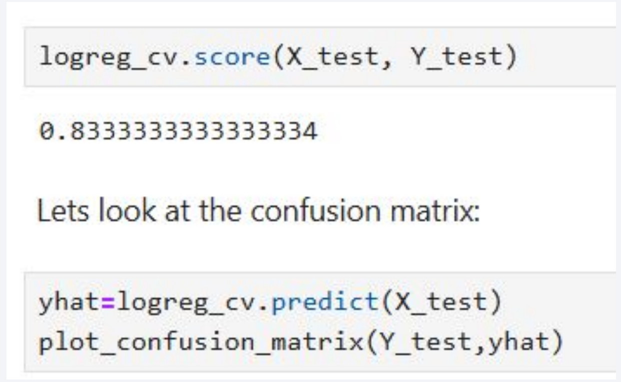
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 2)
```

4. For all 4 models: Logistic Regression., SVM, Decision Tree and KNN
We create the model and apply GridSearchCV with certain parameters and train the mode to the



```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
logreg_cv.best_params_
```

5. Get the accuracy score of the model over the testing set and confusion matrix



```
logreg_cv.score(X_test, Y_test)
```

```
0.8333333333333334
```

Lets look at the confusion matrix:

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

Source:
https://github.com/AntoData/IBM_capstone_project/blob/main/Module%204%20-%20Predictive%20Analysis%20(Classification)/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

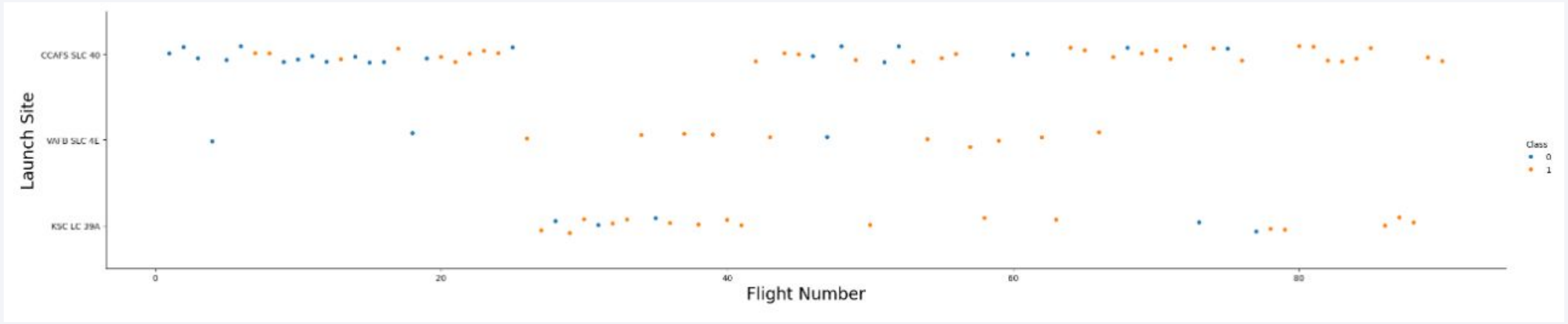
Results

- Logistic Regression, SVM, Decision Trees and KNN all perform similarly, their accuracy being around 83%
- As time passes, success rates for launches increase
- Orbits ES-L1, GEO, HEO, SSO have the highest success rate
- For heavy payloads the successful landing rate is mostly for Polar, LEO and ISS.
- For launch site VAFB-SLC there are no rockets launched for with a payload mass greater than 10000

Section 2

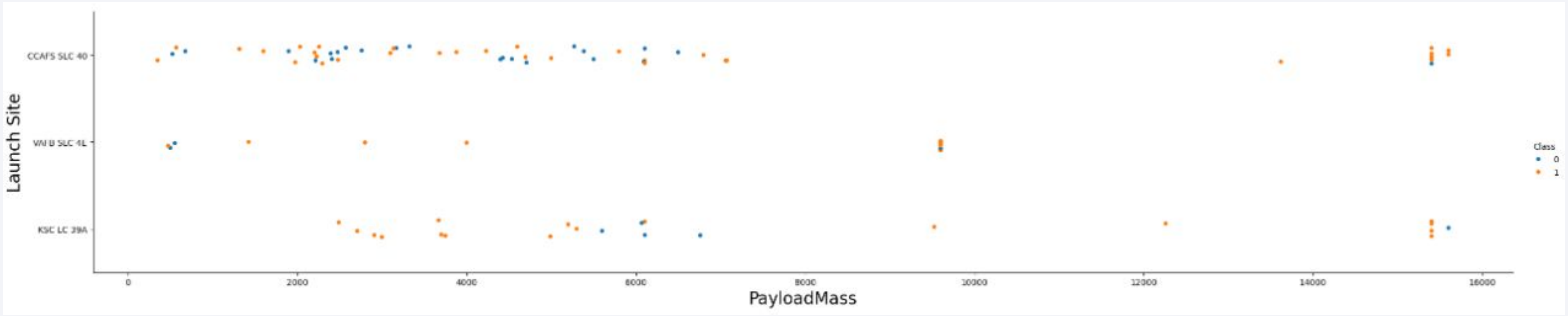
Insights drawn from EDA

Flight Number vs. Launch Site



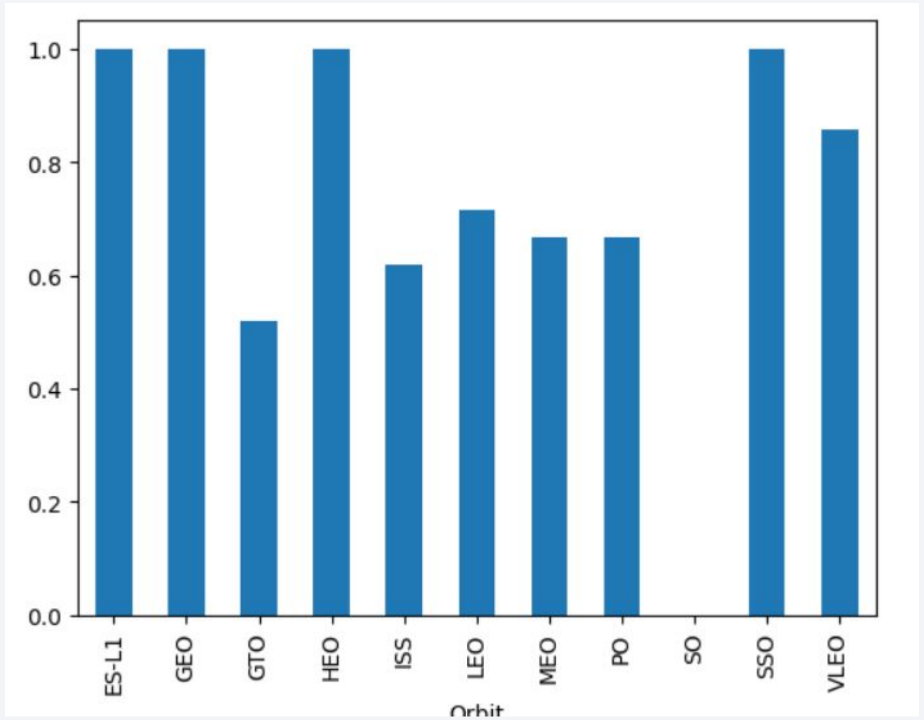
Flight numbers and launch sites are almost perfectly synched so several close launches are done within the same launch site. That means that if flight numbers are chronological, it looks like rocket launches happen in seasonally in each launch site. Probably due that a rocket launch usually includes several tests and more than one launch

Payload vs. Launch Site



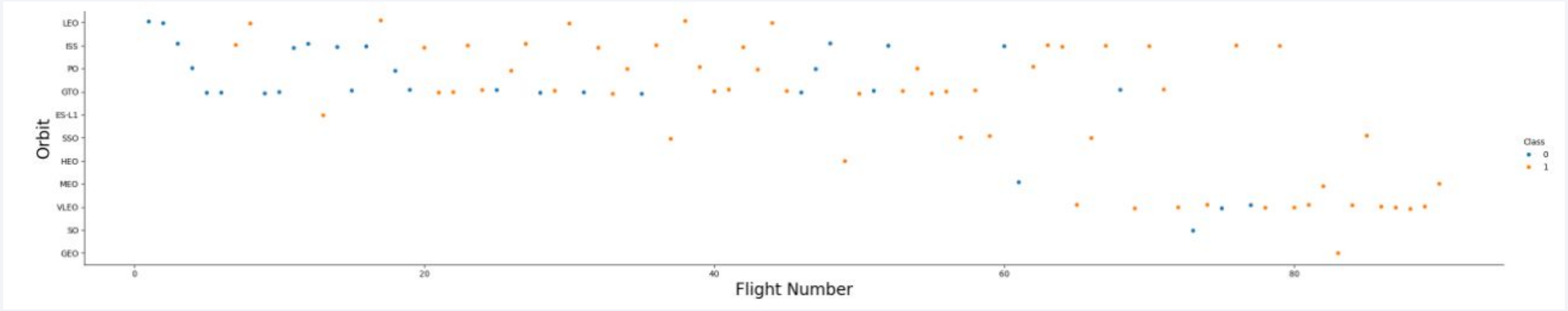
For launch site VAFB-SLC there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type



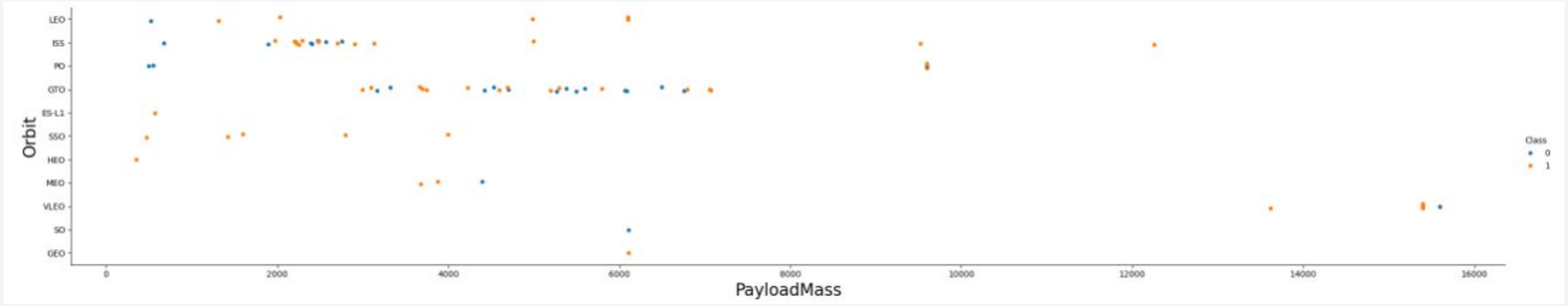
Orbits ES-L1, GEO, HEO, SSO have the highest success rate, 100%

Flight Number vs. Orbit Type



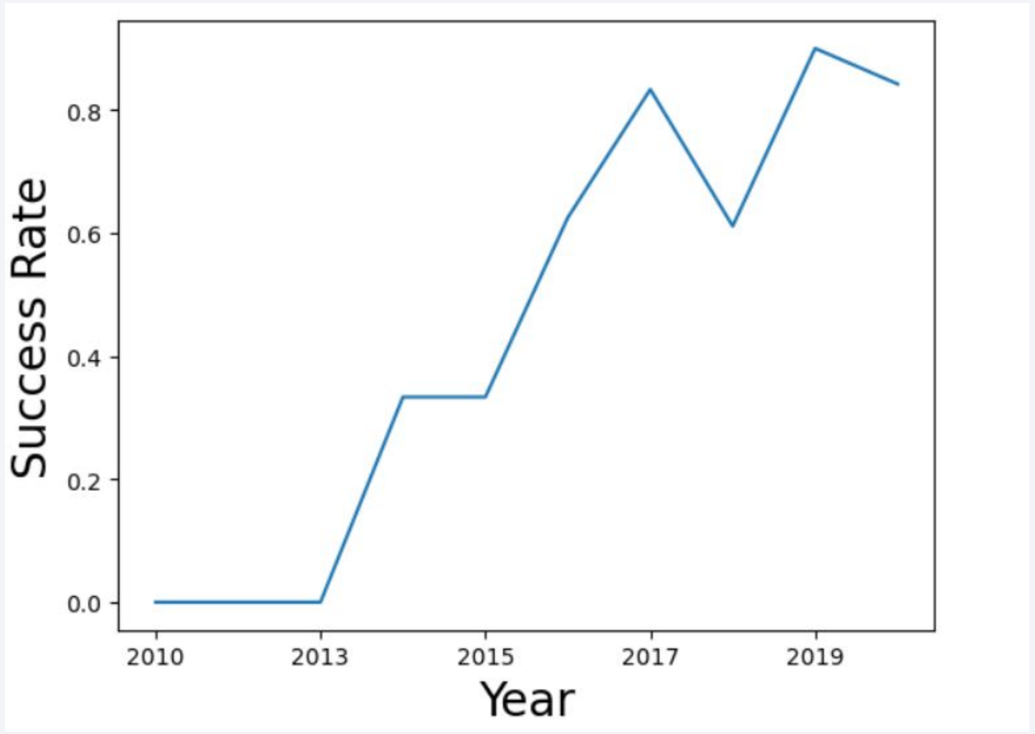
In orbit of type LEO, success seems to be related to the flight numbers. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

Payload vs. Orbit Type



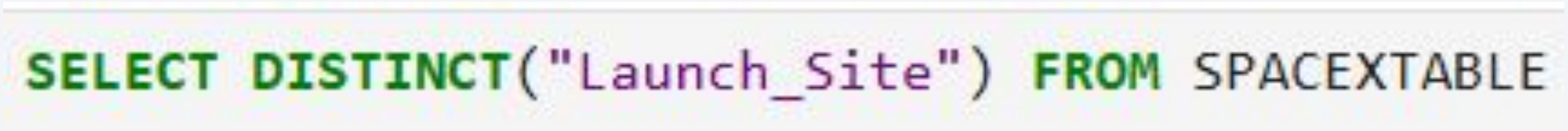
For heavy payloads the successful landing or positive landing rates are higher for Polar,LEO and ISS.
However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend



Success rate since 2013 kept increasing till 2018 to increase again in 2019 with a final dive in 2020

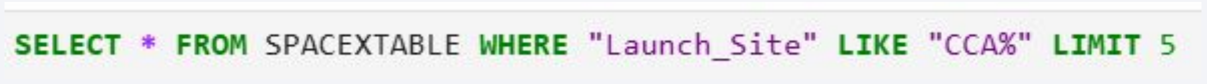
All Launch Site Names



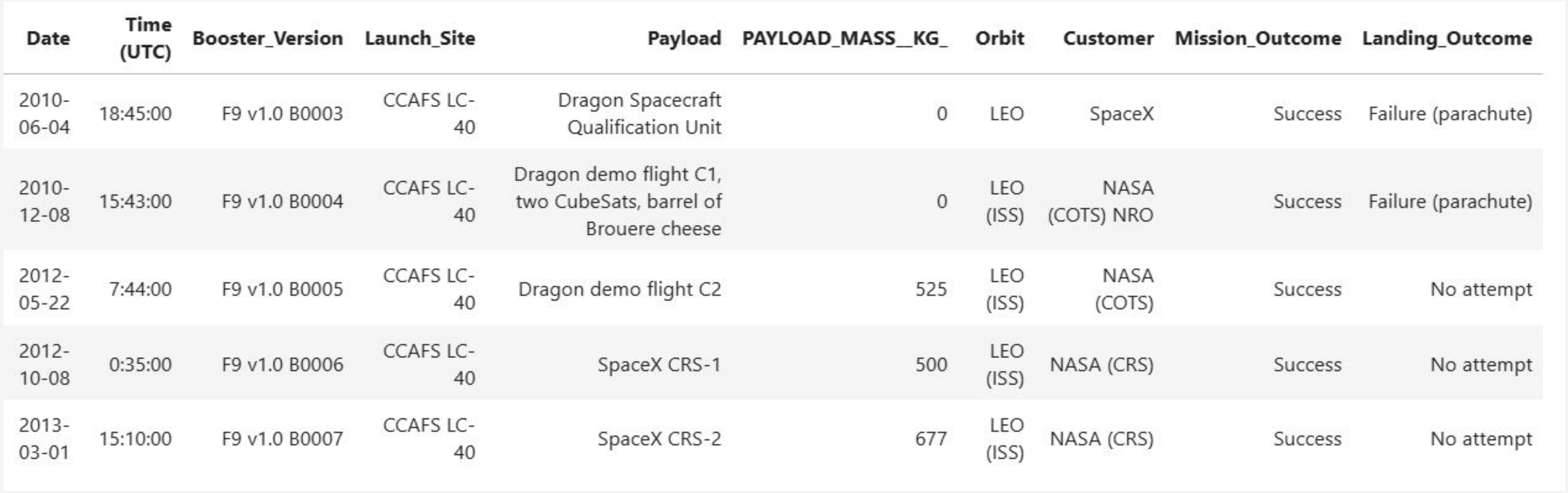
```
SELECT DISTINCT("Launch_Site") FROM SPACEXTABLE
```

- SELECT DISTINCT("Launch_Site") FROM SPACEXTABLE
    - CCAFS LC-40
    - VAFB SLC-4E
    - KSC LC-39A
    - CCAFS SLC-40

Launch Site Names Begin with 'CCA'



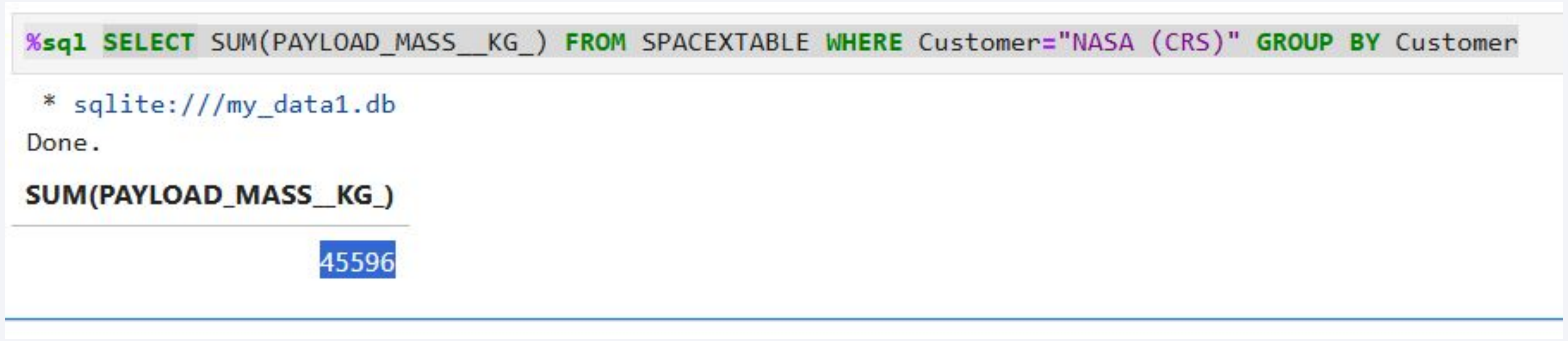
```
SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE "CCA%" LIMIT 5
```



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

Total Payload Mass

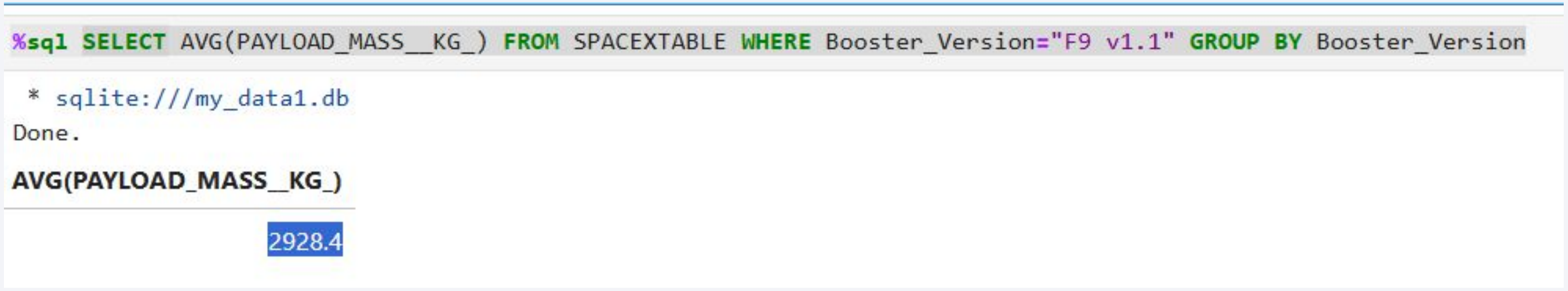
- SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer="NASA (CRS)" GROUP BY Customer
    - 45596



