IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Corinna
1 Jan 2022

Outline

Executive Summary

- Summary of methodologies
    - Data collection
    - Data wrangling
    - Exploratory data analysis (EDA) using visualization and SQL
    - Interactive visual analytics using Folium and Plotly Dash
    - Predictive analysis using classification models
- Summary of all results
    - EDA and interactive visualizations
    - Evaluation of predictive analysis models

Introduction

- **Project background and context**
  - To predict if Falcon 9 first stage lands successfully
  - SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage.
  - Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.
- **Problem statements to find solutions**
  - What influences the success rate of landing?
  - Relationship of variables (e.g. payload mass, orbit, etc) with rate of successful launches
  - Conditions to ensure best landing

Section 1

Methodology

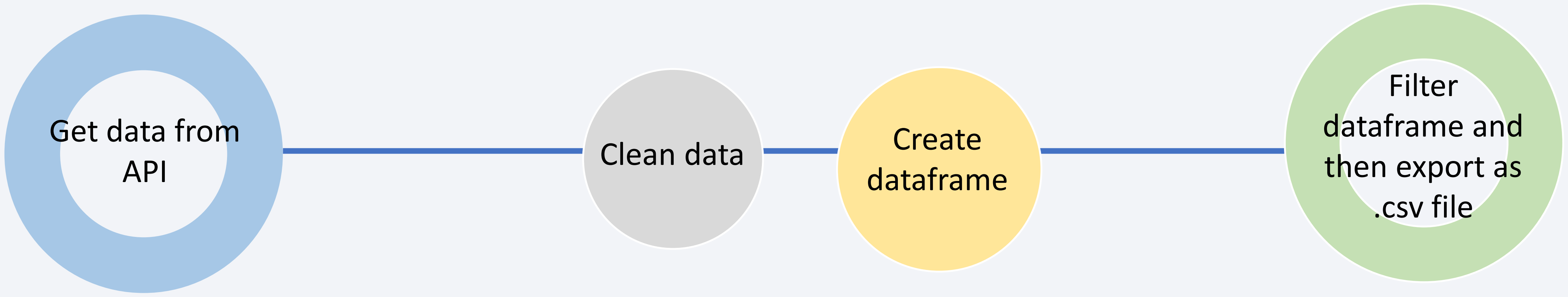
Methodology

Executive Summary

- Data collection methodology:
    - SpaceX Rest API
    - Web Scraping
- Perform data wrangling
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection

- SpaceX data sets were collected from SpaceX REST API .



Data Collection – SpaceX API

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

```
response = requests.get(spacex_url)
```

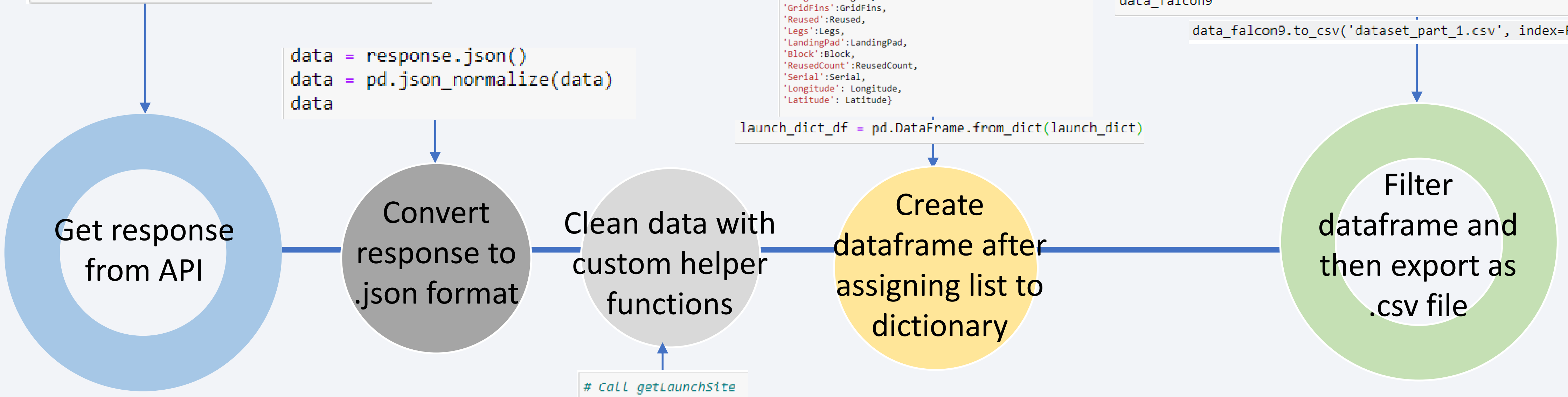
```
data = response.json()
data = pd.json_normalize(data)
data
```

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

```
launch_dict_df = pd.DataFrame.from_dict(launch_dict)
```

```
data_falcon9 = launch_dict_df[launch_dict_df['BoosterVersion']=='Falcon 9']
data_falcon9
```

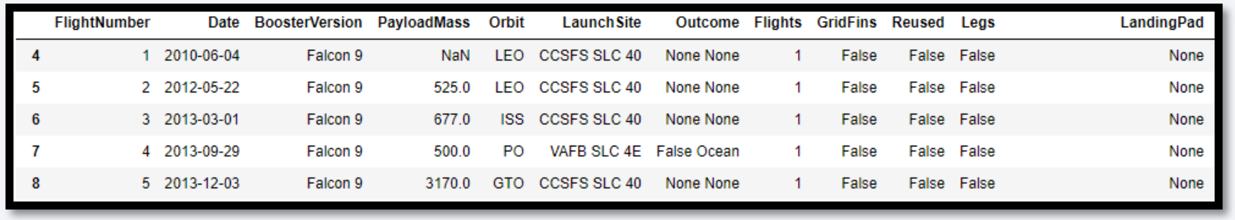
```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```


Get response from API
Convert response to .json format
Clean data with custom helper functions
Create dataframe after assigning list to dictionary
Filter dataframe and then export as .csv file

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```



	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None

Github URL

Data Collection - Scraping

1. Get response from HTML page

```
response = requests.get(static_url)
```

2. Create a BeautifulSoup object from the response

```
soup = BeautifulSoup(response.text, "html.parser")
```

3. Find all tables on the HTML page

```
html_tables = soup.find_all('table')
```

```
first_launch_table = html_tables[2]
```

4. Collect all relevant column names

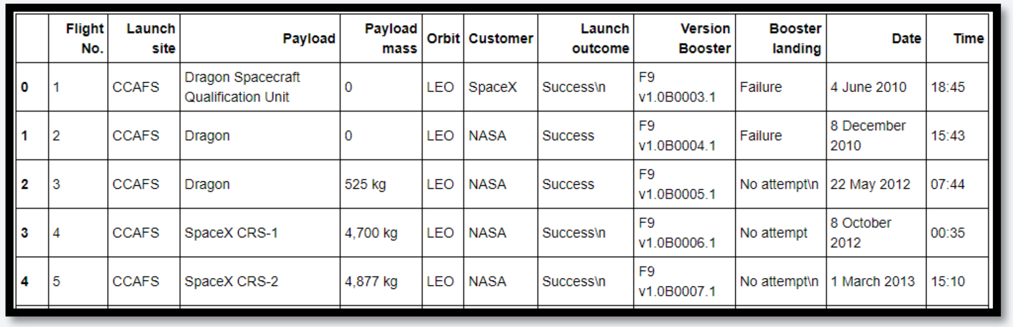
```
for tag in first_launch_table.find_all("th"):
    name = extract_column_from_header(tag)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

5. Create a dataframe from dictionary

```
launch_dict= dict.fromkeys(column_names)
```

```
df = pd.DataFrame.from_dict(launch_dict,orient='index')
df = df.transpose()
```

6. Export dataframe to CSV



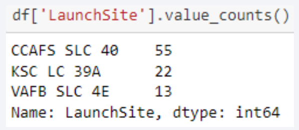
	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

Github URL

Data Wrangling

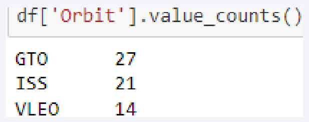
- Data wrangling is process of cleaning and structuring raw, messy datasets into a convenient format for downstream analysis.

