IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<AC>
<1st Dec 2024>

Outline

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

Executive Summary

- Summary of methodology:
  - Using API to get the launch data and results from SpaceX
  - Grid search, logistic regression, decision tree, Support Vector Machine (SVM), k-nearest neighbors
- Summary of all results
- Launch success rate has been increasing
- ES-L1, GEO, HEO and SSO orbit type has had 100% success rate
- KSC LC-39A had the highest launch success rate of 76.9%
- Decision tree classification model has the highest accuracy

Introduction

- Goal: to understand the relationships between factors like payload mass and launch site, and the launch outcome of Falcon 9
- Problems to find answers:
  - Launch success rate trend
  - What are the orbit types with the highest success rate?
  - Which launch site has the highest launch success rate?
  - Which classification model has the highest accuracy?

Section 1

Methodology

Methodology

Executive Summary

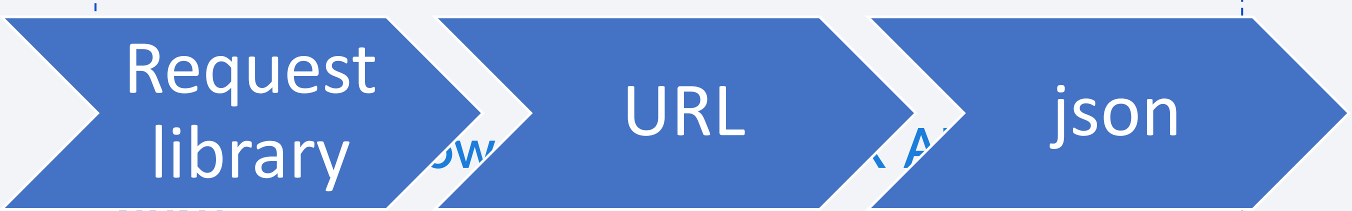
- Data collection methodology:
  - Use SpaceX API to download launch data of Falcon 9
- Perform data wrangling
  - Re-classifying the outcomes of launch into dummy variables 1 and 0
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Use logistic regression, decision tree, Support Vector Machine (SVM), k-nearest neighbors
  - Scale the input data and fit them in different classification models
  - Compare the scores of test data in different classification models

Data Collection

- Data sets were collected via SpaceX API or wiki page

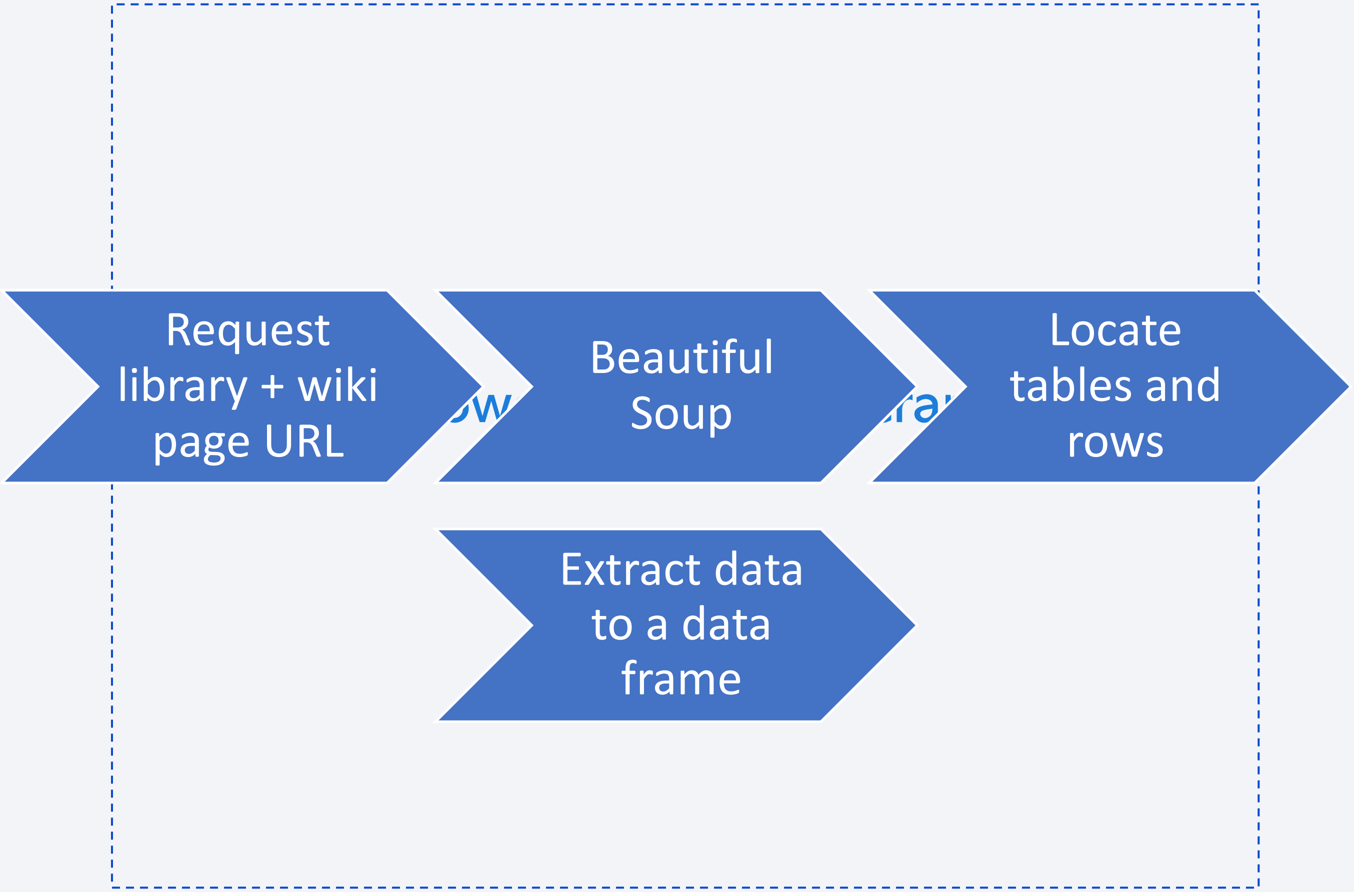
Data Collection – SpaceX API

- spacex_url=https://api.spacexdata.com/v4/launches/past
- response = requests.get(spacex_url)
- data_falcon9_ ['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, load_mean)
- Github link to data collection API notebook



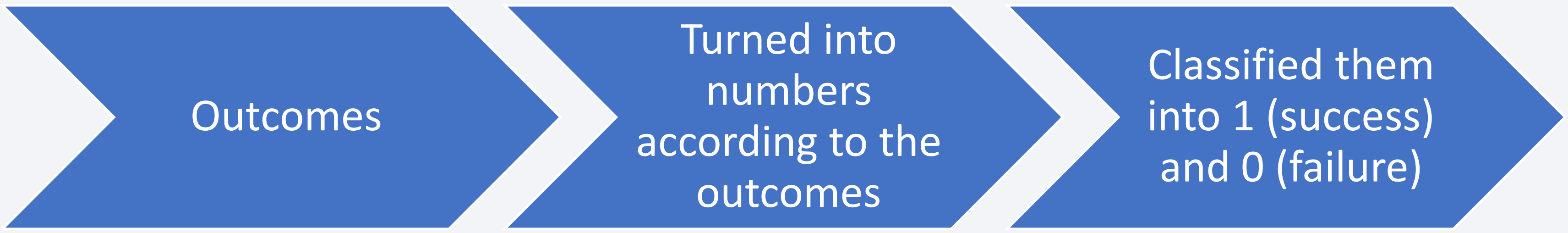
Data Collection - Scraping

- response = requests.get(static_url)
- soup = BeautifulSoup(response.text, 'html.parser')
- html_tables = soup.find_all('table', class_="wikitable plainrowheaders collapsible")
- [Github link to data collection scraping notebook]



Data Wrangling

- Missing values of Payload Mass have been replaced by the mean
- Outcomes of launches have been converted to 1 (success) and 0 (failure) and stored in ‘class’
- Github link to data wrangling notebook



EDA with Data Visualization

- **Flight number and launch site and success or not**: if there is a pattern of using a certain launch site and the success, and to see if a launch site has more success, and if the launches become more likely to succeed
- **Payload mass and launch site and success or not**: find out if there is a pattern of payload mass affecting success rate
- **Orbit type and success rate**: which orbit type is more likely to succeed: how the launch outcomes progress over time in different types of orbit
- **Flight number, orbit type and success or not** & **Payload mass, orbit type and success or not**: how orbit type affects success rate over time and its relationship with payload mass affecting success rate
- **Line chart of yearly success rate**: if success rate increases over the years
- Github link to EDA with data visualization notebook