```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer="NASA (CRS)" GROUP BY Customer

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

SUM(PAYLOAD_MASS__KG_)
45596

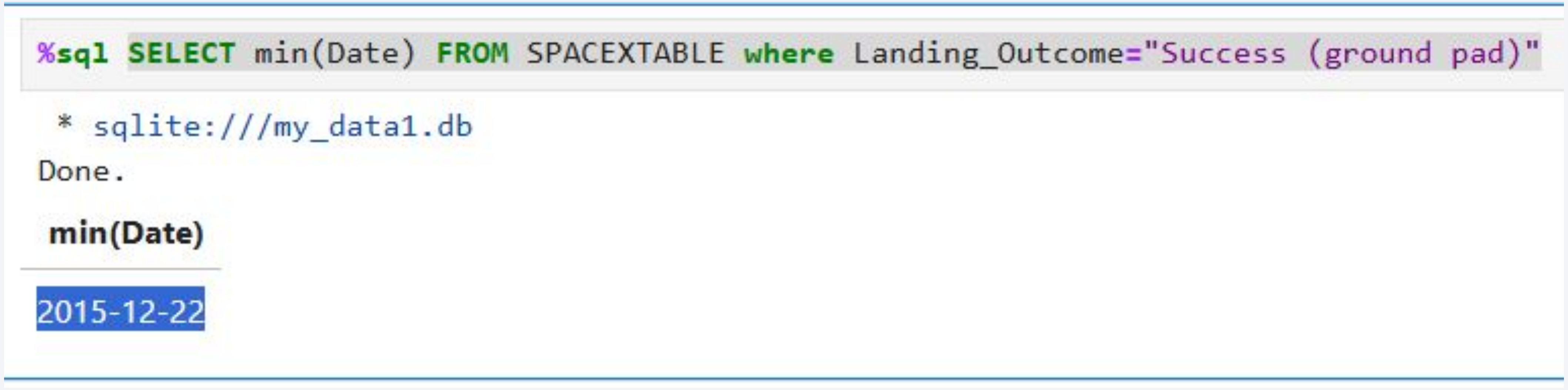
Average Payload Mass by F9 v1.1



```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version="F9 v1.1" GROUP BY Booster_Version

 * sqlite:///my_data1.db
Done.

AVG(PAYLOAD_MASS__KG_)
                2928.4
```

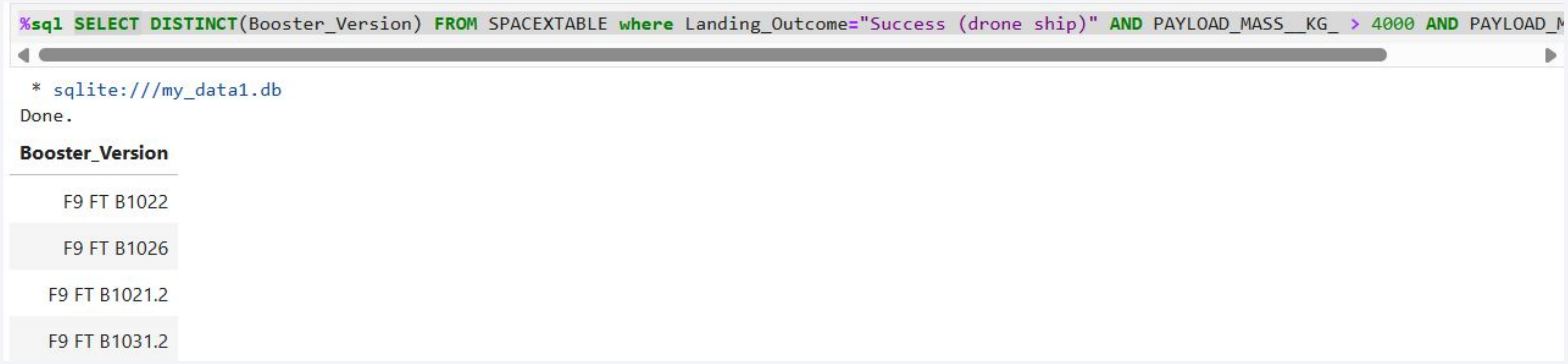
- SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version="F9 v1.1" GROUP BY Booster_Version
    - 2928.4

First Successful Ground Landing Date


```
%sql SELECT min(Date) FROM SPACEXTABLE where Landing_Outcome="Success (ground pad)"

 * sqlite:///my_data1.db
Done.

min(Date)

2015-12-22
```

- SELECT min(Date) FROM SPACEXTABLE where Landing_Outcome="Success (ground pad)"
    - 2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000



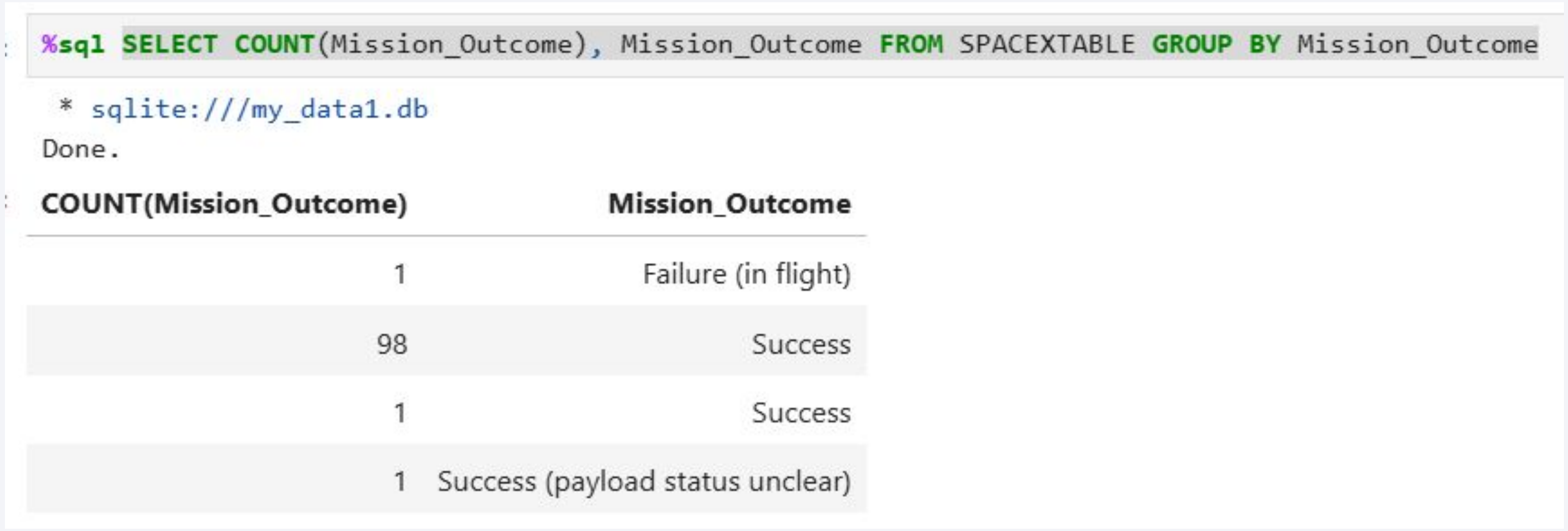
```
%sql SELECT DISTINCT(Booster_Version) FROM SPACEXTABLE where Landing_Outcome="Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_M
```

