IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Marsaoui Ahmed
09/01/2022

Outline*

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

* Make sure you are in the diaporama mode to see the different animations

Executive Summary

- In This project we have used different data collection techniques. After multiple data wrangling and visualization stages, we have built multiple predictions models to end up choosing the best one using different evaluation's metrics
- Obtained results show that using the right launch site with specific payload range and orbit will ensure a potential successful launch and therefore a considerable launch cost decrease.

Introduction

- The commercial space age is a fact
- Sending spacecraft to the ISS is the main activity of this filed
- The launch stage is not always successful ⇒ high cost even in case of the possibility of the reuse of the first stage
- How to predict the success of the first stage landing in order to predict the cost of the launch?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Different techniques were used (Rest APIs and Web scraping)
- Perform data wrangling
  - Applying multiple transformations to make the data readable. Dealing with missed data
- 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

Data Collection

- Two main techniques were used:

  - Rest APIs

  - Web scraping (Beautiful soup)

- Workflows of these technique are available in the next slides

Data Collection – SpaceX API

- The collected data needed to be formatted into Json format
- Most Significant features were extracted
- The data was transformed into a pandas dataframe
- The Null values of the Payload Mass Field was filling with the mean of this column
- Github URL for the [Data Collection – SpaceX API]

Data Collection – SpaceX API workflow

Defining the source
- The given URL

Getting response
- Requests.get(url)

Formatting and normalizing the response
- json()
- json_normalize()

Getting the needed fields
- Applying given functions

Constructing the final dataset from the obtained dictionnary
- pd.DataFrame(
- launch_dict)

Filtring the falcon9 data and Filling the missing PayloadloadMass Field
- .loc()
- .replace()

Data Collection - Scraping

- Building the Beautiful soup object to get the response content
- Extracting The Html tables content:

  1- Headers

  2- Relevant data
- Inserting the collected data in the previously created pandas data frame
- Github URL for the Data Collection - Scraping

Data Collection – Scraping workflow

Request the Falcon9 Launch Wiki page from its URL

- Requests.get(url)

Creating a BeautifulSoup object from the HTML response

- BeautifulSoup(response
- content,"html.parser")

Finding all tables on the wiki page

- soup.find_all('table')

creating an empty dictionary with keys from the extracted column names

- dict.fromkeys(column
- _names)

