IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Harry Mok
30th Dec, 2021

Outline

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

Executive Summary

- Summary of methodologies
  - Utilization of Data Collection, Web Scrapping, Data Wrangling, SQL, Exploratory Data Analysis, Visualization, Interactive Dashboard, Machine Learning for result prediction

- Summary of all results
  - KNN is the most appropriate prediction method for launch result
  - success rate since 2013 kept increasing till 2020

Introduction

- In this capstone, we will predict if the Falcon 9 first stage will land 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. In this module, you will be provided with an overview of the problem and the tools you need to complete the course.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data collected by web scrapping of Space X open-source data on API
- Perform data wrangling
  - Data wrangling is done by encoding the landing results into 0 or 1 landing class.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - KNN, SVM, Logistic Regression, Decision Tree methods for classification
  - GridSearchCV for parameters optimization
  - Accuracy scores for model evaluation

Data Collection

- Describe how data sets were collected.
- You need to present your data collection process use key phrases and flowcharts
  - Web scrapping of Wikipedia
  - requesting rocket launch data from SpaceX API with URL
  - Put data into Pandas Dataframe

Data Collection – SpaceX API

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

- Add the GitHub URL of the completed SpaceX API calls notebook (must include completed code cell and outcome cell), as an external reference and peer-review purpose

1. **Import Libraries and Define Auxiliary Functions**
2. **Request and parse the SpaceX launch data using the GET request**
3. **Filter the data frame to only include Falcon 9 launches**
4. **Dealing with Missing Values**

https://gist.github.com/harry41108/cfb581aaaf0163105ee5ca59d6fd987c

Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts

- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose

1. **Request the Falcon9 Launch Wiki page from its URL**
2. **Extract all column/variable names from the HTML table header**
3. **Create a data frame by parsing the launch HTML tables**

https://gist.github.com/harry41108/cd8436ad3ac009f261b51b606666fbcd

Data Wrangling

- Describe how data were processed
- You need to present your data wrangling process using key phrases and flowcharts
- Add the GitHub URL of your completed data wrangling related notebooks, as an external reference and peer-review purpose

  - **Import Libraries and Define Auxiliary Functions**
  - **Calculate the number of launches on each site**
  - **Calculate the number and occurrence of each orbit**
  - **Calculate the number and occurence of mission outcome per orbit type**
  - **Create a landing outcome label from Outcome column**

https://gist.github.com/harry41108/a33d89574636dc2bcdc584ae0fb65ad3

EDA with Data Visualization

- Summarize what charts were plotted and why you used those charts
- Add the GitHub URL of your completed EDA with data visualization notebook, as an external reference and peer-review purpose

- **Visualize the relationship between Flight Number and Launch Site (Catplot)**
- **Visualize the relationship between Payload and Launch Site (Catplot)**
- **Visualize the relationship between success rate of each orbit type (Barplot)**
- **Visualize the relationship between FlightNumber and Orbit type (Catplot)**
- **Visualize the relationship between Payload and Orbit type (Catplot)**
- **Visualize the launch success yearly trend (Lineplot)**

https://gist.github.com/harry41108/6f4c9373548d08821543fd946d2529fc

EDA with SQL

- Using bullet point format, summarize the SQL queries you performed
- Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose

  - 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 failed landing_outcomes in drone ship, their booster versions, and launch site names for 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

https://gist.github.com/harry41108/72995b0333d5eac42a7616274a5c3b00

Build an Interactive Map with Folium

- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map
- Explain why you added those objects
- Add the GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose

    - Markers
    - Circles
    - Lines

https://gist.github.com/harry41108/cd6065b7ed503b48906c5b3725af18cc

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- Explain why you added those plots and interactions
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

Bar Chart
Dropdown
Slider
Scatter Chart

https://gist.github.com/harry41108/47e64dff1f8f4a938ab28603164f51a5

Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved, and found the best performing classification model
- You need present your model development process using key phrases and flowchart
- Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose

Create a NumPy array from the column Class in data, by applying the method to_numpy() then assign it to the variable Y,make sure the output is a Pandas series (only one bracket df['name of column'])

Standardize the data in X then reassign it to the variable X using the transform provided below.

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

Create a logistic regression object then create a GridSearchCV object logreg_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters

Create a support vector machine object then create a GridSearchCV object svm_cv with cv - 10. Fit the object to find the best parameters from the dictionary parameters.

Create a decision tree classifier object then create a GridSearchCV object tree_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.

Create a k nearest neighbors object then create a GridSearchCV object knn_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.