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

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

- SELECT DISTINCT(Booster_Version) FROM SPACEXTABLE where Landing_Outcome="Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000
    - F9 FT B1022
    - F9 FT B1026
    - F9 FT B1021.2
    - F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes



```
%sql SELECT COUNT(Mission_Outcome), Mission_Outcome FROM SPACEXTABLE GROUP BY Mission_Outcome
```

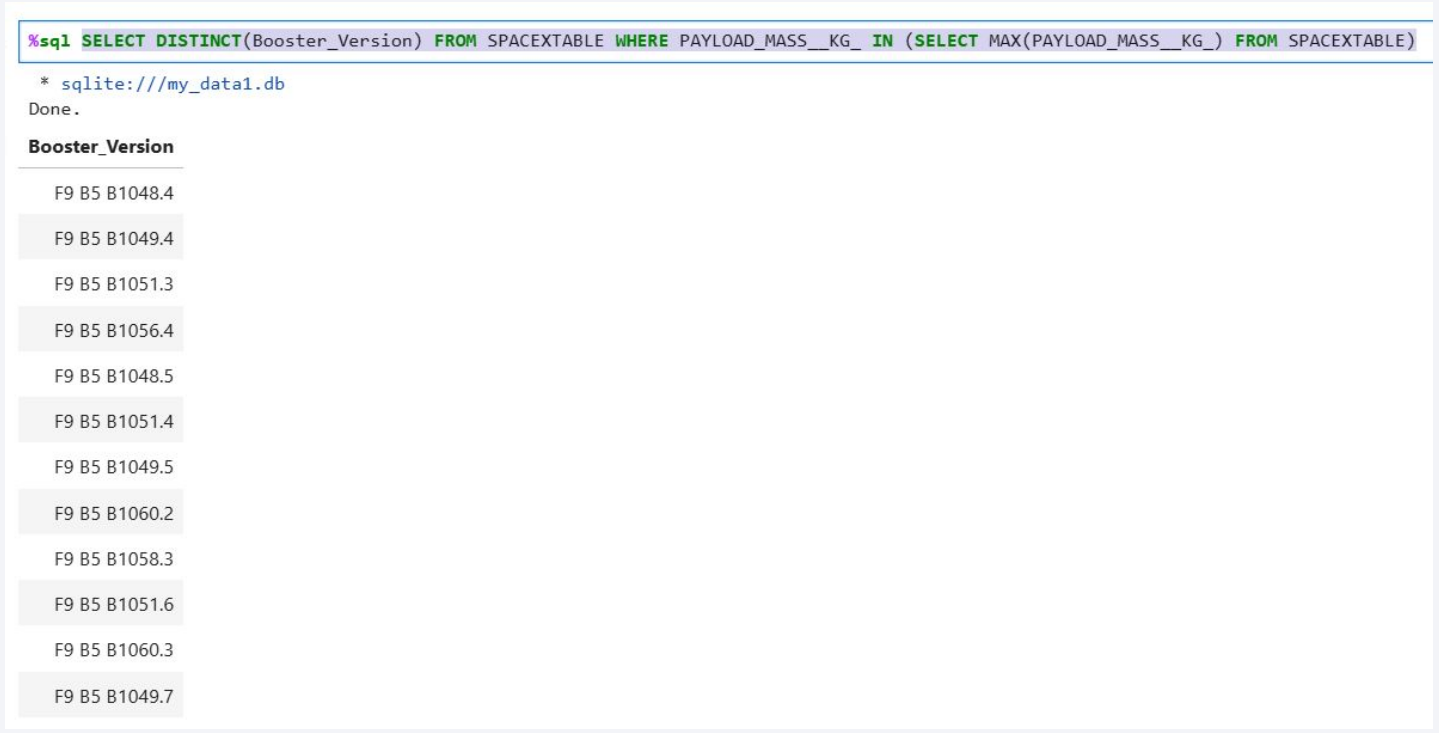
```
 * sqlite:///my_data1.db
Done.
```

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

- SELECT COUNT(Mission_Outcome), Mission_Outcome FROM SPACEXTABLE GROUP BY Mission_Outcome
  - 100 Success vs 1 Failure

Boosters Carried Maximum Payload

- SELECT DISTINCT(Booster_Version) FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ IN (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)

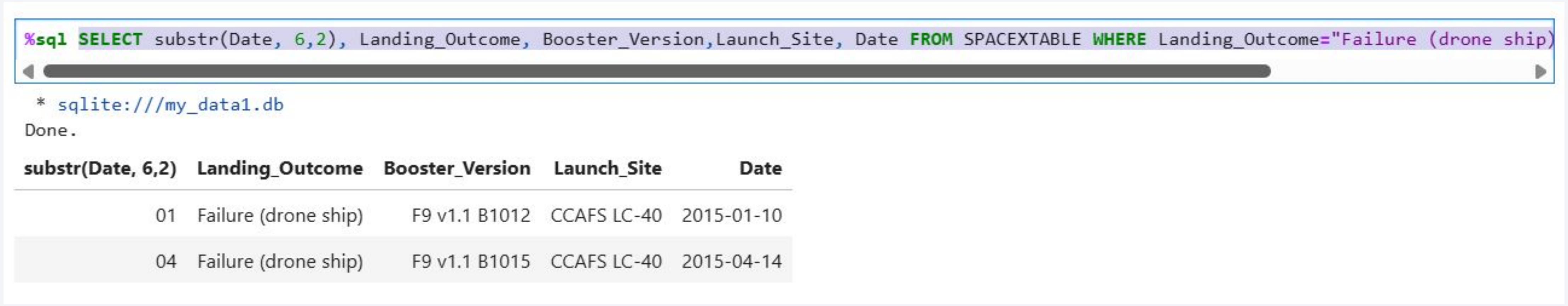


```
%sql SELECT DISTINCT(Booster_Version) FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ IN (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)

 * 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

- SELECT substr(Date, 6,2), Landing_Outcome, Booster_Version,Launch_Site, Date FROM SPACEXTABLE WHERE Landing_Outcome="Failure (drone ship)" AND substr(Date,0,5)="2015"



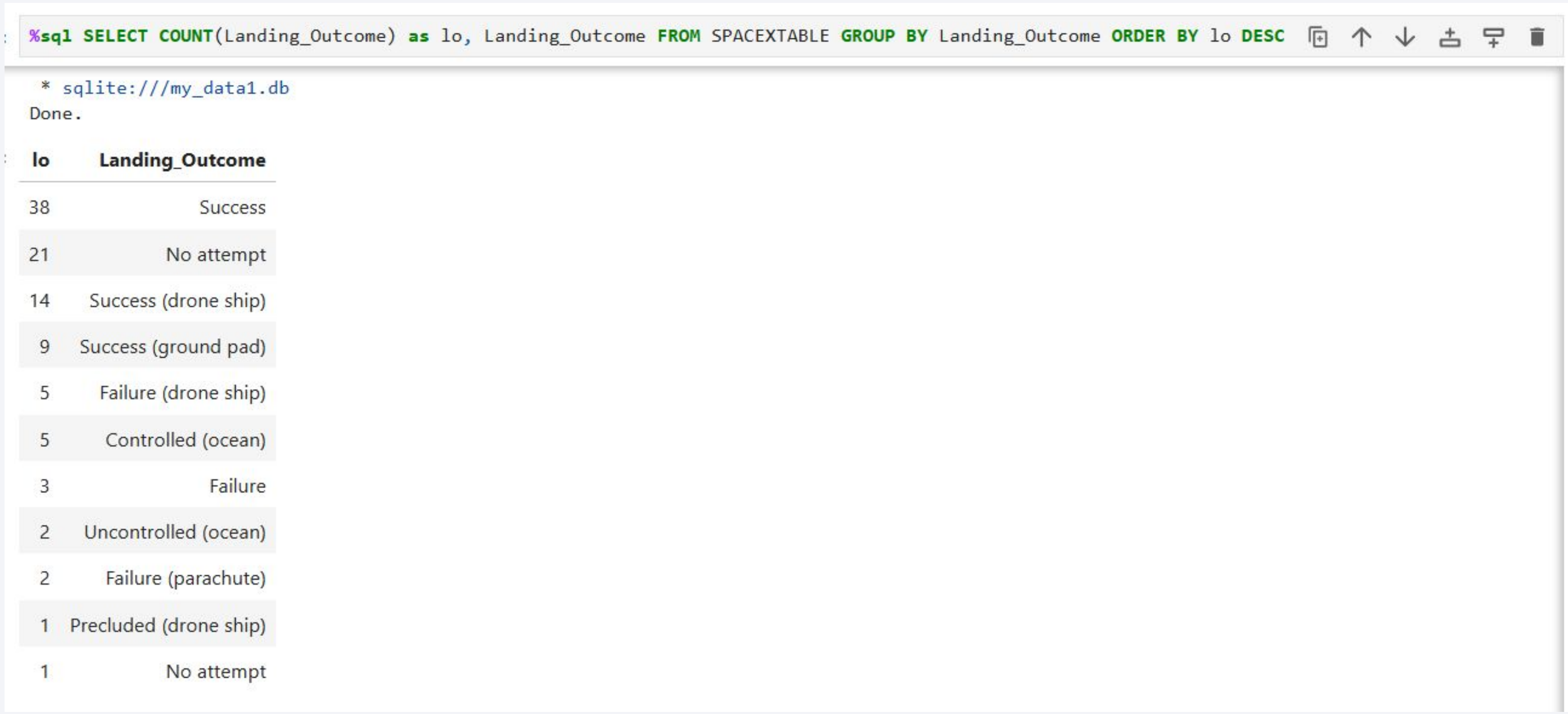
```
%sql SELECT substr(Date, 6,2), Landing_Outcome, Booster_Version,Launch_Site, Date FROM SPACEXTABLE WHERE Landing_Outcome="Failure (drone ship)
```

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

substr(Date, 6,2)	Landing_Outcome	Booster_Version	Launch_Site	Date
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015-01-10
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015-04-14

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

SELECT COUNT(Landing_Outcome) as lo, Landing_Outcome FROM SPACEXTABLE GROUP BY Landing_Outcome ORDER BY lo DESC



