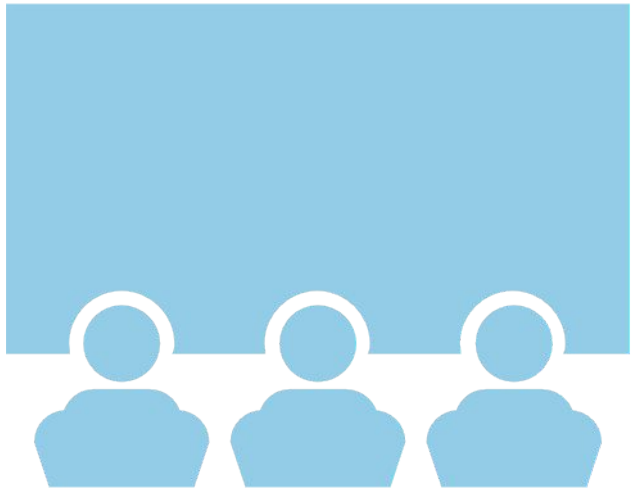
Data Science Capstone project

Nagasai Biginepalli

08/09/2021

Outline



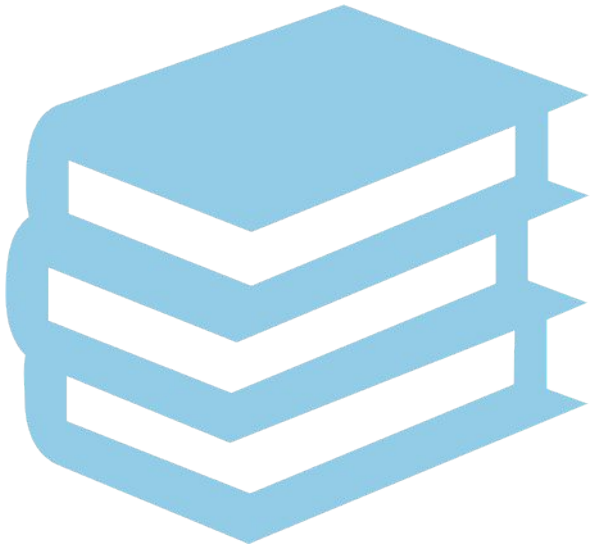
- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

Space X 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 Space X can reuse the first stage.

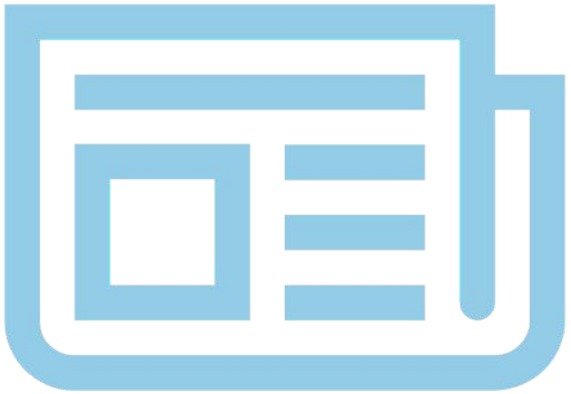
- In this project, we have created four classification models of Logistic regression, Decision Tree, SVM and KNN to predict if the first stage will land.
- To explore and analyze data, we have taken the following methods: cleaning and feature engineering, formatting, normalization and visualization data. To build and evaluate the models, we used the steps such as split data into training/test data, cross validation, tuning parameters and calculate the score of models.
- The accuracy of four models that we receive is same 83.33%. It is pretty good.
- We gained also some insights such as:
  - Launch success yearly trend increases when the number of flights increases.
  - KSC LC-39A site has the largest successful launches and also has the highest launch success rate.
  - Payload range from 2000 to 4000kg has the highest launch success rate, and above 5500kg has lowest launch success rate.
  - F9 Booster version of FT has the highest launch success rate.
  - ES-L1, GEO, HEO, SSO are the orbit types which have the success rate of 100%. On the contrary, SO orbit has not any successfully launches
- The results that we get are very positive, but in fact we have analyzed on a small data set. So in the future when the data of launches is more, we will have a more complete analysis.

Introduction



- Space X 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 Space X 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 space X for a rocket launch.
- In this project, we will create a machine learning pipeline to predict if the first stage will land.

Methodology



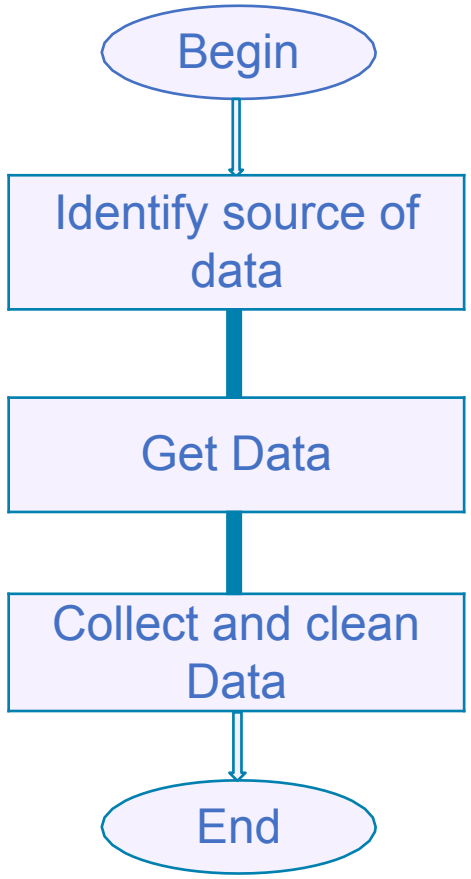
- Data collection methodology:
  - Describe how data were collected
- Perform data wrangling
  - Describe how data were processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Methodology

Data collection

Data collection includes three main steps:

1. **Identify source of data:**
   - Data can come from website, a file, a API, a database,… etc.
   - Identify source of data help us how to get data
2. **Get data:** we can data by:
   - Web scraping, Rest API
   - Read file, or using SQL query
3. **Collect and clean data**:
   - Collect the relevant information to analyze
   - Clean raw data

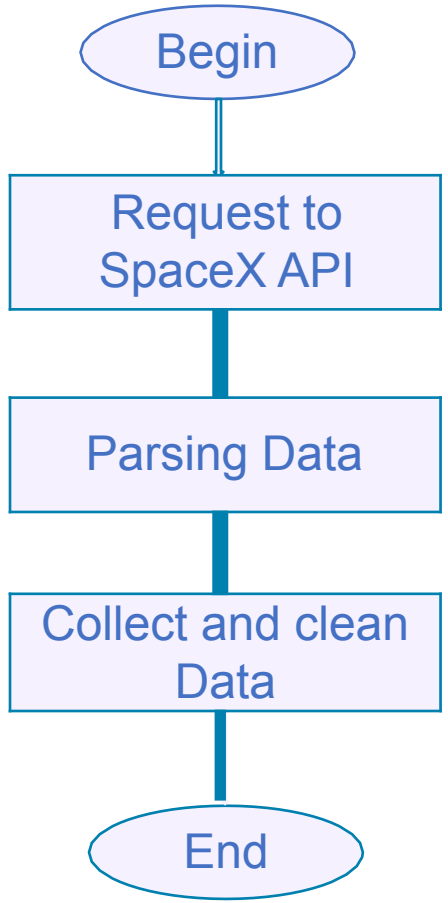


Data collection – SpaceX API

- To collect data from SpaceX API, we need following steps:

1. **Request to SpaceX API:**
   - Using HTTP requests to get data from SpaceX API
2. **Parsing data:**
   - Parse the SpaceX launch data using the GET request to get the json result
   - Use json_normalize method to convert the json result into a dataframe
3. **Collect and clean data:**
   - Get information about the launches using the IDs given for each launch. Specifically we will collect relevant columns to analyze
   - Clean raw data

- GitHub URL of notebook: https://github.com/Nagasai524/Coursera/blob/master/Data%20Collection.ipynb

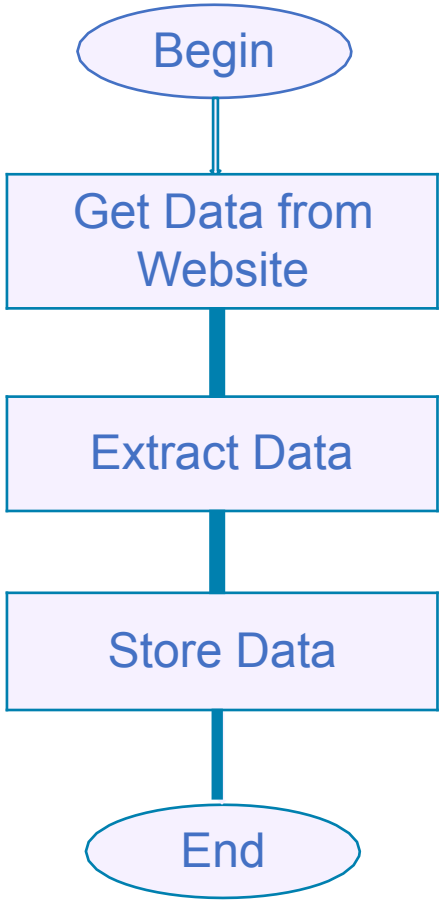


Data collection – Web scraping

- Data collection by web scraping contain 3 steps:

1. **Get data from website**:
   - Using HTTP Get to request website
   - Using BeautifulSoup and HTTP Reponse to get data
2. **Extract data**: parsing HTML table header to collect data
3. **Store data**: load data into dataframe or store it into database