1. **Calculate the number of launches on each site**



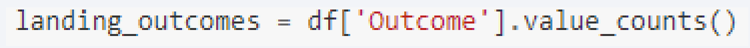
```
df['LaunchSite'].value_counts()

CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

2. **Calculate the number and occurrence of each orbit**



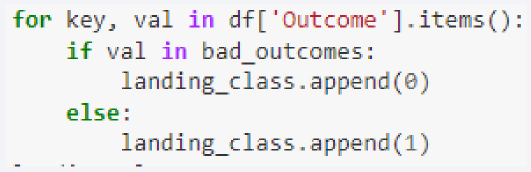
```
df['Orbit'].value_counts()

GTO      27
ISS      21
VLEO     14
```

3. **Calculate the number and occurence of mission outcome per orbit type**



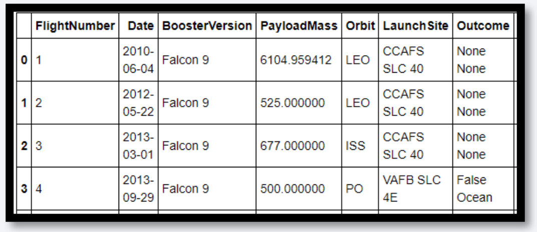
```
landing_outcomes = df['Outcome'].value_counts()
```

4. **Create a landing outcome label from Outcome column**



```
for key, val in df['Outcome'].items():
    if val in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

5. **Export dataframe as CSV**

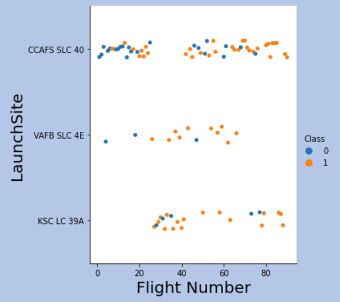


	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean

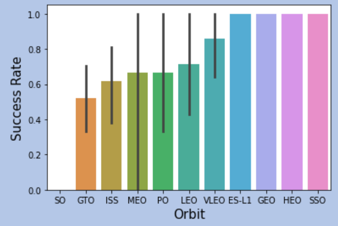
Github URL

EDA with Data Visualization

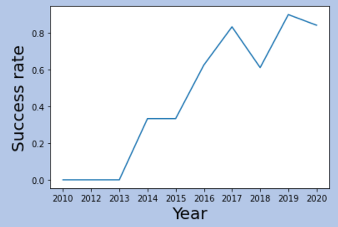
- **Scatterplots**
  - Flight Number vs. Launch Site
  - Payload vs. Launch Site
  - Flight Number vs. Orbit Type
  - Payload vs. Orbit Type
- **Bar charts**
  - Success Rate vs. Orbit Type
- **Line chart**
  - Launch Success Yearly Trend



Scatterplots display the correlation between numerical variables.



Bar charts help to visually check if there are any direct trend or relationship between attributes (e.g. compare outcome in different categories)



Line charts give predictive insight in terms of showing the general trend or pattern.

Github URL

EDA with SQL

SQL QUERIES PERFORMED

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

Github URL

Build an Interactive Map with Folium

Object	**Attribute**	**Output on map**
Map marker	folium.Marker()	Map an object to mark on the map
Circle marker	folium.Circle()	Create a circle where marker is located
Icon marker	folium.Icon()	Create an icon on the map
PolyLine	folium.PolyLine()	Draw a line between selected points
MarkerCluster	MarkerCluster()	Simplify a map containing many markers having the same coordinate.

Github URL

Build a Dashboard with Plotly Dash

- Pie charts
  - Total successful launches for all sites
  - Relative proportions for launch successes vs launch failures per site
- Scatterplots (with slider function to change payload mass range)
  - Payload mass (kg) vs launch outcome for different booster versions
  - Non-linear relationship between payload mass and launch success

Method	Purpose
Dropdown	Create Dropdown menu to select site of interest
RangeSlider()	Create rangeslider for payload mass (kg) range
px.pie()	Create piechart for successful launches at different sites
px.scatter()	Create scatterplot to show relationship between payload mass (kg) and
@app.callback()	Update the property of the output component with what was returned by the executed function

Link to Dashboard

Github URL

Predictive Analysis (Classification)

Build model — **Evaluate model** — **Improve model** — **Find best model**

Build model

- Transform data into Numpy arrays
- Standardize data
- Split data for training and testing
- Set parameters and methods in GridSearchCV
- Train model after fitting data into GridSearchCV

Evaluate model

- Check accuracy per model
- Get best parameters for each method
- Plot the confusion matrix

Improve model

- Tuning the hyperparameters

Find best model

- Choose best performing model based on highest accuracy score

Github URL

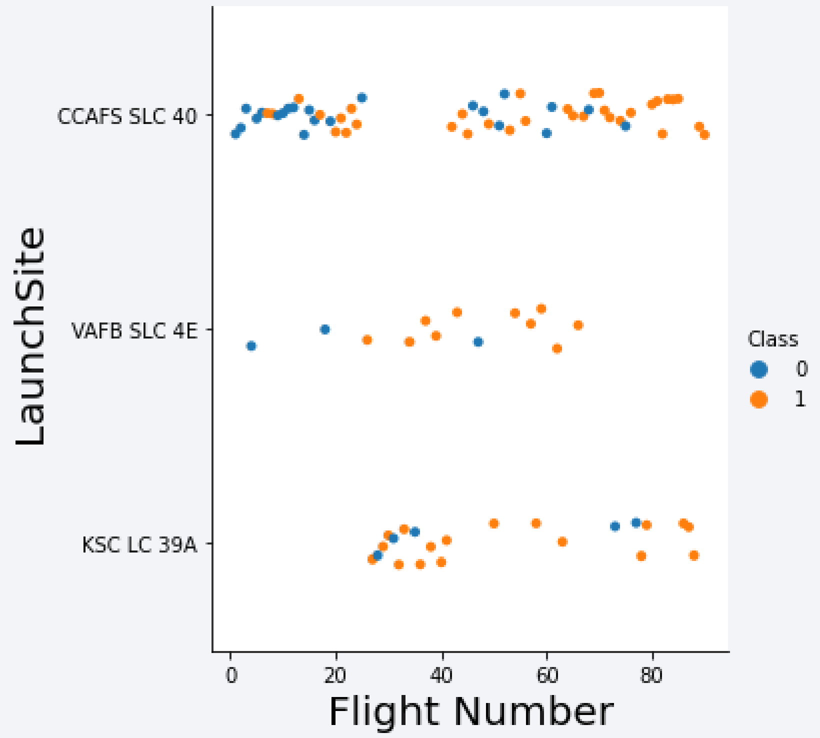
Results

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

Section 2

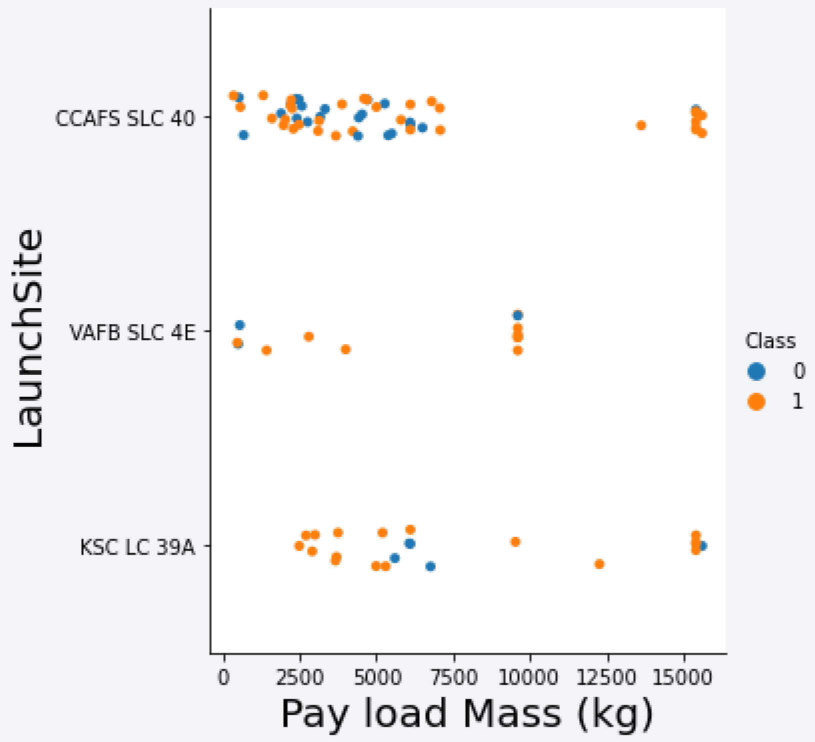
Insights drawn from EDA

Flight Number vs. Launch Site



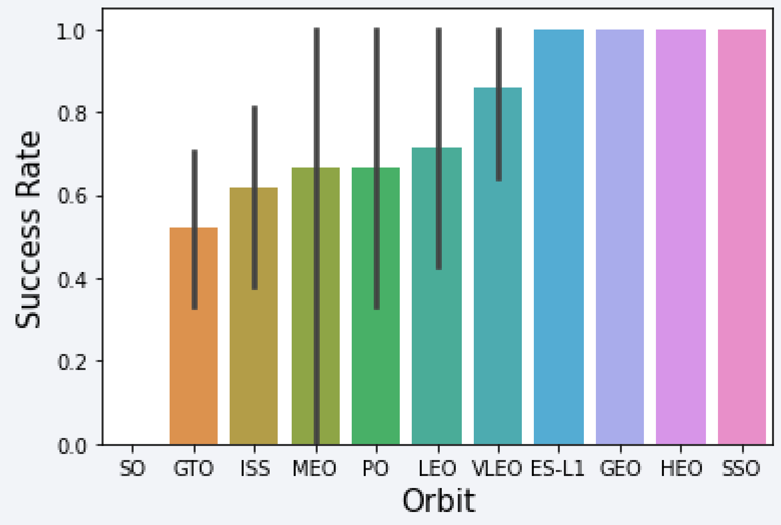
- The greater the number of flights, the higher likelihood of a successful landing at a given launch site.