```
%sql SELECT COUNT(Landing_Outcome) as lo, Landing_Outcome FROM SPACEXTABLE GROUP BY Landing_Outcome ORDER BY lo DESC
```

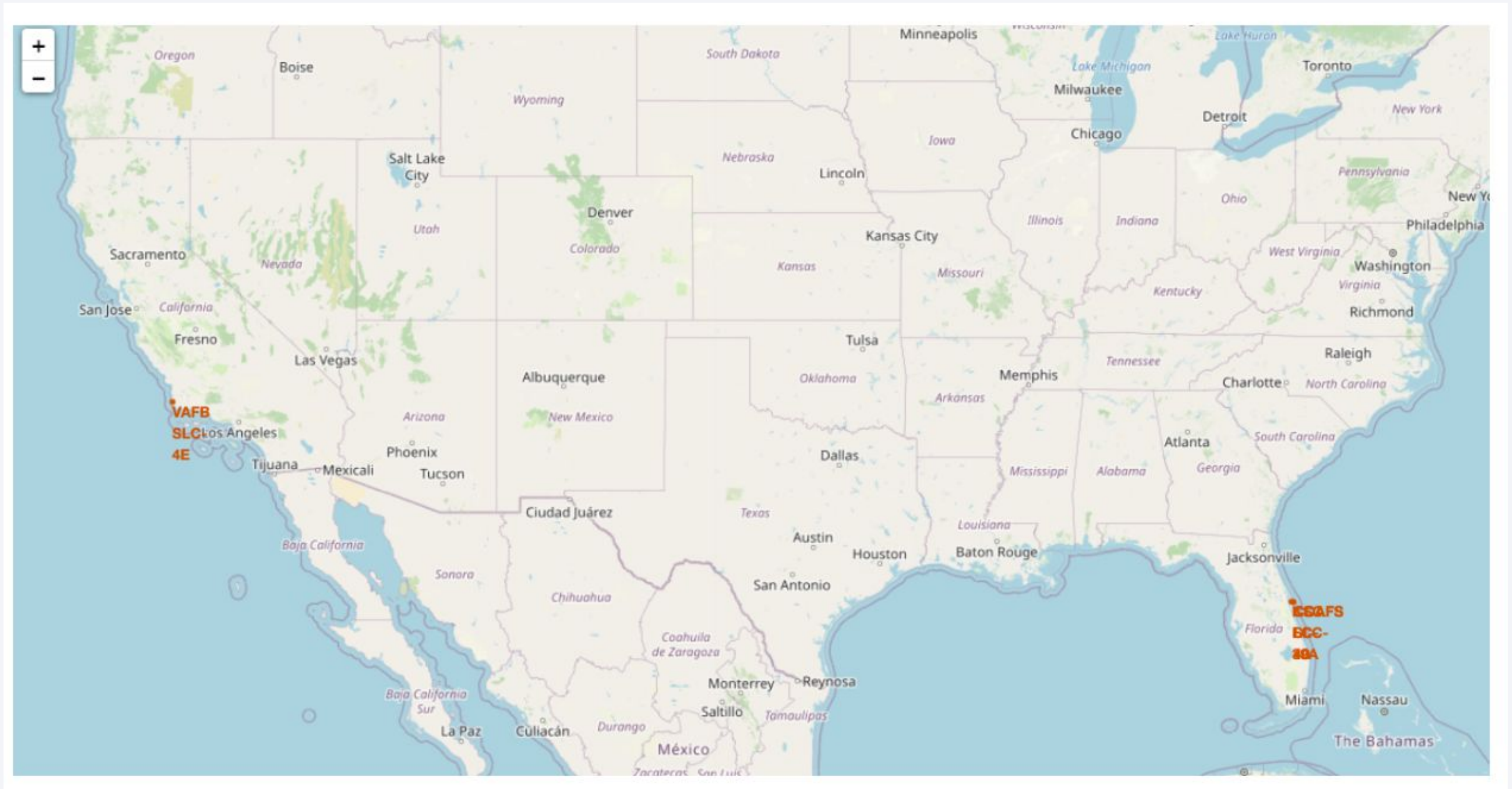
```
 * sqlite:///my_data1.db
Done.
```

lo	Landing_Outcome
38	Success
21	No attempt
14	Success (drone ship)
9	Success (ground pad)
5	Failure (drone ship)
5	Controlled (ocean)
3	Failure
2	Uncontrolled (ocean)
2	Failure (parachute)
1	Precluded (drone ship)
1	No attempt

Section 3

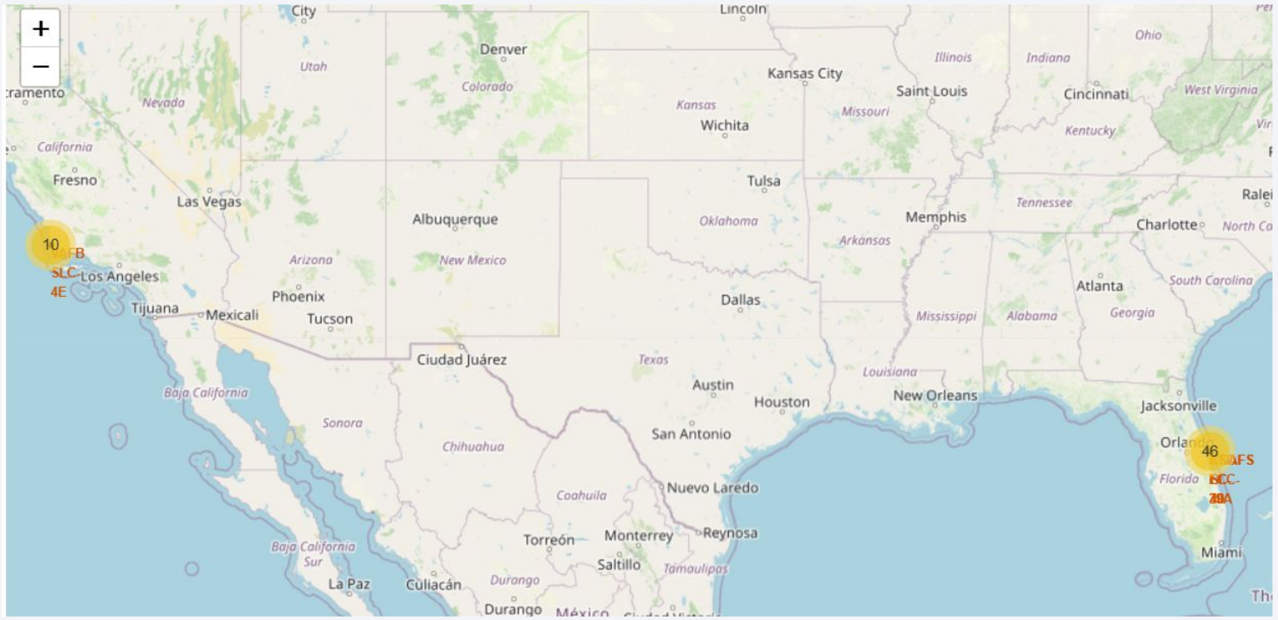
Launch Sites Proximities Analysis

Launch sites on a map



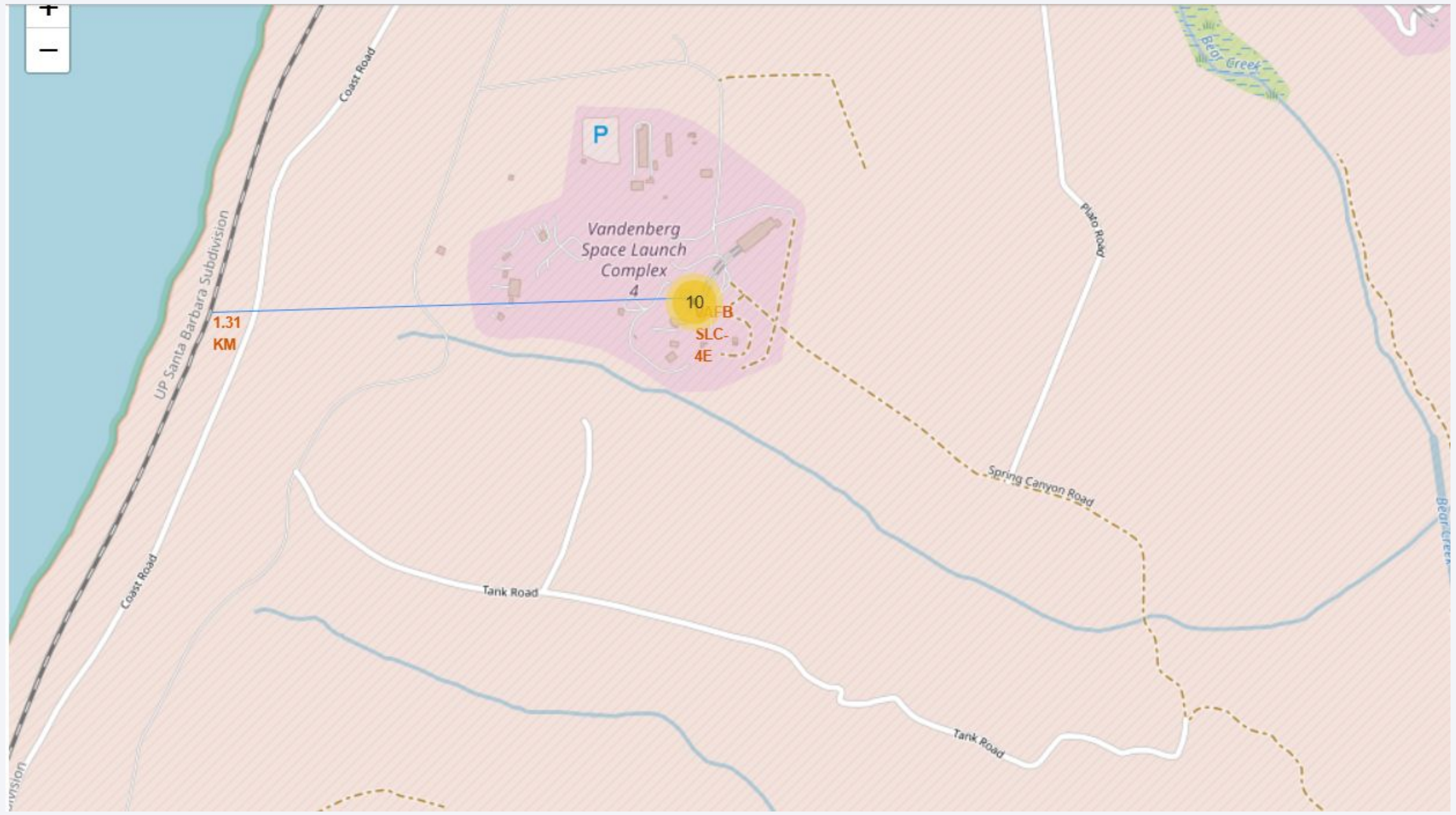
The map contains all the launch site locations as orange circles

Success/failed launches per site

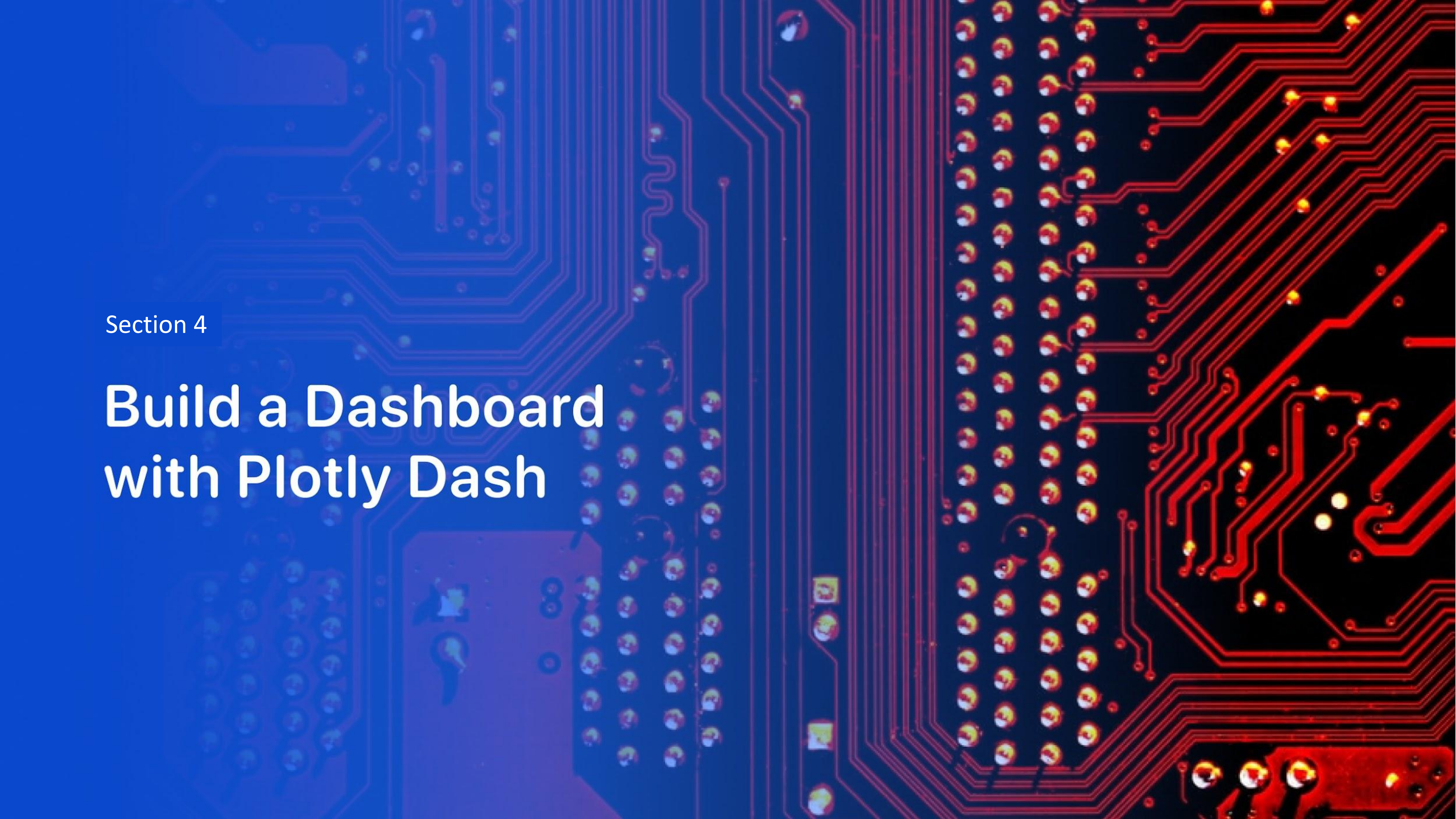


- The map displays circles that return the number of launches in that area. Until we zoom to the location of a certain launch site and then we display the number of launches for that site only. When we select a circle in a launch site other markers are displayed in red to state failed launches and in green to display successful ones

Distances between launch sites to its proximities



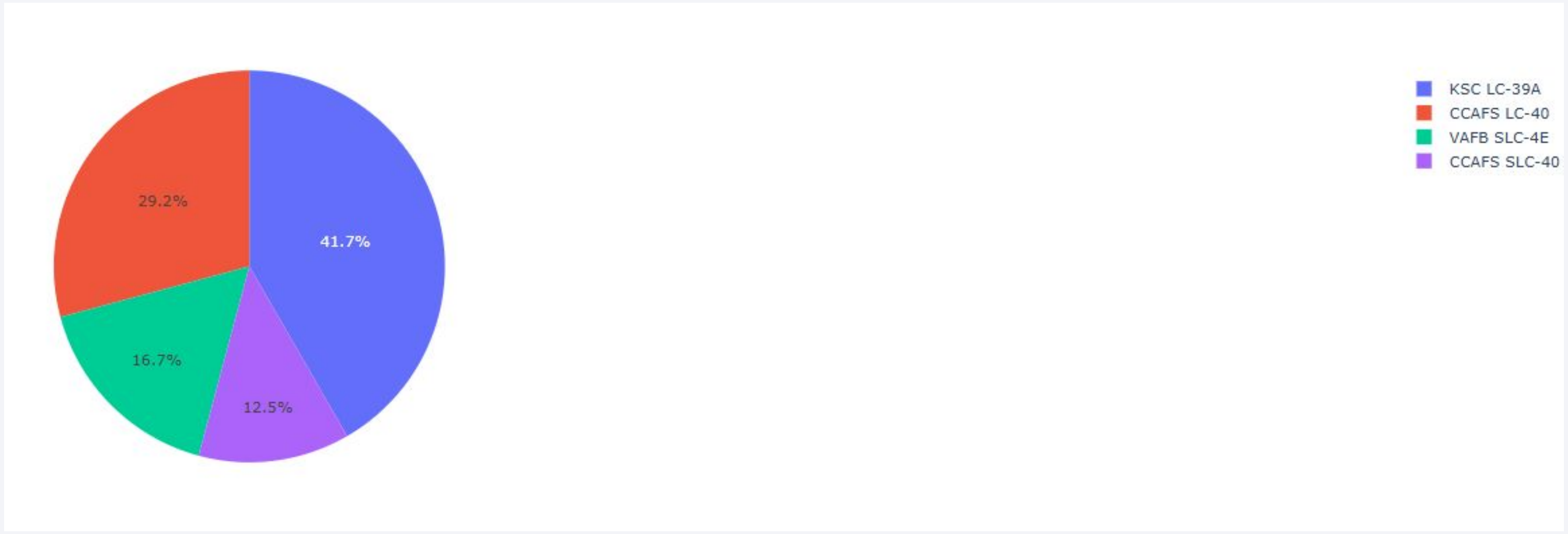