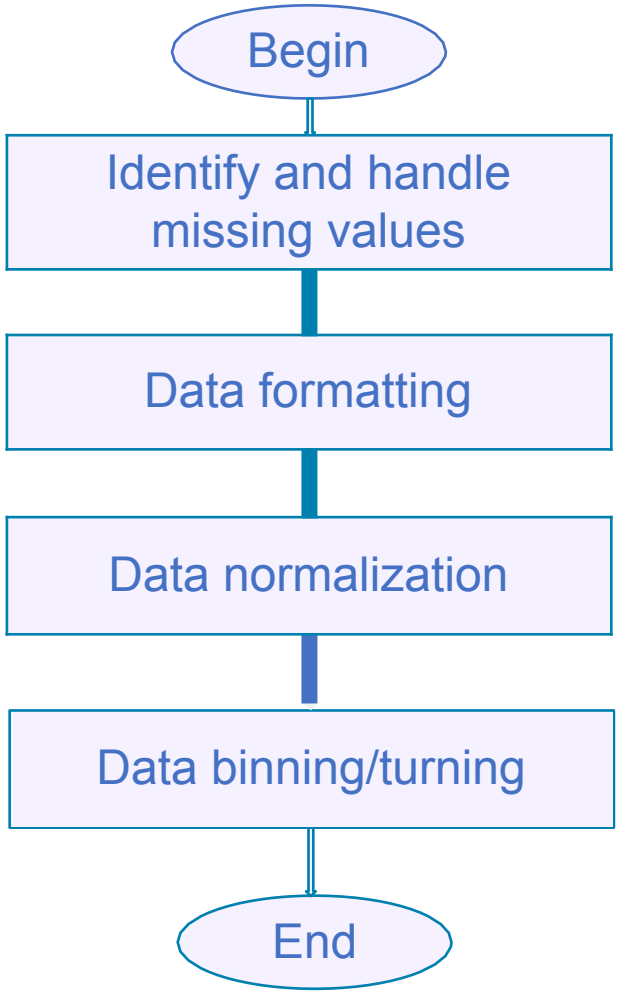
- GitHub URL of notebook: https://github.com/Nagasai524/Coursera/blob/master/WebScraping.ipynb



Data wrangling

Data wrangling includes the following steps:

1. **Identify and handle missing values:**
2. **Data formatting:** Standardize the values into the same format, or unit, or convention.
3. **Data normalization:** Bring all data into a similar range for more useful comparison
4. **Data binning/turning**:
   - Binning creates bigger categories from a set of numerical values
   - Turning Categorical values to numeric variables to make statistical modeling easier



EDA with data visualization

- Data visualization is useful to explore data and feature engineering. In this context, we used following charts:
  - Scatter plot
  - Bar chart
  - Line chart
- These charts help us:
  - To visualize the relationship between variables
  - To obtain some preliminary insights about how each important variable would affect the success rate, we will select the features that will be used in success prediction in the future module.

- GitHub URL of notebook: https://github.com/Nagasai524/Coursera/blob/master/EDA_with_Visualization.ipynb

EDA with SQL

- To understand the Spacex DataSet, we performed some samples of SQL queries such as:
  - Select
  - Sum()
  - Avg()
  - Min()
  - Max()
  - Count()
  - Sub query

- GitHub URL of notebook: https://github.com/Nagasai524/Coursera/blob/master/EDA%20with%20SQL.ipynb

Build an interactive map with Folium

- To build an interactive map with Folium, we created and added some map objects such as Marker, MarkerCluster, Circle, Line.
- These objects used to:
  - Mark all launch sites on map
  - Mark the success/failed launches for each site on the map
  - Calculate the distances between a launch site to its proximities

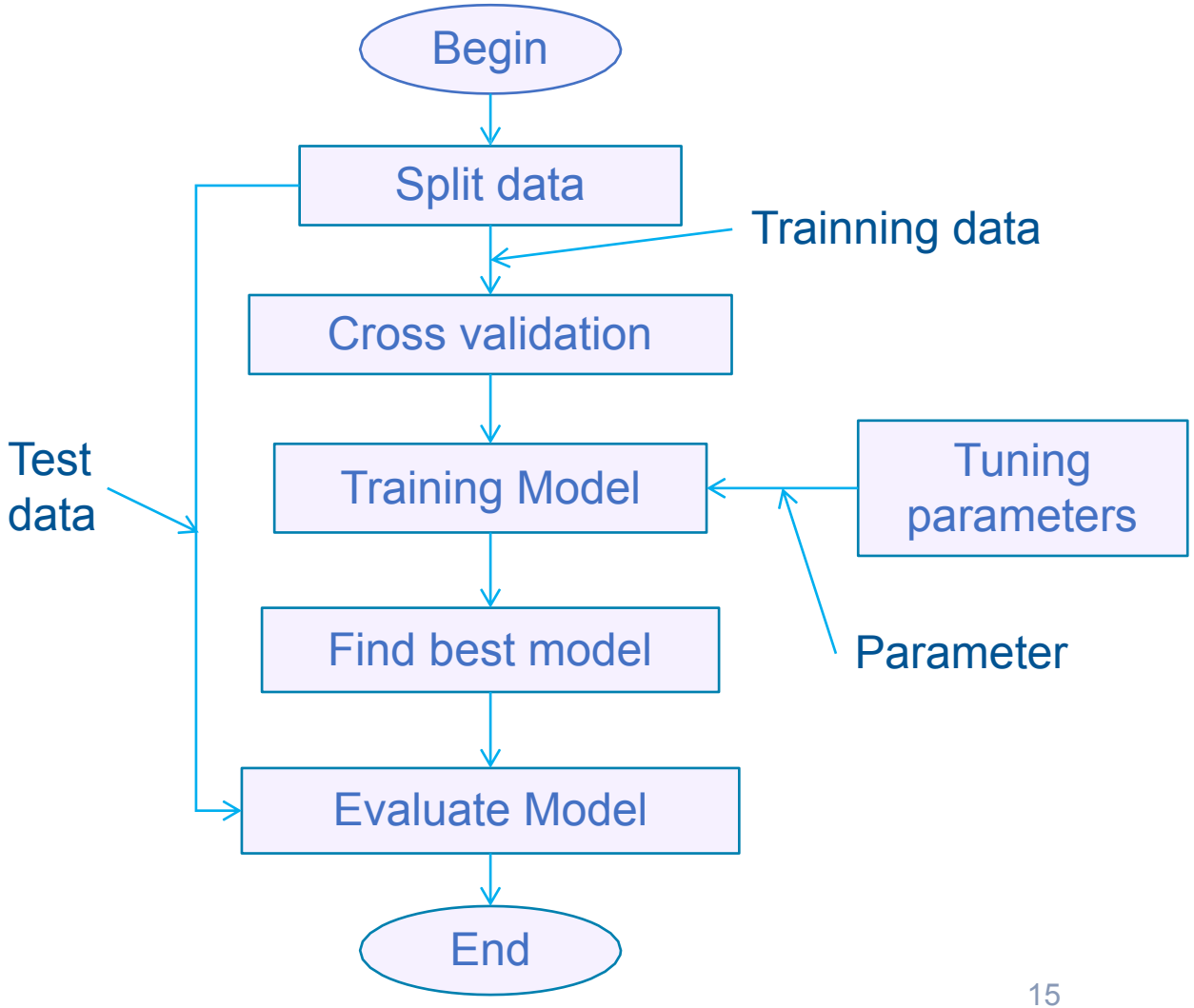
- GitHub URL of notebook: https://github.com/Nagasai524/Coursera/blob/master/Interactive_Visual_data_analytices.ipynb

Build a Dashboard with Plotly Dash

- The dashboard contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart.
- By using these objects, we can obtain some insights to answer the following questions:
  1. Which site has the largest successful launches ?
  2. Which site has the highest launch success rate ?
  3. Which payload range(s) has the highest launch success rate ?
  4. Which payload range(s) has lowest launch success rate ?
  5. Which F9 Booster version has the highest launch success rate ?
- GitHub URL of file code: https://github.com/longevite8/Data-Science/blob/ca37de212bbf659bb9972d1b47652a90485dcb2d/spacex_dash_app.py

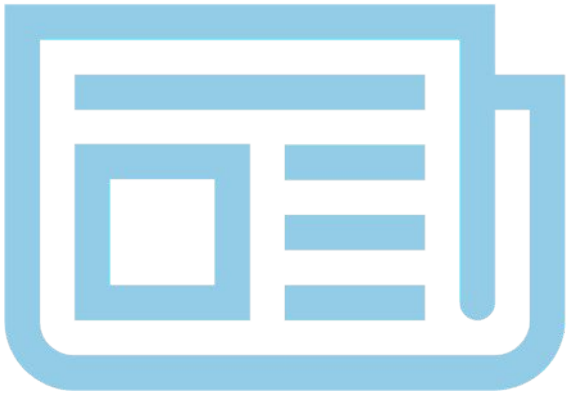
Predictive analysis (Classification)

- After cleaning data, the process of predictive analysis includes the steps such as the flowchart here
- We apply this process for all four classification models Logistic Regression, Decision Tree, SVM and KNN.