Payload vs. Launch Site



- The more massive the payload, the less likely the first stage will return.
- However, there is no definitive conclusion as to whether the La

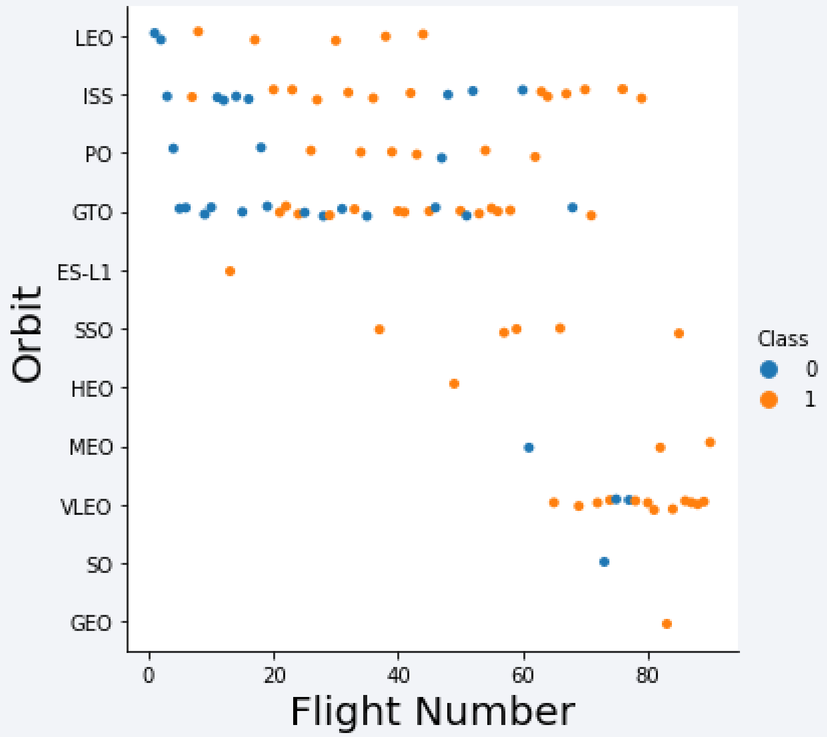
Success Rate vs. Orbit Type



Arranged by increasing success rates

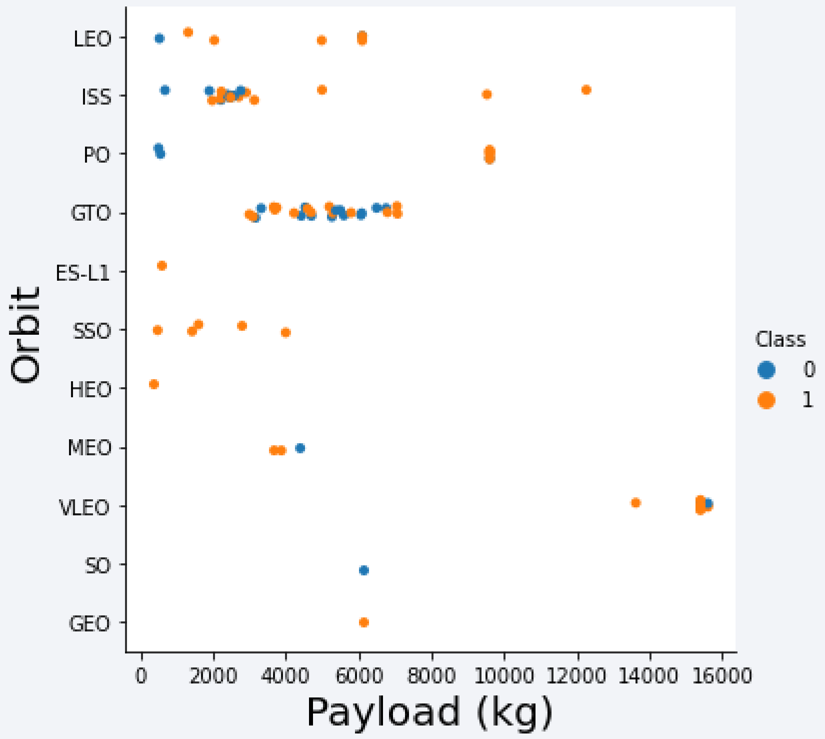
- Orbit types ES-L1, GEO, HEO and SSO have the highest success rates.

Flight Number vs. Orbit Type



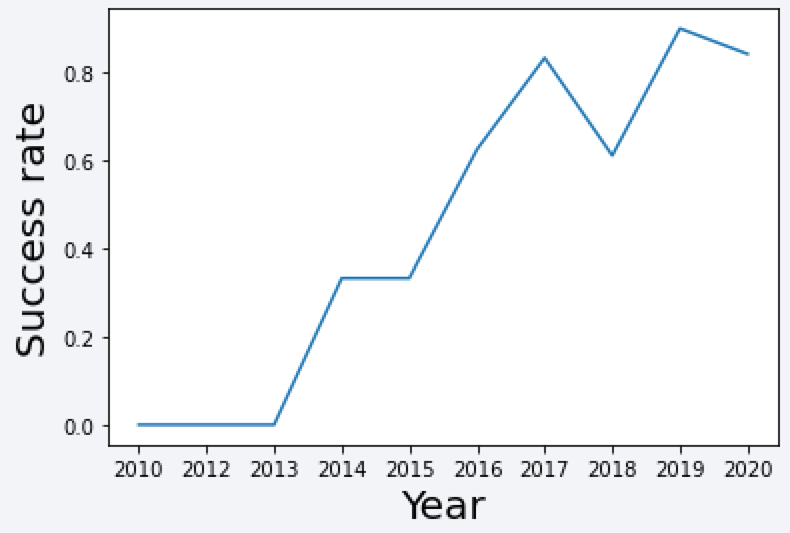
- For the LEO orbit, increasing number of flights result in better success rate.
- The GTO orbit does not show this relationship between flight number and successful landing.

Payload vs. Orbit Type



- With heavy payloads, there is higher successful landing rate for PO, LEO and ISS.
- However, this is not applicable for GTO as there is no distinct relationship between successful missions with payload mass.