A line was drawn from a launch site in California to its closes railway



Section 4

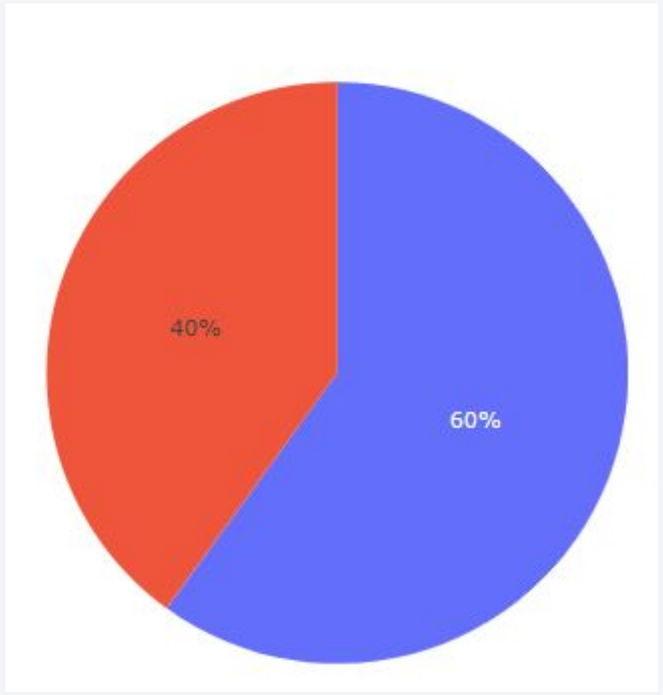
Build a Dashboard with Plotly Dash

Launch site vs success rate



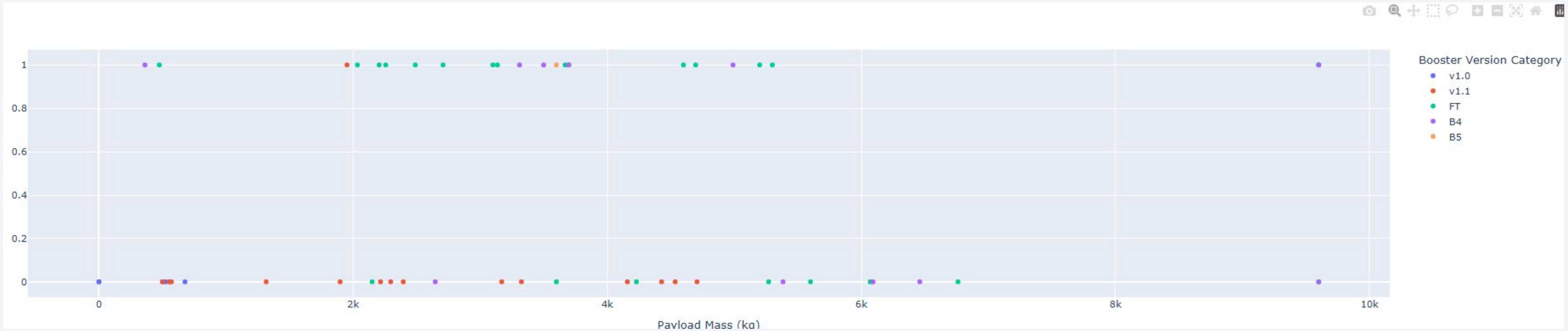
- In this case no particular launch site is selected: Percentage of successful launches in that site (vs other sites) is displayed
- Allows us to analyse the chances of success in each launch site

Success vs Failure rate in a launch site



- In this case, a particular launch site was selected: Percentage of successful launches vs failures would is displayed
- Allows us to analyse the chances of success in each launch site

Payload vs Launch Outcome

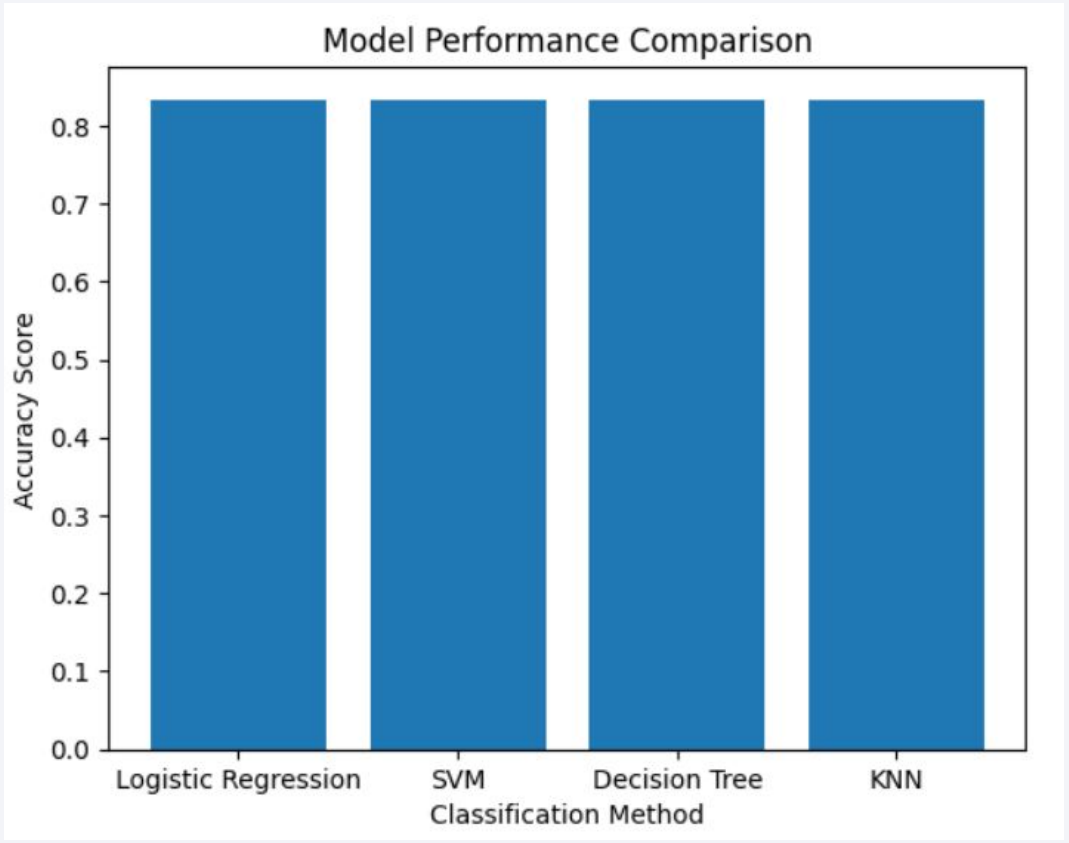


- Dropdown allows to filter by launch site
- Slider allows to filter by payload mass
- The point color is the booster version
- Allows to explore the different outcomes depending on launch site, payload mass and booster version

Section 5

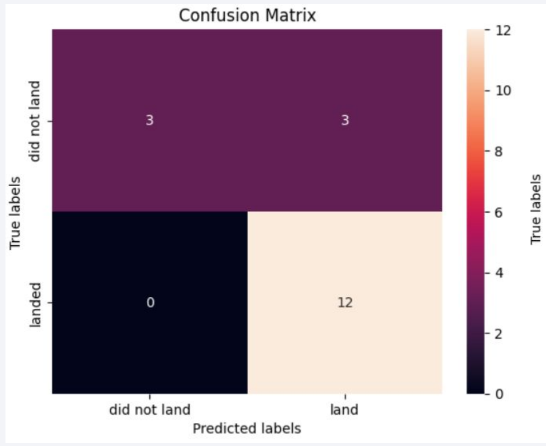
Predictive Analysis (Classification)

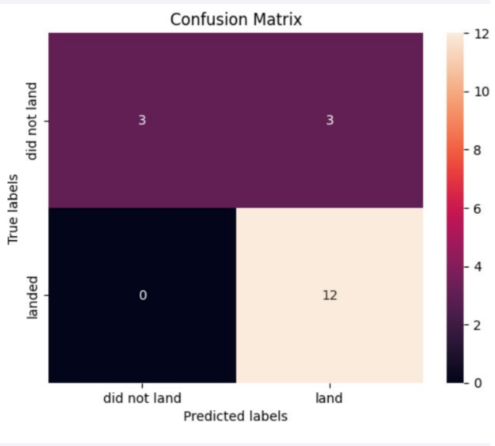
Classification Accuracy

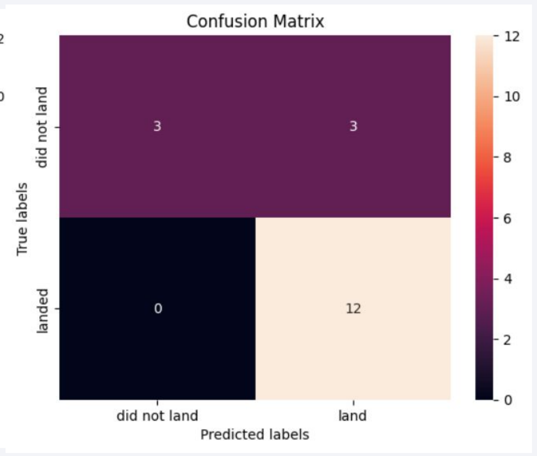