- GitHub URL of notobook: https://github.com/Nagasai524/Coursera/blob/master/Machine%20Learning%20Prediction%20Lab.ipynb

Results

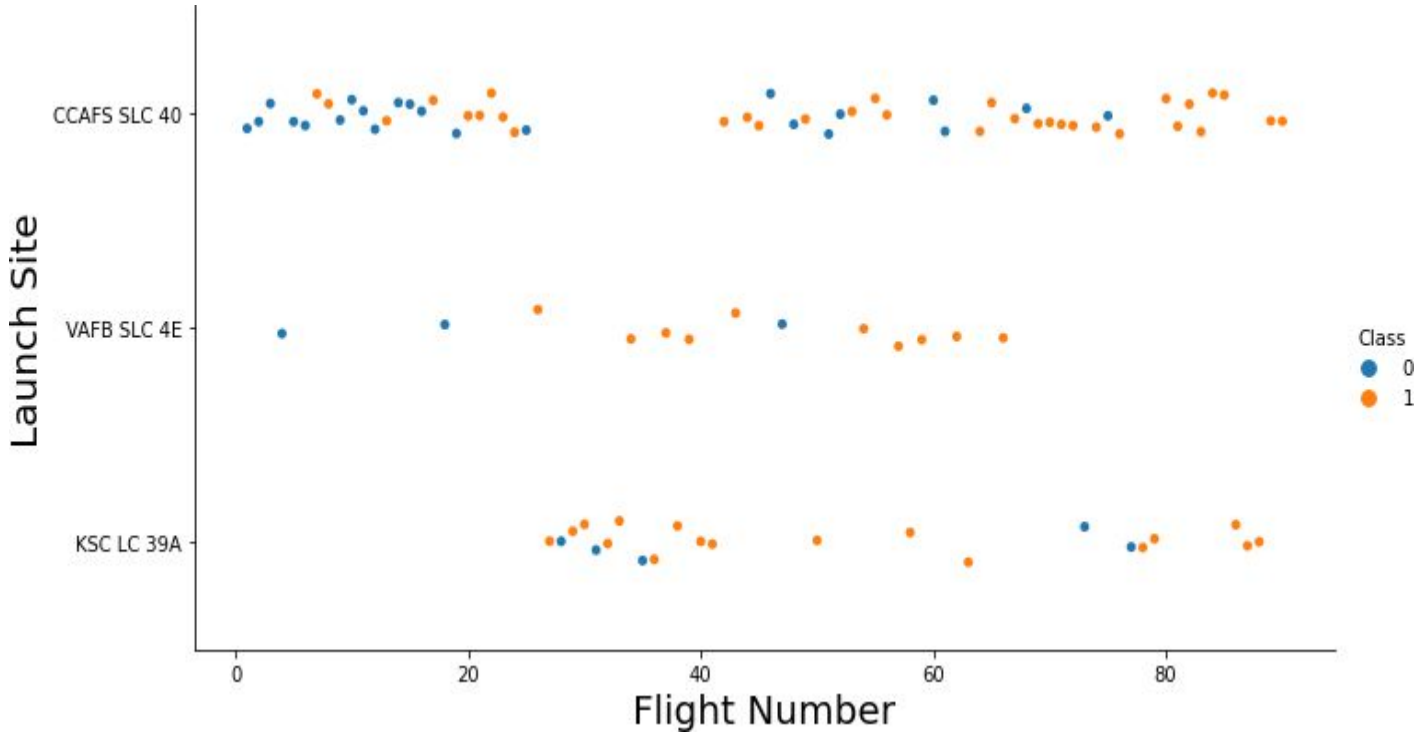


- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

EDA with Visualization

Flight Number vs. Launch Site

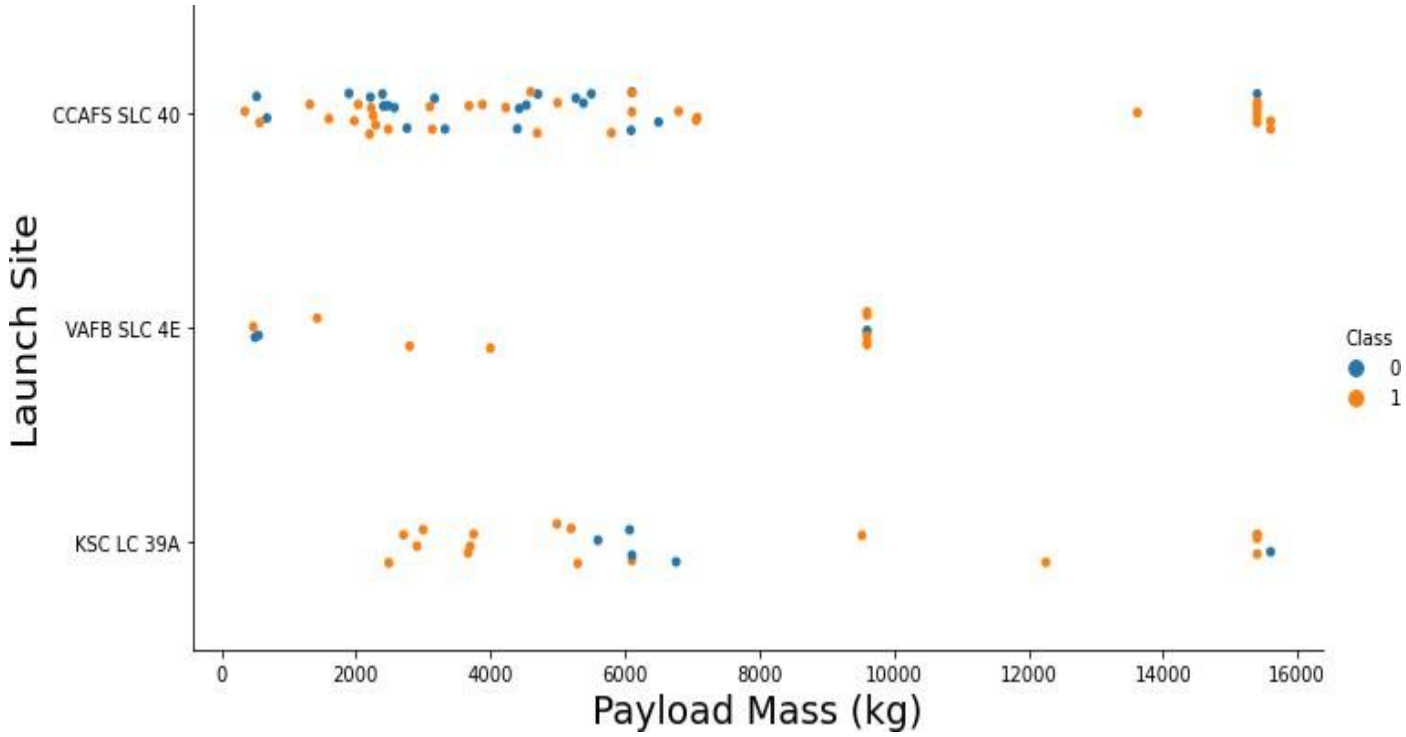
We can see that: when the number of flights increases, so the success rate increases too.



Payload vs. Launch Site

The results show us that:

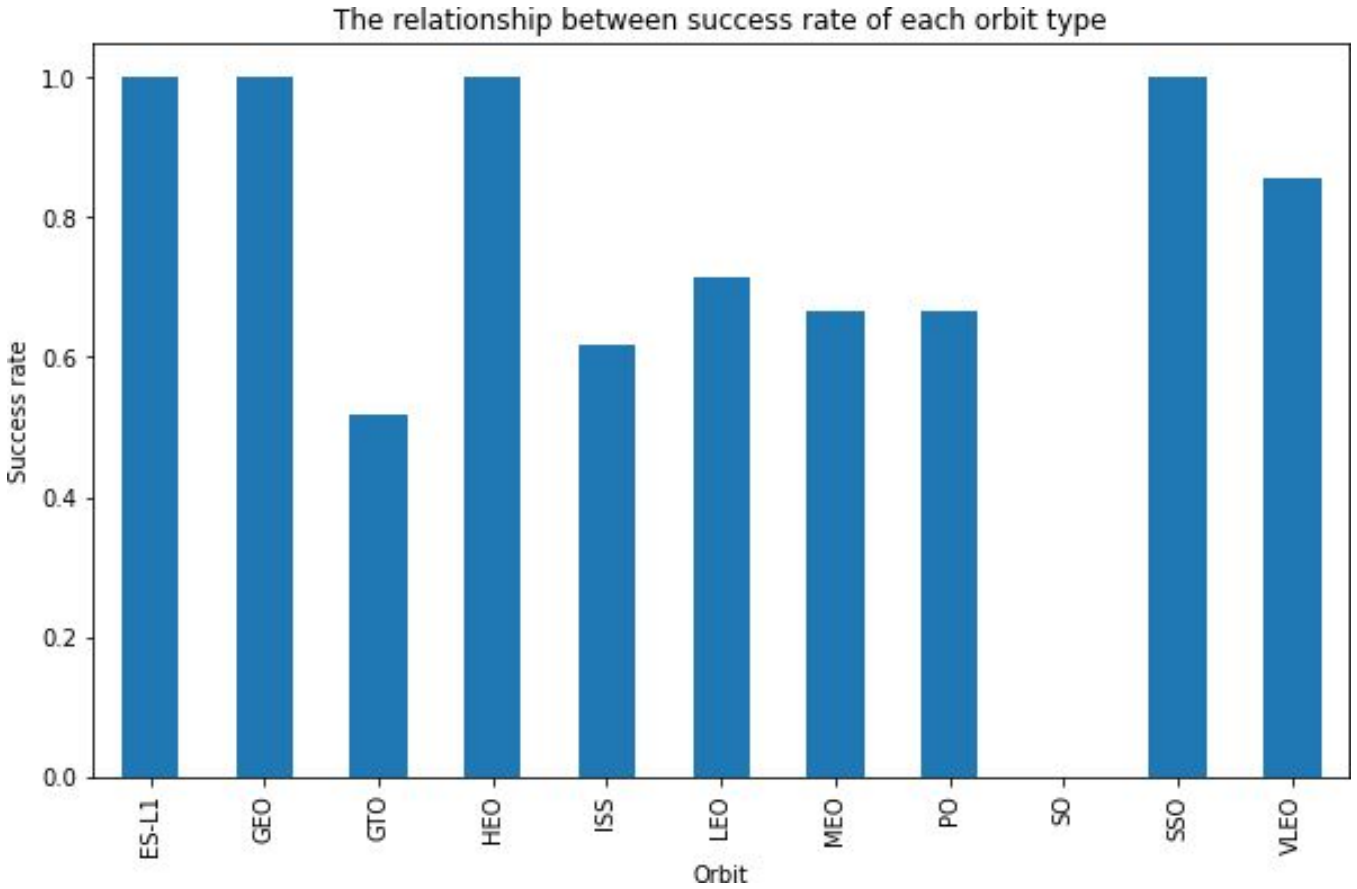
- When the payload mass is greater than 7000kg, the success rate is high
- When the payload mass is less than 5500kg, the success rate is 100% at the site KSC LC-39A