https://gist.github.com/harry41108/e9eb0a65f05a8bfa995069bc756d2c5d

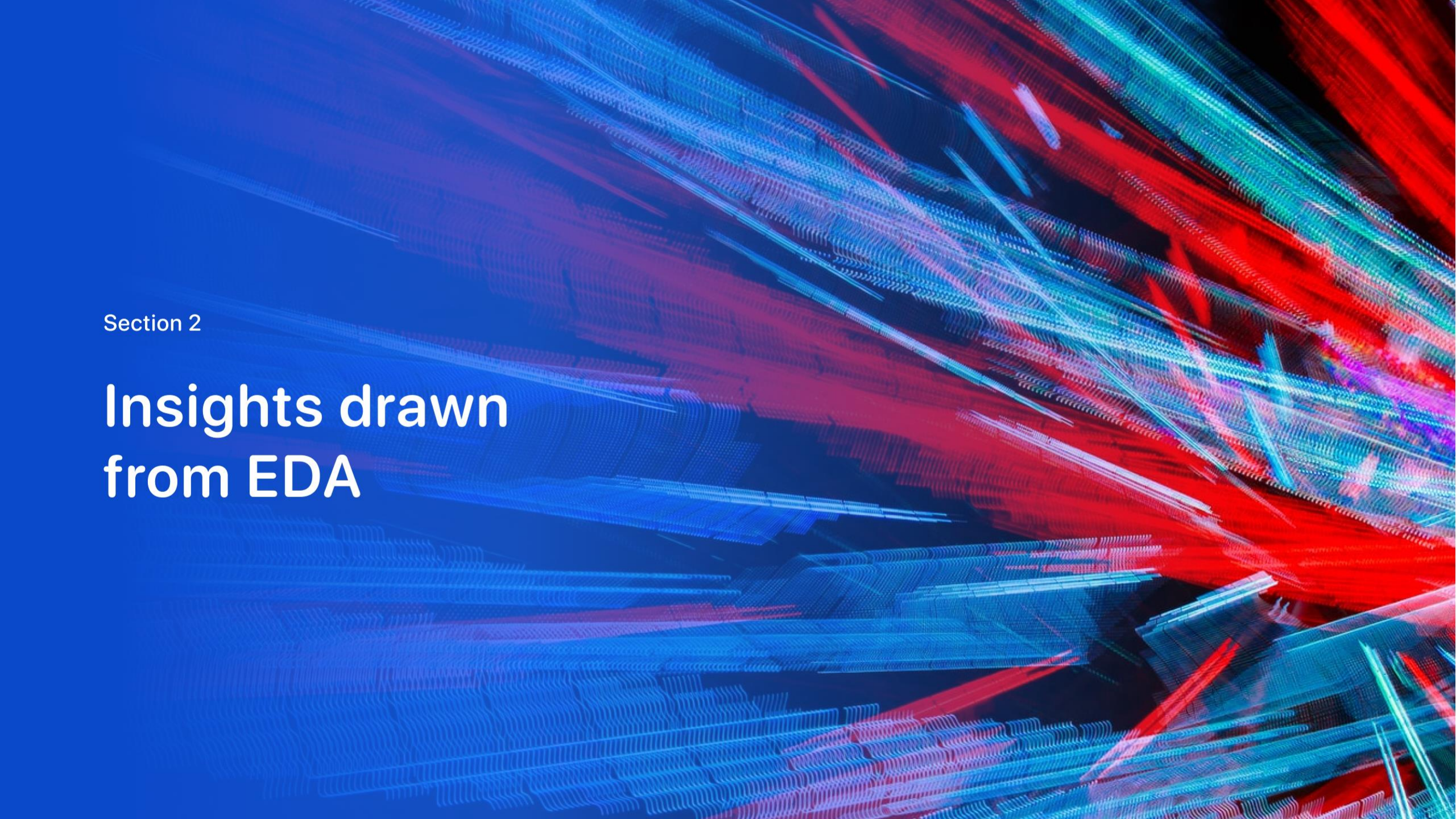
Results

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

  - KNN is the most appropriate prediction method for launch result
  - success rate since 2013 kept increasing till 2020

Section 2

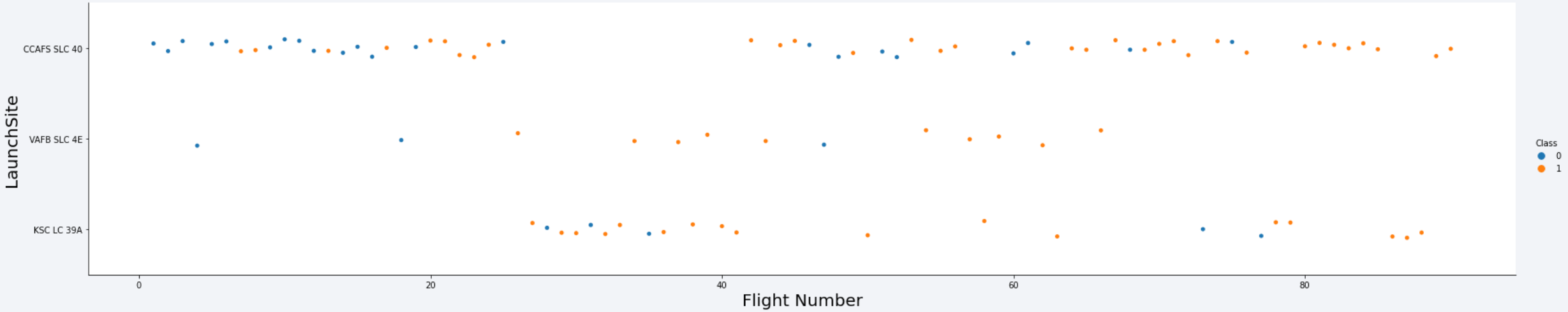
Insights drawn from EDA



Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site

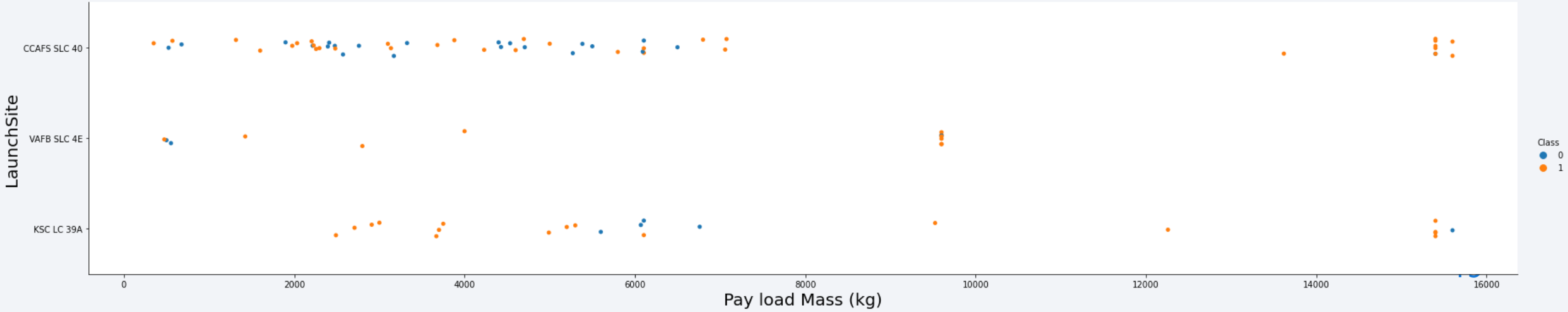
- Show the screenshot of the scatter plot with explanations



