Winning Space Race with Data Science

Babu Naga Ramesh Nekkanti

13-Dec-2021

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection using API and Web scraping
  - Data wrangling
  - Exploratory Data Analysis using SQL and Visualization
  - Perform Interactive Visual Analysis using Folium & Plotly Dash tools
  - Perform Predictive Analysis using Classification Models
- Summary of all results

Introduction

- Project background and context
  - SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards 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 successfully. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.
- Problems you want to find answers
  - Whether the Falcon 9 first stage will land successfully and with what factors?
  - What are the conditions that will aid SpaceX to achieve the best results.
  - Find the effect of each factor of rocket launching variables on the outcome.

Section 1

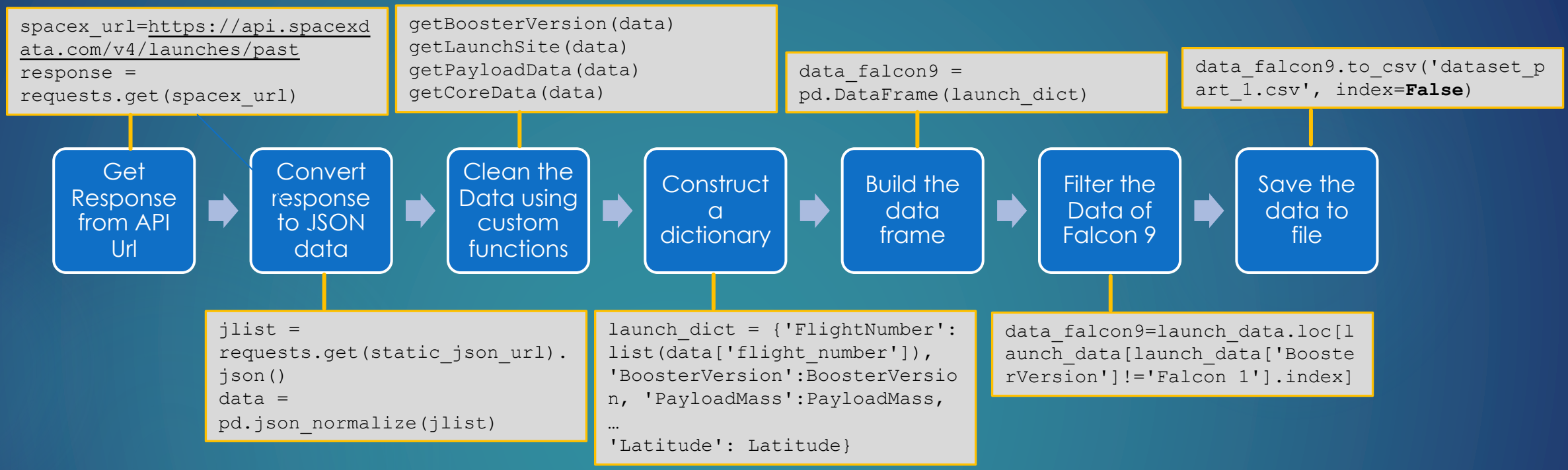
Methodology

Methodology

- Data collection methodology:
  - Through SpaceX API
  - Web scraping from wikipedia
- Perform data wrangling
  - Dropping the data rows of Falcon 1
  - Dropping irrelevant columns
  - One hot encoding of data fields
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Bar graphs and scatter graphs to show the patterns between the data
  - Using IBM DB2 SQL queries to analyze the data using aggregate functions and grouping.
- Perform interactive visual analytics using Folium and Plotly Dash
  - Using Folium interactive maps and map markers for success and failure launches
  - Using Plotly Dash – build interactive dashboard to analyze launch site and payload affect on results
- Perform predictive analysis using classification models
  - Using different classification models such as SVM, Decision Trees, KNN and Logistic Regression

Data Collection

Data Collection – SpaceX API



https://github.com/rameshnbn/SpaceX-Data-Sience-Project/blob/main/Week1a%20-%20Data%20Collection%20API%20Lab.ipynb

Data Collection - Scraping

Get HTML Response from Url

```
static_url =
https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922
response = requests.get(static_url).text
```

Construct BeautifulSoup object from HTML response string

```
soup = Beautifudf=pd.DataFrame(launch_dict)lSoup(response, 'html.parser')
```

Find the HTML table element

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

Extract the column names from <th> elements using custom function

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

Iterate through each row of the table and create a dictionary of Data

```
for table_number,table in enumerate(soup.find_all('table',"wikitable
plainrowheaders collapsible")):
    for rows in table.find_all("tr"):
            row=rows.find_all('td’)
                ..
```

Convert it to a DataFrame

```
df=pd.DataFrame(launch_dict)
```

Save the data to file

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

https://github.com/rameshnbn/SpaceX-Data-Sience-Project/blob/main/Week1c%20-%20Data%20Collection%20with%20Web%20Scraping%20lab.ipynb

Data Wrangling

Launch Counts at each site

```
In [5]: # Apply value_counts() on column L
        df['LaunchSite'].value_counts()

Out[5]: CCAFS SLC 40     55
        KSC LC 39A       22
        VAFB SLC 4E      13
        Name: LaunchSite, dtype: int64
```

Launch count for each orbit

```
In [6]: # Apply value_counts on Orb
        df['Orbit'].value_counts()

Out[6]: GTO        27
        ISS        21
        VLEO       14
        PO          9
        LEO         7
        SSO         5
        MEO         3
        ES-L1       1
        HEO         1
        SO          1
        GEO         1
        Name: Orbit, dtype: int64
```

Mission outcome counts

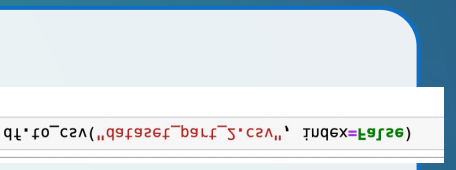
```
In [7]: # landing_outcomes = values on Outcome column
        landing_outcomes = df['Outcome'].value_counts()
        landing_outcomes

Out[7]: True ASDS      41
        None None      19
        True RTLS      14
        False ASDS      6
        True Ocean      5
        False Ocean     2
        None ASDS       2
        False RTLS      1
        Name: Outcome, dtype: int64
```

Create Outcome Label

```
In [2]: #df['Class']=landing_class
        bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
        landing_class=lambda outcome: 0 if outcome in bad_outcomes
        df['Class'] = df['Outcome'].apply(landing_class)
        df[['Class']].head(8)

Out[2]:
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

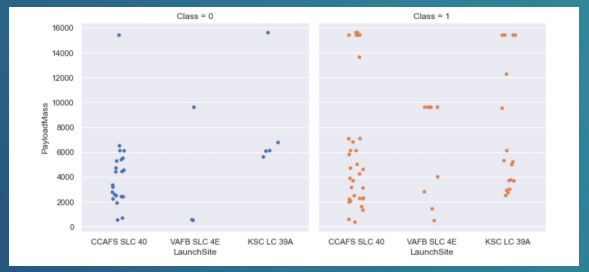
Export the data

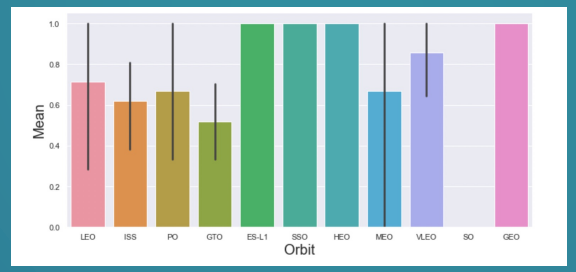


https://github.com/rameshnbn/SpaceX-Data-Sience-Project/blob/main/Week1b%20-%20%20Data%20Wrangling%20lab.ipynb

EDA with Data Visualization

- Different charts plotted for Exploratory Data Analysis are
  - Scatter chart for outcome analysis for different variables, such as
    - Payload and Launch site
    - FlightnNmber and Lauch site
    - FlightNumber and Orbit type
    - Payload and Orbit type
  - Barchart – To find the orbits with highest and lowest success rate.
  - Line chart - To find launch success yearly trend





https://github.com/rameshnbn/SpaceX-Data-Sience-Project/blob/main/Week2b%20-%20EDA%20with%20Visualization%20lab.ipynb

EDA with SQL

- Summary of SQL queries / clauses used in this EDA
  - DISTINCT
  - PREDICATES
    - LIKE
    - =, >, <, >=, <=, BETWEEN
    - AND
  - GROUP BY CLAUSE
  - ORDER BY CLAUSE
  - Aggregate Queries
    - COUNT
    - SUM
    - AVG
    - MIN
    - MAX
  - CASE WHEN .. THEN .. ELSE .. END
  - Sub Queries and Nested Select statements
  - Different functions
- A detailed list of queries and out comes will follow later in this report
- https://github.com/rameshnbn/SpaceX-Data-Sience-Project/blob/main/Week2a%20-%20EDA%20with%20SQL%20lab.ipynb