Filling up with launch records extracted from table rows

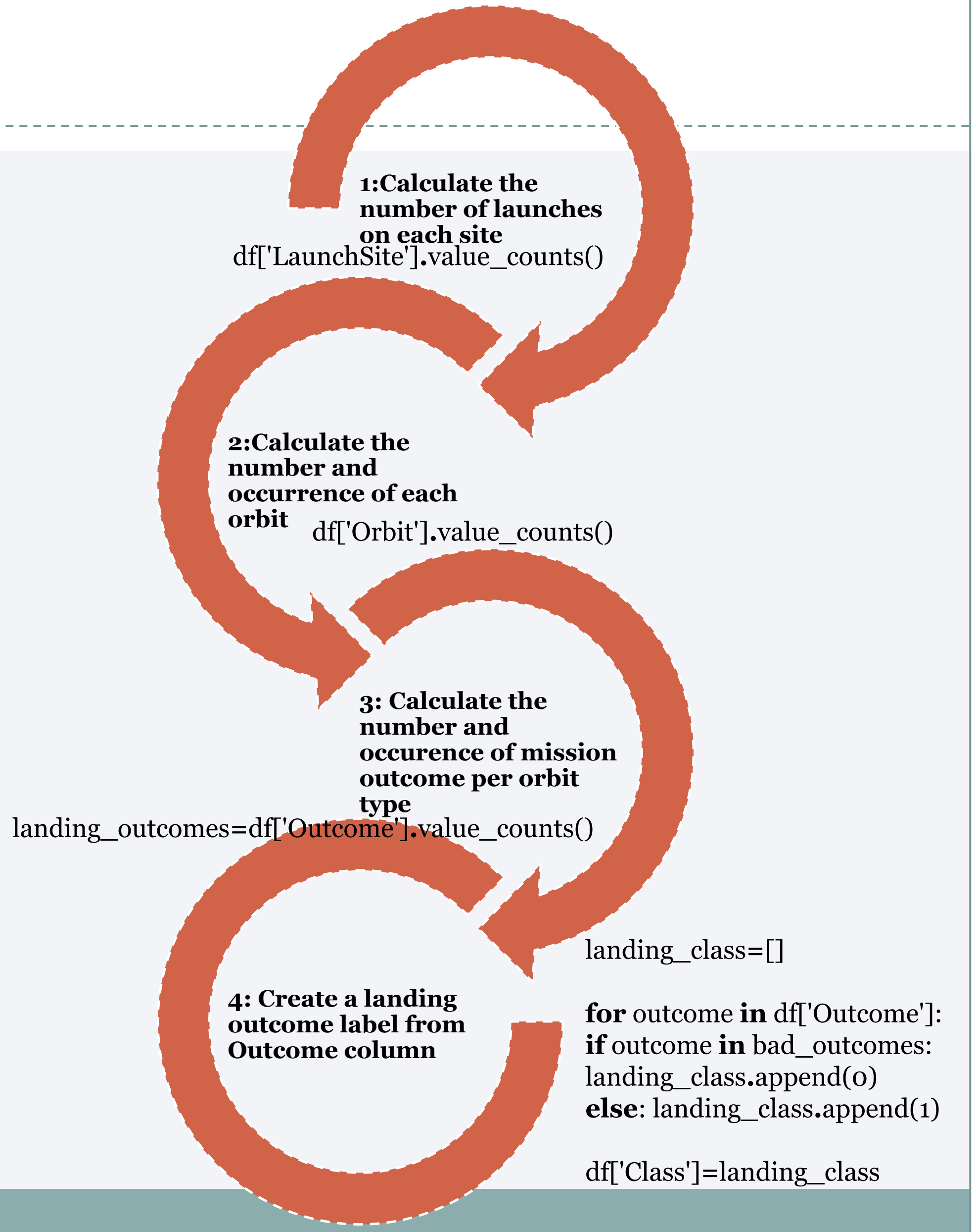
- Completing the given code snippet

Creating the dataframe and saving it to a csv file

- pd.DataFrame(
- launch_dict)
- to_csv()

Data Wrangling

- Getting first insights using DATA Analyse
- Stage1: Determining the launch frequency of each site
- Stage 2: Investigating the use of each possible orbit
- Stage3: determining the number of landing outcomes
- Stage 4: Adding the outcome binary column (the one which be used for predicting)
- GitHub URL for [data wrangling]



EDA with Data Visualization

- Multiple charts were plotted for various aims

Charts	Purpose
Scatter using catplot	-FlightNumber vs. PayloadMass using the class as hue -FlightNumber vs LaunchSite using the class as hue
Scatter charts	-Payload vs Launch Site using the class as the color -FlightNumber vs Orbit type using the class as hue -Payload vs Orbit type using the class as hue
Bar chart	The success rate of each orbit type
Line chart	The launch success yearly trend

- GitHub URL of completed EDA with data

EDA with SQL

- Investigating the obtained dataset via SQL commands
- Understand further the data distribution
- Commands included:

  1- Aggregation functions (sum(), avg(), min() and max())

  2- Subqueries

  3- Where clauses

  4-String expressions (like)

- Results will be shown later in this same report
- GitHub URL of EDA with SQL

Build an Interactive Map with Folium

- Multiple map objects were used for different purposes

Map object	Purpose
Circle	*Circle object marked each launch site based on its coordinate (Lat, Long) values.*
Icon	*Showing the launch site's name whenever a cercle object was created*
Marker	Mark the presence of the launching site empowered with the other map objects
MousePosition	To easily show the coordinates of any points of interests (such as railway)
PolyLine	Making the distance between the launch site and the chosen point of interest

- GitHub URL of interactive map with Folium map

Build a Dashboard with Plotly Dash

- To generate our interactive dash report, we created 5 elements with these interactions:

Element	Purpose
HTML H1 element	For the Dashboard Title
Dropdown	To have a searchable dropdown list containing all the launch sites and the default value of “All sites”
Pie Chart	-To show the total success launches if ALL sites are selected -To show the success (class=1) count and failed (class=0) count for the selected site
Range slider	To select the payload range. Default displays the min and max payloads
Scatter chart	To display all values for variable Payload Mass (kg) and variable class (Depends on the Drop down and range slider selection)