Success rate vs. Orbit type

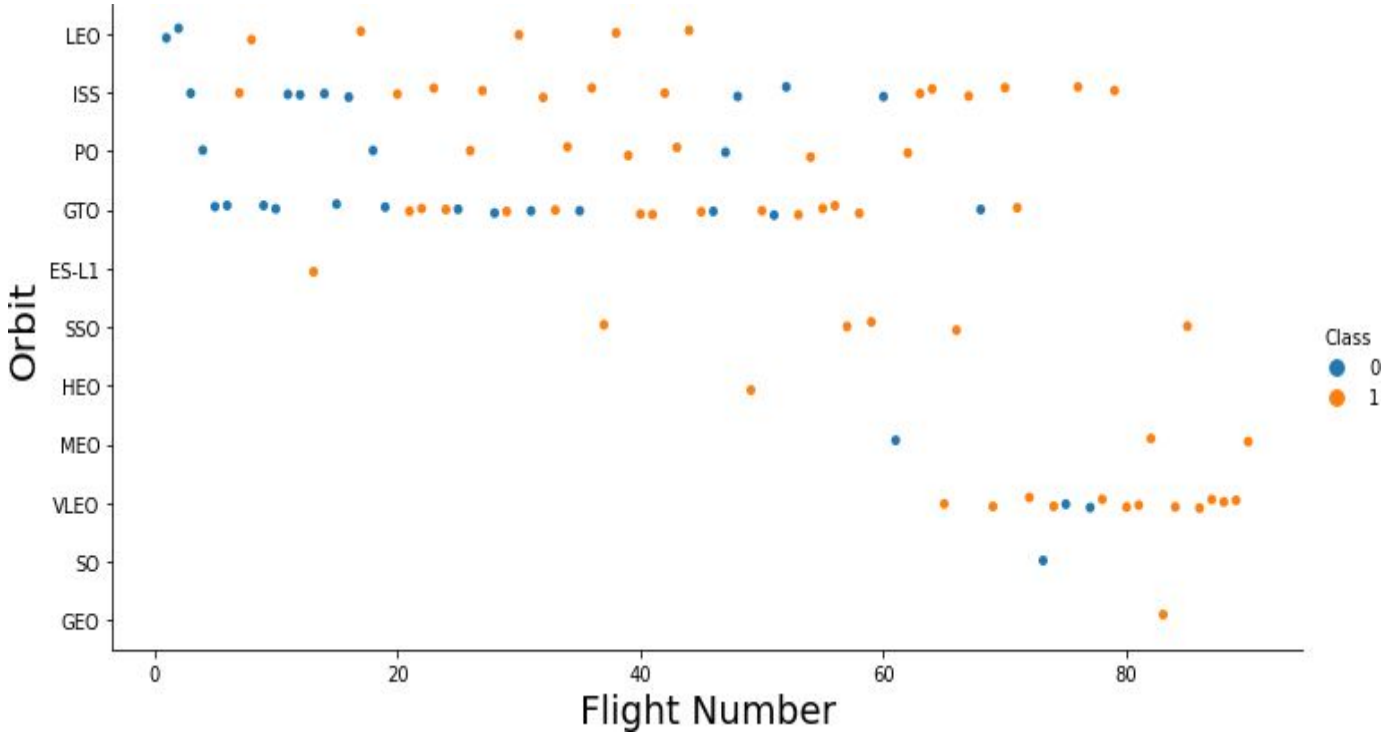
The results show that:

- ES-L1, GEO, HEO, SSO are the orbit types which have the success rate of 100%
- SO orbit has not any successfully launches



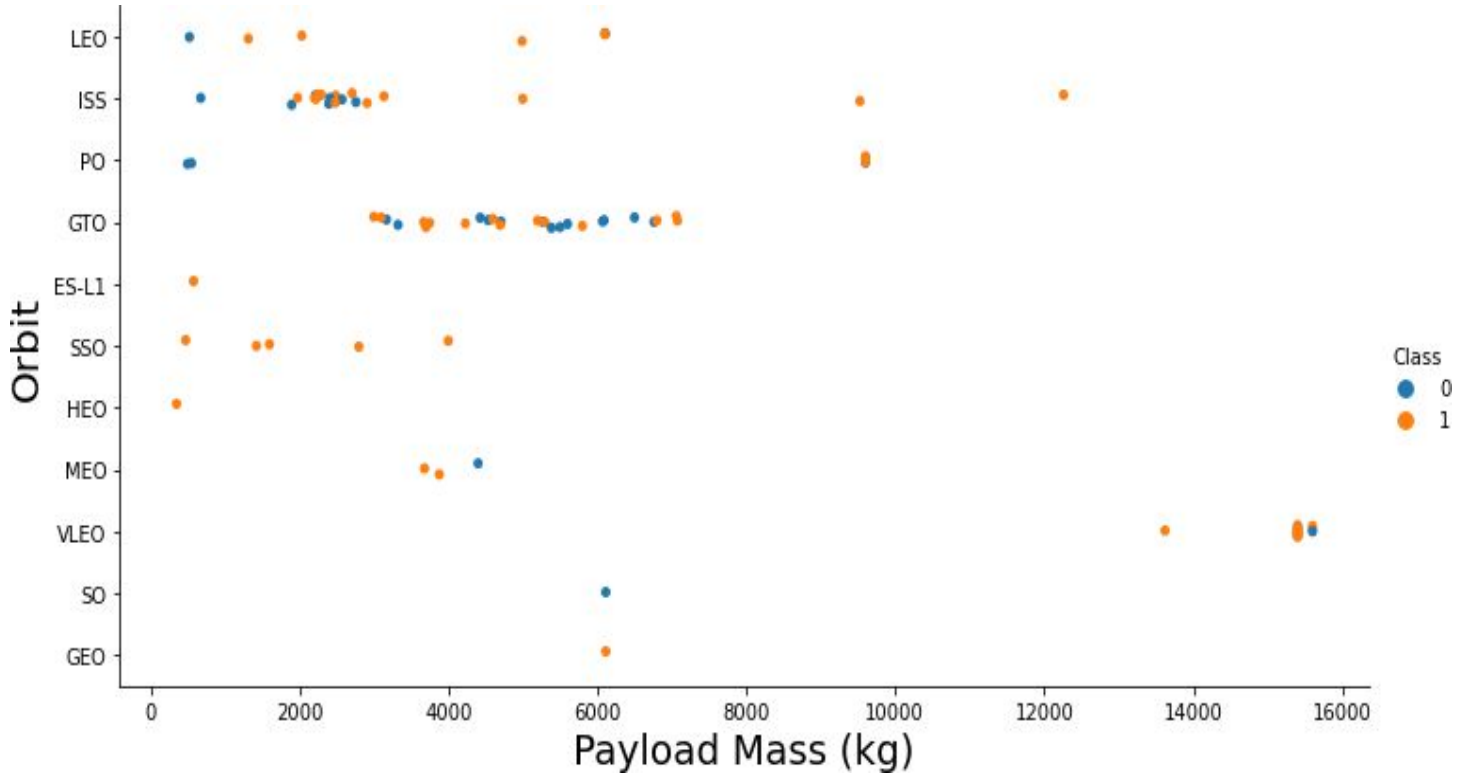
Flight Number vs. Orbit type

When flight number increases, the success rate increases too.