```
SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;

SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by
Booster Version F9 v1.1" FROM SPACEXTBL WHERE BOOSTER_VERSION
= 'F9 v1.1';

SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING__OUTCOME =
'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND
PAYLOAD_MASS__KG_ < 6000;

SELECT MISSION_OUTCOME,COUNT(MISSION_OUTCOME) AS "Count" FROM
SPACEXTBL GROUP BY MISSION_OUTCOME;

SELECT sum(CASE WHEN MISSION_OUTCOME LIKE '%Success%' THEN 1
ELSE 0 END) AS "Successful Mission", \
    sum(CASE WHEN MISSION_OUTCOME LIKE '%Failure%' THEN 1 ELSE
0 END) AS "Failure Mission" \
FROM SPACEXTBL;

SELECT LANDING__OUTCOME as "Landing Outcome",
COUNT(LANDING__OUTCOME) AS "Total Count", \
ROW_NUMBER() OVER () AS "RANK" FROM SPACEXTBL \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING__OUTCOME \
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```

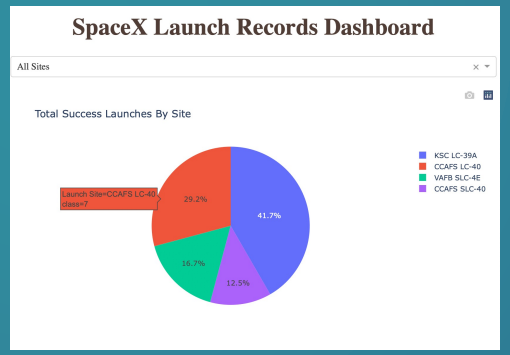
Build an Interactive Map with Folium

- Summary of different objects added to the folium map
  - Map Markers – To show the launch sites
  - Icon Marker – To create icon
  - Circles – to create a circle where a marker is placed
  - Marker Cluster – To show the launches representing the outcome
  - Lines – To show the distance between two points

- https://github.com/rameshnbn/SpaceX-Data-Sience-Project/blob/main/Week3a%20-%20Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

- We have added a Pie chart and Scatter Chart to the interactive Plotly Dashboard Python APP.
  - Pie chart and Scatter charts are linked to the Site selection dropdown
    - Pie chart shows the distribution of successful missions for all or for one selected site
    - Slider dynamic control for Payload is added to Scatter plot is to show the success rate for the selected range of payload.



https://github.com/rameshnbn/SpaceX-Data-Sience-Project/tree/main/Week3b%20-%20Interactive%20Plotly%20Dash%20App

Predictive Analysis (Classification)

Build different classification Models

- Load the features data
- Transform the data as X and y
- Split the data into Train and test sets
- On the train set using GridSearchCV with classifiers of
  - KNN,
  - Decision Tree,
  - SVM and
  - Logistic Regression

Evaluate each Model

- For each model built
  - Predict the outcome using the test set
  - Plot the confusion matrix
  - Find the accuracy of each model

Find best performing classification model

- Compile the results of each model
- Sort the results based on the accuracy (performance)
- Find the best fitting model for the given data sets.

https://github.com/rameshnbn/SpaceX-Data-Sience-Project/blob/main/Week4%20-%20Machine%20Learning%20Prediction%20Lab.ipynb

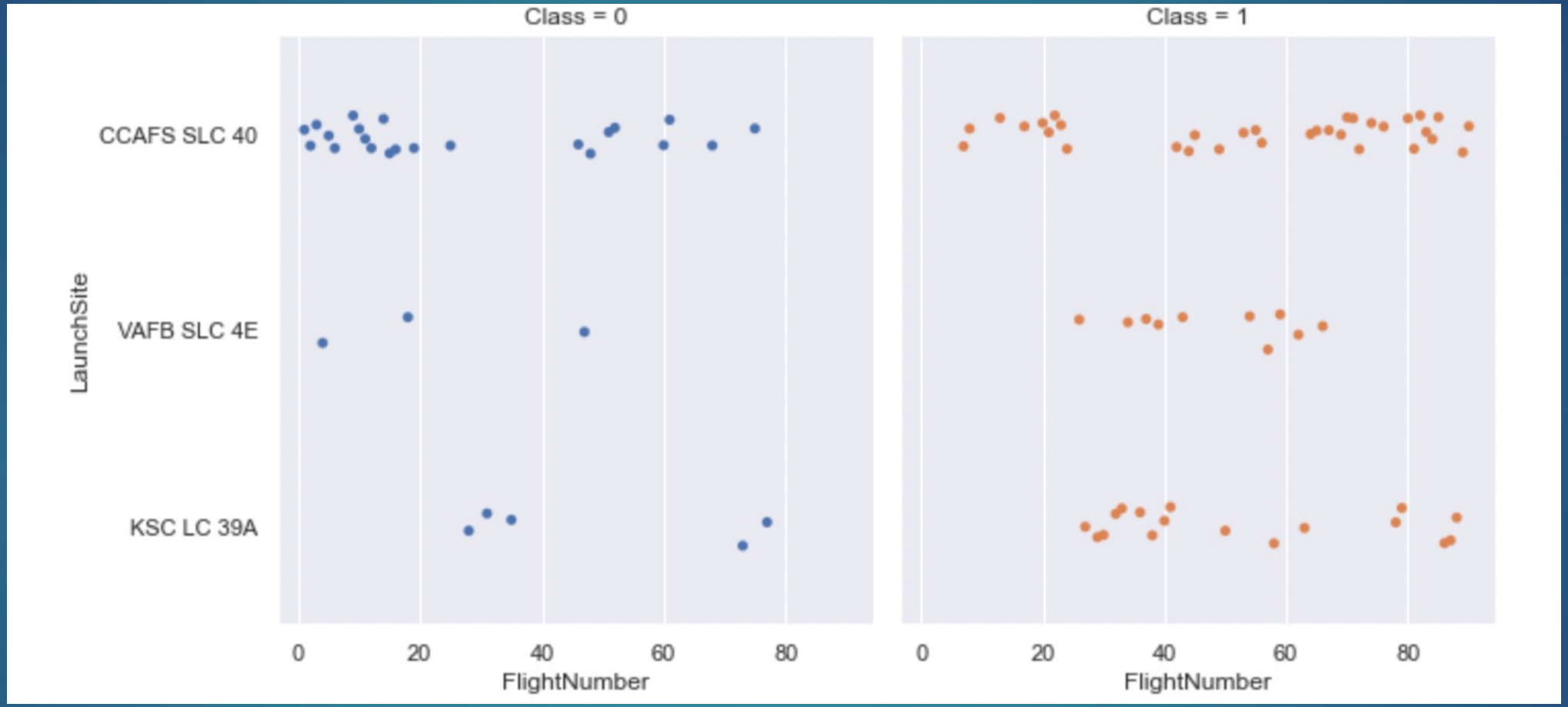
Results

- Exploratory data analysis results are shown in the next slides
  - Insights Drawn from EDA and SQL
- Interactive analytics demo in screenshots
  - These are shown in Plotly Dash section
- Predictive analysis results