Payload vs. Launch Site

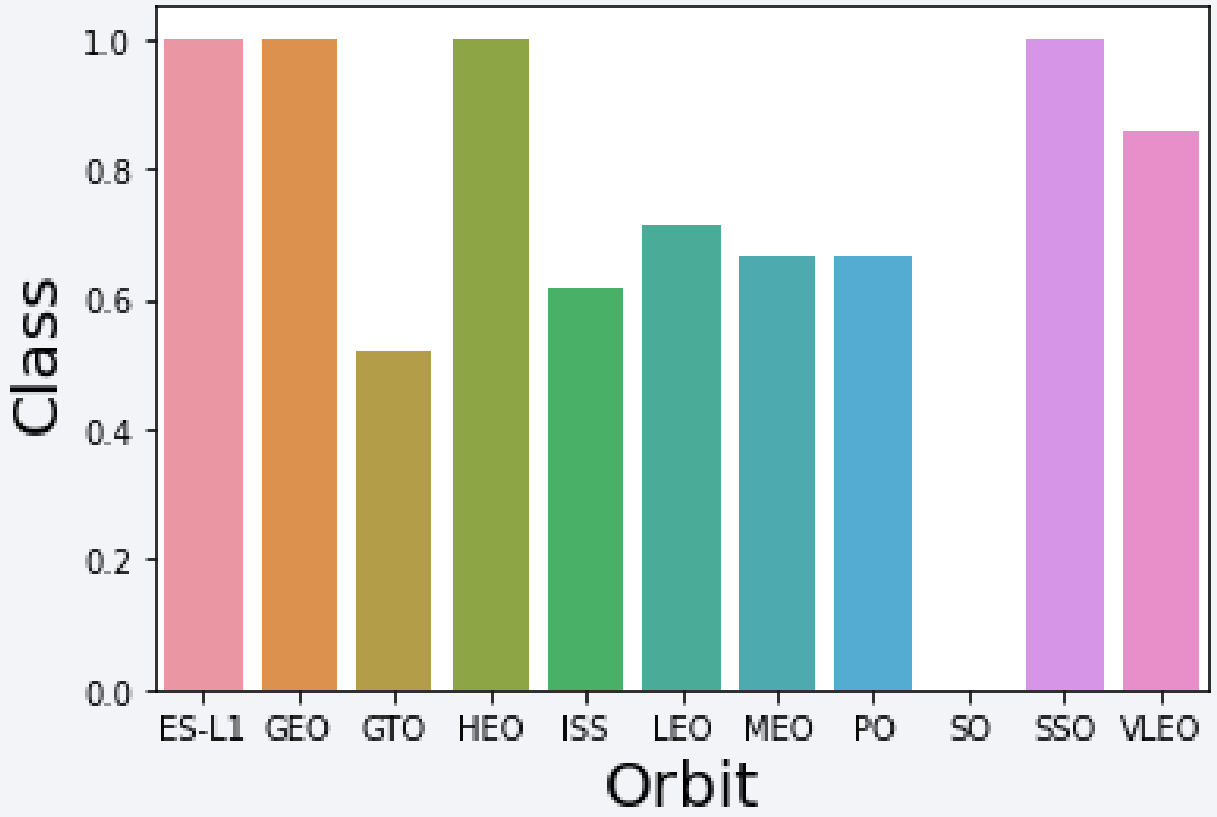
- Show a scatter plot of Payload vs. Launch Site

- Show the screenshot of the scatter plot with explanations



Success Rate vs. Orbit Type

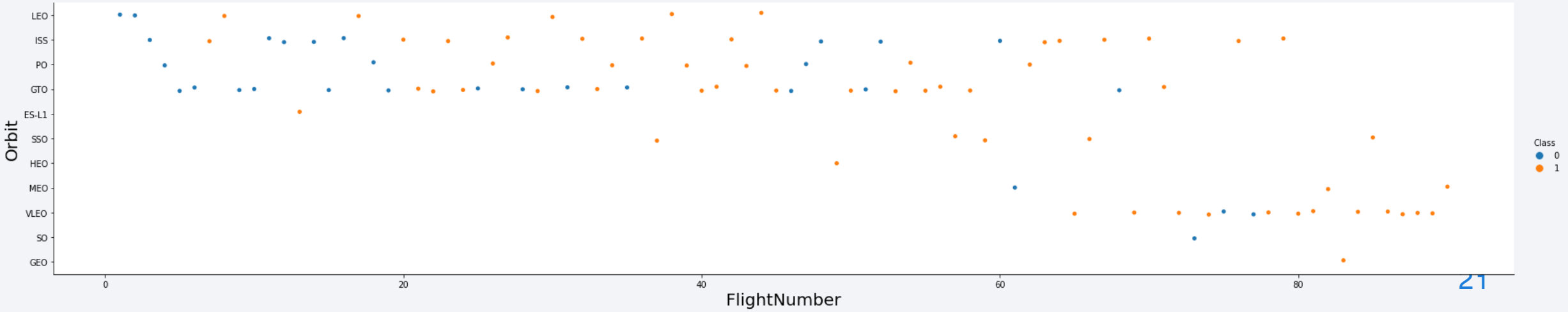
- Show a bar chart for the success rate of each orbit type
- Show the screenshot of the scatter plot with explanations



Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type

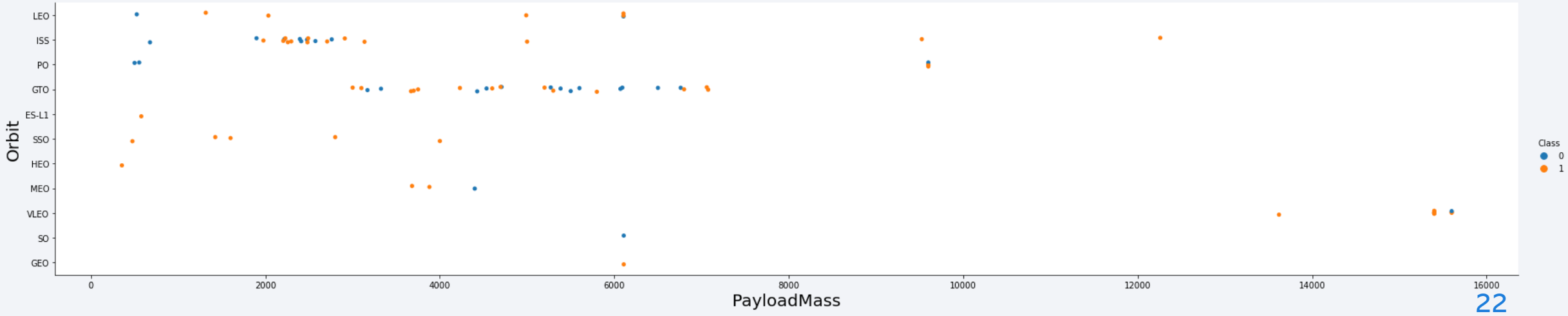
- Show the screenshot of the scatter plot with explanations