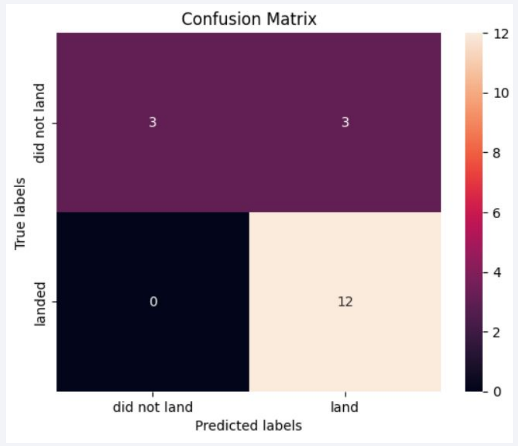
- All 4 models have the same accuracy score

Confusion Matrix









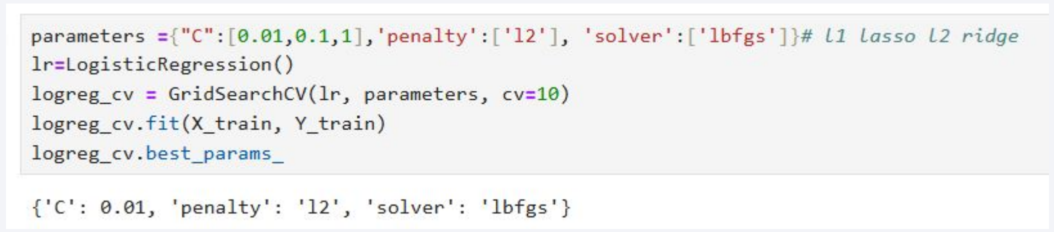
- For all 4 models the performance and the confusion matrix are the same. We predict landings correctly but sometimes we predict launches where the first stage did not land properly as it had landed properly. We are a little biased to return false positives

Conclusions

- All 4 models perform similarly to the point where their confusion matrixes are the same.
- These models tend to predict successful launches with 100% accuracy, however they sometimes predict and successful launches instances where the first stage will not land properly
- So we will be a little biased towards thinking launches will end with the first stage landing. Not allowing us to predict some cases where we will lose the first launch and therefore not being able to avoid it
- As time passes, success rates for launches increase
- Orbits ES-L1, GEO, HEO, SSO have the highest success rate
- For heavy payloads the successful landing rate is higher for Polar, LEO and ISS.

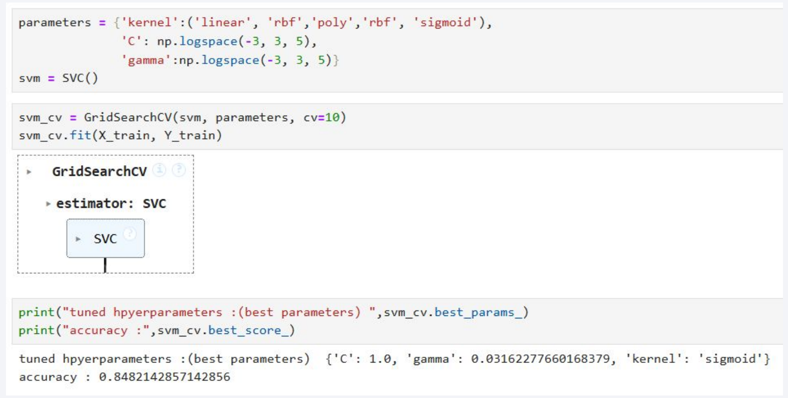
Appendix

Best parameters for each model