Section 2

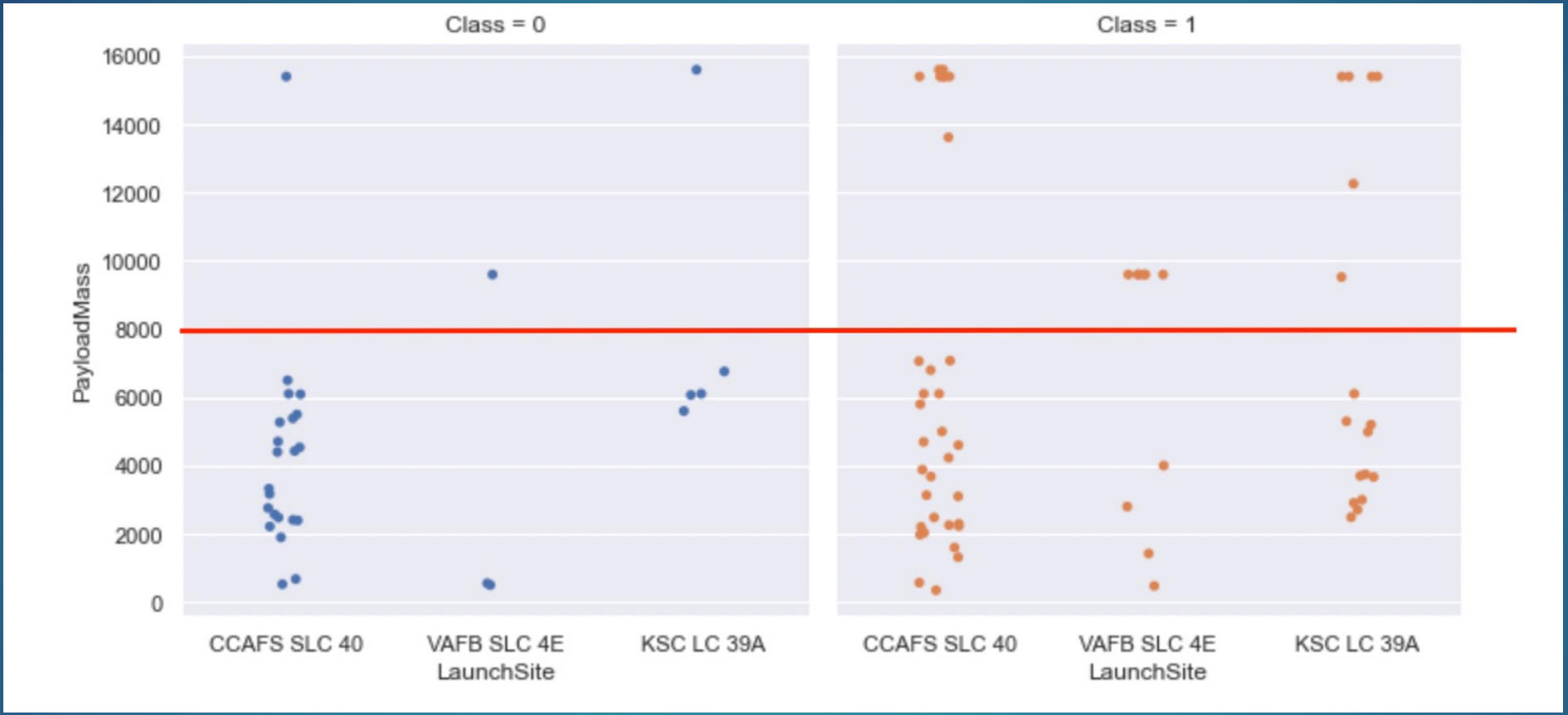
Insights drawn from EDA

Flight Number vs. Launch Site



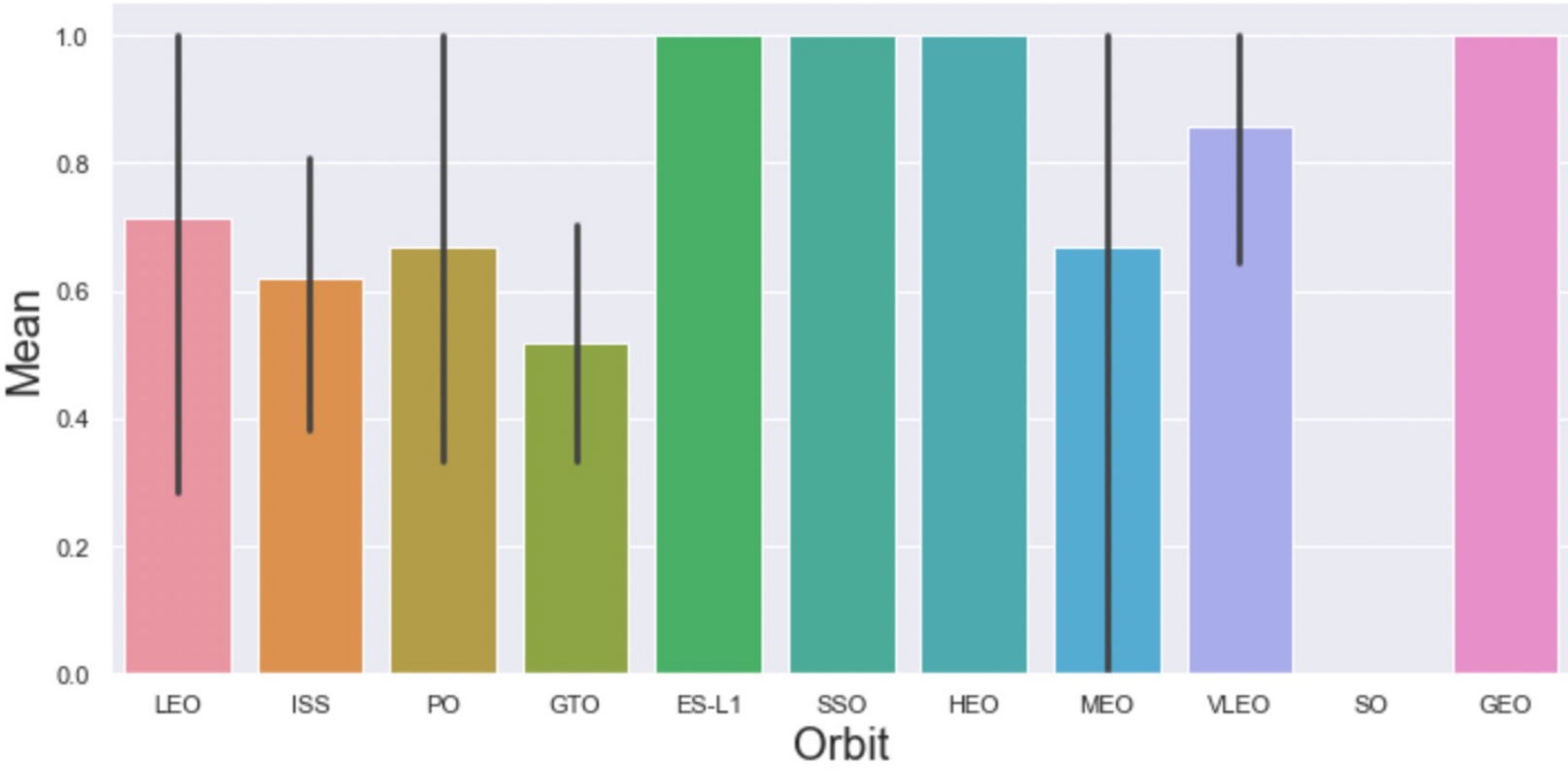
- Rocket success rate is increasing with more flight numbers (>30).
- It looks like "KSC LC 39A" and "VAFB SLC 4E" sites has higher launch success rates

Payload vs. Launch Site



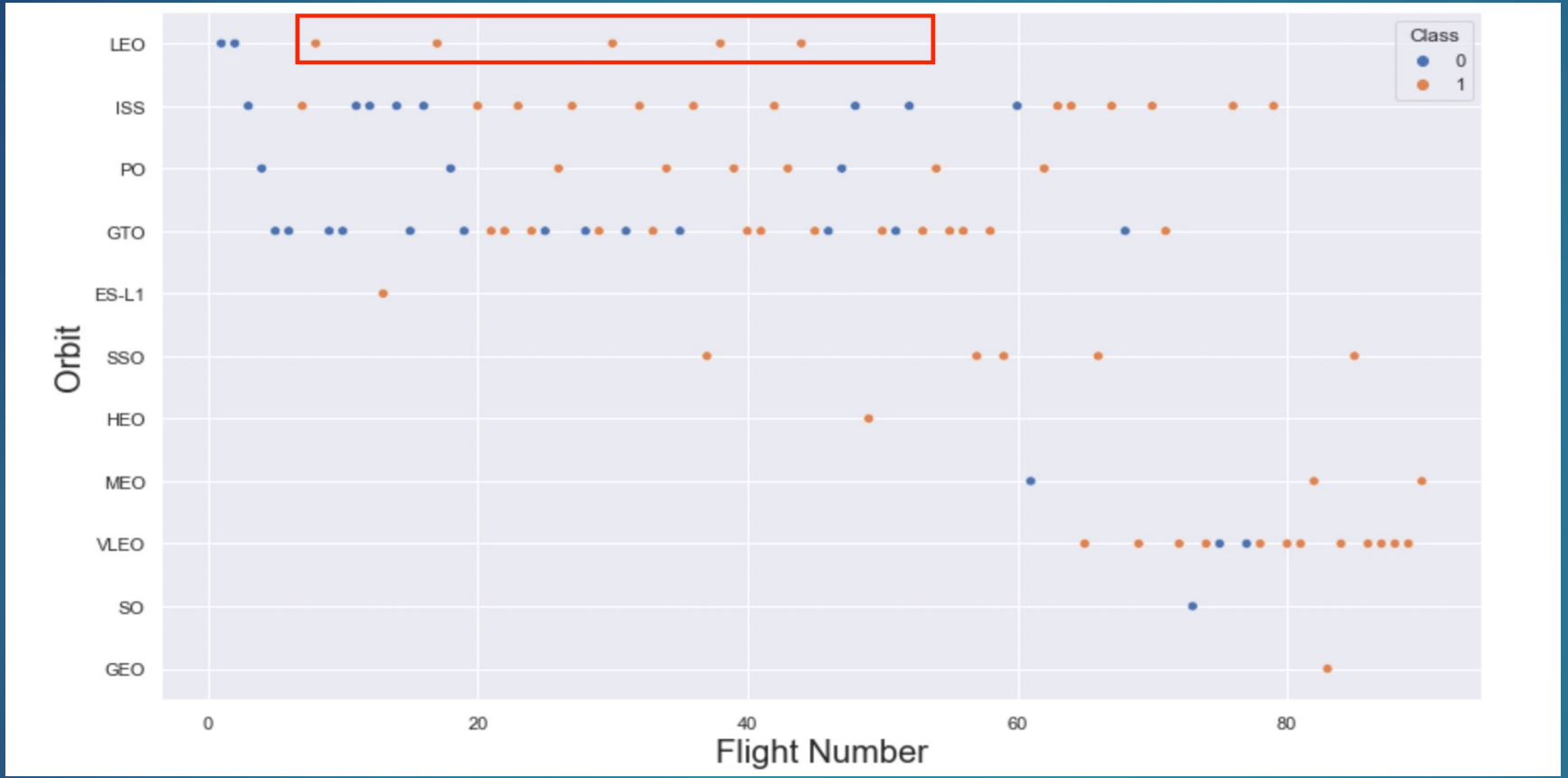
- The success rate is higher if the the payload mass of the rocket is more than 8000kg.
- *But there is no clear pattern found with Launch Site is dependent on Pay Load Mass for a success launch.*