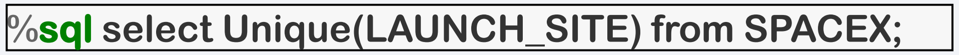
Launch Success Yearly Trend



- Success rate since 2013 kept increasing till 2020

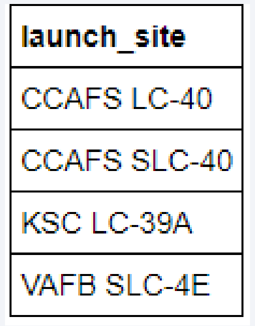
All Launch Site Names

- SQL Query



```
%sql select Unique(LAUNCH_SITE) from SPACEX;
```

- Output →

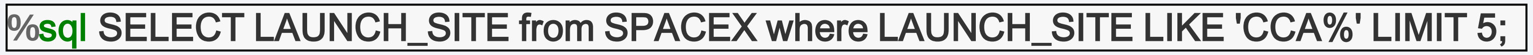


launch_site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

- Pull unique values for launch sites using the UNIQUE constraint from the column LAUNCH_SITE from the table SPACEX

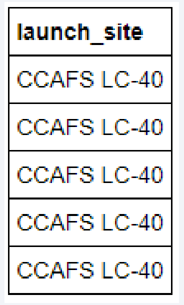
Launch Site Names Begin with 'CCA'

- SQL Query



```
%sql SELECT LAUNCH_SITE from SPACEX where LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

- Output →



launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

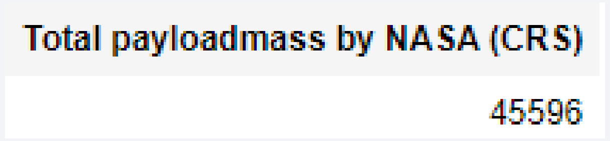
- Fetch 5 records from the table SPACEX where launch sites begin with `CCA` using the condition LIKE keyword with the wildcard 'CCA%' for pattern matching.
- The '%' symbol at the end of the wildcard indicates that the launch site name must start and match with the character string before '%'.

Total Payload Mass

- SQL Query

```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEX where CUSTOMER = 'NASA (CRS)'
```

- Output →

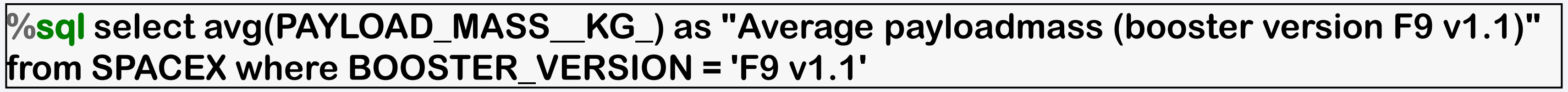


Total payloadmass by NASA (CRS)
45596

- The function sum calculates total values in the column PAYLOAD_MASS__KG_ from table SPACEX and filters the data to fetch Customer name ‘NASA (CRS)’ with the WHERE clause.

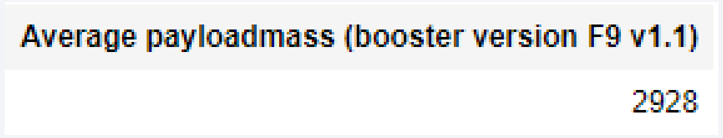
Average Payload Mass by F9 v1.1

- SQL Query



```
%sql select avg(PAYLOAD_MASS__KG_) as "Average payloadmass (booster version F9 v1.1)"
from SPACEX where BOOSTER_VERSION = 'F9 v1.1'
```

- Output →



Average payloadmass (booster version F9 v1.1)
2928

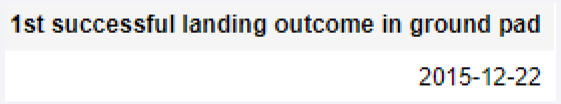
- The function AVG calculates the average value in the the column PAYLOAD_MASS__KG_ while the WHERE clause filters the data to only capture the values from booster version F9 v1.1.

First Successful Ground Landing Date

- SQL Query

```
%sql select min(DATE) as "1st successful landing outcome in ground pad" from SPACEX
where landing__outcome = 'Success (ground pad)'
```

- Output →



1st successful landing outcome in ground pad
2015-12-22

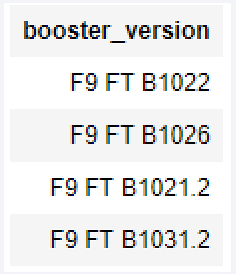
- The function MIN finds the minimum date of the landing__outcome column while the WHERE clause filters the landing__outcome column containing the string 'Success (ground pad)'

Successful Drone Ship Landing with Payload between 4000 and 6000

- SQL Query

```
%sql select BOOSTER_VERSION from SPACEX where LANDING__OUTCOME LIKE 'Success
(drone ship)%' and payload_mass__kg_ BETWEEN 4000 and 6000;
```

- Output →



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

- Select the BOOSTER_VERSION column from table SPACEX.
- The WHERE clause adds additional filters to restrict the landing__outcome column to contain the string 'Success (drone ship)', along with the AND clause to include the additional filter for payload_mass__kg_ to be in the range values (greater than 4000 but less than 6000) specified by the BETWEEN clause.

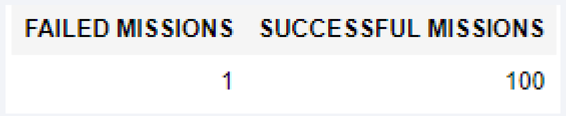
Total Number of Successful and Failure Mission Outcomes

- SQL Query

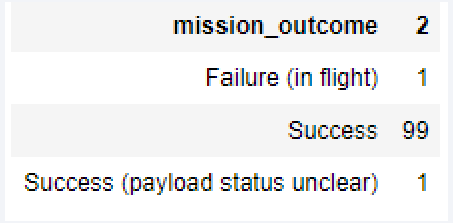
```
%sql SELECT sum(case when MISSION_OUTCOME LIKE 'Failure%' then 1 else 0 end) AS "FAILED MISSIONS",
sum(case when MISSION_OUTCOME LIKE 'Success%'then 1 else 0 end) AS "SUCCESSFUL MISSIONS"
FROM SPACEX
```

```
%sql select MISSION_OUTCOME, count(*) from SPACEX GROUP BY MISSION_OUTCOME
```

- Output →



FAILED MISSIONS	SUCCESSFUL MISSIONS
1	100



mission_outcome	2
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

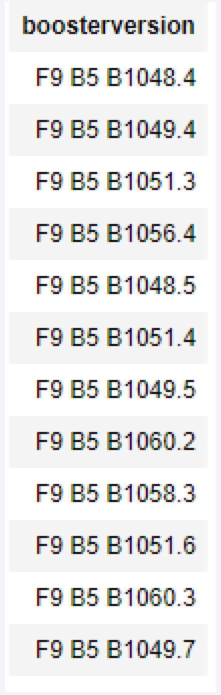
- We can use the GROUP BY clause to group the types available in the mission_outcome column and COUNT(*) to get the number of times a specific type appears in the selected column MISSION_OUTCOME
- For a more specific query to include only success and failure outcomes, we can use the LIKE wildcard to include the strings 'Success%' or 'Failure%' stated along with the CASE statement to return the values if the condition is met.

Boosters Carried Maximum Payload

- SQL Query

```
%sql select BOOSTER_VERSION as boosterversion from SPACEX where
PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEX);
```

- Output →



boosterversion
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

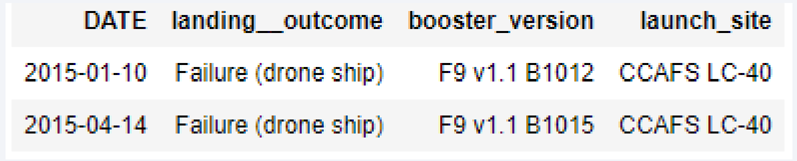
We can use MAX function to calculate the maximum payload in the PAYLOAD_MASS__KG_ column with the WHERE clause to filter the BOOSTER_VERSION containing that maximum payload.

2015 Launch Records

- SQL Query

```
%sql select DATE, landing__outcome, booster_version, launch_site from SPACEX WHERE
landing__outcome LIKE 'Fail%' and DATE LIKE '2015%'
```

- Output →



DATE	landing__outcome	booster_version	launch_site
2015-01-10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
2015-04-14	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

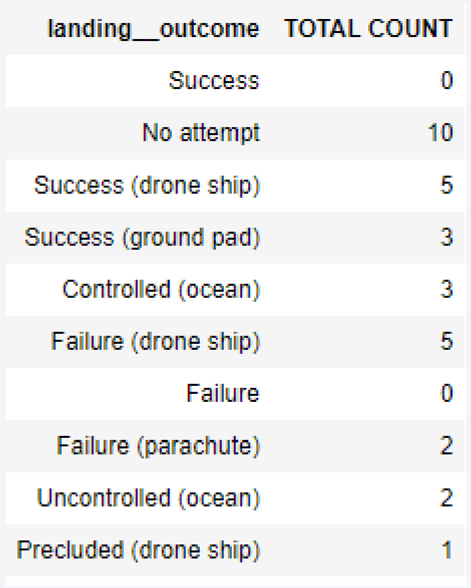
- We can list the failed landing_outcomes in drone ship, their booster versions, and launch site names in the year 2015 and the corresponding dates using the LIKE operator with wildcard characters '%'

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

- SQL Query