EDA with SQL

- Find out total and sum of payload mass of a certain customer and booster version
- Date of first successful ground pad landing: SELECT MIN(Date) FROM SPACEXTABLE WHERE "Landing_Outcome" = \'Success (ground pad)\’
- Finding out booster version of successful drop ship landings given payload mass is between 4000 and 6000 kg
- Find out the month that there was landing failure in 2015
- Find out the most frequent landing outcomes and number of occurrences between 2010 June 4th and 2017 March 20th
- Gibhub link to EDA with SQL notebook

Build an Interactive Map with Folium

- Marker and circles: location of launch sites, adding each launch at its launch site on the map => see clear distribution of launches
- Adding ‘mouse position’ to the map so the coordinates of the position the mouse hovering will be shown => easy to find out coordinates to calculate distance
- Lines: displaying the distance from the launch site and other locations, e.g. railway => how close or far are the launch sites from other locations
- Github link to Folium map notebook

Build a Dashboard with Plotly Dash

- Pie chart of all sites : success launches of each launch site
- Or a pie chart of a selected site: success rate of launch at the site
- => how success rate changes in different sites. Find out the launch site with highest success rate
- Scatter chart of launches' payload mass and whether it's successful or not, along with the booster version
- => relationship between payload mass the launch outcome
- Github link to plotly dash

Predictive Analysis (Classification)

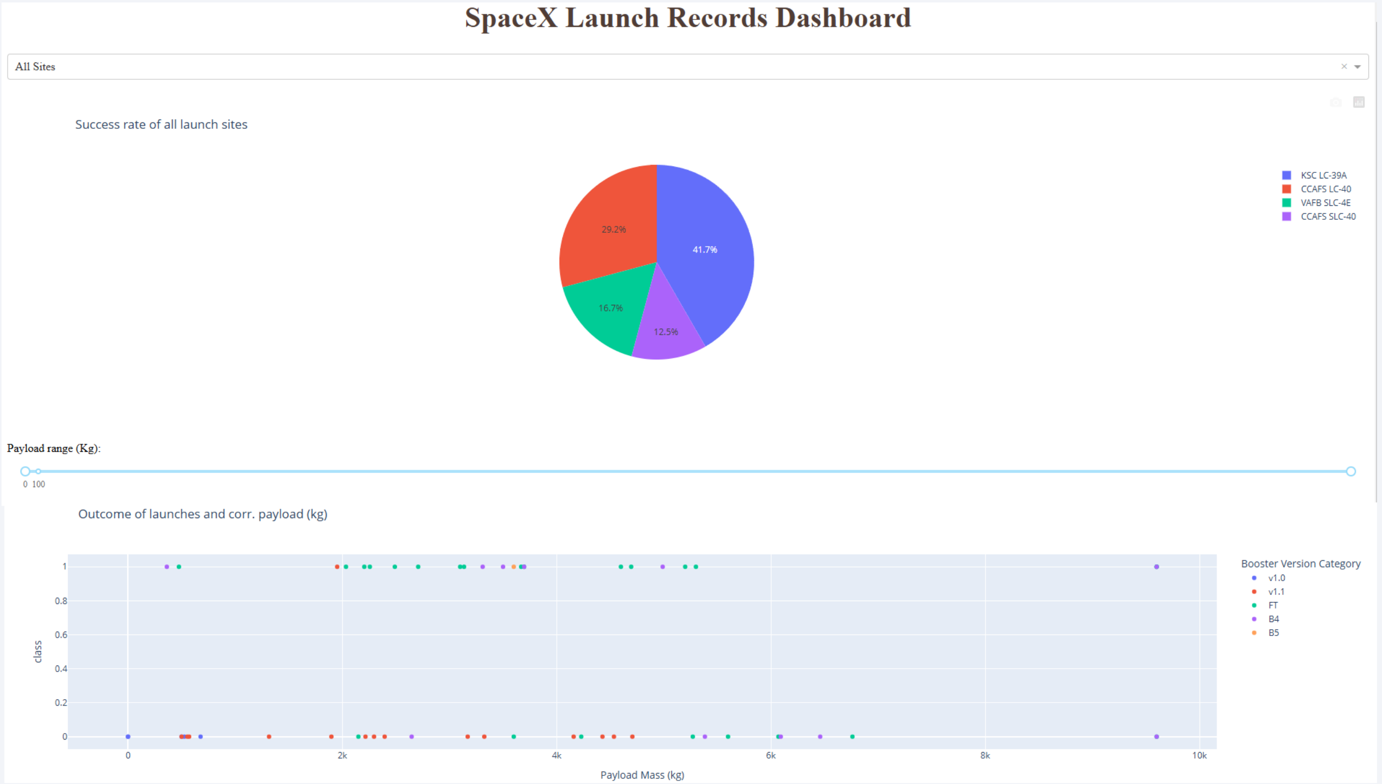
- Scale the input data
- Use Grid search to explore different parameters and cross validation of the classification models used: logistic regression, decision tree, supporting vector machine (SVM) and k-nearest neighbors
- Compare the score of fitting the train data sets and the test data set => choose the model with the highest score
- Github link to classification notebook

Results

- High success rate when payload mass >8000 kg; 100% success rate at CCAFS SLC 40
- ES-L1, GEO, HEL and SSO orbit types have 100% success rate; while SO with 0% success rate
- In the earlier stage, more launches were of LEO, PO and GTP orbit types; in more recent stage, launches are more of SSO, MEO, VLEO
- For payload mass < 8000 kg, orbit types are mainly GTO and ISS; for payload mass > 8000 kg, it's more likely to be VLEO
- Success rate has been generally increasing since year 2014

Results

- Interactive dashboard:



Results

- Predictive analytics:

	Accuracy of train data	Test data score
Logistic regression	0.8464	0.8333
Support Vector Machine (SVM)	0.8482	0.8333
Decision tree	0.875	0.8333
K-nearest neighbors	0.8482	0.8333

- Decision tree has the highest accuracy of train data, and highest test data score. So it should be chosen as the classification model

Section 2

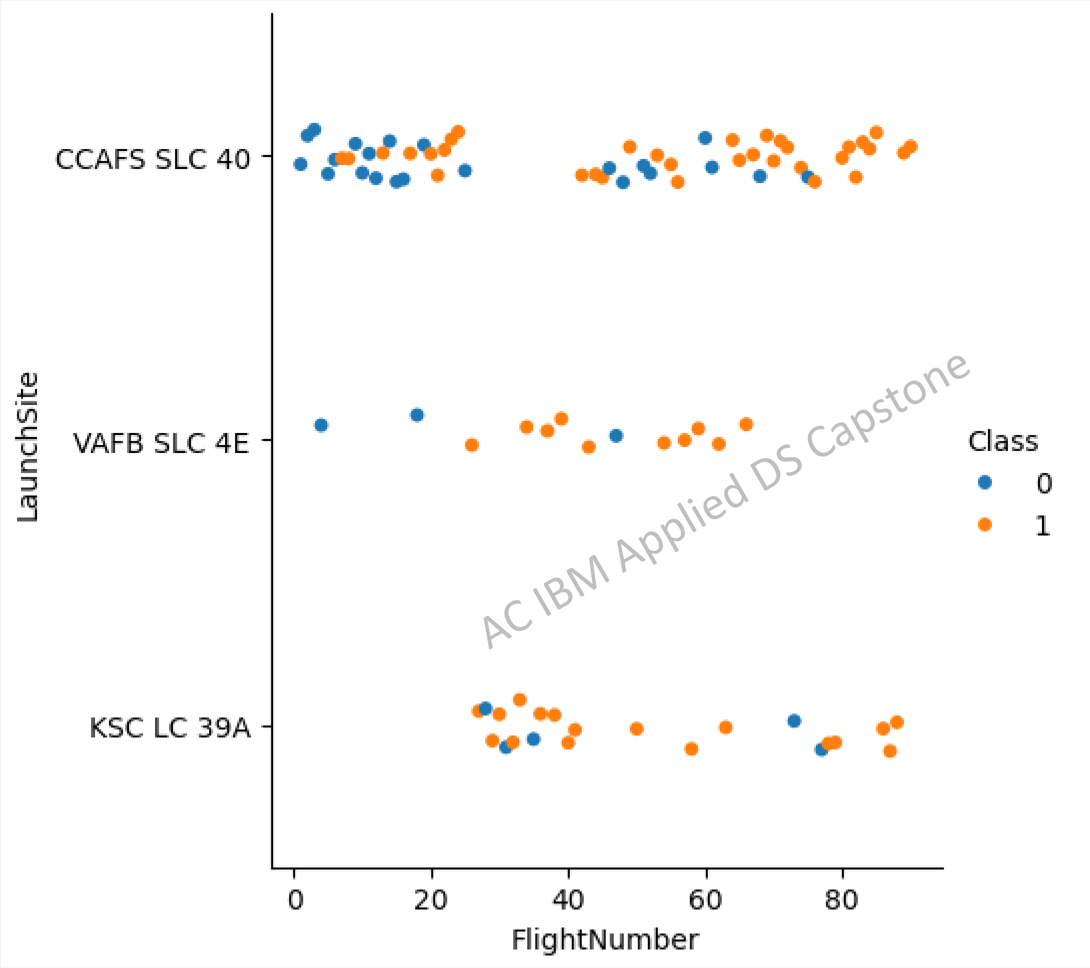
Insights drawn from EDA

Flight Number vs. Launch Site

- a scatter plot of Flight Number vs. Launch Site

- CCAFS SLC 40 has been the most used launch site since the beginning; CAFB SLC 4E is more used in earlier launches, while KSC LC 39A is more used in later launches