Success Rate vs. Orbit Type



- Orbits ES-L1, SSO, HEO, GEO have highest Success rates.
- Whereas SO orbit has lowest success rate.

Flight Number vs. Orbit Type

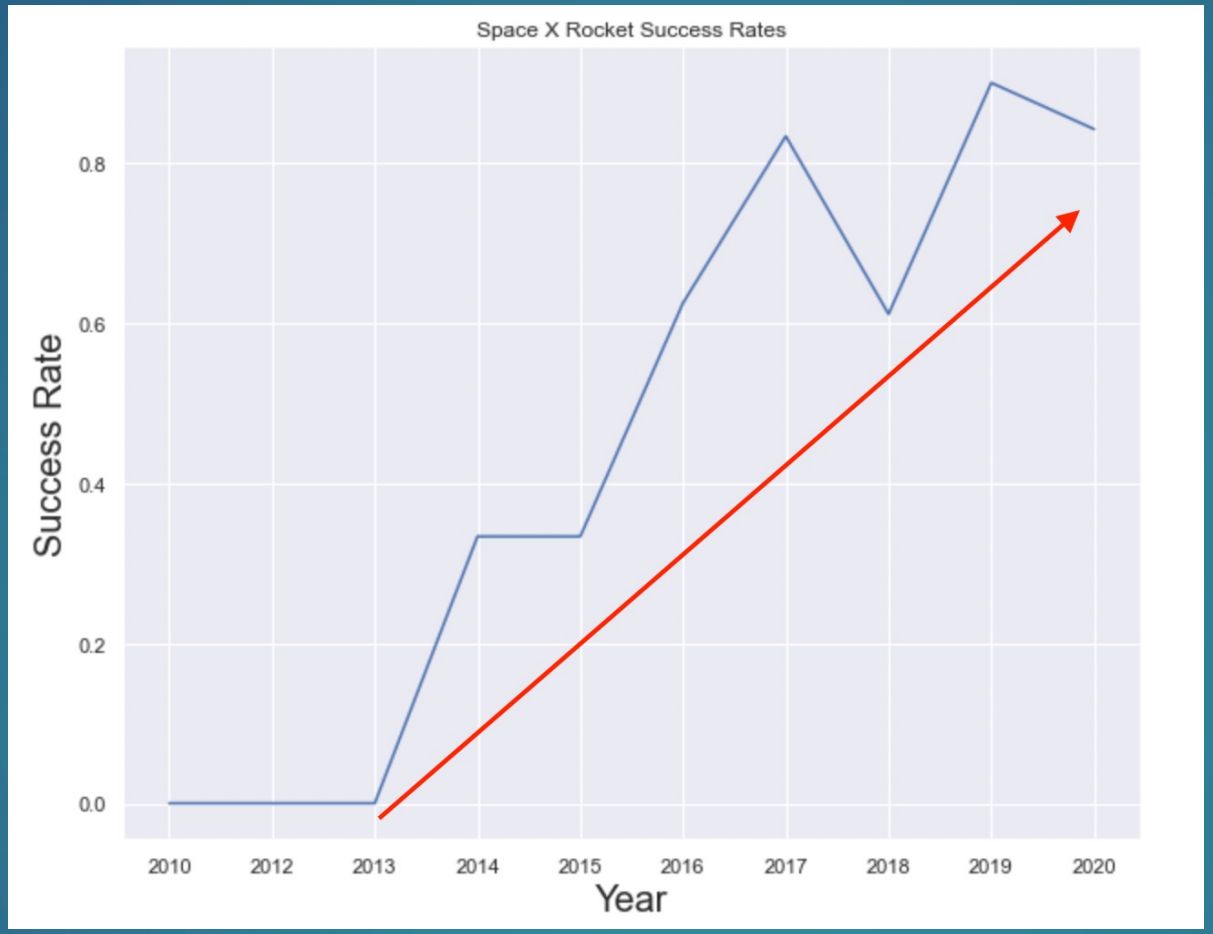


- The LEO orbit the Success appears related to the number of flights;
- On the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- Also we can find that ES-L1, SSO and HEO have success with lighter payloads.
- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend



- the success rate since 2013 kept increasing till 2020

All Launch Site Names


```
In [8]: %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
 * ibm_db_sa://tzg30766:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

Out[8]:

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

- There are 4 distinct sites available for rocket launch

Launch Site Names Begin with 'CCA'

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

```
In [9]: %sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
 * ibm_db_sa://tzg30766:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

Out[9]:

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

- The limit query restricts the number of output rows to be shown
- The LIKE operator is used to apply the string patterns in SQL, here in this case the Site names starts with CCA

Total Payload Mass

In [10]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)" \
FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';
```

```
 * ibm_db_sa://tzg30766:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

Out[10]:

Total Payload Mass by NASA (CRS)
45596

- Sum of all NASA launches is around 46000 KG.

Average Payload Mass by F9 v1.1


```
In [12]: %sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster Version F9 v1.1" \
         FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';

          * ibm_db_sa://tzg30766:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:307
         56/bludb
         Done.

Out[12]:
```

Average Payload Mass by Booster Version F9 v1.1
2928

- The Average payload carried by F9 v1.1 is around 2000KG, which is low

First Successful Ground Landing Date


```
In [13]: %sql SELECT MIN(DATE) AS "First Succesful Landing Outcome in Ground Pad" \
         FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (ground pad)';

          * ibm_db_sa://tzg30766:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:307
         56/bludb
         Done.

Out[13]:
```

First Succesful Landing Outcome in Ground Pad
2015-12-22

- The first successful launch was on 22nd Dec 2015

Successful Drone Ship Landing with Payload between 4000 and 6000

In [14]:
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

```
 * ibm_db_sa://tzg30766:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

Out[14]:

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

- There are 4 booster versions which has success for the payload between 4000 and 6000

Total Number of Successful and Failure Mission Outcomes

In [21]:
```
%sql SELECT sum(CASE WHEN MISSION_OUTCOME LIKE '%Success%' THEN 1 ELSE 0 END) AS "Successful Mission", \
    sum(CASE WHEN MISSION_OUTCOME LIKE '%Failure%' THEN 1 ELSE 0 END) AS "Failure Mission" \
FROM SPACEXTBL;
```

```
 * ibm_db_sa://tzg30766:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

Out[21]:

Successful Mission	Failure Mission
100	1

- Out of 101 missions, there are 100 successes and 1 failure

Boosters Carried Maximum Payload

In [24]:
```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

```
 * ibm_db_sa://tzg30766:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

Out[24]:

Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

- There are 12 distinct booster versions that have carried highest payload

2015 Launch Records

In [27]:
```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL \
WHERE DATE LIKE '2015-%' AND LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://tzg30766:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

Out[27]:

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

- Two booster versions in 2015 have registered the failures and both are from the same launch site.

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

In [28]:
```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count", \
ROW_NUMBER() OVER () AS "RANK" FROM SPACEXTBL \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING__OUTCOME \
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```

```
 * ibm_db_sa://tzg30766:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

Out[28]:

Landing Outcome	Total Count	RANK
No attempt	10	1
Failure (drone ship)	5	2
Success (drone ship)	5	3
Controlled (ocean)	3	4
Success (ground pad)	3	5
Failure (parachute)	2	6
Uncontrolled (ocean)	2	7
Precluded (drone ship)	1	8

- Both failure and success counts for the drone ship are equal

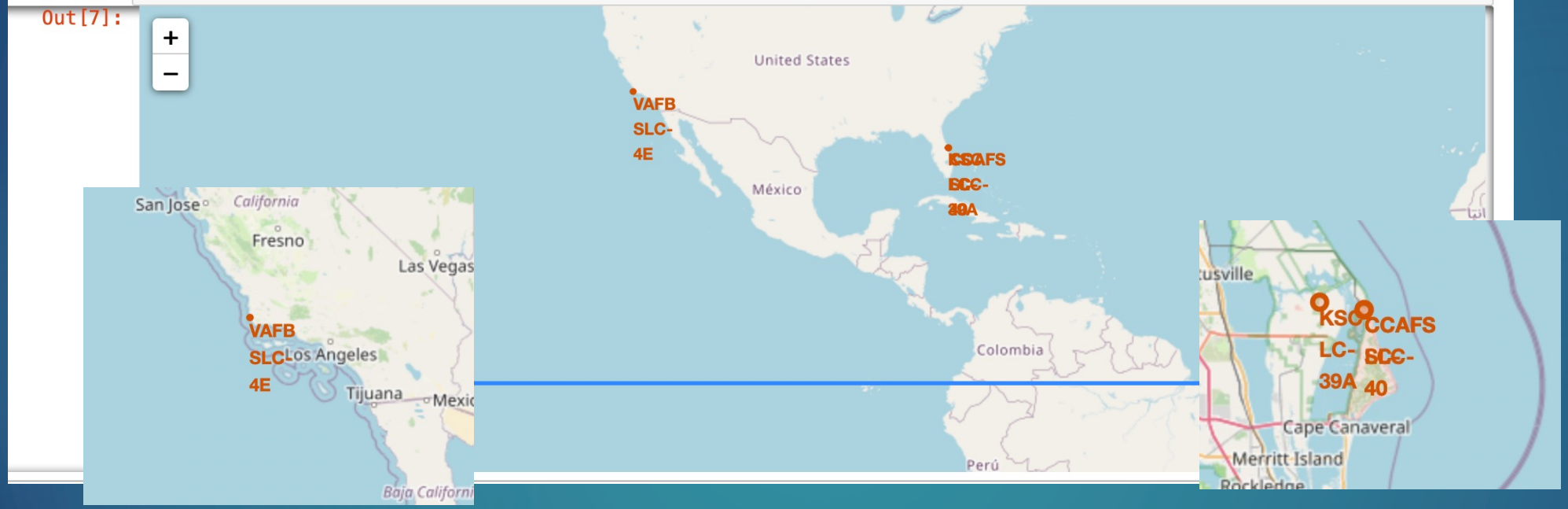
Section 4

Launch Sites Proximities Analysis

Launch Sites in USA

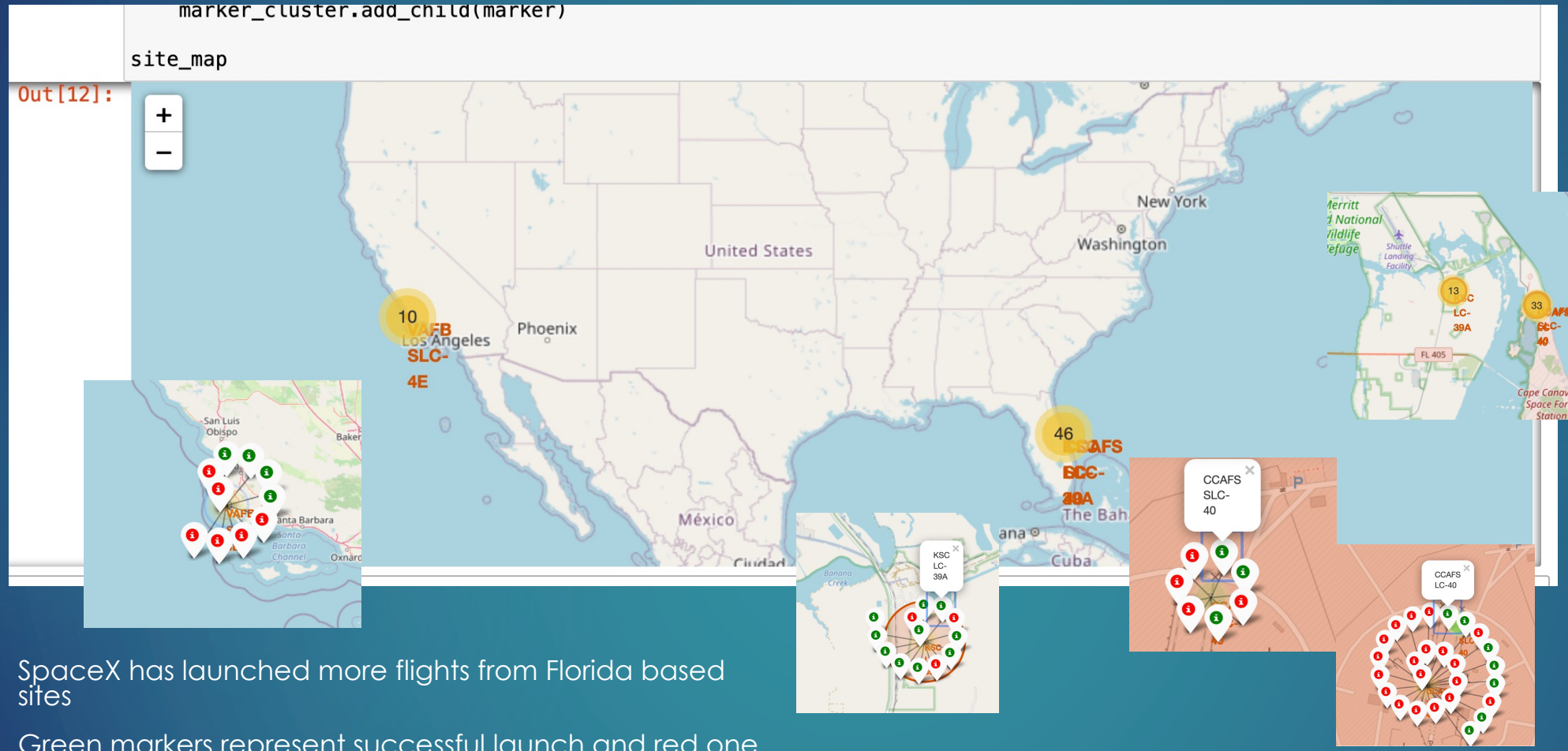