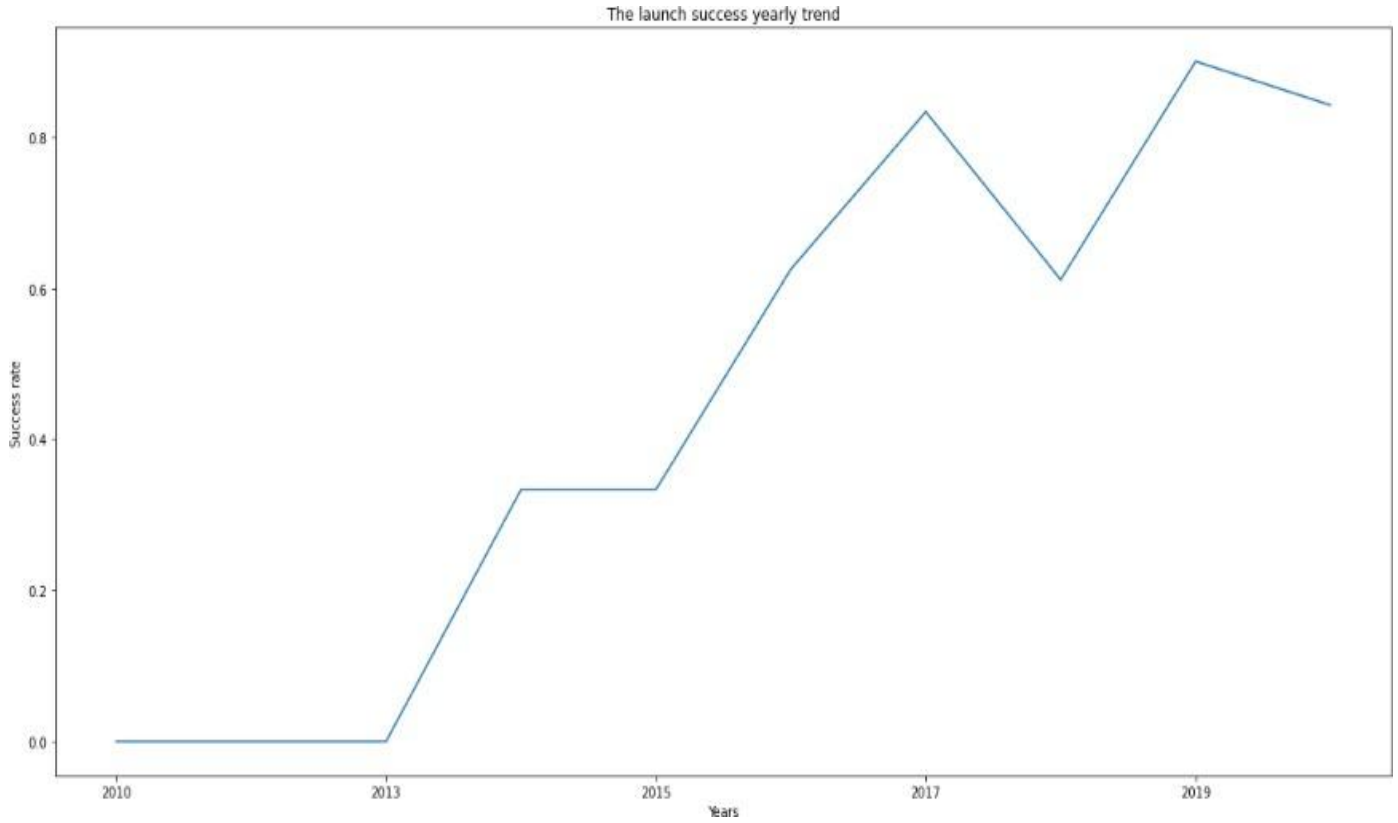
Payload vs. Orbit type

When the payload mass is greater than 7000kg, the success rate is high



Launch success yearly trend

We can observe that the success rate since 2013 kept increasing till 2020



EDA with SQL

All launch site names

- Find the names of the unique launch sites:

```
%sql select distinct Launch_Site from SPACEXDATASET
```

```
 * ibm_db_sa://nqf22321:***@0c77d6f2-5da9-48a9-81f8-86b520
Done.
```

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

- Need to use ‘distinct’ in the query for getting the unique launch sites.

Launch site names begin with `CCA`

- Find all launch sites begin with `CCA`

```
%sql select distinct Launch_Site from SPACEXDATASET where Launch_Site like 'CCA%'
```

```
 * ibm_db_sa://nqf22321:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90108kqb1od8lcg.datab
Done.
```

launch_site
CCAFS LC-40
CCAFS SLC-40

- By using ‘distinct’ and ‘%’ in this case to get the result.

Total payload mass

- Calculate the total payload carried by boosters from NASA

```
%sql select SUM(PAYLOAD_MASS__KG_) from SPACEXDATASET where CUSTOMER LIKE '%NASA%'
```

```
 * ibm_db_sa://nqf22321:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90108kqb1od8lcg.data
Done.
```

1
107010

- In this query, we use ‘sum()’ function and ‘like’operator.

Average payload mass by F9 v1.1

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

```
%sql select AVG(PAYLOAD_MASS__KG_) from SPACEXDATASET where Booster_Version = 'F9 v1.1'
```

```
 * ibm_db_sa://nqf22321:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90108kqb1od8lcg.databases.ap
Done.
```

1
2928

- Using Avg() function to get the average payload mass

First successful ground landing date

- Find the date when the first successful landing outcome in ground pad

```
%sql select MIN(Date) from SPACEXDATASET where Landing__Outcome = 'Success (ground pad)'
```

```
 * ibm_db_sa://nqf22321:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90108kqb1od8lcg.databases.app
Done.
```

1
2015-12-22

- With Min(Date) function, we can find the first date when landing outcome in ground pad is
  successful

Successful drone ship landing with payload between 4000 and 6000

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

```
%sql select distinct Booster_Version from SPACEXDATASET where (Landing__Outcome = 'Success (drone ship)') and (PAYLOAD_MASS__KG_ > 4000 and
```

```
 * ibm_db_sa://nqf22321:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

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

- Full query: Select distinct Booster_Version from SPACEXDATASET where (Landing Outcome = 'Success (drone ship)') and (PAYLOAD_MASS KG_ > 4000 and PAYLOAD_MASS KG_ < 6000)

Total number of successful and failure mission outcomes

- Calculate the total number of successful and failure mission outcomes

```
%sql select count(*) as "Total number", Mission_Outcome from SPACEXDATASET where (Mission_Outcome LIKE '%Success%') or (Mission_Outcome LIKE
```

```
 * ibm_db_sa://nqf22321:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90108kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

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

- Full query: Select count(*) as "Total number", Mission_Outcome from SPACEXDATASET where (Mission_Outcome LIKE '%Success%') or (Mission_Outcome LIKE '%Failure%') group by Mission_Outcome order by "Total number" desc, Mission_Outcome desc

Boosters carried maximum payload

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

```
%sql select distinct Booster_Version from SPACEXDATASET where PAYLOAD_MASS__KG_ = (select MAX(PAYLOAD_MASS__KG_) from SPACEXDATASET )
```

```
 * ibm_db_sa://nqf22321:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

booster_version
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

2015 launch records

- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015

```
%sql SELECT MONTHNAME(Date) as "Month Name", Landing__Outcome, Booster_Version, Launch_Site FROM SPACEXDATASET WHERE (YEAR(Date) = '2015') a
```

```
 * ibm_db_sa://nqf22321:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90108kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

Month Name	landing_outcome	booster_version	launch_site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- Full query: SELECT MONTHNAME(Date) as "Month Name", Landing Outcome, Booster_Version, Launch_Site FROM SPACEXDATASET WHERE (YEAR(Date) = '2015') and (Landing Outcome LIKE '%Failure%') ORDER BY Month(Date) ASC