```
%sql SELECT LANDING__OUTCOME, sum(case when DATE BETWEEN '2010-06-04' AND
'2017-03-20' then 1 else 0 end) AS "TOTAL COUNT" FROM SPACEX GROUP BY
LANDING__OUTCOME ORDER BY COUNT(LANDING__OUTCOME) DESC;
```

- Output →



landing__outcome	TOTAL COUNT
Success	0
No attempt	10
Success (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Failure (drone ship)	5
Failure	0
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

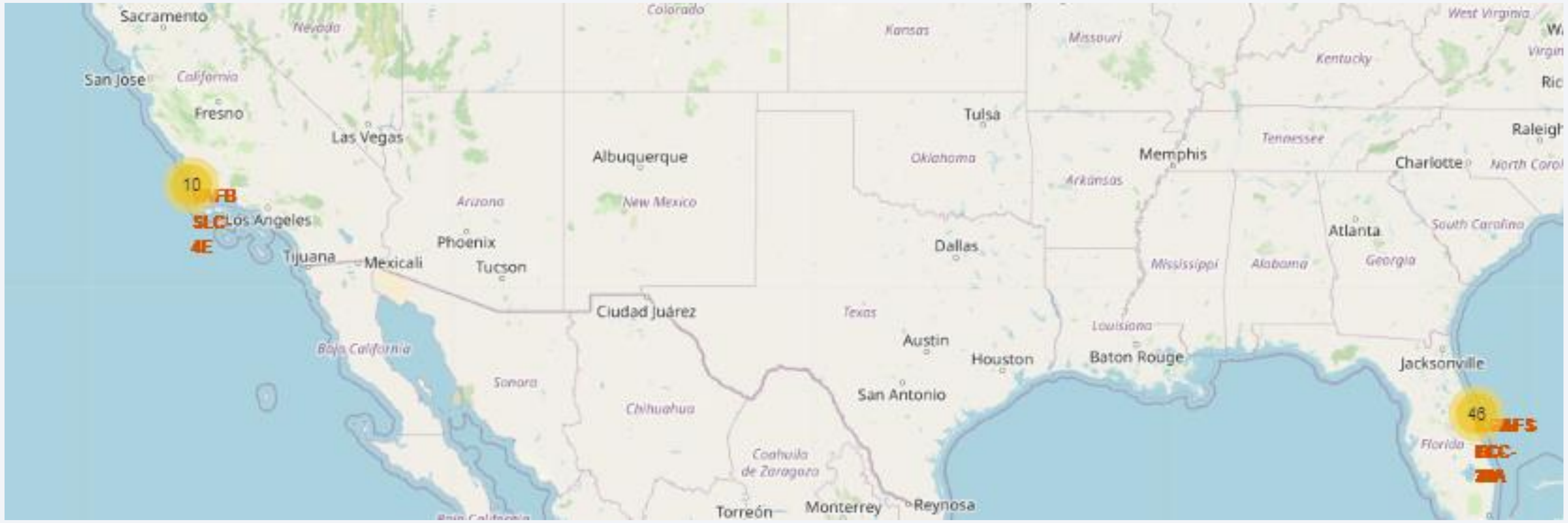
- First, select the LANDING__OUTCOME column and use the CASE statement to fulfil the condition of getting the counts of landing outcomes between the date 2010-06-04 and 2017-03-20
- GROUP BY the types listed in LANDING__OUTCOME column
- We can then rank the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order with the ORDER BY and DESC clauses.

Section 4

Launch Sites Proximities Analysis

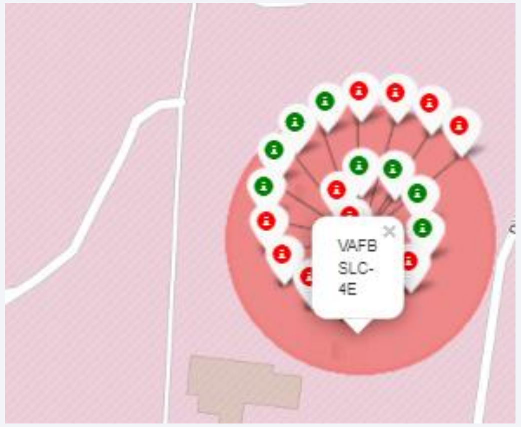
SpaceX launch sites on the global map

- On the global map, SpaceX launch sites are located at the coastlines of U.S. states California and Florida.

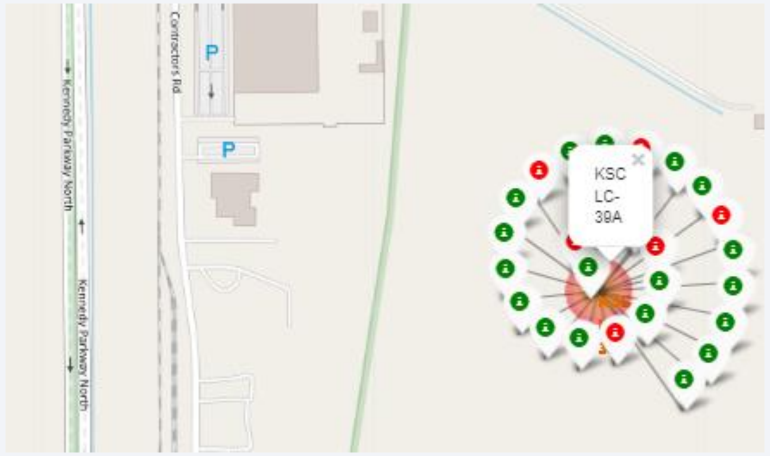


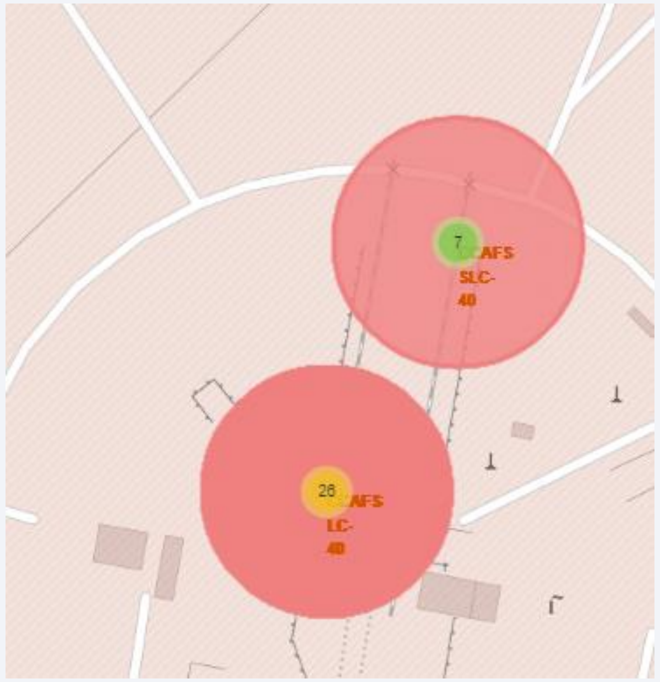
Colour-labeled launch outcomes

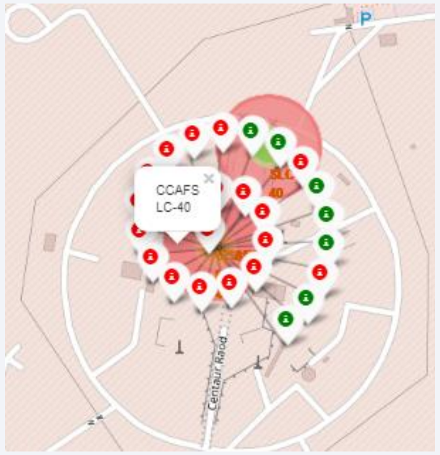
California

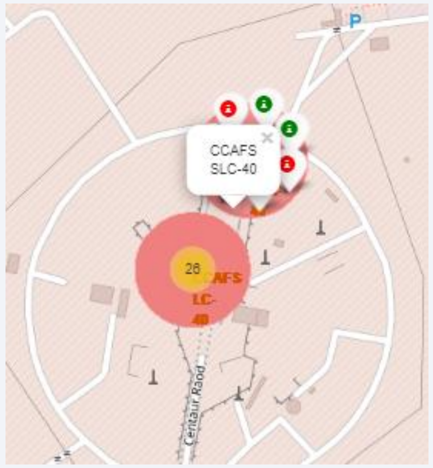


Florida