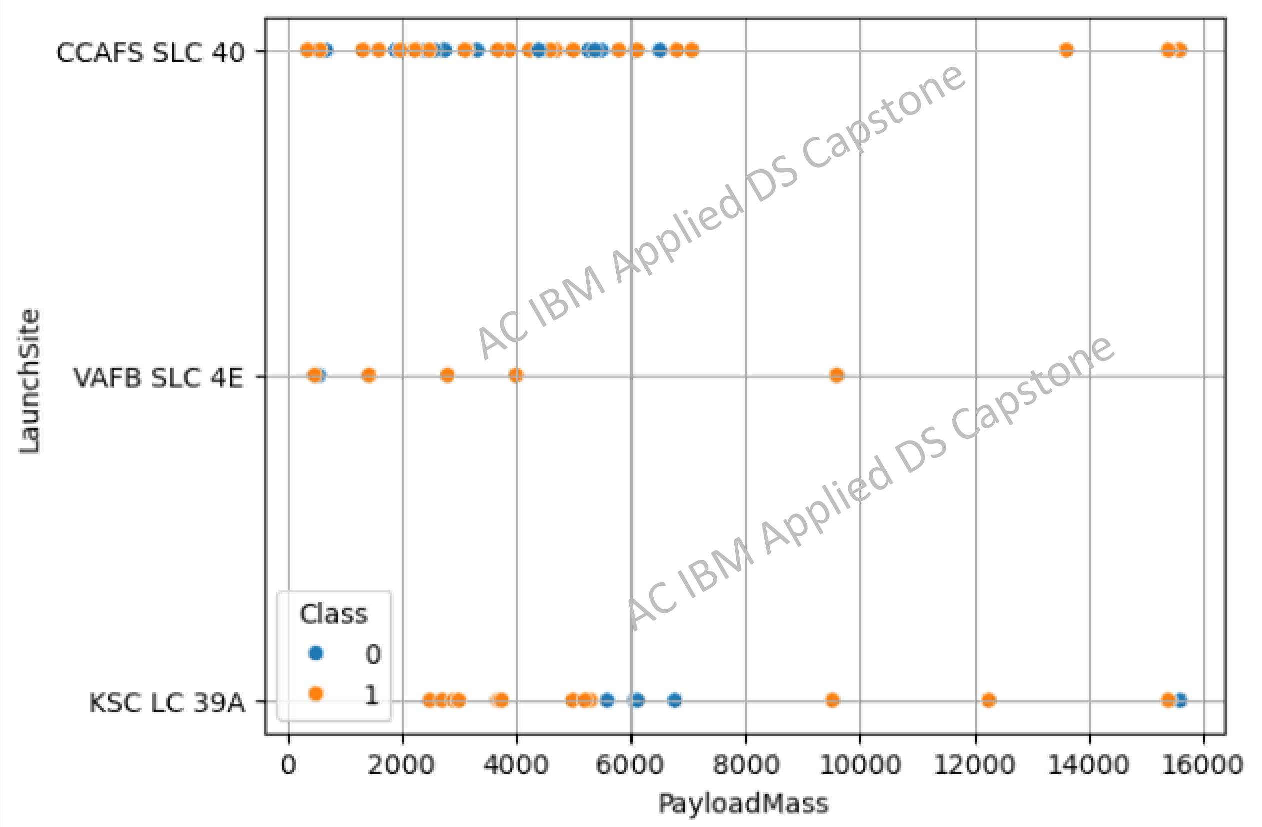
- Most recent launches have relatively high success rate



Payload vs. Launch Site

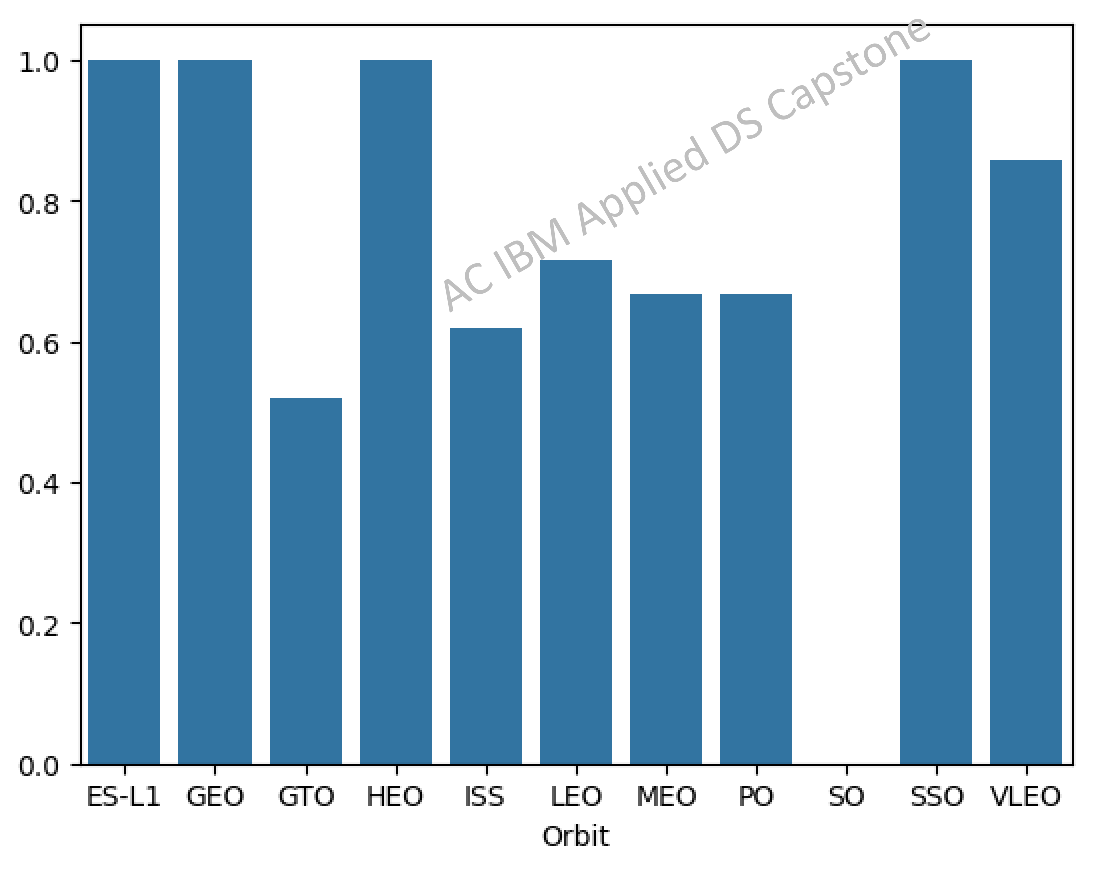
- scatter plot of Payload vs. Launch Site