Rank success count between 2010-06-04 and 2017-03-20

- Rank the count ofsuccessful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

```
%sql SELECT COUNT(*) AS "COUNT", Landing__Outcome FROM SPACEXDATASET WHERE (Date BETWEEN '2010-06-04' AND '2017-03-20') AND (Landing__Outcom
```

```
 * ibm_db_sa://nqf22321:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90108kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

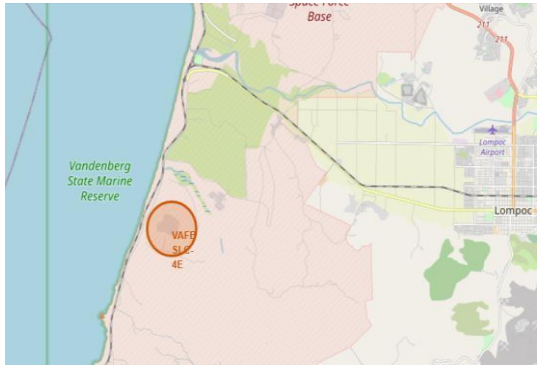
COUNT	landing_outcome
5	Success (drone ship)
3	Success (ground pad)

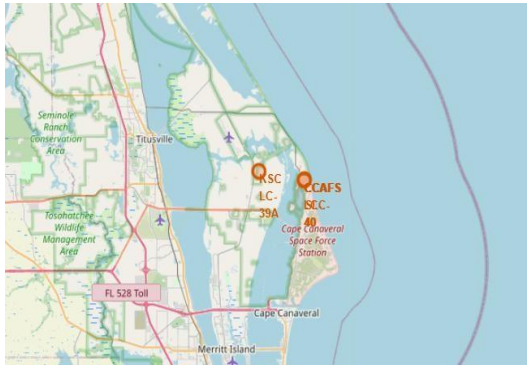
- Full query: SELECT COUNT(*) AS "COUNT", Landing Outcome FROM SPACEXDATASET WHERE (Date BETWEEN '2010-06-04' AND '2017-03-20') AND (Landing Outcome LIKE '%Success%') GROUP BY Landing__Outcome ORDER BY "COUNT" DESC

Interactive map with Folium

Location of all launch sites

- All launch sites:
  - Near the equator
  - Near the beach
  - Far from residential area





The success/failed launches for each site

	(1)	(2)	(3)	(4)
KSC LC-LC-39A	13	10	3	**0.769**
CCAFS SLC-40	7	3	4	**0.429**
VAFB SLC-4E	10	4	6	**0.4**
CCAFS LC-40	26	7	19	**0.269**

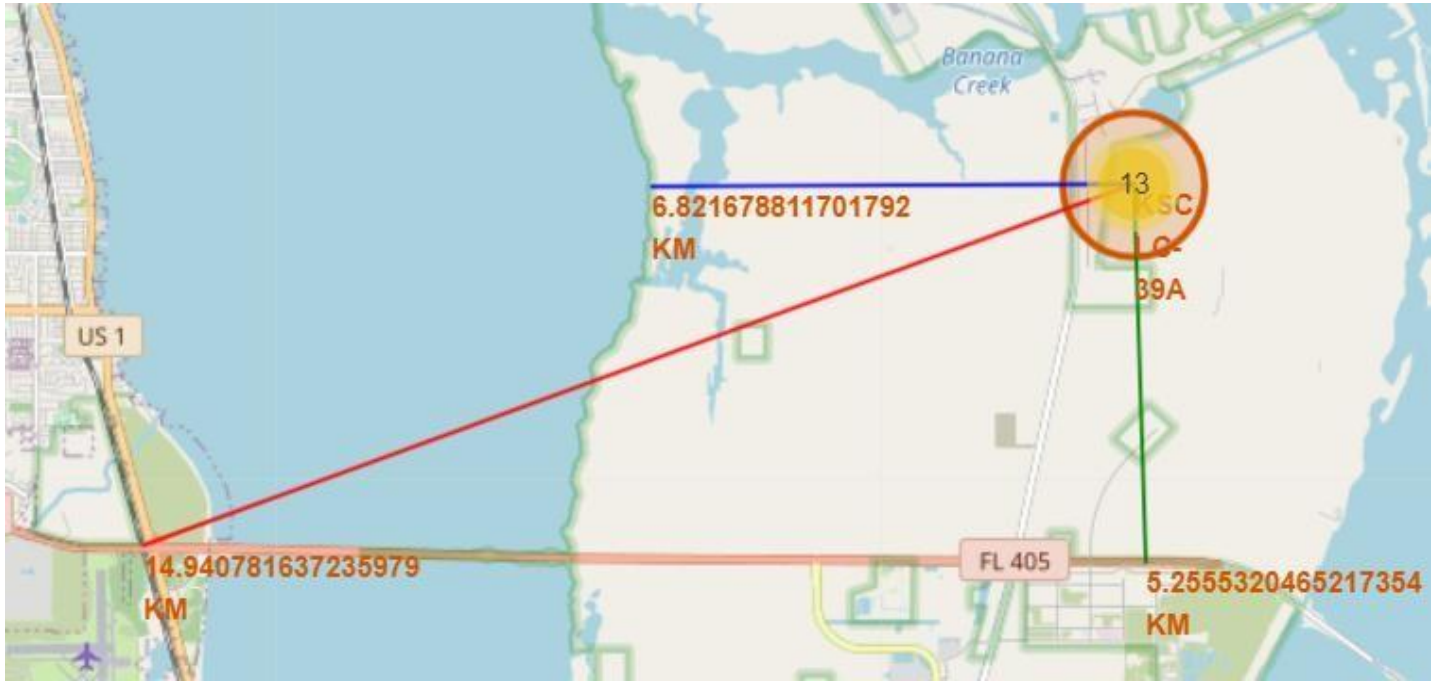
(1) Total launches
(2) Successful launches
(3) Failed launches
(4) % success (=(2)/(1))

=> KSC LC-39A is the site which have the highest success rate

Distance from site to its proximities

The site launch is:

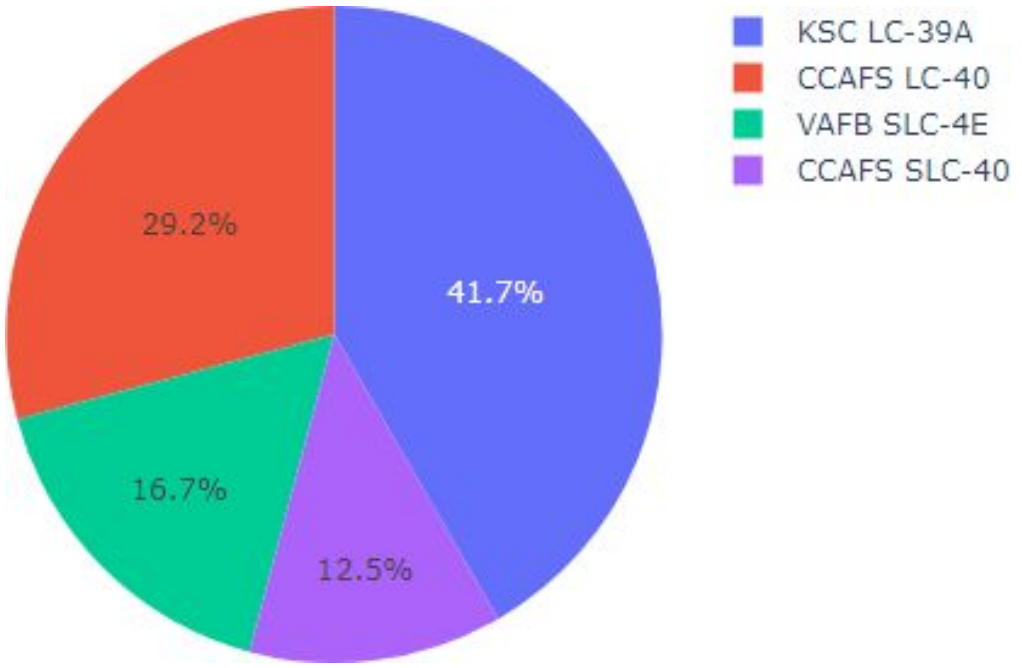
- near the beach
- far from residence area



Build a Dashboard with Plotly Dash

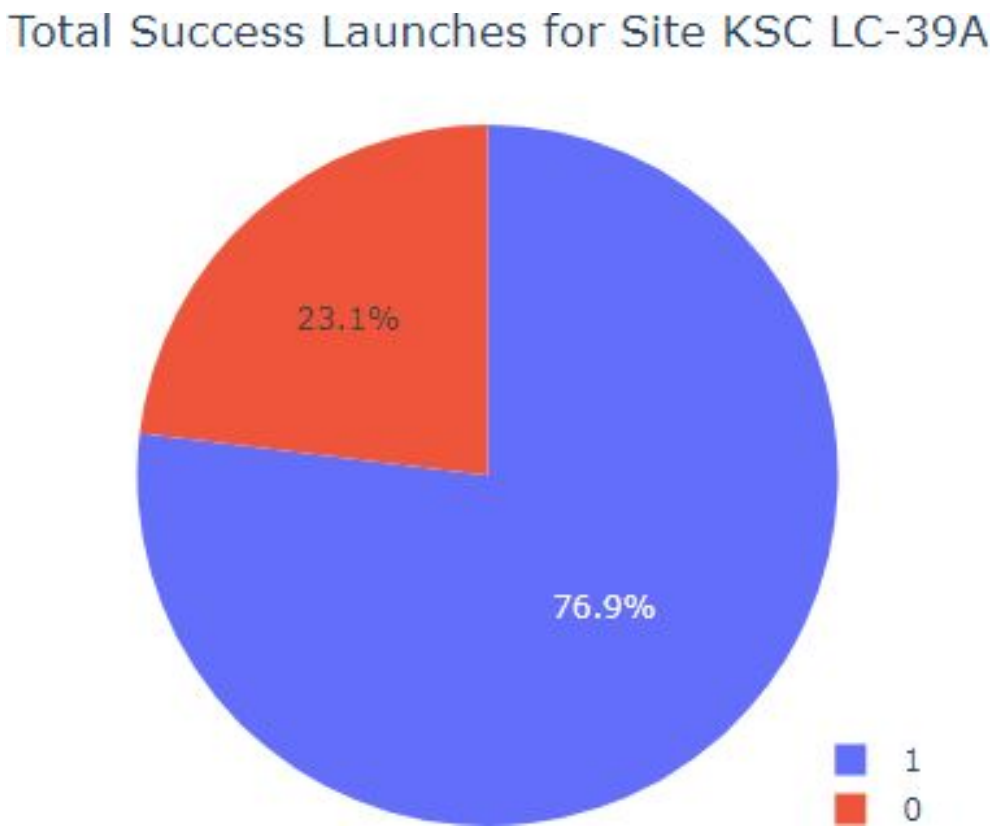
Total success launches by all sites

- For all sites:
  - KSC LC-39A site has the most number of successful launches, whereas CCAFS SLC-40 has the least number of successful launches



The launch site with highest launch success ratio

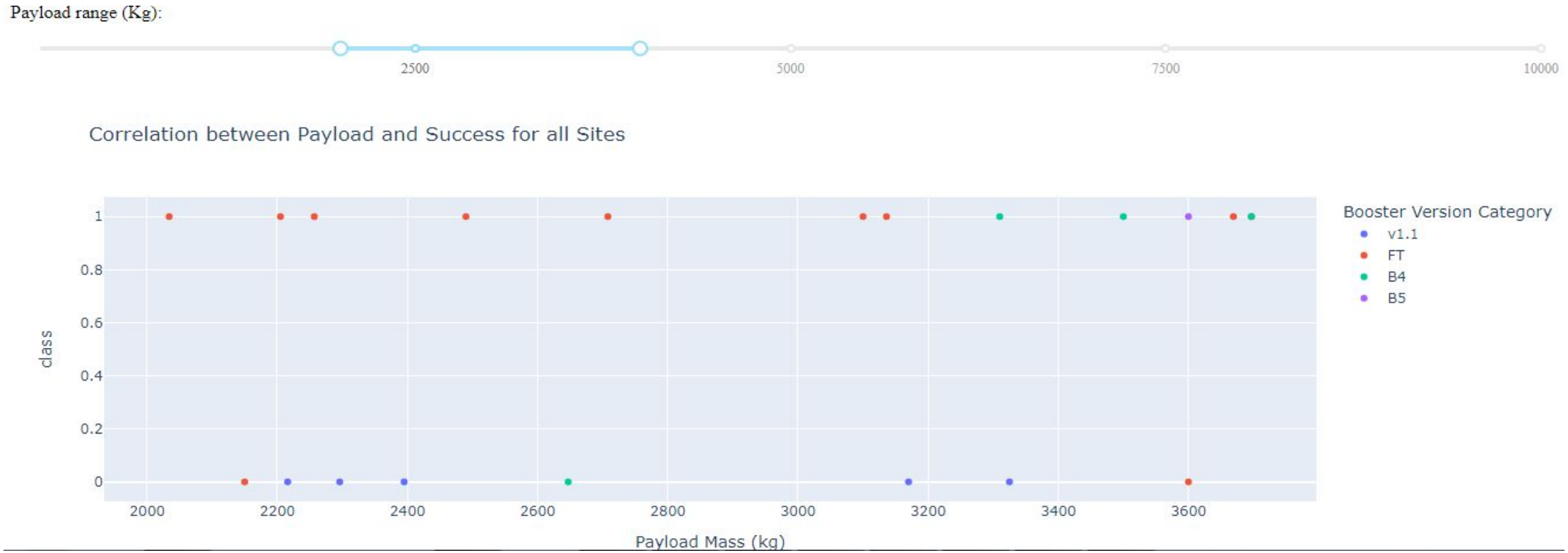
- KSC LC-39A is the site with highest launch success ratio, it gain 76.9%



Payload vs Launch Outcome scatter plot for all sites

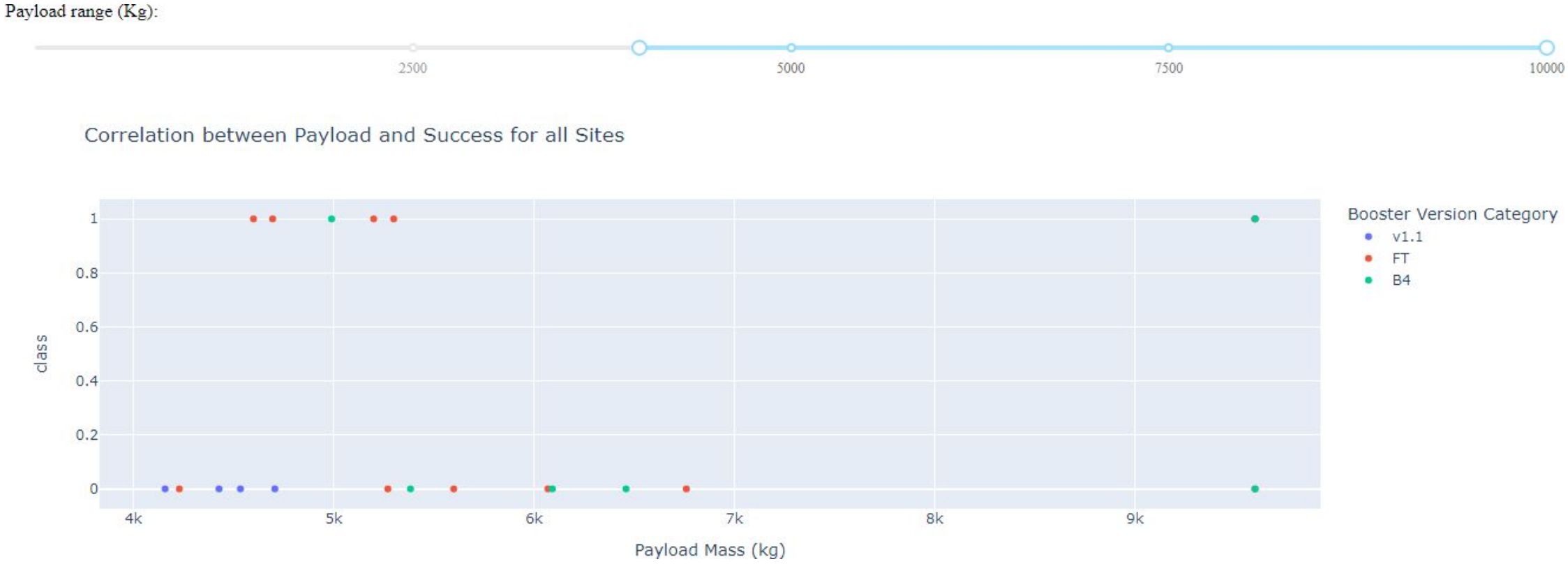