Payload vs. Orbit Type

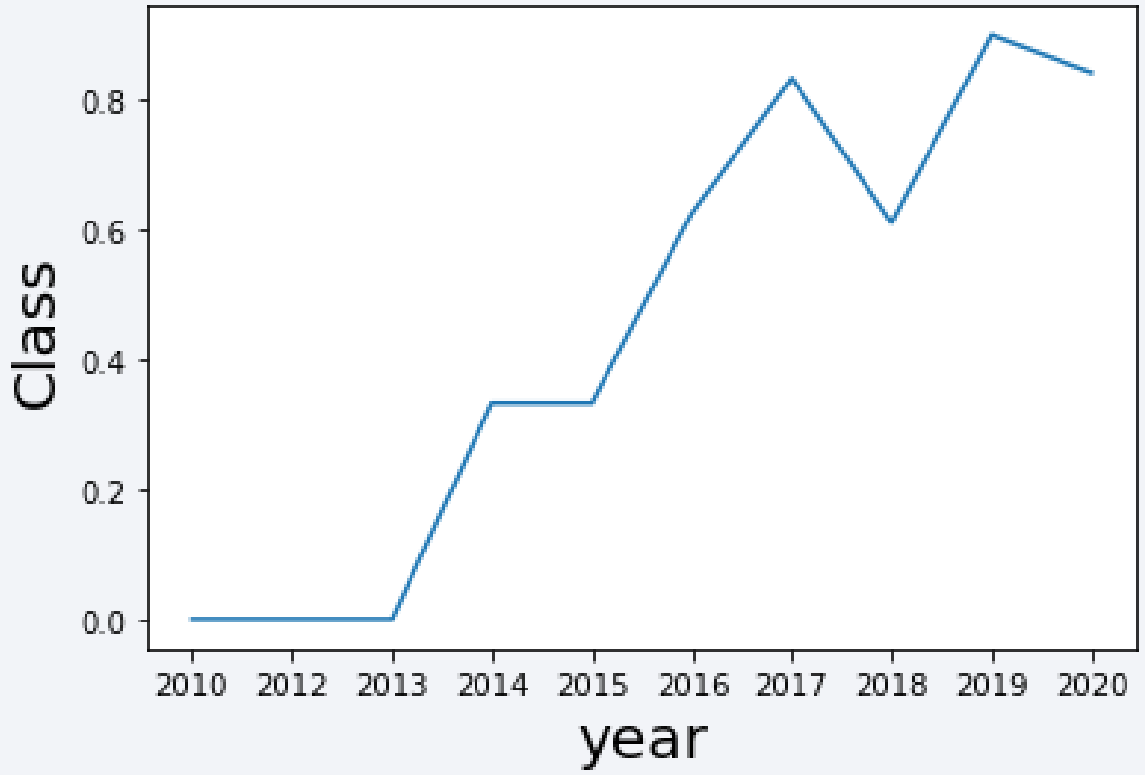
- Show a scatter point of payload vs. orbit type

- Show the screenshot of the scatter plot with explanations



Launch Success Yearly Trend

- Show a line chart of yearly average success rate
- Show the screenshot of the scatter plot with explanations



All Launch Site Names

- Find the names of the unique launch sites
- Present your query result with a short explanation here

Task 1

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

In [8]:
```
%sql SELECT DISTINCT LAUNCH_SITE FROM FHQ39276.SPACEXTBL;
```

```
 * ibm_db_sa://fhq39276:***@98538591-7217-4024-b027-8baa776ff
ludb
Done.
```

Out[8]:

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- Present your query result with a short explanation here

Task 2

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

In [9]:
```
%sql SELECT * FROM FHQ39276.SPACEXTBL WHERE LAUNCH_SITE LIKE '%CCA%' limit 5;
```

```
 * ibm_db_sa://fhq39276:***@98538591-7217-4024-b027-8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30875/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

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Present your query result with a short explanation here

Task 3

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

In [10]:
```
%sql SELECT SUM(payload_mass__kg_) FROM FHQ39276.SPACEXTBL WHERE CUSTOMER LIKE '%CRS%';
```

```
 * ibm_db_sa://fhq39276:***@98538591-7217-4024-b027-8baa776ffad1.c3n41cmd0nqnrk39u98g.data
ludb
Done.
```

Out[10]:

1
48213

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Present your query result with a short explanation here

Task 4

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

In [11]:
```
%sql SELECT AVG(payload_mass__kg_) FROM FHQ39276.SPACEXTBL WHERE booster_version LIKE '%F9 v1.1%';
```

```
 * ibm_db_sa://fhq39276:***@98538591-7217-4024-b027-8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdom
ludb
Done.
```

Out[11]:

1
2534

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Present your query result with a short explanation here

Task 5

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

Hint:Use min function

In [13]:
```
%sql SELECT MIN(DATE) FROM FHQ39276.SPACEXTBL WHERE landing__outcome LIKE '%Success (ground pad)%';
```

```
 * ibm_db_sa://fhq39276:***@98538591-7217-4024-b027-8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdom
ludb
Done.
```

Out[13]:

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

- Present your query result with a short explanation here

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

In [14]:
```
%sql SELECT DISTINCT booster_version FROM FHQ39276.SPACEXTBL WHERE landing__outcome LIKE '%Success (drone ship)%' AND
```

```
 * ibm_db_sa://fhq39276:***@98538591-7217-4024-b027-8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30875/b
ludb
Done.
```

Out[14]:

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Present your query result with a short explanation here

Task 7

List the total number of successful and failure mission outcomes

In [24]:
```
%sql SELECT COUNT(mission_outcome) AS Count1, mission_outcome FROM FHQ39276.SPACEXTBL GROUP BY mission_outcome
```

```
 * ibm_db_sa://fhq39276:***@98538591-7217-4024-b027-8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:
ludb
Done.
```

Out[24]:

count1	mission_outcome
1	Failure (in flight)
99	Success
1	Success (payload status unclear)

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Present your query result with a short explanation here

Task 8

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

In [31]:
```
%sql SELECT booster_version, payload_mass__kg_ FROM FHQ39276.SPACEXTBL ORDER BY payload_mass__kg_ DESC
```