- GitHub URL of Plotly Dash lab

Predictive Analysis (Classification)

- The model building had the following process:

Getting the class as a numpy array

```
Y=data['Class'].to_numpy()
```

Preprocessing the data by normalizing it

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

Splitting tha data into train and test data sets

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

Creating the model object for the asked algorithms

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}
lr=LogisticRegression()
logreg_cv=GridSearchCV(lr, parameters,cv=10, iid=None)
logreg_cv.fit(X_train, Y_train)
```

Fine tuning the model using gridsearch

Using the best model to predict the test data class

```
yhat = svm_cv.predict(X_test)
```

Evaluaing via the accuracy on test data

```
print("the accuracy on the test data ",tree_cv.best_estimator_.score(X_test, Y_test))
```

Evaluting via the confusion matrix using the test data

```
plot_confusion_matrix(Y_test,yhat)
```

- Github URL for <u>the predictive analysis</u>

Results (1): Exploratory data analysis results

- Different launch sites have different success rates.

 CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%

- No rockets launched for heavy payload mass(greater than 10000) for the VAFB-SLC launch site

- For the CCAFS SLC 40 launch site, higher payload indicates higher success rate

- For KSC LC 39A launch site, lower payload (<5000) indicates higher success rate, Otherwise there is no correlation between the 2 variables

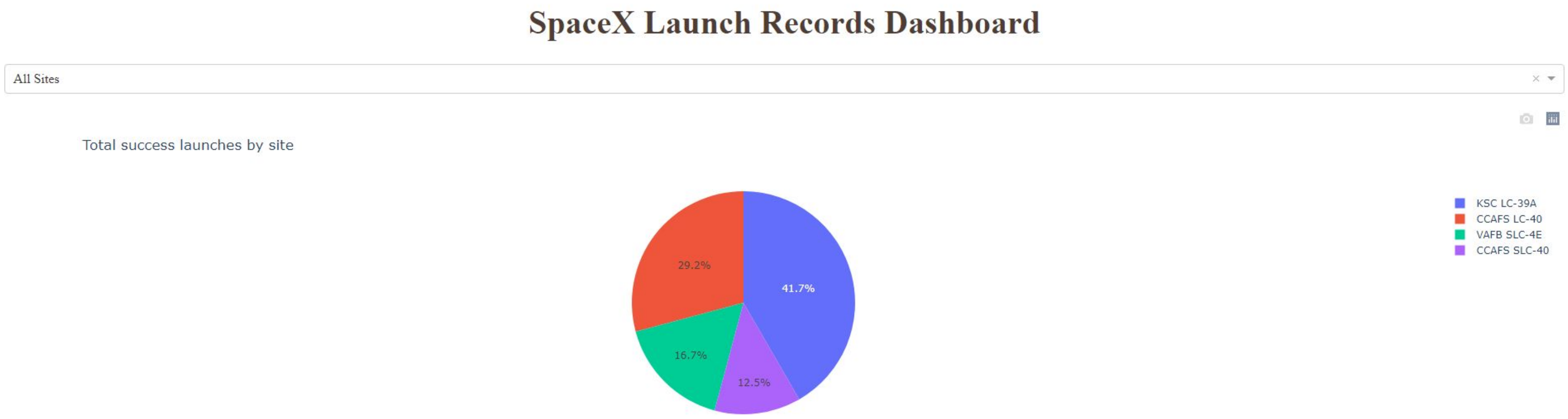
- The ES-L1, GEO,HEO and SSO have a success rate of 1 which mean all of the launch towards them were successful

- The success rate increases along with the launch year

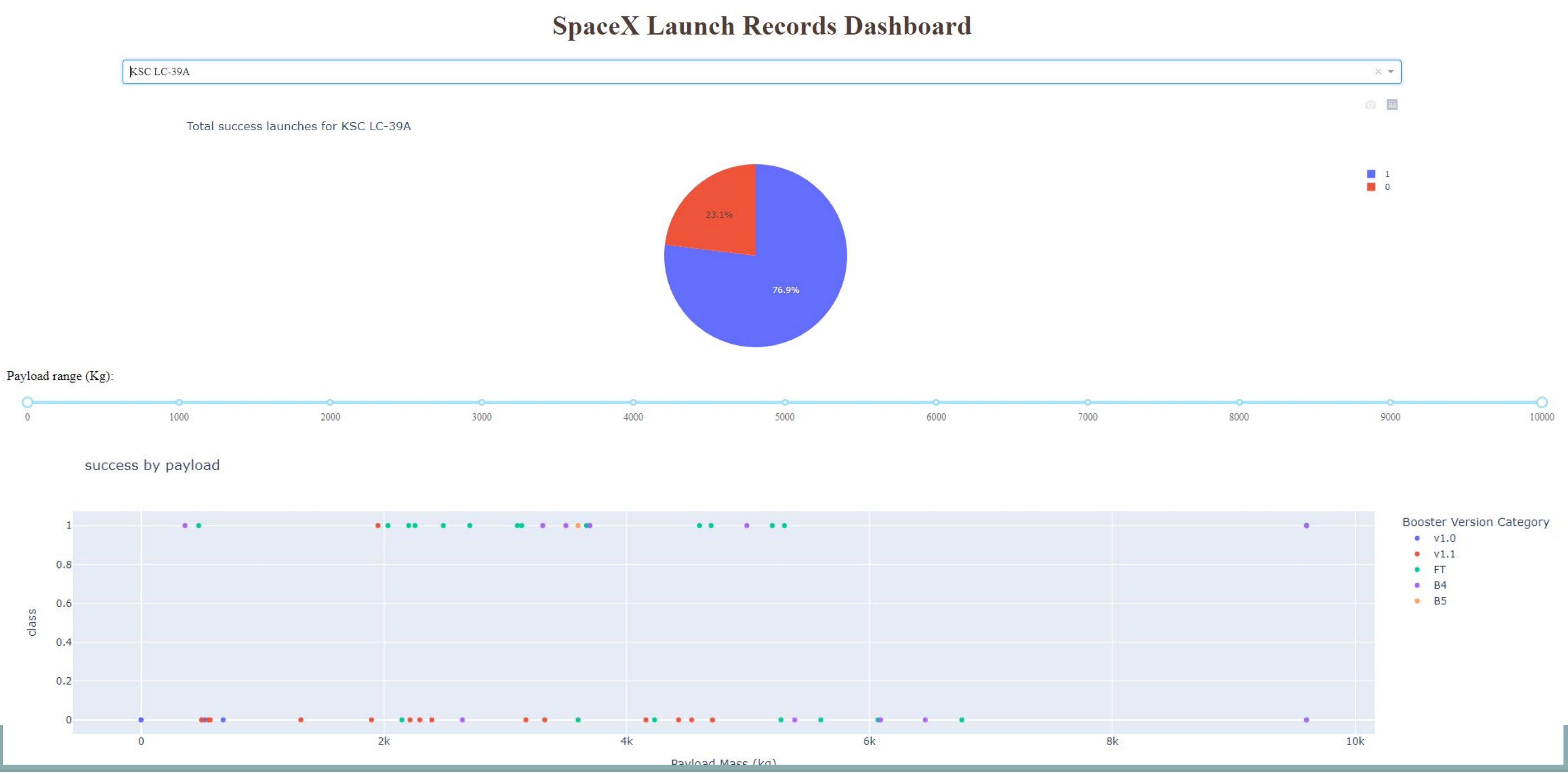
✔ **The choice of the Launch site, the payload and the orbit is very important to have a successful launch over the years**

Results (2): Interactive analytics demo in screenshots

- Among all sites, the highest number of successful launches correspond “KSC LC-39A”
- The highest success rate corresponds also to this site
- Using all sites and all payload ranges, the **FT booster version category** has the highest success rate. Manipulating different ranges of payloads is demonstrated later on this report



Results (2): Interactive analytics demo in screenshots



Results (3): Predictive analysis results

- Using the test dataset, all the models have the same jaccard_score and f1_score so the best model will be the one having the least Logloss: the decision tree model

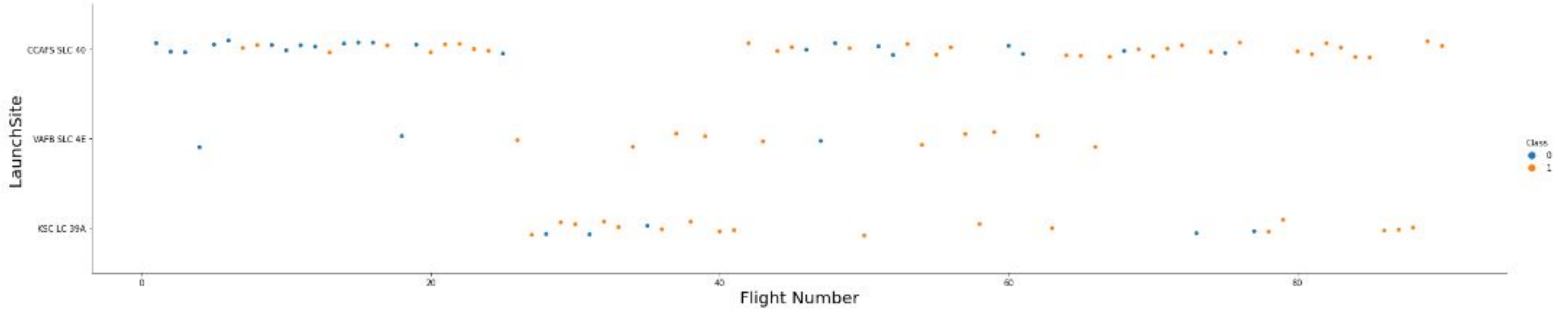
	Algorithm	jaccard	F1_score	LogLoss
0	KNN	0.5	0.814815	0.366219
1	Decision Tree	0.5	0.814815	0.287438
2	SVM	0.5	0.814815	NaN
3	LogisticRegression	0.5	0.814815	0.478667

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site