We find that: when the payload mass is bigger, the success rate is smaller.

Payload vs Launch Outcome scatter plot with payload range from 2000 to 4000kg



In payload range from 2000 to 4000kg, we have the most success rate.

Payload vs Launch Outcome scatter plot with payload range above 4000kg

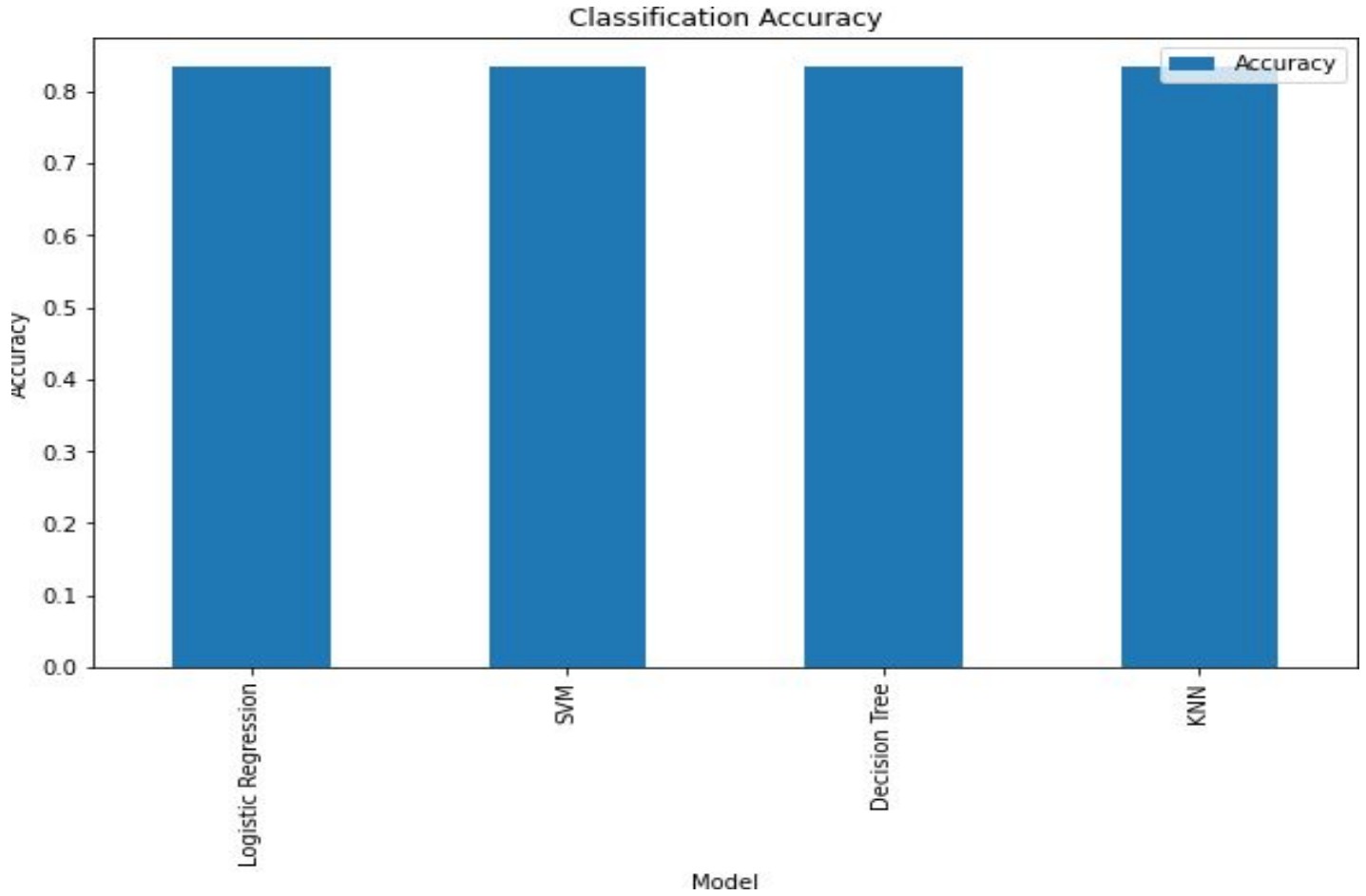


In payload range above 4000kg, the successfully ratio is small. Especially if above 5500kg then there is only one launch of success.

Predictive analysis (Classification)

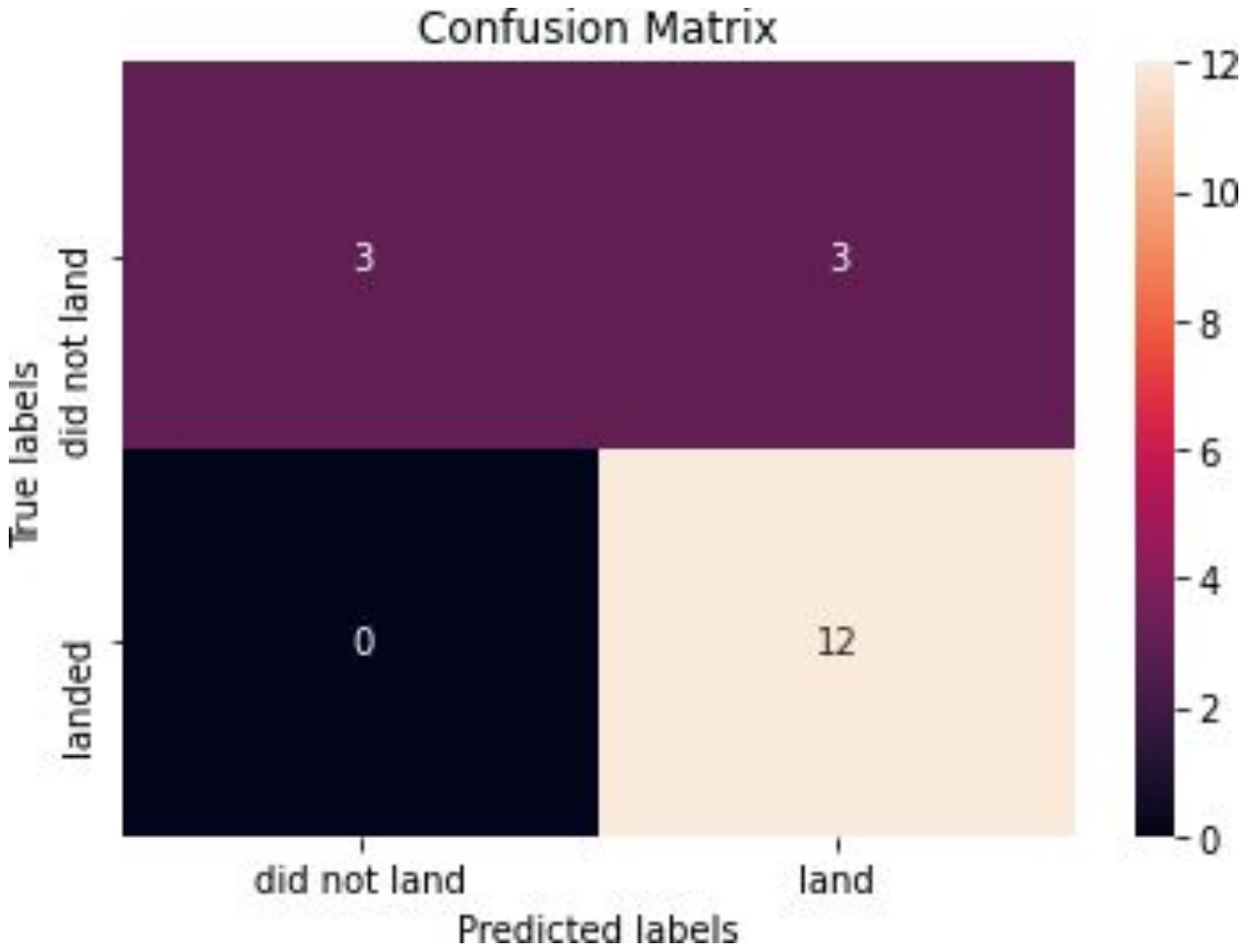
Classification Accuracy

- All models have the same accuracy of 0.833
- This result may appear when the data set which is used to build and evaluate the model is too small.



Confusion Matrix

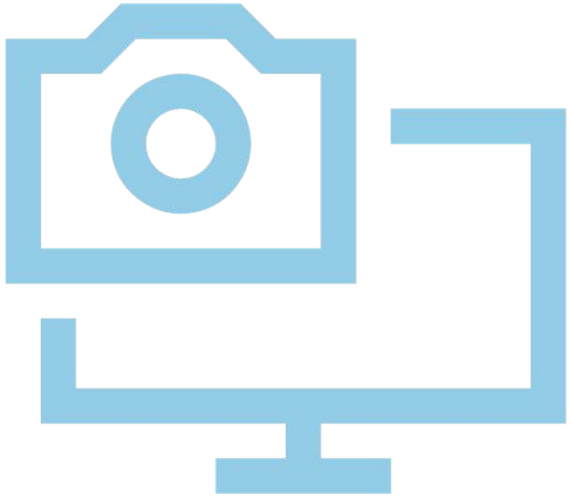
- Since all models have the same accuracy, they also have the same Confusion Matrix
- This result is pretty good when number of true positive is high and true negative is low



CONCLUSION

- In this project, we have created four classification models of Logistic regression, Decision Tree, SVM and KNN to predict if the first stage will land. And the accuracy of them is same 83.33%. This result is pretty good.
- We gained also some insights such as:
  - Launch success yearly trend increases when the number of flights increases.
  - KSC LC-39A site has the largest successful launches and also has the highest launch success rate.
  - Payload range from 2000 to 4000kg has the highest launch success rate, and above 5500kg has lowest launch success rate.
  - F9 Booster version of FT has the highest launch success rate.
  - ES-L1, GEO, HEO, SSO are the orbit types which have the success rate of 100%. On the contrary, SO orbit has not any successfully launches
- The results that we get are very positive, but in fact we have analyzed on a small data set. So in the future when the data of launches is more, we will have a more complete

APPENDIX – Data Wrangling

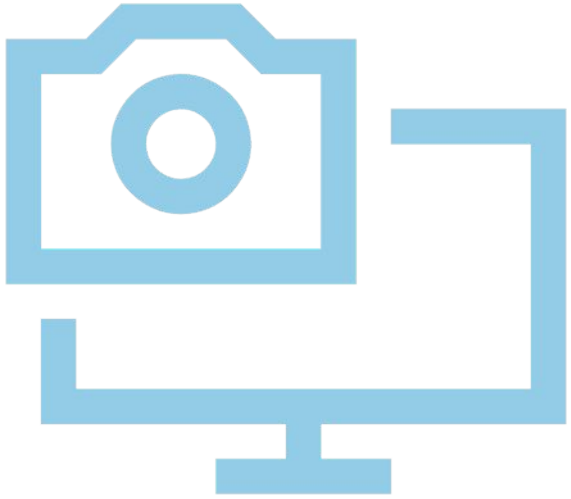


Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