- For payload mass >8000 kg, the success rate has been 100%



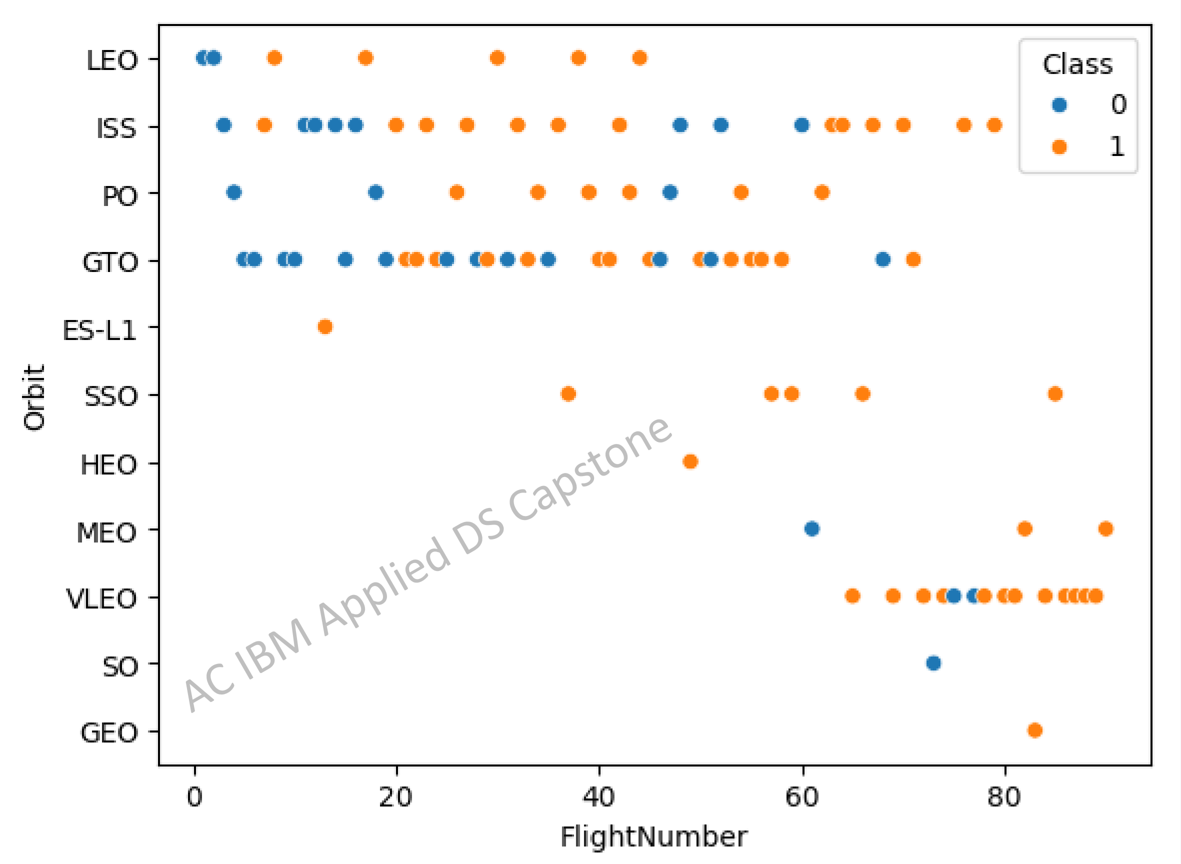
Success Rate vs. Orbit Type

- bar chart for the success rate of each orbit type
- ES-L1, GEO, HEO and SSO orbit type has had 100% success rate
- SO has 0% success rate (only 1 launch)



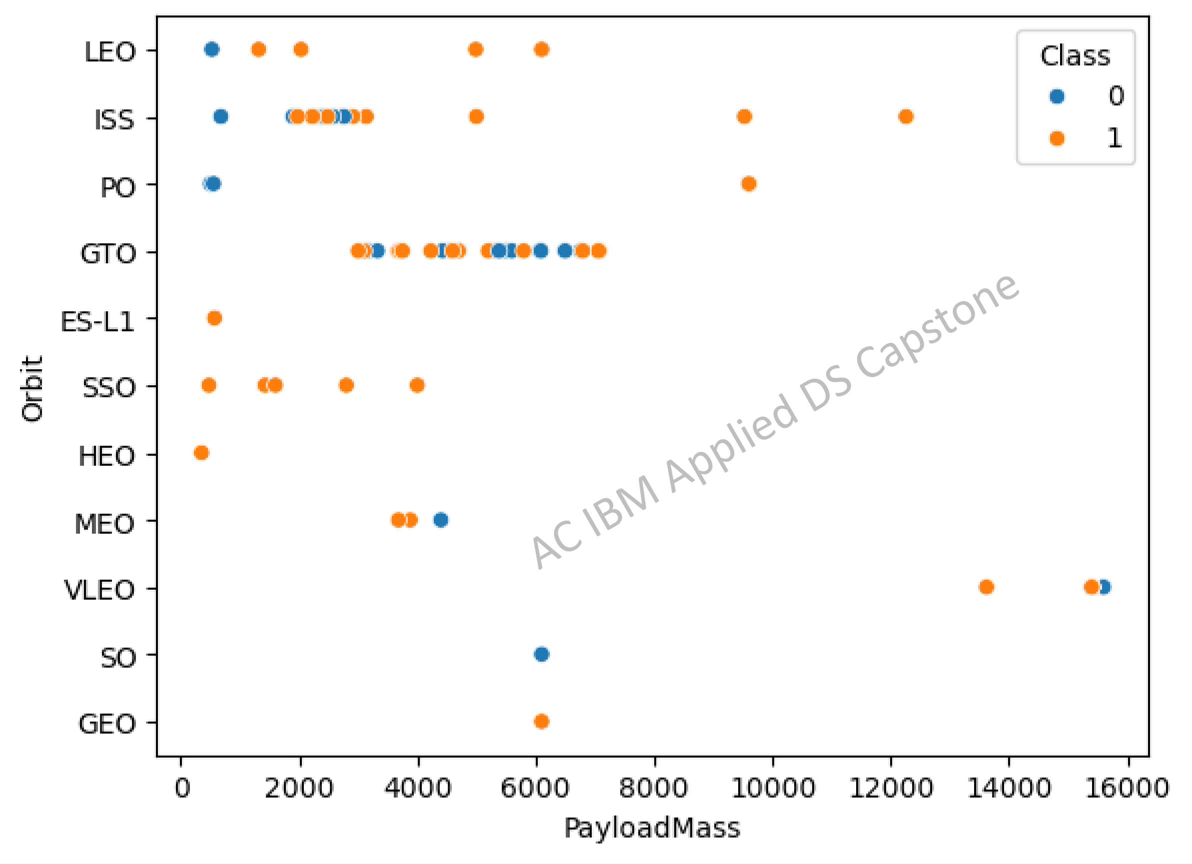
Flight Number vs. Orbit Type

- a scatter point of Flight number vs. Orbit type
- VLEO has been used in recent launches
- Recent launches after flight number 80 has been 100%



Payload vs. Orbit Type

- a scatter point of payload vs. orbit type
- High success rate of payload mass >8000 kg
- ISS and PO have 100% success rate for payload mass >8000 kg

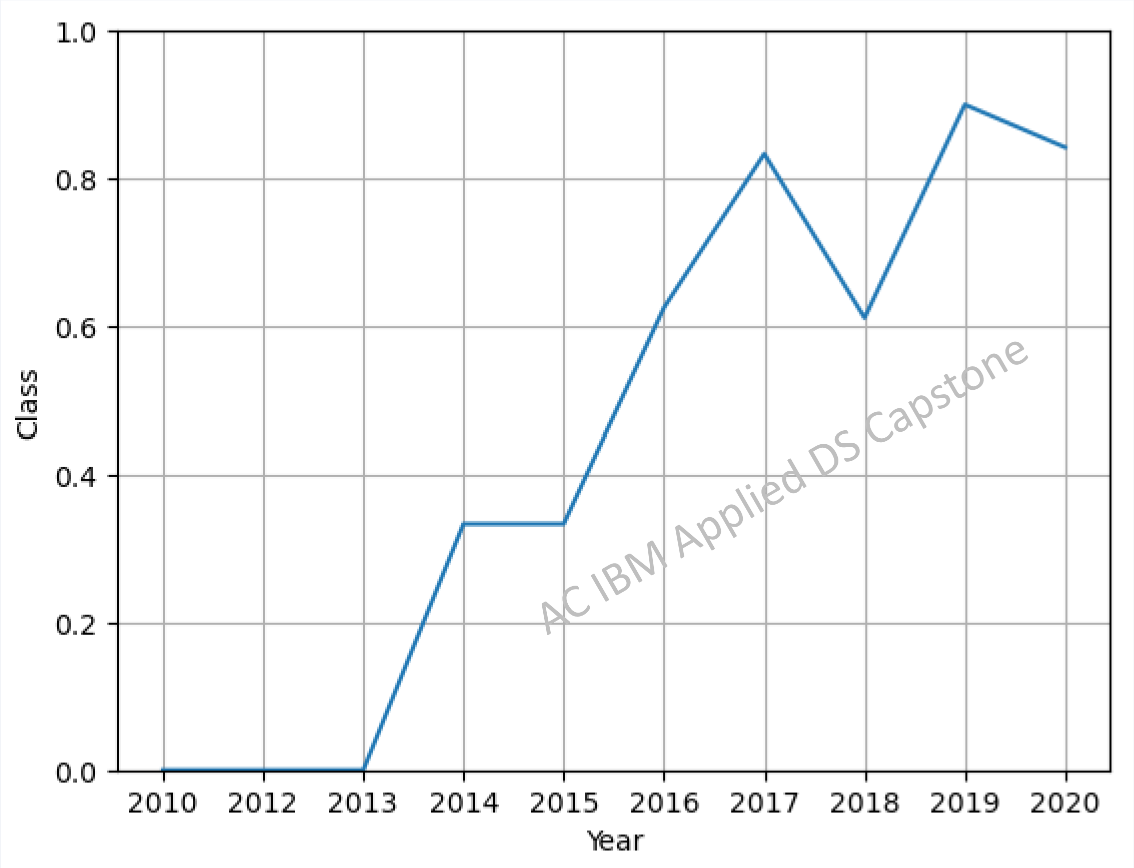


Launch Success Yearly Trend

- line chart of yearly average success rate

- Success rate has been increasing since 2014

- 2019 has the highest success rate from 2010 to 2020



All Launch Site Names

- names of the unique launch sites:

Task 1

Display the names of the unique launch sites in the space mission

```
[19]: sta = 'SELECT DISTINCT("Launch_Site") FROM SPACEXTABLE'
      cur.execute(sta)
      druck = cur.fetchall()
      druck
```

```
[19]: [('CCAFS LC-40',), ('VAFB SLC-4E',), ('KSC LC-39A',), ('CCAFS SLC-40',)]
```

- Using DISTINCT() to find out the list of names of unique launch sites

Launch Site Names Begin with 'CCA'

- 5 records where launch sites begin with `CCA`
- Using 'LIKE' and wildcard '%' to find out the records
- LIMIT 5 limits the results to only 5 records

```
[22]: sta = 'SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE \'CCA%\' LIMIT 5'
      cur.execute(sta)
      druck = cur.fetchall()
      druck
```

```
[22]: [('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

- total payload carried by boosters from NASA is 45596 kg

Task 3 ¶

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

```
[23]: sta = 'SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer = \'NASA (CRS)\' '
      cur.execute(sta)
      druck = cur.fetchall()
      druck

[23]: [(45596,)]
```

- Using SUM() to find out the total payload and specifying customer is NASA in the where clause

Average Payload Mass by F9 v1.1

- the average payload mass carried by booster version F9 v1.1 is 2928.4

Task 4

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

```
[24]: sta = 'SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE "Booster_Version" = \'F9 v1.1\' '
      cur.execute(sta)
      druck = cur.fetchall()
      druck

[24]: [(2928.4,)]
```

- Using AVG() to find out the average, stating the booster version in the where clause

First Successful Ground Landing Date

- Date of the first successful landing outcome on ground pad is Dec 22nd 2015

Task 5

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

Hint:Use min function

```
[25]: sta = 'SELECT MIN(Date) FROM SPACEXTABLE WHERE "Landing_Outcome" = \'Success (ground pad)\' '
      cur.execute(sta)
      druck = cur.fetchall()
      druck

[25]: [('2015-12-22',)]
```

- Using MIN(DATE) to find out the earliest date; specifying the landing outcome is success ground pad in where clause

Successful Drone Ship Landing with Payload between 4000 and 6000

- the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000:
- F9 FT B1022, F9 FT B1026, F9 FT B1021.2, F9 FT B1031.2

Task 6

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

```
[26]: sta = 'SELECT "Booster_Version" FROM SPACEXTABLE WHERE "Landing_Outcome" = \'Success (drone ship)\' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000'
cur.execute(sta)
druck = cur.fetchall()
druck
```

```
[26]: [('F9 FT B1022',), ('F9 FT B1026',), ('F9 FT  B1021.2',), ('F9 FT  B1031.2',)]
```

- Specifying the type of landing outcome and the range of payload mass in the where clause

Total Number of Successful and Failure Mission Outcomes

- total number of successful and failure mission outcomes is 61

Task 7

List the total number of successful and failure mission outcomes

```
[27]: sta = 'SELECT COUNT(*) FROM SPACEXTABLE WHERE "Landing_Outcome" LIKE \'Success%\' OR "Landing_Outcome" LIKE \'Faillure%\''
      cur.execute(sta)
      druck = cur.fetchall()
      druck

[27]: [(61,)]
```

- Using COUNT(*) to count the number of times of success and failure of launches. Specified the condition using 'LIKE' and '%' in the where clause

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass:

Task 8

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

[28]:

```
sta = 'SELECT "Booster_Version" FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ in (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE) '
cur.execute(sta)
druck = cur.fetchall()
druck
```

[28]:

```
[('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 ',)]
```

- Selecting the records where the payload mass is maximum using MAX() in a subquery, then select the booster version with the maximum payload mass from that subquery
- 12 boosters

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
[35]: sta = 'SELECT substr(Date, 6, 2), "Landing_Outcome" FROM SPACEXTABLE WHERE substr(Date, 1, 4) = \'2015\' AND "Landing_Outcome" LIKE \'Failure%\''
      cur.execute(sta)
      druck = cur.fetchall()
      druck

[35]: [('01', 'Failure (drone ship)'), ('04', 'Failure (drone ship)')]
```

- Only 2 records fit the requirement
- Using substr(Date, position to start, length to extract) to extract the month and specify the year in where clause

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

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

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.

[41]:
```
sta = 'SELECT "Landing_Outcome", COUNT("Landing_Outcome") FROM SPACEXTABLE WHERE Date BETWEEN  \'2010-06-04\' AND \'2017-03-20\' GROUP BY "Landing_Outcome" ORDER BY COUNT("Landing_Outcome") DESC'
cur.execute(sta)
druck = cur.fetchall()
druck
```

[41]:
```
[('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)]
```

- The landing outcome took place most frequently is ‘No attempt’ with 10 times, then Success (drone ship) and Failure (drone ship), both occurred 5 times