```
 * ibm_db_sa://fhq39276:***@98538591-7217-4024-b027-8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32716/bludb
Done.
```

Out[31]:

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1049.7	15600
F9 B5 B1060.3	15600
F9 B5 B1051.6	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600
F9 B5 B1049.5	15600
F9 B5 B1051.4	15600
F9 B5 B1048.5	15600
F9 B5 B1056.4	15600
F9 B5 B1051.3	15600
F9 B5 B1049.4	15600

2015 Launch Records

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

- Present your query result with a short explanation here

Task 9

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

In [34]:
```
%sql SELECT DATE, landing__outcome, booster_version, launch_site FROM FHQ39276.SPACEXTBL WHERE landing__outcome LIKE '
```

```
 * ibm_db_sa://fhq39276:***@98538591-7217-4024-b027-8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30875/bludb
Done.
```

Out[34]:

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

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

- Present your query

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

In [39]:
```
%sql SELECT landing__outcome, COUNT(landing__outcome) AS count2 FROM FHQ39276.SPACEXTBL WHERE DATE BETWEEN '2010-06-04
```

```
 * ibm_db_sa://fhq39276:***@98538591-7217-4024-b027-8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30875/b
ludb
Done.
```

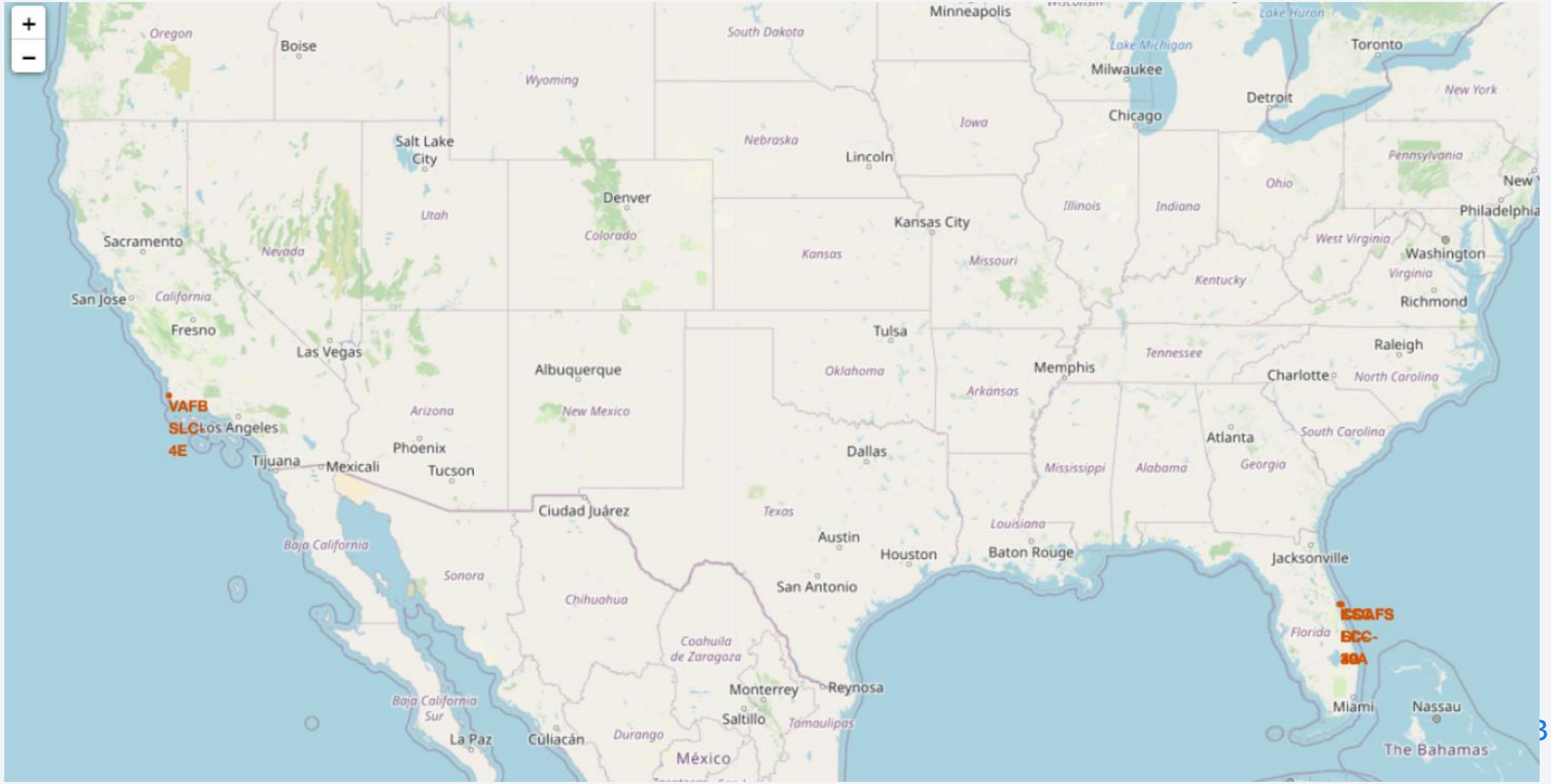
Out[39]:

landing__outcome	count2
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

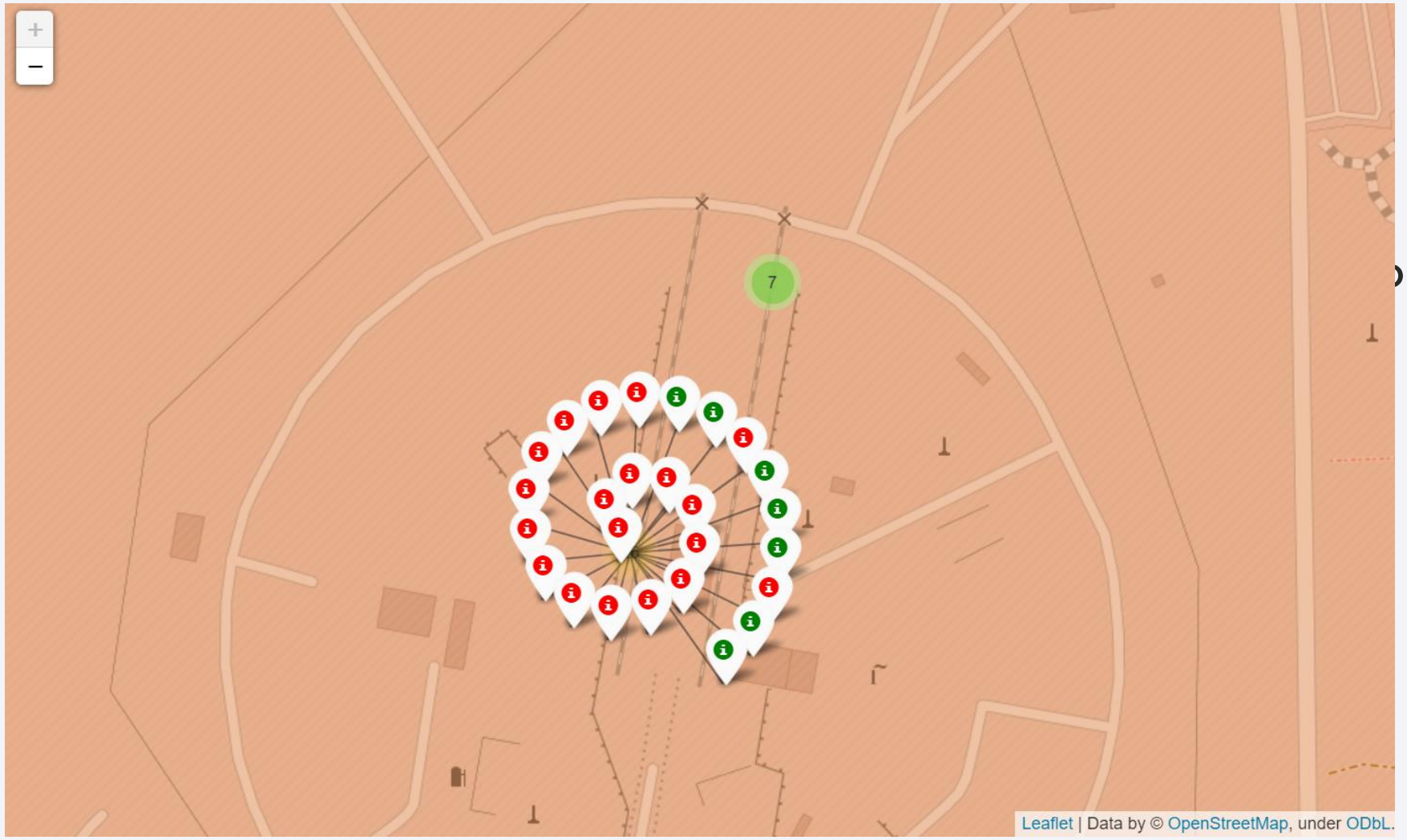
Section 4

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>



<Folium Map Screenshot 2>

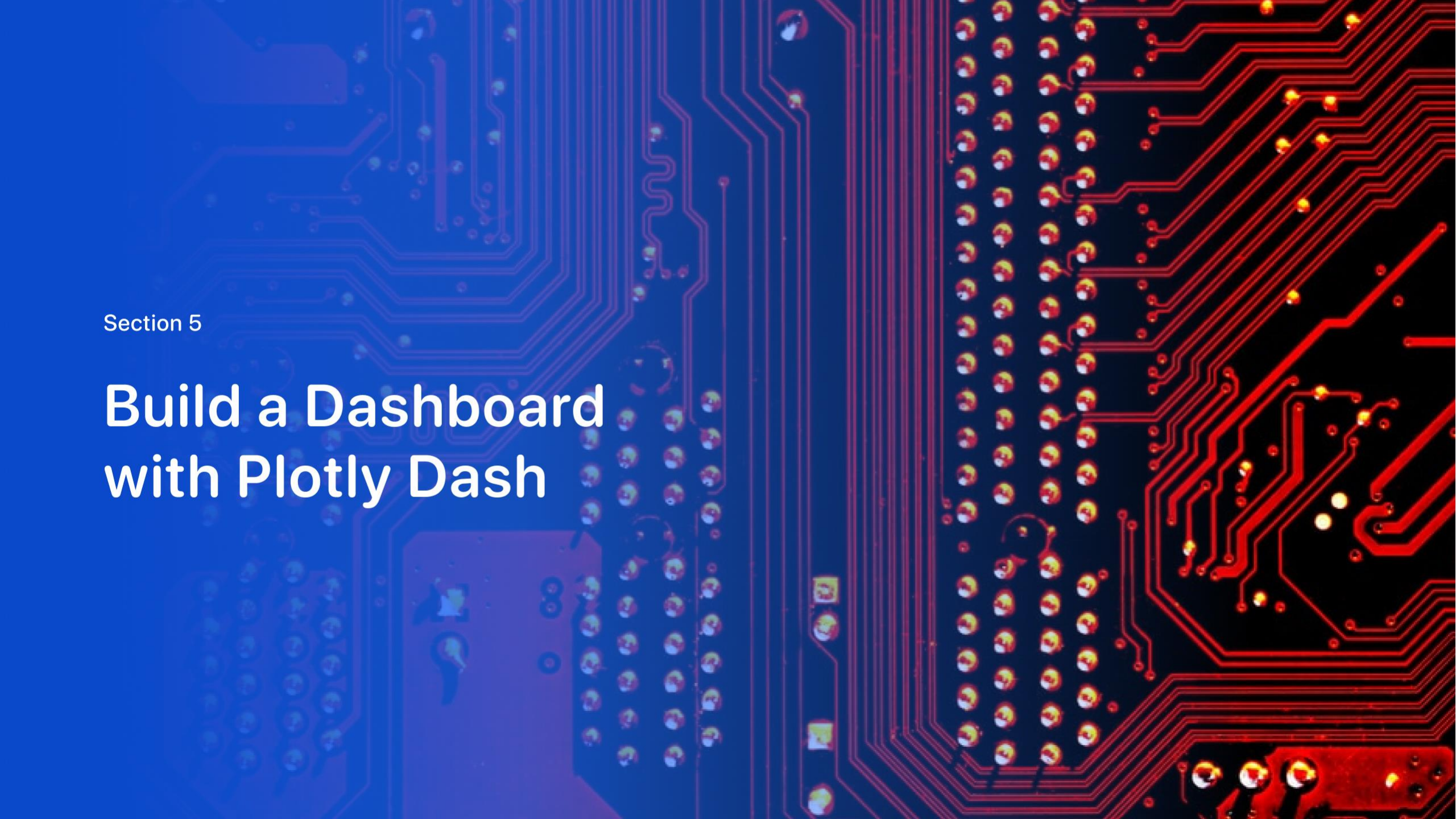
- 

- olor-
-

<Folium Map Screenshot 3>