```
#Add equator line
equator_coordinates=[ [0, 195],
    [0, -450]]
equator_line=folium.PolyLine(locations=equator_coordinates,weight=3)
site_map.add_child(equator_line)
site_map
```

Out[7]:



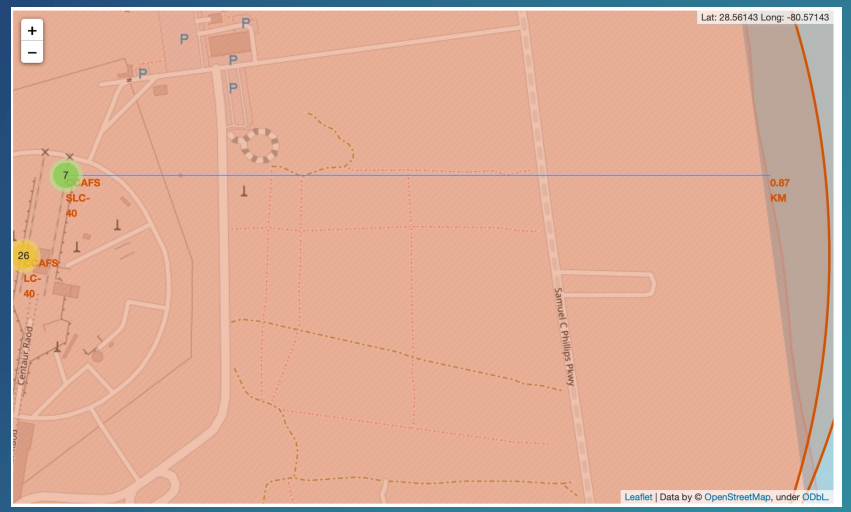
- Equator line is shown in Blue color
- Are all launch sites in proximity to the Equator line? **No, the launch sites are not in proximity to the equator line.**
- Are all launch sites in very close proximity to the coast? **Yes, we can see the same in the zoomed images.**

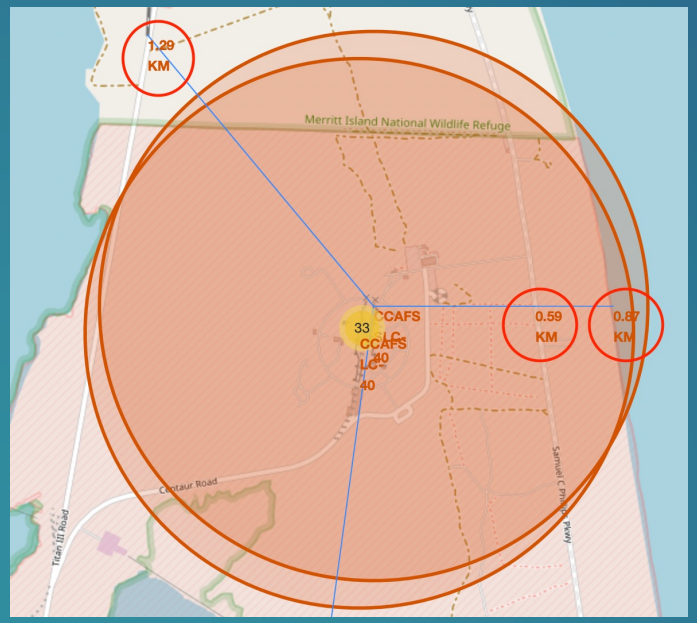
SpaceX - Launches

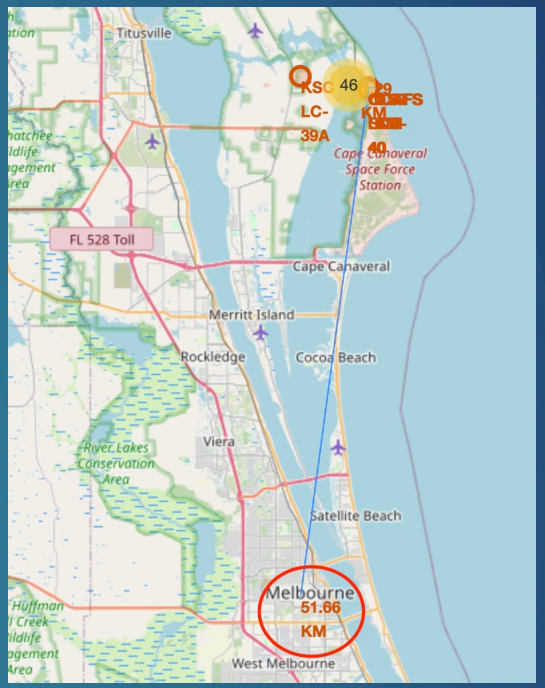


- SpaceX has launched more flights from Florida based sites
- Green markers represent successful launch and red one for failure
- KSC LC – 39A has more successful launches
- Out of 4 sites3 are located near to East coast

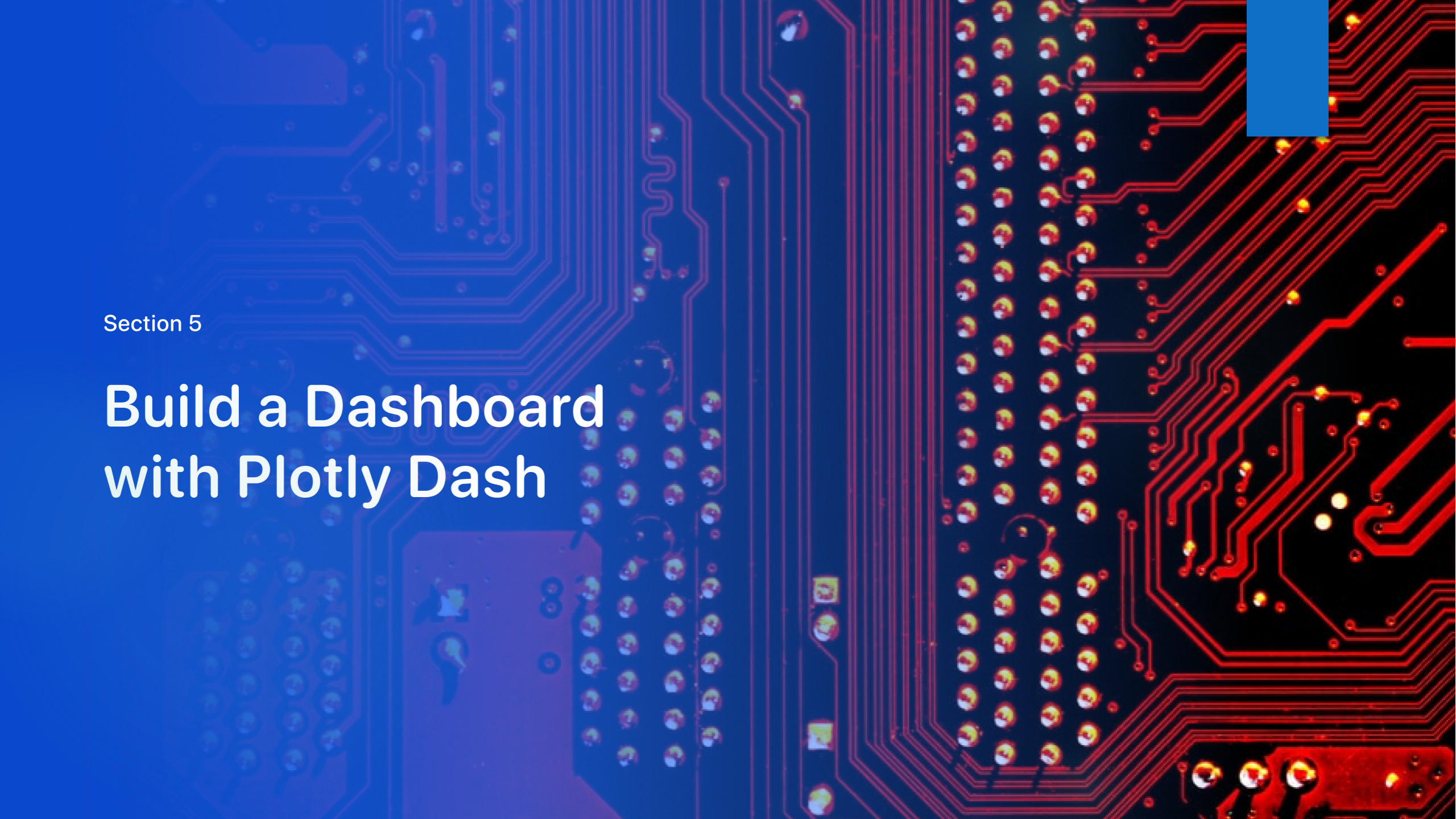
Distances from site CCAFS SLC - 40







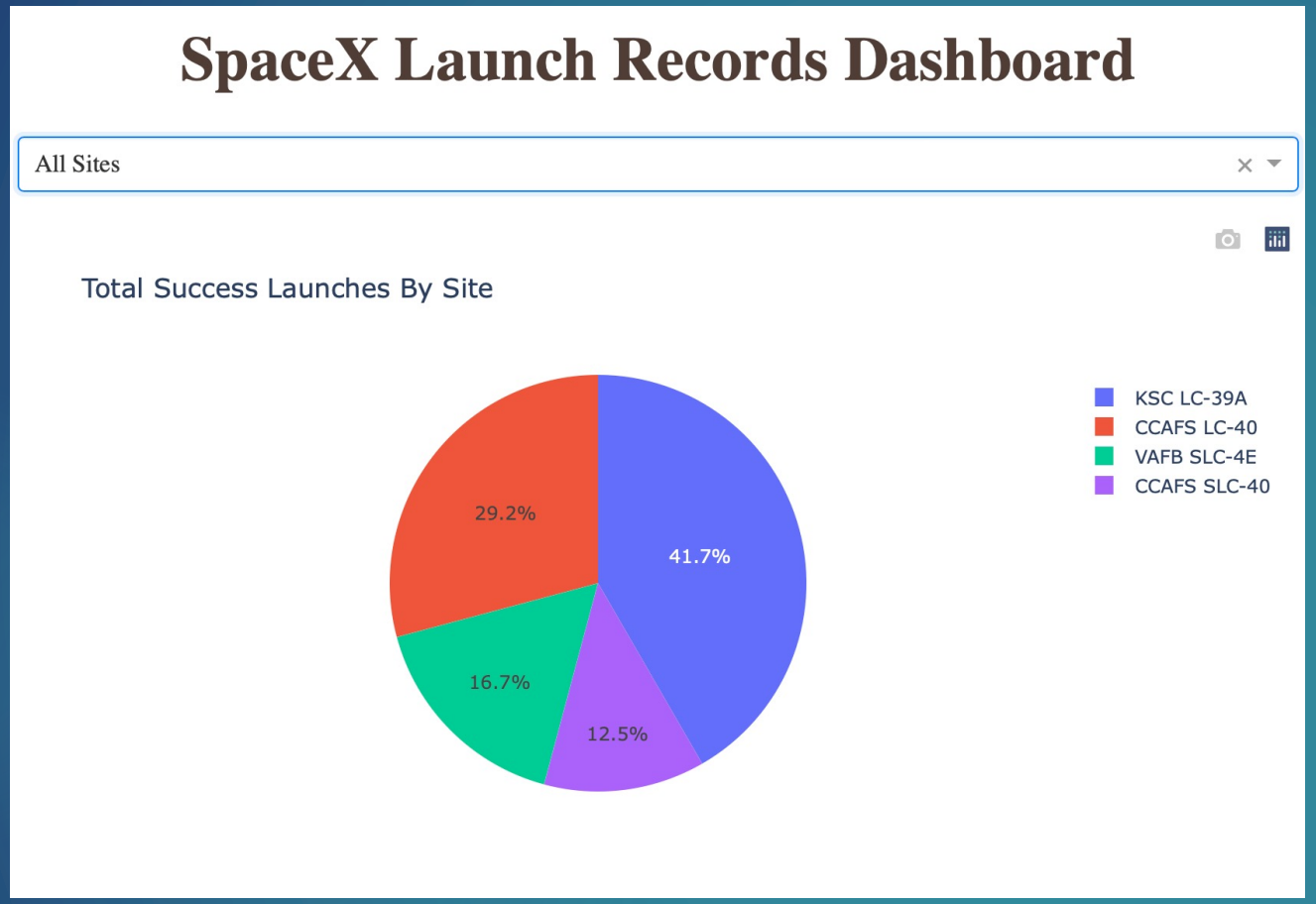
- The selected site CCAFS SLC – 40 distances from Coast, Railway and Highway are
  - Under 0.9KM, 1.3KM and 0.6 KM respectively.
- The Melbourne city from the same site is bit far (51KM)
- Most of the sites are in close proximity to the transport lines such as Coast, Railway and Highways

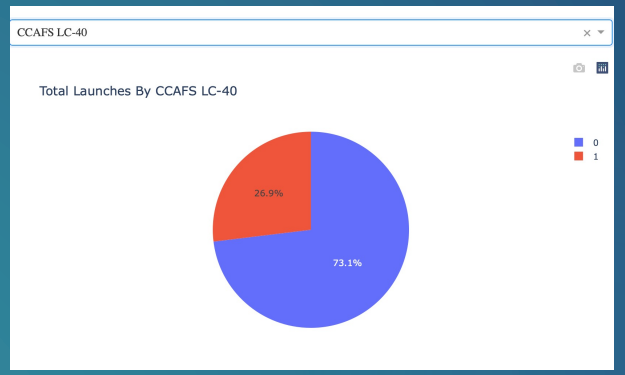


Section 5

Build a Dashboard with Plotly Dash

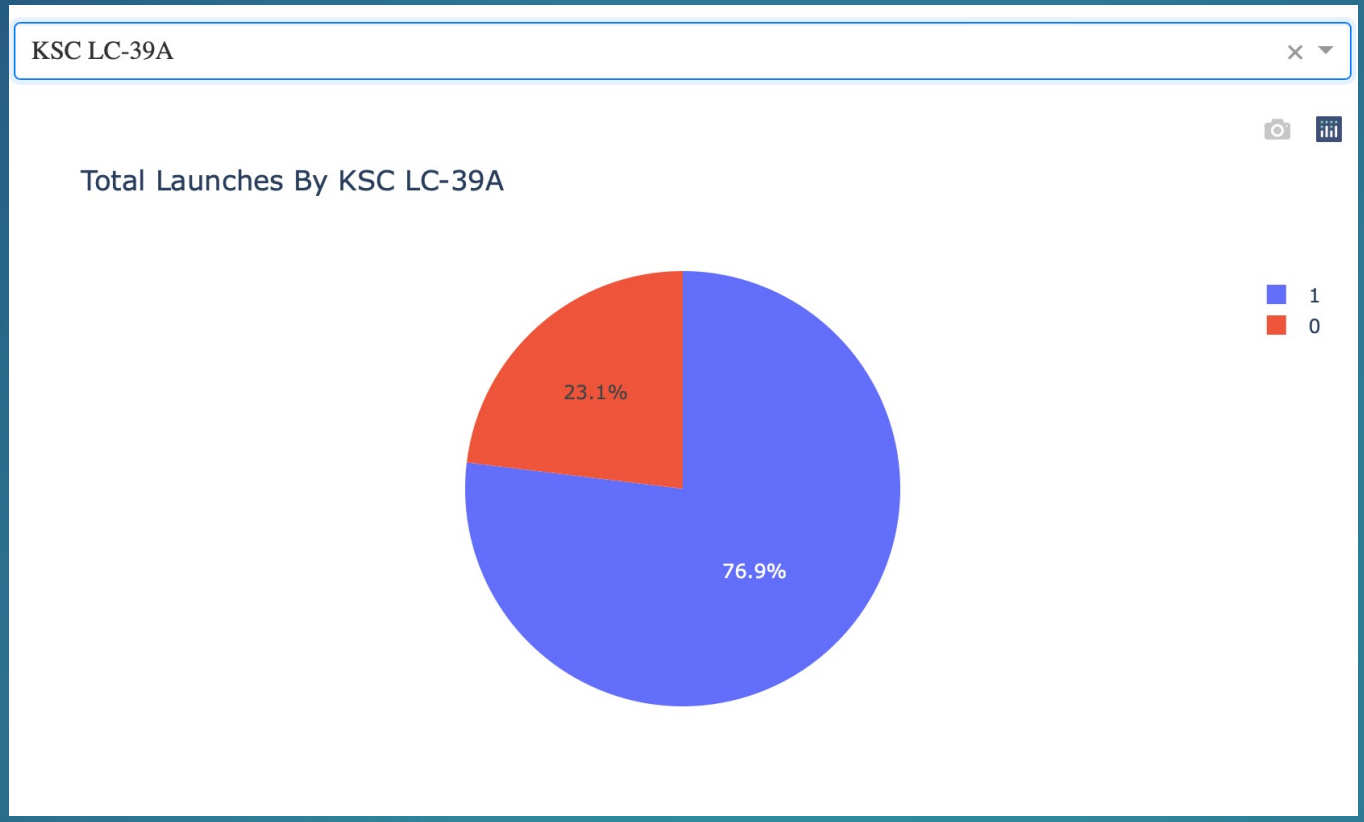
SpaceX Launches by Site





- KSC LC-39A is the leader of the board

KSC LC – 39A Site Success Rate



- KSC LC-39A site has 76.9% successful launches

Correlation Between Payload and Success

- We can observe that the FT booster version has good success rate for lesser payloads
- The FT booster success rate deteriorated for higher payloads
- v1.1 booster has low success rate

Section 6

Predictive Analysis (Classification)

Logistic Regression - Classification