- Used COUNT(), GROUP BY, ORDER BY and DESC

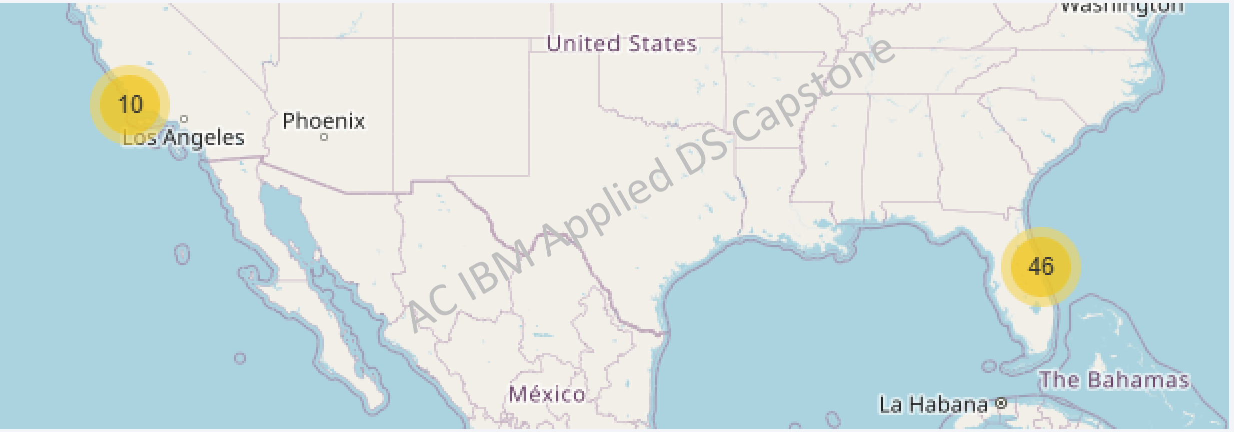
Section 3

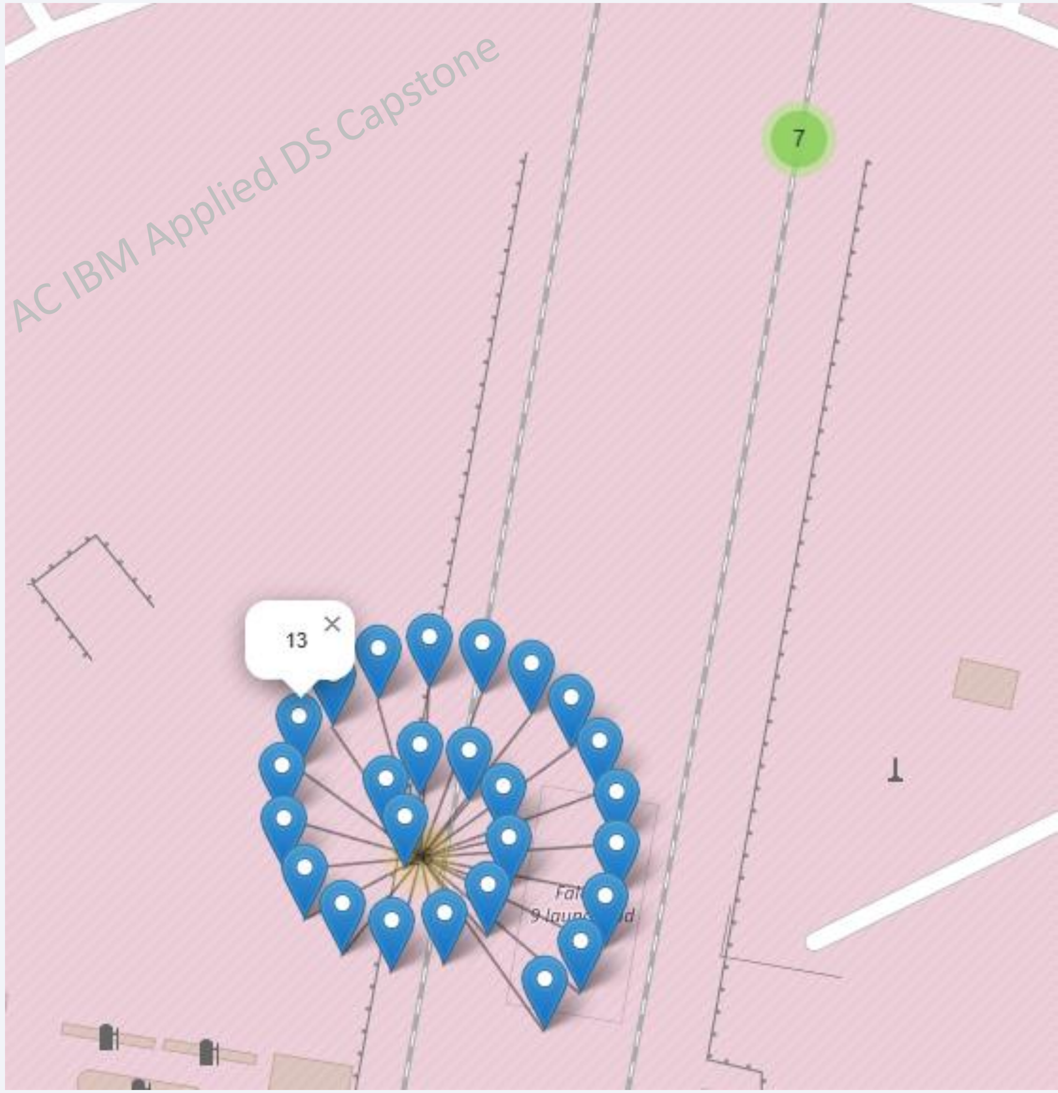
Launch Sites Proximities Analysis

All Launch Sites’ Location

- 3 launch sites are in the east coast while 1 launch site is in the west coast

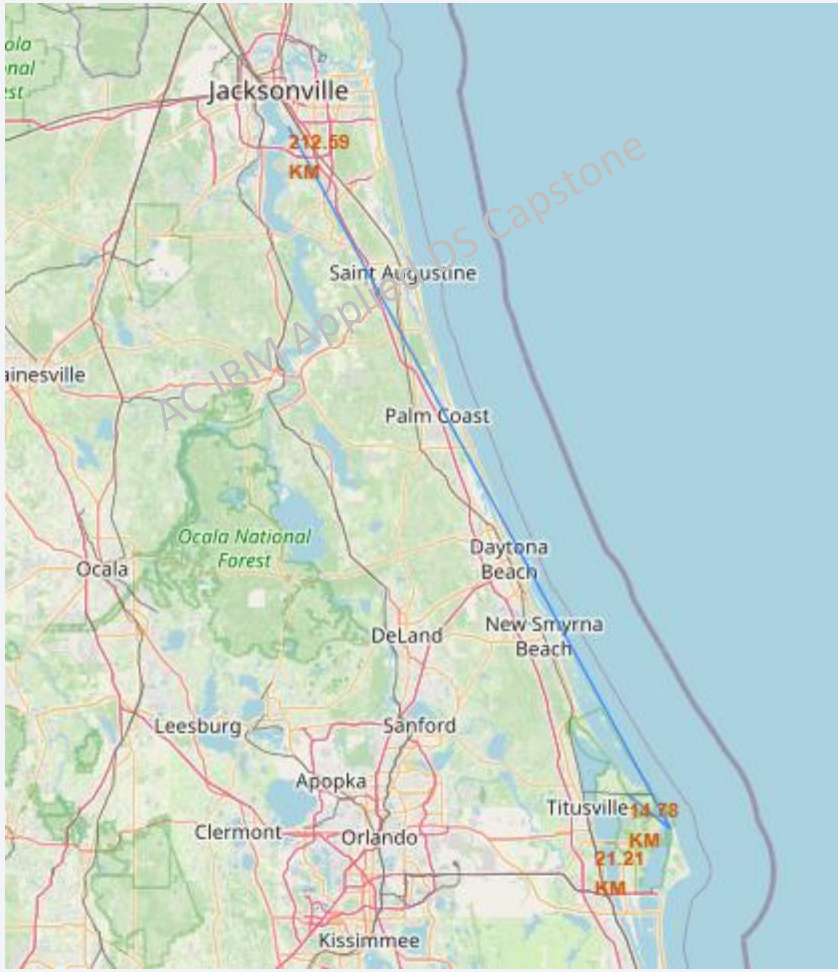
Launch outcomes at launch sites





- All launches are grouped according to their launch site
- More launches were in the east coast