Section 5

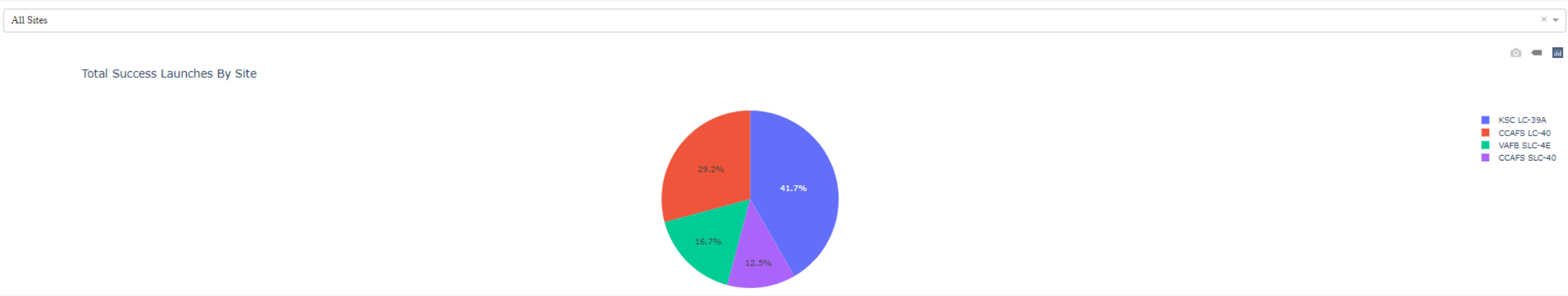
Build a Dashboard with Plotly Dash



<Dashboard Screenshot 1>

- Replace <Dashboard screenshot 1> title with an appropriate title

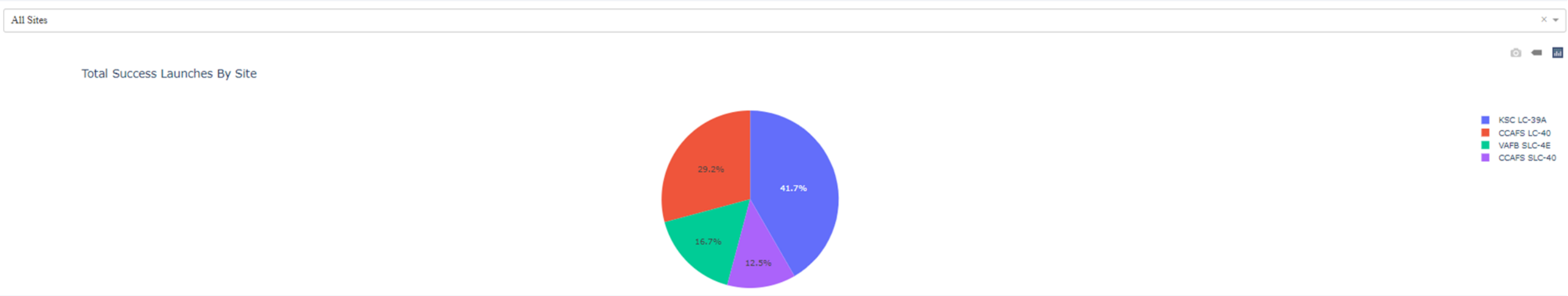
- Show the screenshot of launch success count for all sites, in a piechart



<Dashboard Screenshot 2>

- Replace <Dashboard screenshot 2> title with an appropriate title

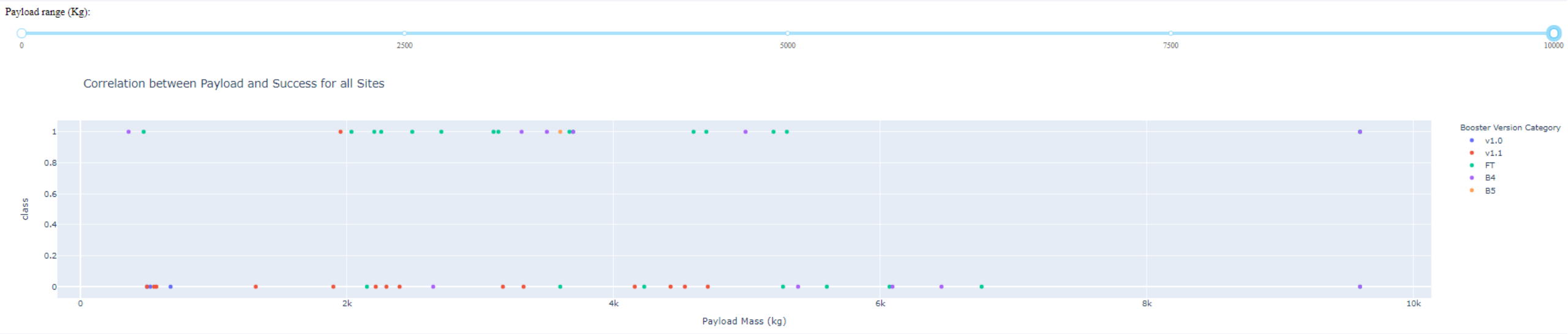
- Show the screenshot of the piechart for the launch site with highest launch success ratio



<Dashboard Screenshot 3>

- Replace <Dashboard screenshot 3> title with an appropriate title

- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider



Section 6

Predictive Analysis (Classification)

Classification Accuracy

Logistic Regression (2)

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