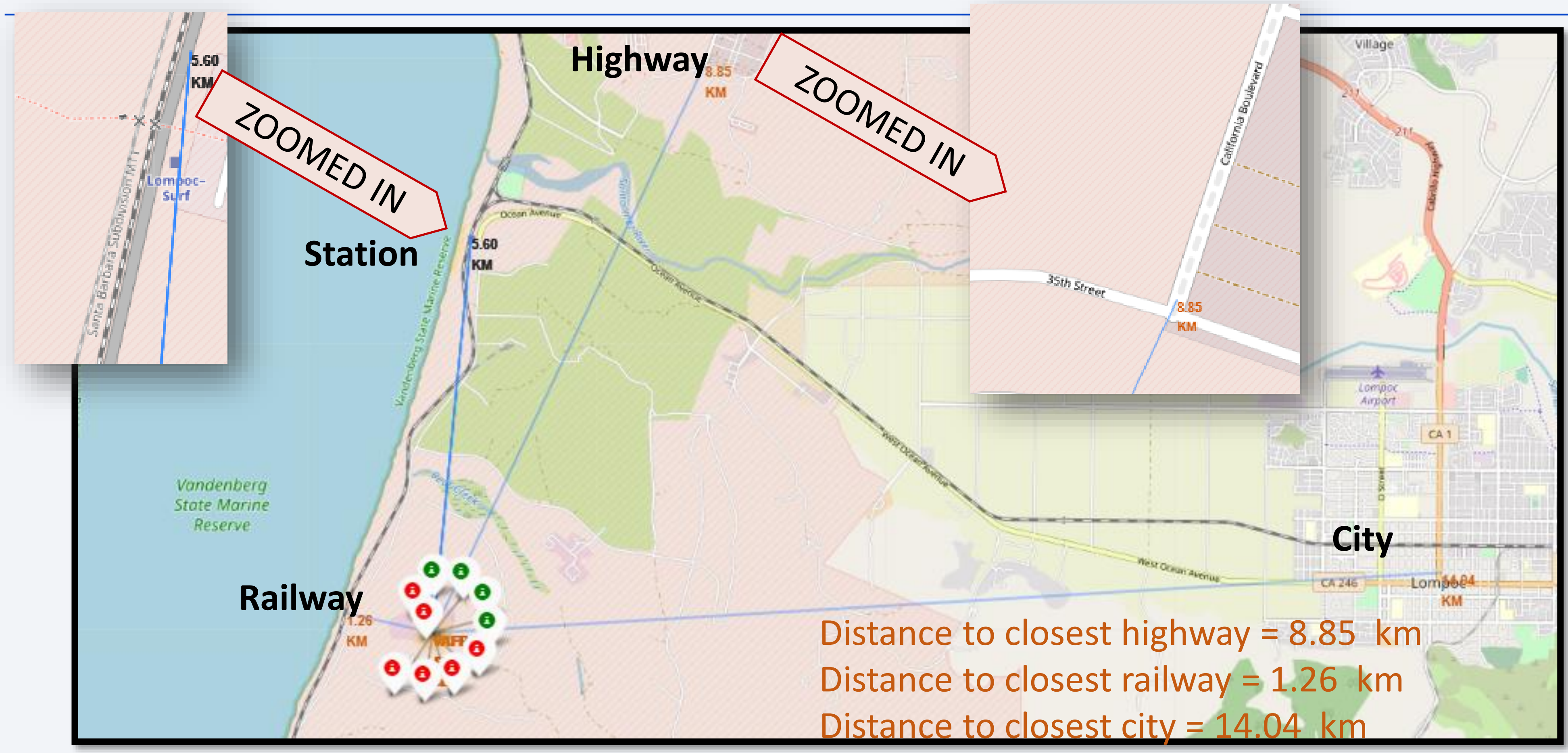






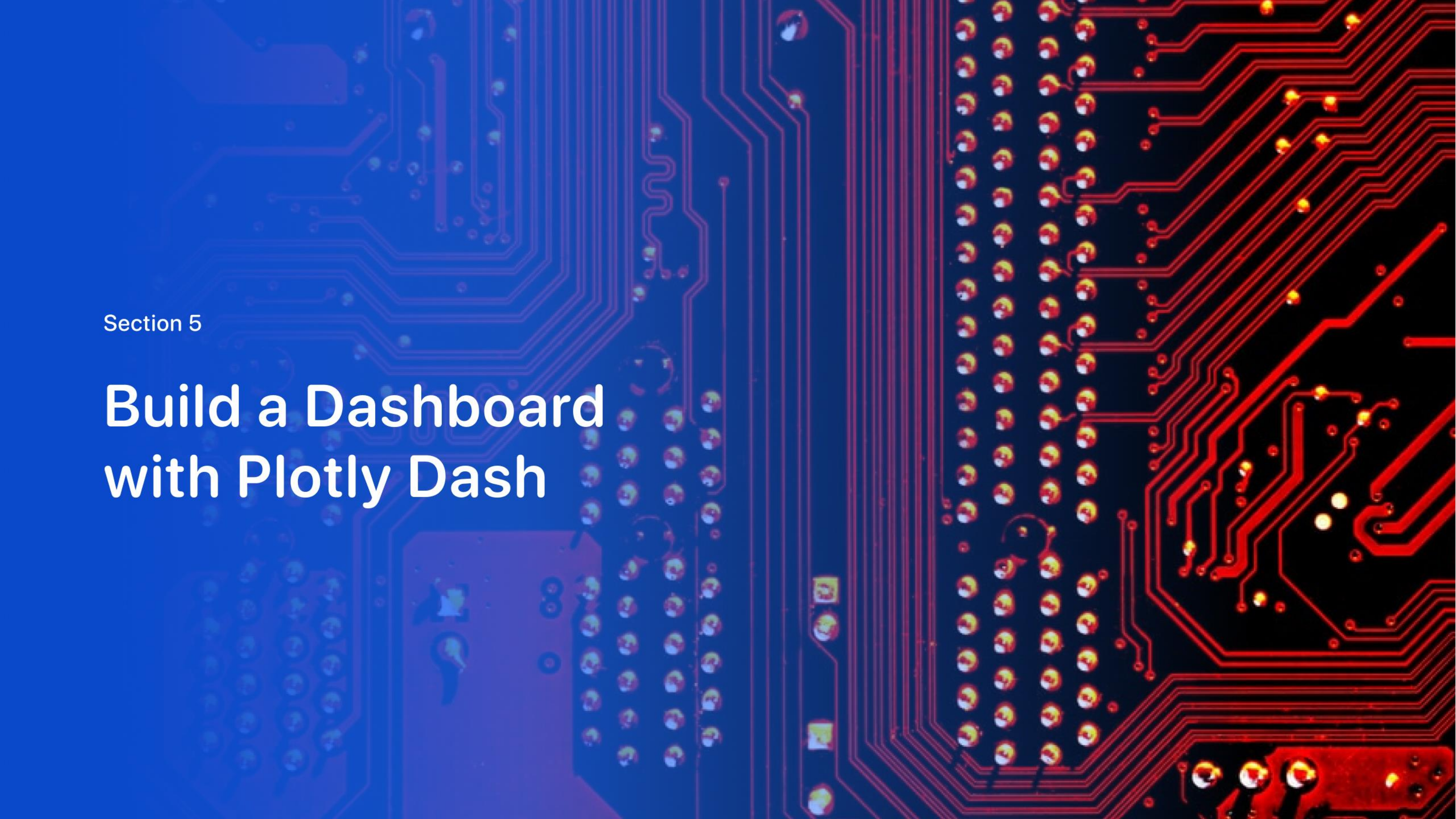
Green markers indicate successful launches while red markers indicate failed launches

Launch site proximities to railway, highway, coastline



Distance to closest highway = 8.85 km
Distance to closest railway = 1.26 km
Distance to closest city = 14.04 km

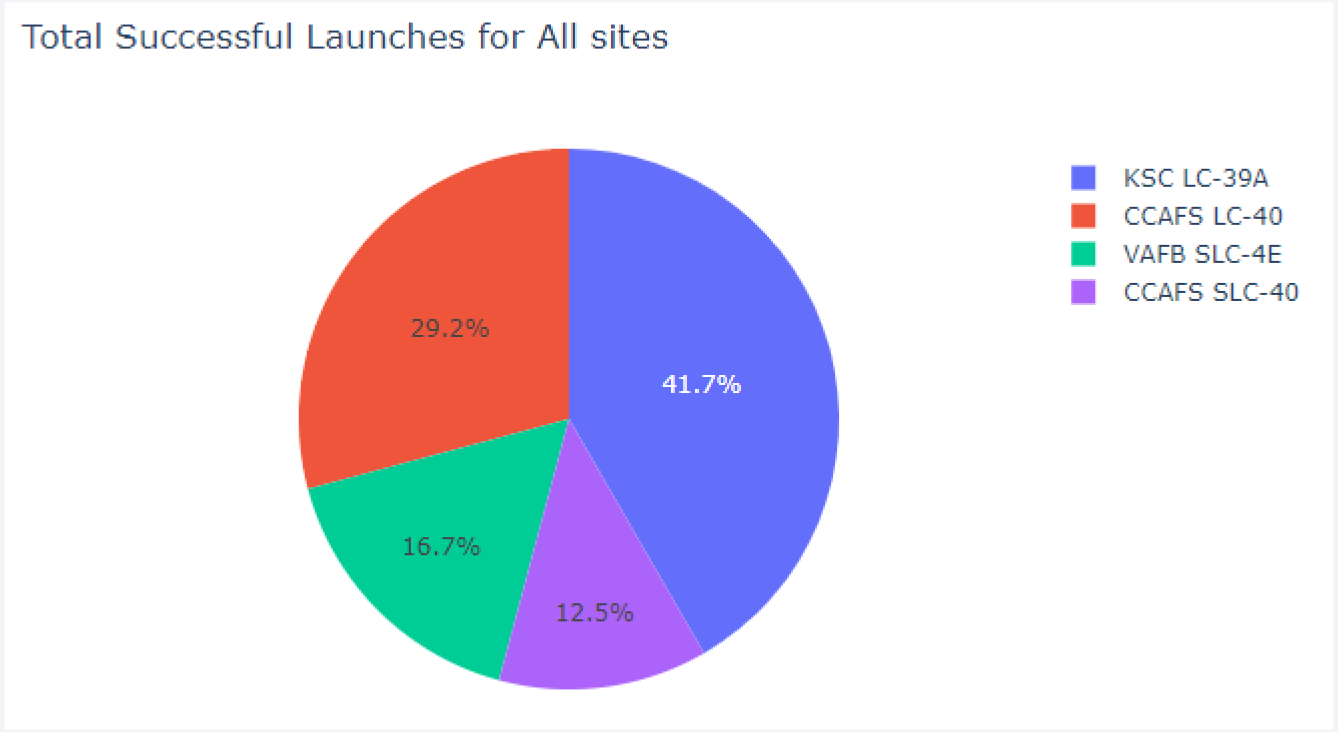
- Using VAFB SLC-4E launch site as a reference, the its proximities to railway, highway, coastline can be displayed with distance calculated.
- Launch site is situated away from cities and highways but railroads might be an exception if those railroads are along the coastal lines.



Section 5

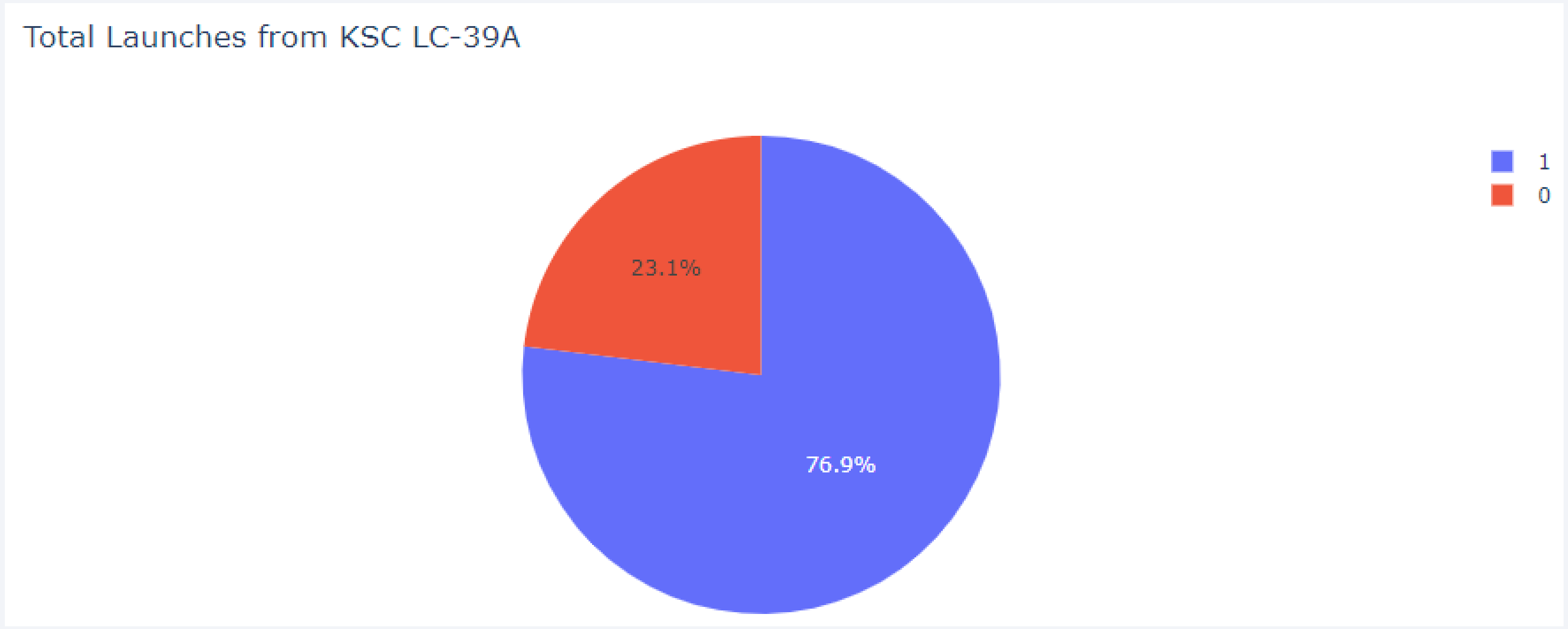
Build a Dashboard with Plotly Dash

<Dashboard> Percentage of launch success for all sites



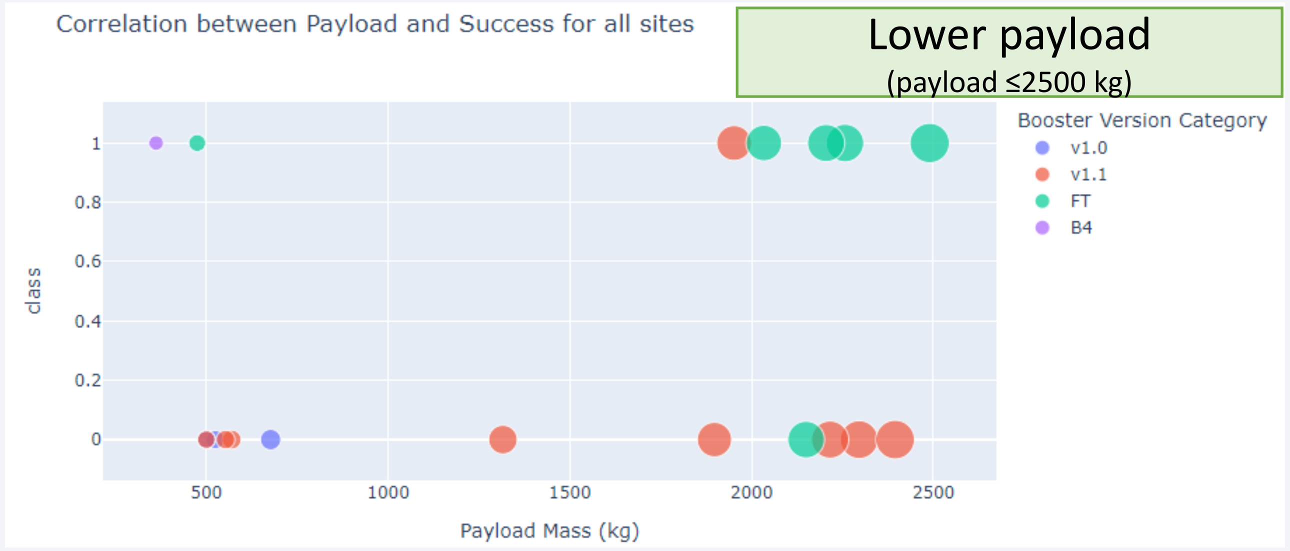
- KSC LC-39-A had the most successful launches while CCAFS SLC-40 had the least.

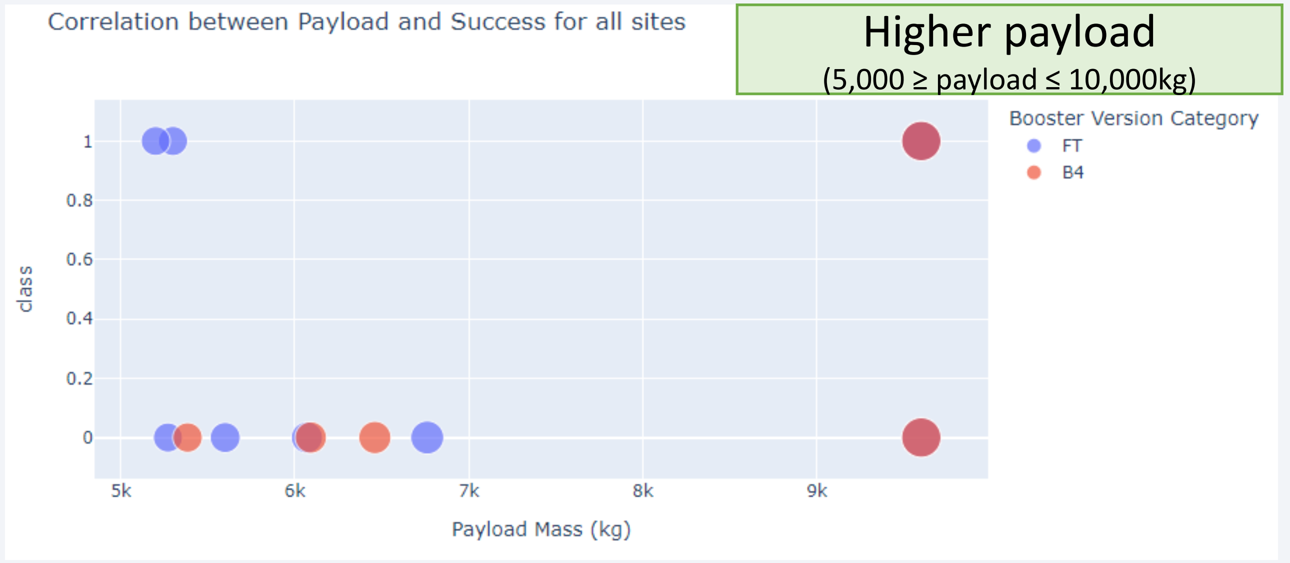
<Dashboard> Site with highest launch success ratio



- KSC LC-39-A had the highest launch success ratio (76.9% success, 23.1% failure).

<Dashboard> Payload vs Launch Outcome for all sites





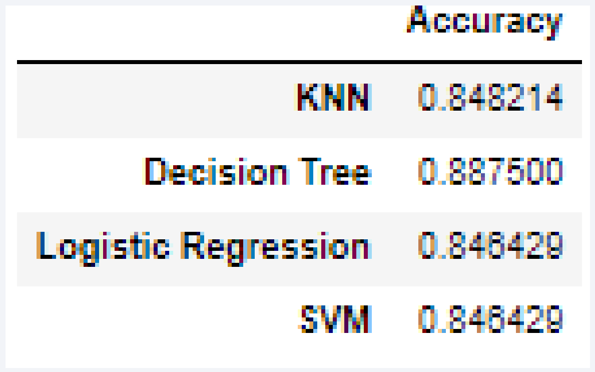
- Lower payloads result in higher success rate for all sites.

Section 6

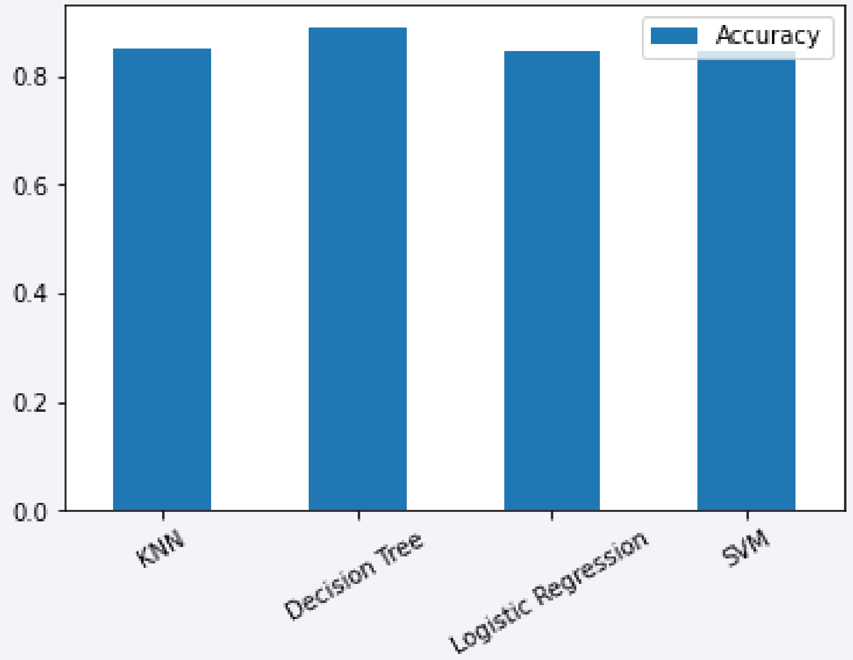