```
# Calculate the mean value of PayloadMass column
mean_payload = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].fillna(mean_payload, inplace=True)
```

APPENDIX - Preparing Data Feature Engineering



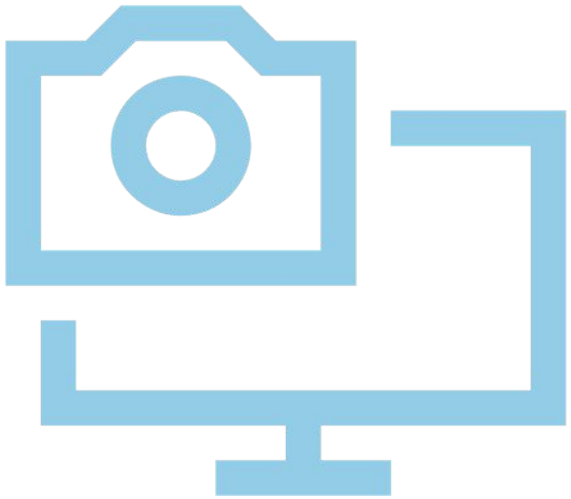
Features Engineering

By now, you should obtain some preliminary insights about how each important variable would affect the success rate, we will select the features that will be used in success prediction in the future module.

```
features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Legs', 'LandingPad
features.tail()
```

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
85	86	15400.0	VLEO	KSC LC 39A	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	2	B1060
86	87	15400.0	VLEO	KSC LC 39A	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	2	B1058
87	88	15400.0	VLEO	KSC LC 39A	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	5	B1051
88	89	15400.0	VLEO	CCAFS SLC 40	3	True	True	True	5e9e3033383ecbb9e534e7cc	5.0	2	B1060
89	90	3681.0	MEO	CCAFS SLC 40	1	True	False	True	5e9e3032383ecb6bb234e7ca	5.0	0	B1062

APPENDIX – Dummy variables to categorical columns



Create dummy variables to categorical columns

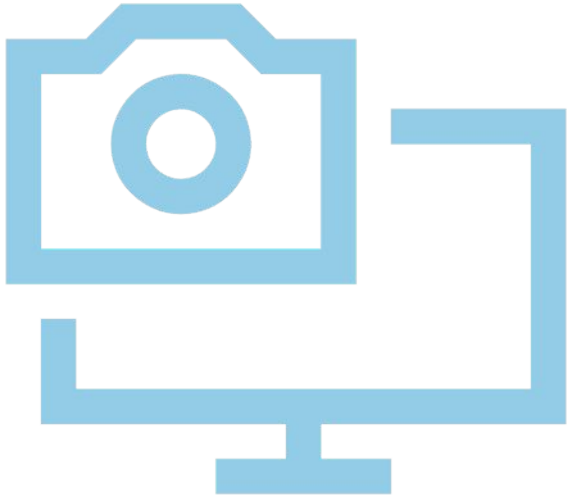
Use the function `get_dummies` and `features` dataframe to apply OneHotEncoder to the column `Orbits`, `LaunchSite`, `LandingPad`, and `Serial`. Assign the value to the variable `features_one_hot`, display the results using the method head. Your result dataframe must include all features including the encoded ones.

```
# HINT: Use get_dummies() function on the categorical columns
one_hot_or = pd.get_dummies(features['Orbit'])
one_hot_ls = pd.get_dummies(features['LaunchSite'])
one_hot_lp = pd.get_dummies(features['LandingPad'])
one_hot_se = pd.get_dummies(features['Serial'])
features_one_hot = pd.concat([one_hot_or, one_hot_ls, one_hot_lp, one_hot_se], axis=1)
features_one_hot.head()
```

	ES-L1	GEO	GTO	HEO	ISS	LEO	MEO	PO	SO	SSO	...	B1048	B1049	B1050	B1051	B1054	B1056	B1058	B1059	B1060	B1062
0	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 72 columns

APPENDIX – Standardize and split data



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

```
# students get this
transform = preprocessing.StandardScaler()
```

```
X= transform.fit(X).transform(X)
```

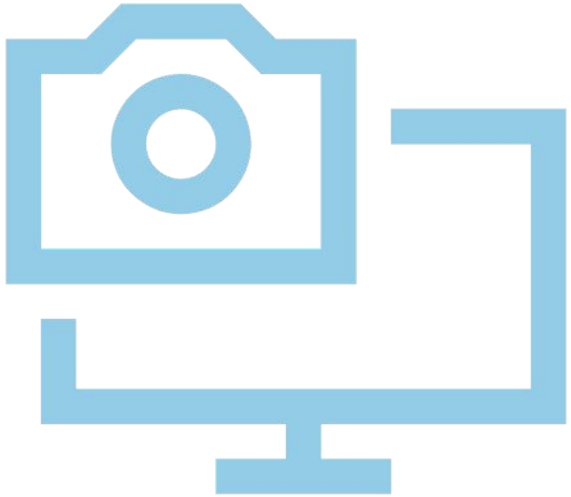
We split the data into training and testing data using the function `train_test_split`. The training data is divided into validation data, a second set used for training data; then the models are trained and hyperparameters are selected using the function `GridSearchCV`.

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.

`X_train, X_test, Y_train, Y_test`

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

APPENDIX – tuning parameters and cross validation



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

```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

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

```
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

```
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
```