```
In [12]: print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
         print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

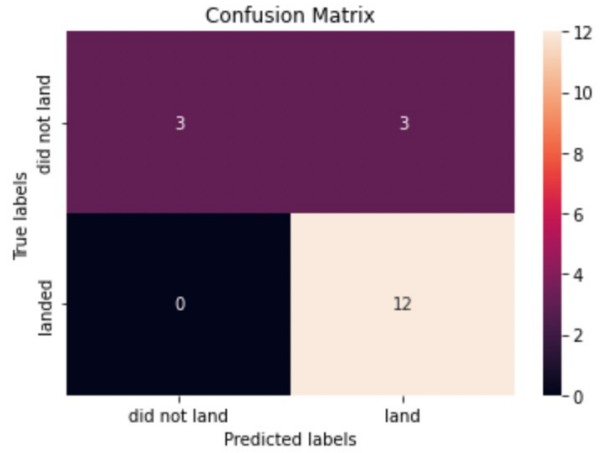
Calculate the accuracy on the test data using the method `score`:

```
In [13]: print('Accuracy for logistic regression using the method score:', logreg_cv.score(X_test, Y_test))
```

```
Accuracy for logistic regression using the method score: 0.8333333333333334
```

Lets look at the confusion matrix:

```
In [14]: yhat=logreg_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



Accuracy for logistic regression using the method is at 83%

SVM Classification

```
In [17]: print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
         print("accuracy :",svm_cv.best_score_)
```

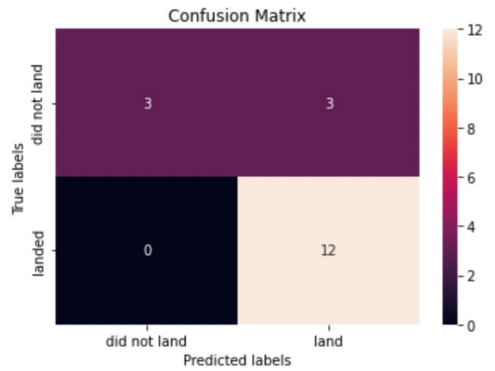
```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

```
In [18]: print("Accuracy for support vector machine on the test data using the method score:", svm_cv.score(X_test, Y_test))
```

```
Accuracy for support vector machine on the test data using the method score: 0.8333333333333334
```