<Folium Map Screenshot 3>



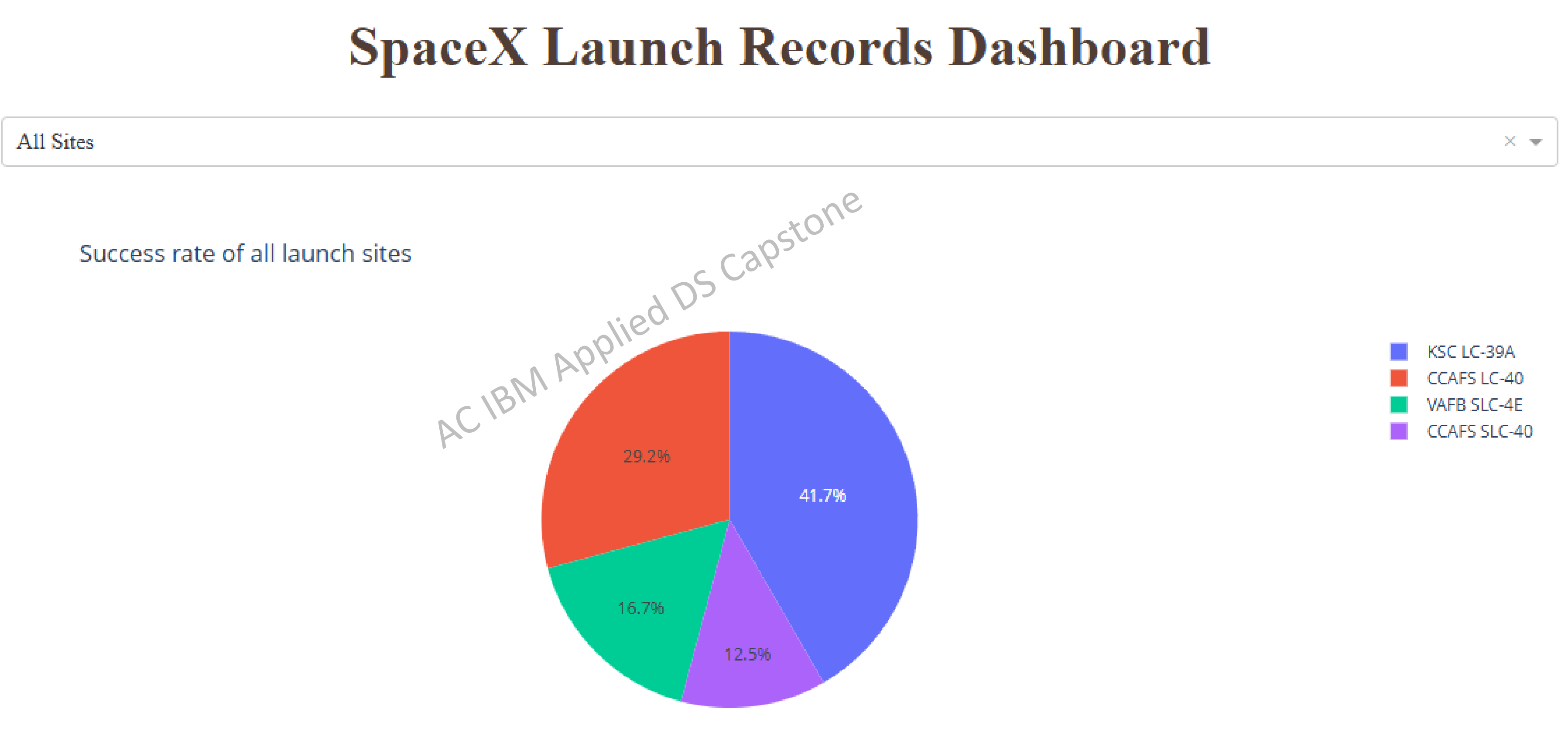
- 213 km from Jacksonville

- 14.78 km from railway
- 21.21 km from highway

Section 4

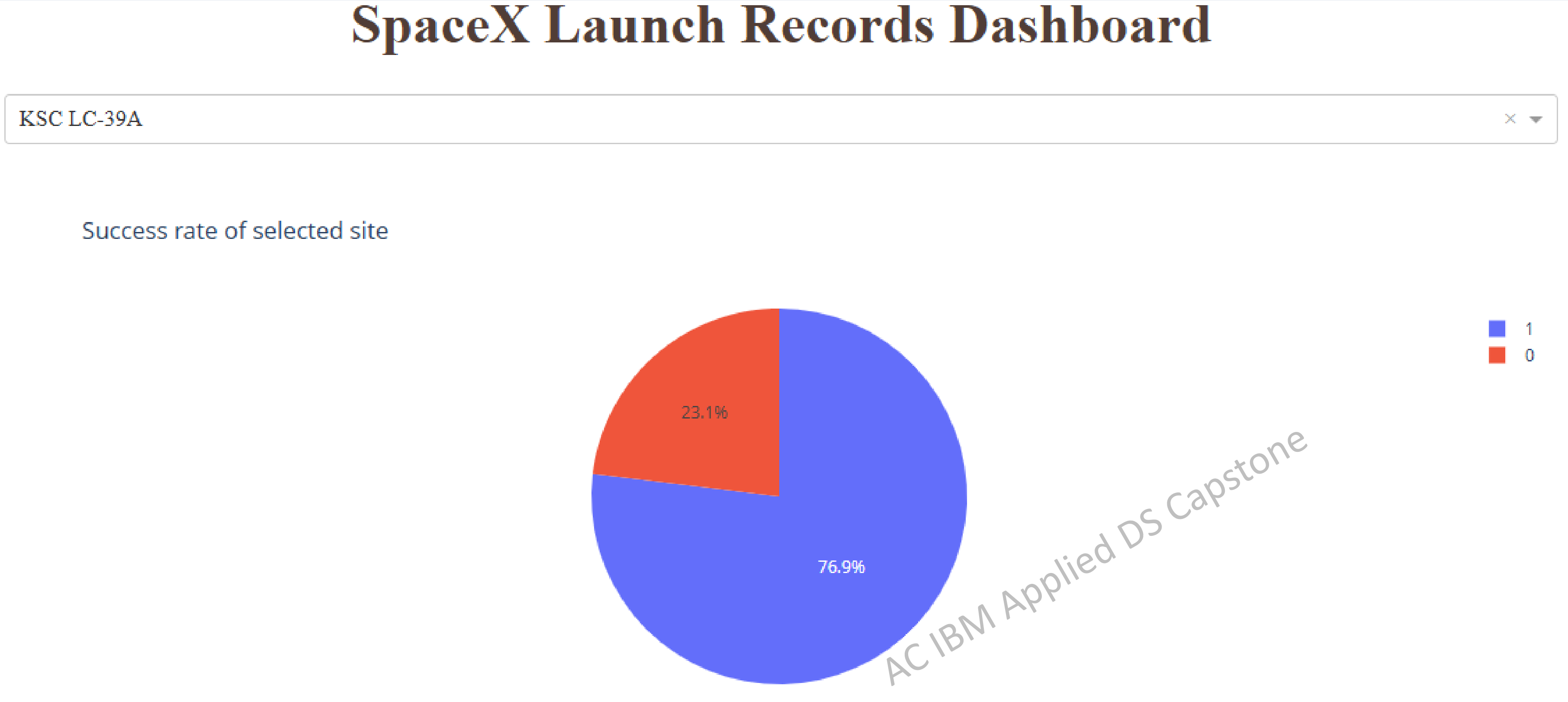
Build a Dashboard with Plotly Dash

Percentages of Successful Launches from All Sites



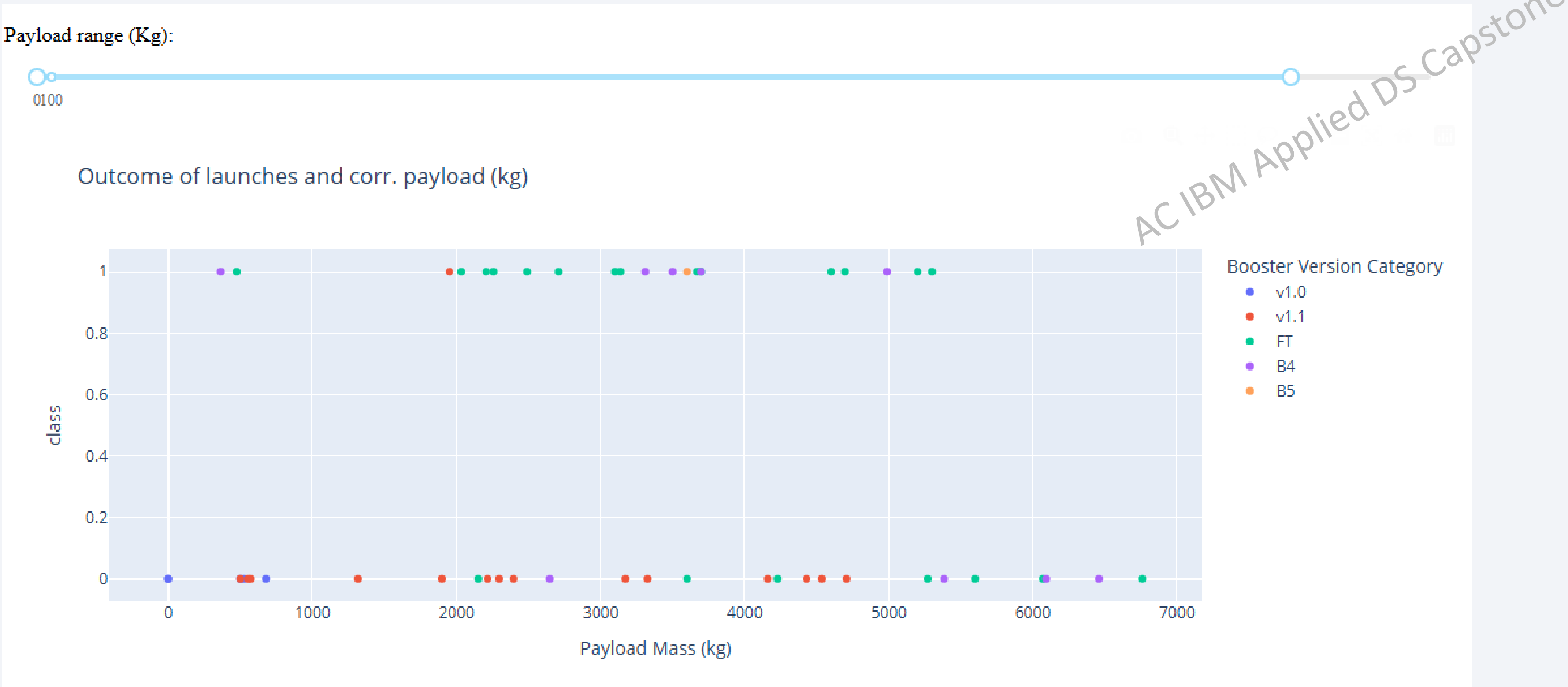
- Most successful launches were at KSC LC-39A with 41.7%, while CCAFS SLC-40 had the lowest success rate of 12.5%