```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
logreg_cv.best_params_
```

```
{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
```

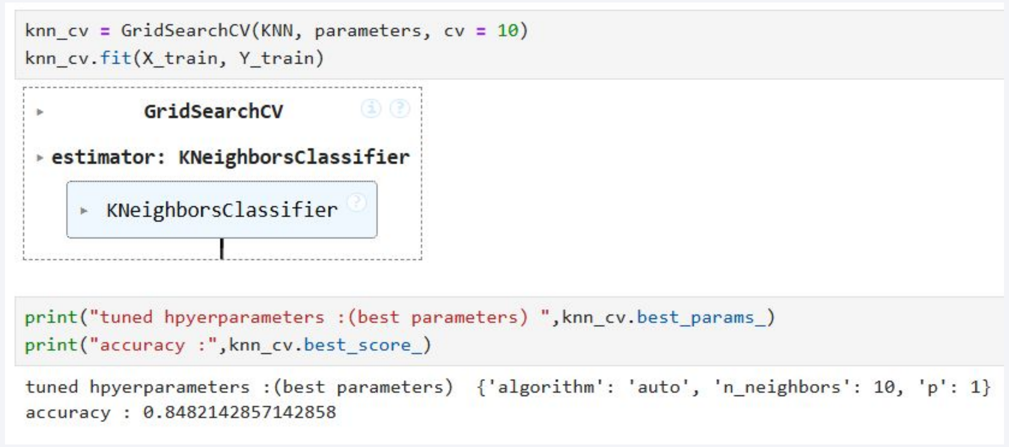