```
# old fashion ML, using params (C, penalty, solver) from

LR = LogisticRegression(C=0.01, penalty="l2", solver='lbf
yhat = LR.predict(X_test) # return prediction
yhat_prob = LR.predict_proba(X_test) # return prob

print("Logistic Train set Accuracy: ", metrics.accuracy_s
print("Logistic Test set Accuracy: ", metrics.accuracy_sc

print("Logistic Jaccard: ", jaccard_score(Y_test, yhat, a
print("Logistic F1 : ", f1_score(Y_test, yhat, average='w
print("Logistic Log Loss: ", log_loss(Y_test, yhat_prob))
print("score",LR.score(X_test,Y_test))

# score = test accuracy
```

```
Logistic Train set Accuracy:  0.8888888888888888
Logistic Test set Accuracy:  0.7222222222222222
Logistic Jaccard:  0.5208333333333333
Logistic F1 :  0.6868686868686869
Logistic Log Loss:  0.6429597753292977
score 0.7222222222222222
```

Support Vector Machine (2)

Calculate the accuracy on the test data using the m

```
# old fashion ML, using params from Grid Se

clf = SVC(C=1, gamma=0.03162277660168379, k
yhat = clf.predict(X_test)
print("SVM Train set Accuracy: ", metrics.a
print("SVM Test set Accuracy: ", metrics.ac

print("SVM Jaccard: ", jaccard_score(Y_test
print("SVM F1 : ", f1_score(Y_test, yhat, a
print("score",clf.score(X_test,Y_test))

# score = test accuracy
```

```
SVM Train set Accuracy:  0.9166666666666666
SVM Test set Accuracy:  0.7777777777777778
SVM Jaccard:  0.6071428571428572
SVM F1 :  0.7592592592592591
score 0.7777777777777778
```

Decision Tree (2)

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

```
# old fashion ML, using params from Grid Search

Ks = 20
mean_acc = np.zeros((Ks - 1))
std_acc = np.zeros((Ks - 1))
for n in range(1, Ks):
    drugTree = DecisionTreeClassifier(criterion="entropy", max_dep
    drugTree.fit(X_train, Y_train)
    predTree = drugTree.predict(X_test)
    mean_acc[n - 1] = metrics.accuracy_score(Y_test, predTree)
    std_acc[n - 1] = np.std(predTree == Y_test) / np.sqrt(yhat.sha
print(mean_acc)
print( "The best accuracy was with", mean_acc.max(), "with max_dep


drugTree = DecisionTreeClassifier(criterion="entropy", max_depth =
                                  min_samples_leaf = 2, min_samples_
drugTree.fit(X_train,Y_train)
predTree = drugTree.predict(X_test)
print("DecisionTree Train set Accuracy: ", metrics.accuracy_score(
print("DecisionTree Test set Accuracy: ", metrics.accuracy_score(Y_

print("DecisionTree Jaccard: ", jaccard_score(Y_test, predTree, av
print("DecisionTree F1 : ", f1_score(Y_test, predTree, average='we
print("score",drugTree.score(X_test,Y_test))

# score = test accuracy
```

```
[0.72222222 0.72222222 0.77777778 0.83333333 0.83333333 0.83333333
 0.77777778 0.83333333 0.83333333 0.83333333 0.83333333 0.83333333
 0.83333333 0.83333333 0.77777778 0.83333333 0.83333333 0.83333333
 0.83333333]
The best accuracy was with 0.8333333333333334 with max_depth= 4
DecisionTree Train set Accuracy:  0.9305555555555556
DecisionTree Test set Accuracy:  0.7222222222222222
DecisionTree Jaccard:  0.5208333333333333
DecisionTree F1 :  0.6868686868686869
score 0.7222222222222222
```

KNN (2)

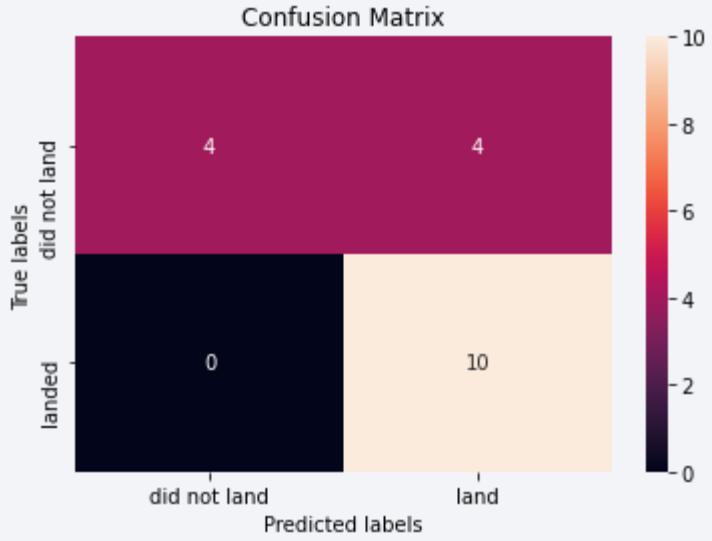
Calculate the accuracy of tree_cv on the test data using the method `score`:

```
# old fashion ML, using params from Grid Search

Ks = 15
mean_acc = np.zeros((Ks - 1))
std_acc = np.zeros((Ks - 1))
for n in range(1, Ks):
    neigh = KNeighborsClassifier(n_neighbors=n).fit(X_train, Y_trai
    yhat = neigh.predict(X_test)
    mean_acc[n - 1] = metrics.accuracy_score(Y_test, yhat)
    std_acc[n - 1] = np.std(yhat == Y_test) / np.sqrt(yhat.shape[0]
print(mean_acc)
print("The best accuracy was with", mean_acc.max(), "with k=", mean

k = 4
neigh = KNeighborsClassifier(n_neighbors = k, p = 1, algorithm = 'a
yhat = neigh.predict(X_test)
print("KNN Train set Accuracy: ", metrics.accuracy_score(Y_train, r
print("KNN Test set Accuracy: ", metrics.accuracy_score(Y_test, yha
print("KNN Jaccard: ", jaccard_score(Y_test, yhat,average="macro"))
print("KNN F1: ", f1_score(Y_test, yhat, average="weighted"))
print("score",neigh.score(X_test,Y_test))

# score = test accuracy
```

```
[0.61111111 0.66666667 0.66666667 0.72222222 0.66666667 0.66666667
 0.55555556 0.55555556 0.55555556 0.55555556 0.55555556 0.55555556
 0.55555556 0.55555556]
The best accuracy was with 0.7222222222222222 with k= 4
KNN Train set Accuracy:  0.8888888888888888
KNN Test set Accuracy:  0.7777777777777778
KNN Jaccard:  0.6071428571428572
KNN F1:  0.7592592592592591
score 0.7777777777777778
```

SVM & KNN both have highest accuracy – 77.8%

Confusion Matrix



- confusion matrix of the best performing model – KNN, highest test accuracy (same as SVM) plus having closer train & test accuracy

Conclusions

- KNN is the most appropriate prediction method for launch result
- success rate since 2013 kept increasing till 2020

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!