Success Rate of Launch Site with the Highest Launch Success Ratio

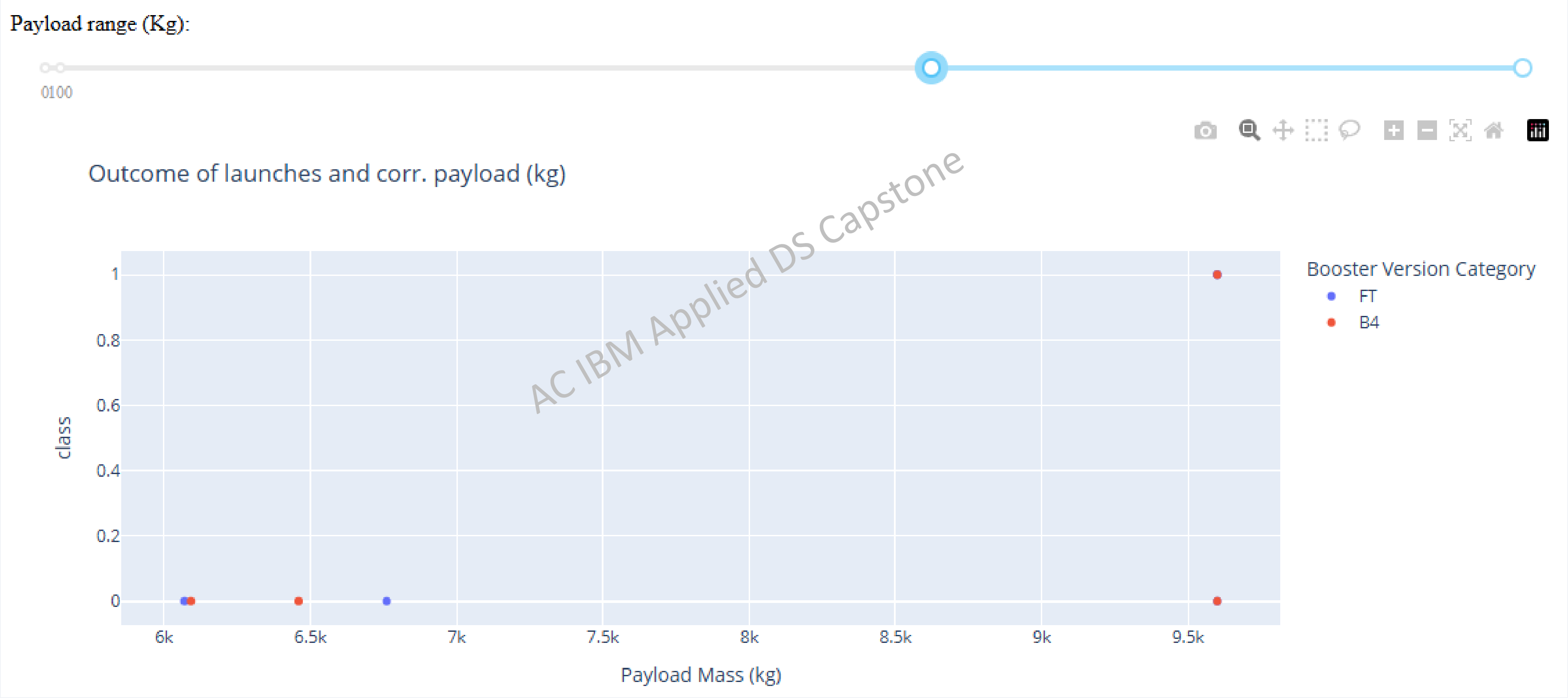


- KSC LC-39A had the highest launch success rate of 76.9%

Payload mass range and outcomes



Payload mass range and outcomes

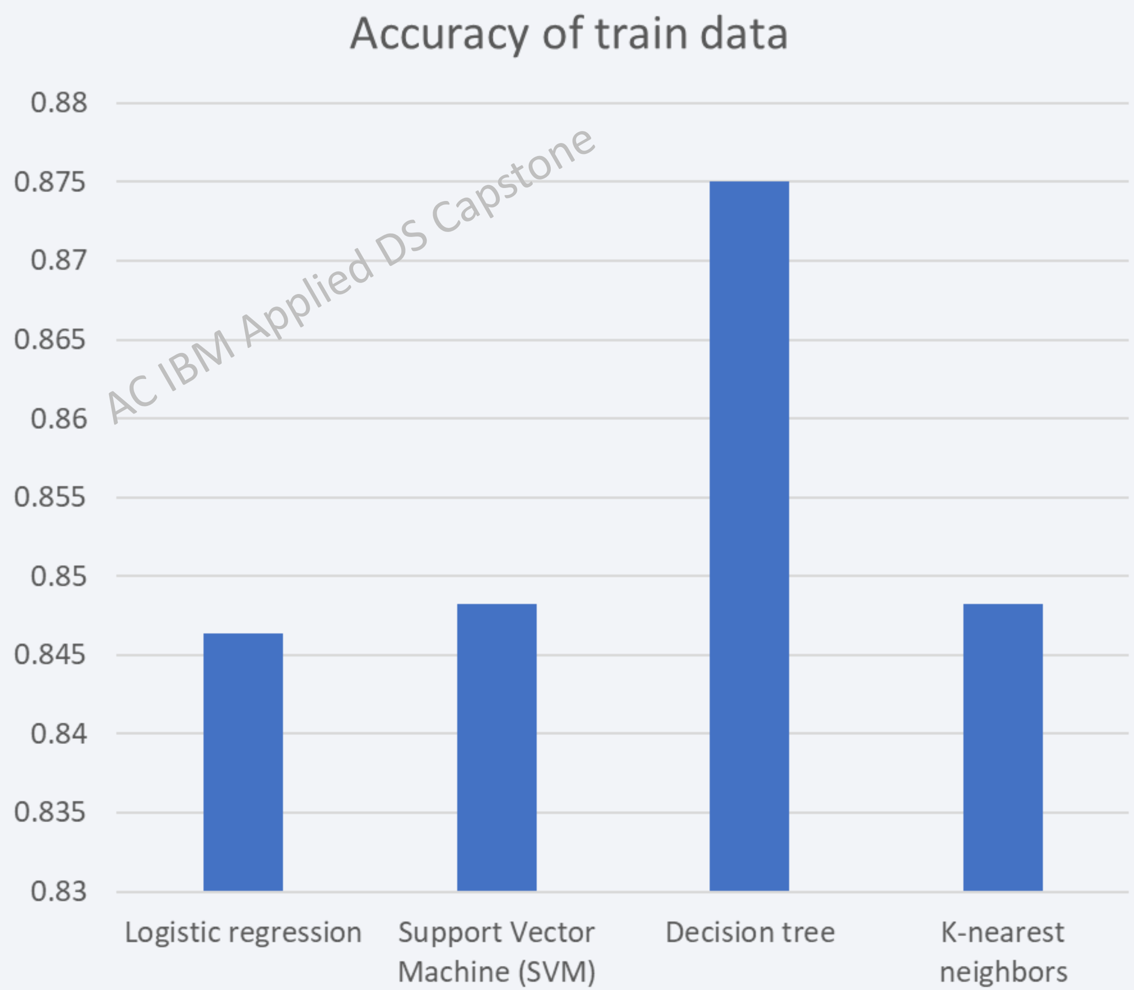


Section 5

Predictive Analysis (Classification)

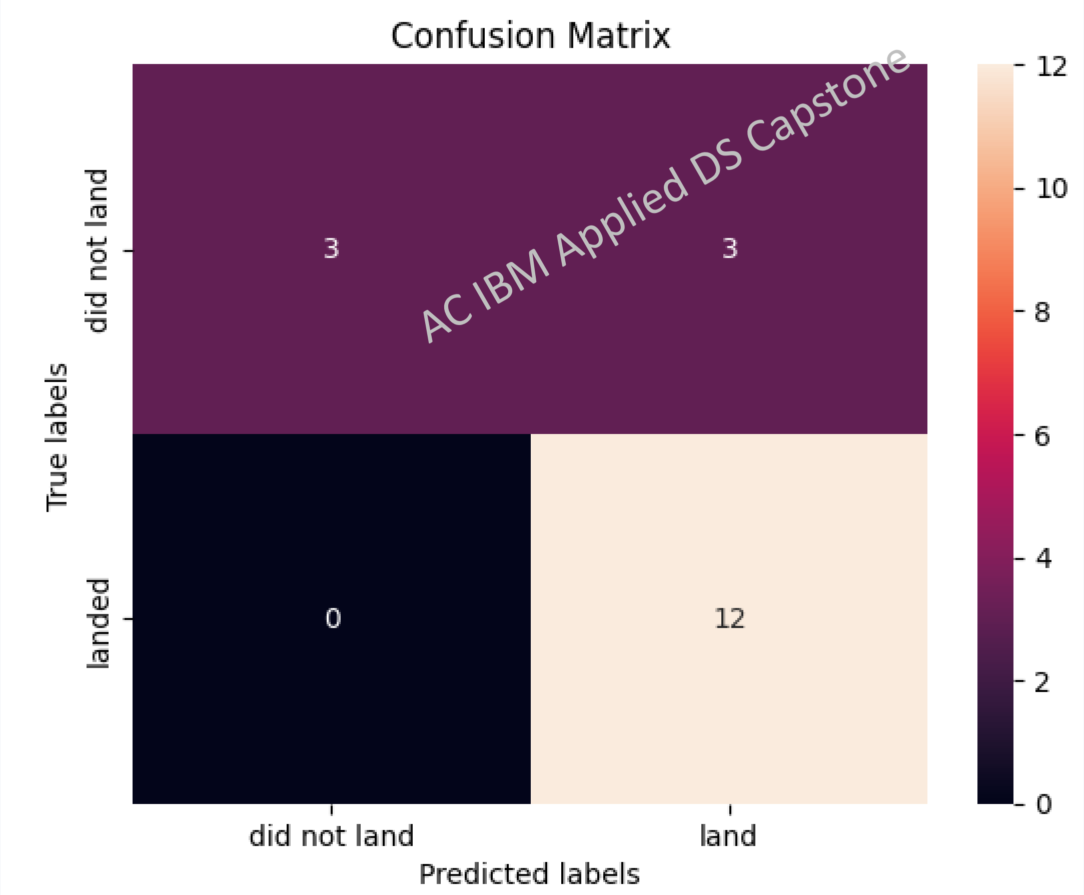
Classification Accuracy

- Decision tree classification model has the highest accuracy



Confusion Matrix

- Decision tree classification model correctly predicted those actually landed
- True Positive: 3, True Negative: 12
- But 3 launches that did not land were predicted to “land” successfully



Conclusions

- Launch success rate has been increasing
- ES-L1, GEO, HEO and SSO orbit type has had 100% success rate
- KSC LC-39A had the highest launch success rate of 76.9%
- Decision tree classification model has the highest accuracy

Thank you!