```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

```
svm_cv = GridSearchCV(svm, parameters, cv=10)
svm_cv.fit(X_train, Y_train)
```

```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

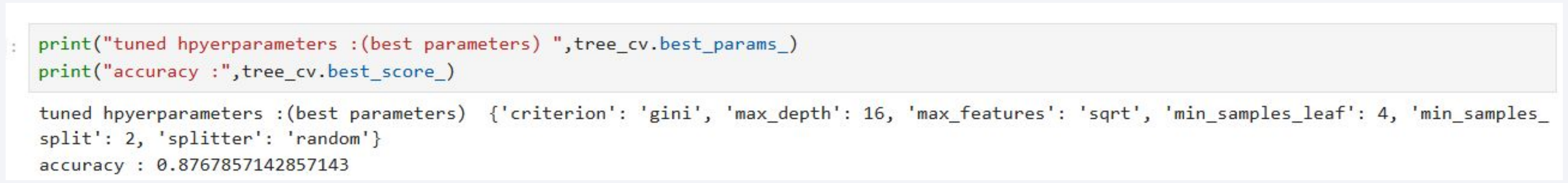
```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```



```
knn_cv = GridSearchCV(KNN, parameters, cv = 10)
knn_cv.fit(X_train, Y_train)
```

```
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```



```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 16, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_
split': 2, 'splitter': 'random'}
accuracy : 0.8767857142857143
```

Thank you!