Predictive Analysis (Classification)

Classification Accuracy

- Decision Tree scored the best (accuracy of 88.75% out of the 4 methods.



	Accuracy
KNN	0.848214
Decision Tree	0.887500
Logistic Regression	0.846429
SVM	0.846429

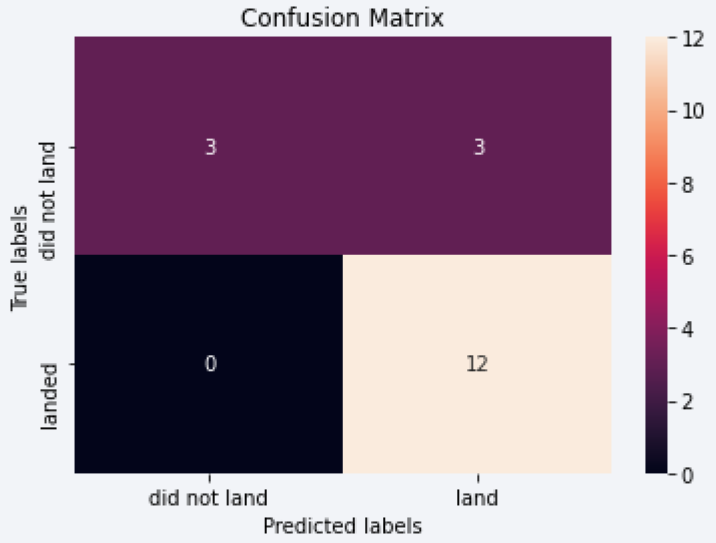


METHOD	TUNED HYPERPARAMS
Logistic regression	{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
Support vector machine	{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
Decision tree	{'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'random'}
KNN	{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}

Confusion Matrix

- The confusion matrix describes the performance of a classification model for a test data set where the true values are known.
- Based on a table of 4 different combinations of predicted and actual values, this indicates the correct and wrong predictions made by the classifier.
- In this case, all models produced the same confusion matrix (number of correctly/incorrectly predicted labels).

Best model: Decision Tree



Predicted labels

Actual labels

TN true negative	FP false positive
FN false negative	TP true positive

Conclusions

- Orbits ES-L1, GEO, HEO, SSO have the highest success rates.
- The success rates for SpaceX launches are improving over the years.