We can plot the confusion matrix

```
In [19]: yhat=svm_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



Accuracy SVM classification model is also at 83%

Tuning with Sigmoid kernel parameter is good

Decision Tree Classification

```
In [22]: print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
         print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqrt', 'min_sam
ples_leaf': 4, 'min_samples_split': 10, 'splitter': 'best'}
accuracy : 0.875
```

```
In [23]: print("Accuracy for decision tree classifier on the test data using the method score:", tree_cv.score(X_test, Y_test
```

```
Accuracy for decision tree classifier on the test data using the method score: 0.8333333333333334
```

We can plot the confusion matrix

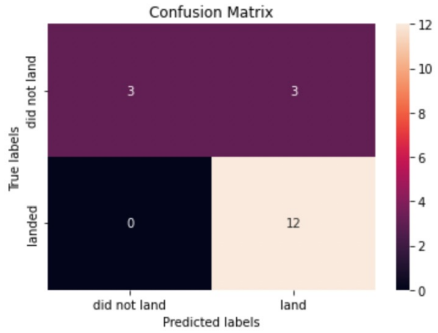
```
In [24]: yhat = tree_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```

Decision Tree classification model is at 88%

KNN Classification

```
In [25]: parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                       'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                       'p': [1,2]}

         KNN = KNeighborsClassifier()

In [26]: gs_cv = GridSearchCV(KNN, parameters, scoring='accuracy', cv=10)
         knn_cv = gs_cv.fit(X_train, Y_train)

In [27]: 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
```

```
In [28]: print("Accuracy for k nearest neighbors on the test data using the method score: ",knn_cv.score(X_test, Y_test))

         Accuracy for k nearest neighbors on the test data using the method score:  0.8333333333333334

         We can plot the confusion matrix

In [29]: yhat = knn_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



Accuracy for KNN classification is at 83%

Classification Models - Comparison

In [30]:
```
algorithms = {'KNN':knn_cv.best_score_,'Decision Tree':tree_cv.best_score_,'Logistic Regression':logreg_cv.best_scor
best_algorithm = max(algorithms, key= lambda x: algorithms[x])

print('The method which performs best is \"',best_algorithm,'\" with a score of',algorithms[best_algorithm])
```

```
The method which performs best is " Decision Tree " with a score of 0.875
```

In [31]:
```
algo_df = pd.DataFrame.from_dict(algorithms, orient='index', columns=['Accuracy'])
```

In [32]:
```
algo_df.sort_values(['Accuracy'], inplace=True)
algo_df
```

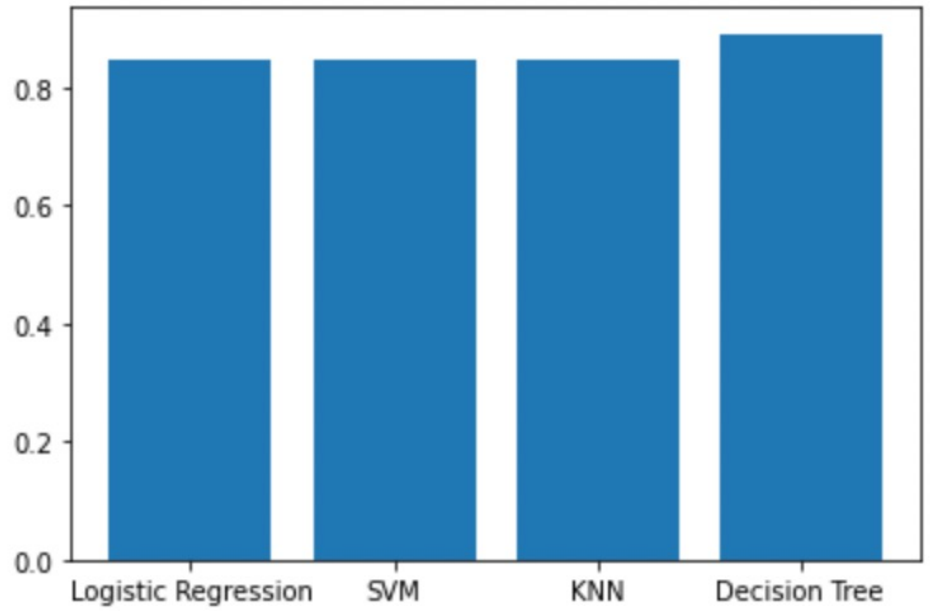
Out[32]:

	Accuracy
Logistic Regression	0.846429
SVM	0.848214
KNN	0.848214
Decision Tree	0.875000

- Decision Tree Classification has good accuracy

Classification Accuracy – Bar Chart


```
In [34]: plt.bar(algo_df.index,algo_df['Accuracy'])
Out[34]: <BarContainer object of 4 artists>
```



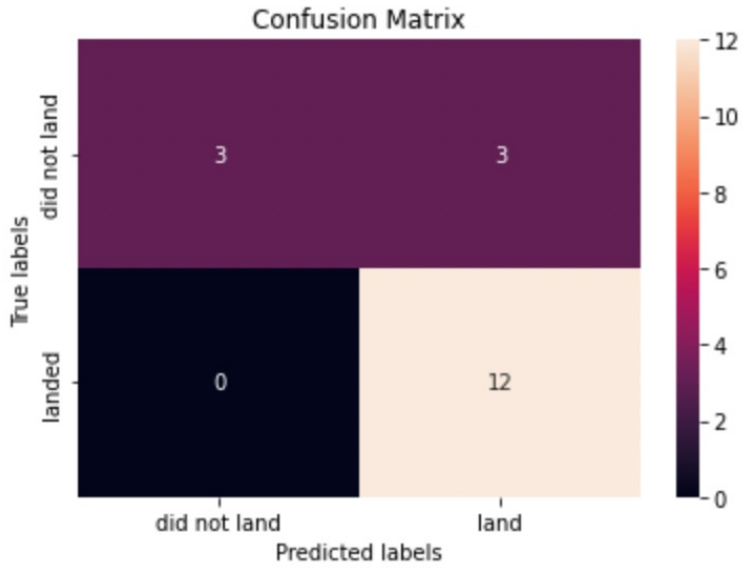
Out[33]:

	Accuracy
Logistic Regression	0.846429
SVM	0.848214
KNN	0.848214
Decision Tree	0.891071

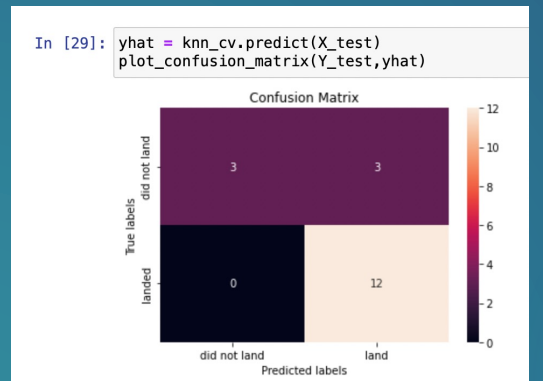
- Decision Tree Classification has good classification accuracy on train set

Confusion Matrices

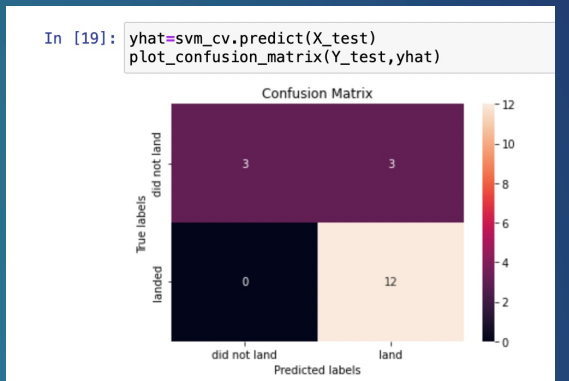

```
In [24]: yhat = tree_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



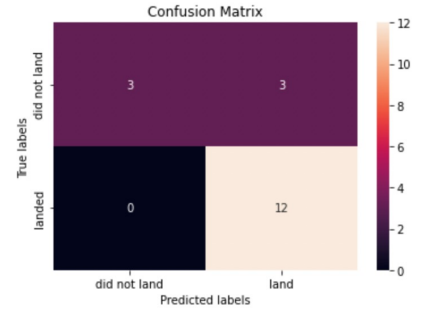

```
In [29]: yhat = knn_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```




```
In [19]: yhat=svm_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```




```
In [14]: yhat=logreg_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



- Decision Tree Classification accuracy is matching with other classifications with 83%

Conclusions

- Success rate have been increasing since 2013 for SpaceX, which is a good sign
- Orbits GEO, HEO, SSO and ES-L1 has higher success rates
- KSC LC – 39A site has most successful launches
- Prefer using Decision Tree classification prediction model for machine learning
- The sites are in close proximity with regard to different transport lines
- Most of the launches are made from Eastern launch sites

Thank you!