- KSC LC-39A had the most successful launches from all the sites.
- Lower payloads are associated with more successful launches.
- Launch sites are typically located away from cities and highways, but more closely located to the coastlines.
- The Decision Tree is the best classifier for Machine Learning Prediction for this dataset.

Appendix

- PythonAnywhere for hosting interactive web application
  - Config file (wsgi.py)
  - Executable file (plotly_dashboard.py)
- Reformatting data

```
LaunchSites = []
LaunchSites.append({'label': 'All Sites', 'value': 'ALL'})
for site in uniqueLaunchSites:
    LaunchSites.append({'label': site, 'value': site})
```

- SQL queries

```
%sql SELECT sum(case when MISSION_OUTCOME LIKE 'Failure%' then 1 else 0 end) AS "FAILED MISSIONS",
sum(case when MISSION_OUTCOME LIKE 'Success%'then 1 else 0 end) AS "SUCCESSFUL MISSIONS"
FROM SPACEX

%sql select DATE, landing__outcome, booster_version, launch_site from SPACEX WHERE landing__outcome LIKE 'Fail%' and DATE LIKE '2015%'

%sql SELECT LANDING__OUTCOME, sum(case when DATE BETWEEN '2010-06-04' AND '2017-03-20' then 1 else 0 end) AS "TOTAL COUNT" FROM
SPACEX GROUP BY LANDING__OUTCOME ORDER BY COUNT(LANDING__OUTCOME) DESC;

%sql select MISSION_OUTCOME, count(*) from SPACEX GROUP BY MISSION_OUTCOME
```

Thank you!