```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```

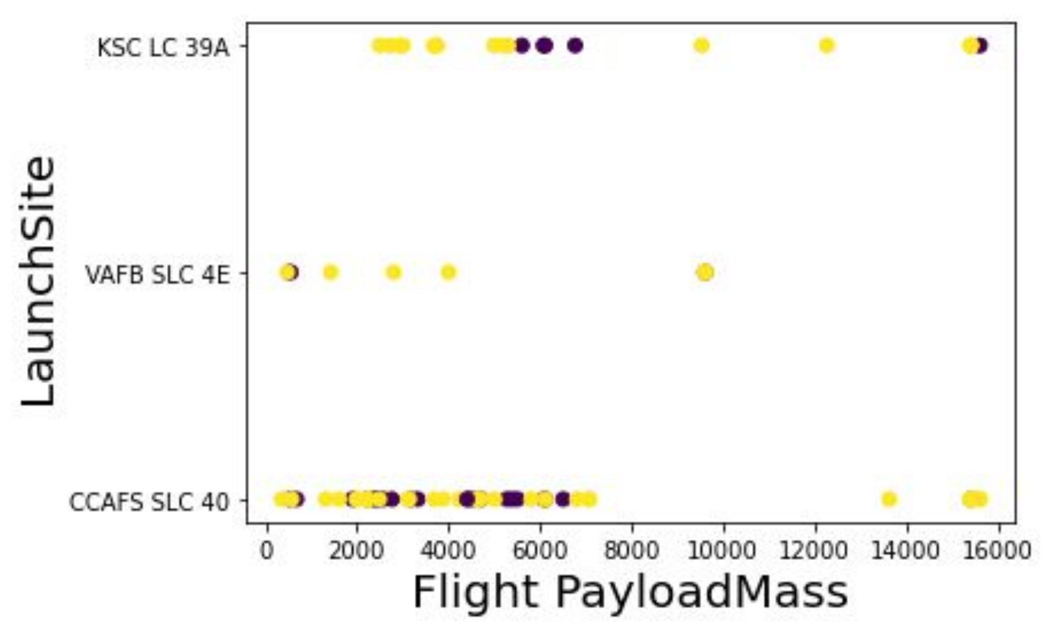


- We see that different launch sites have different success rates
- We see that as the flight number increases, the first stage is more likely to land successfully.
- The launch site is very important as there is a success rate difference between them

Payload vs. Launch Site

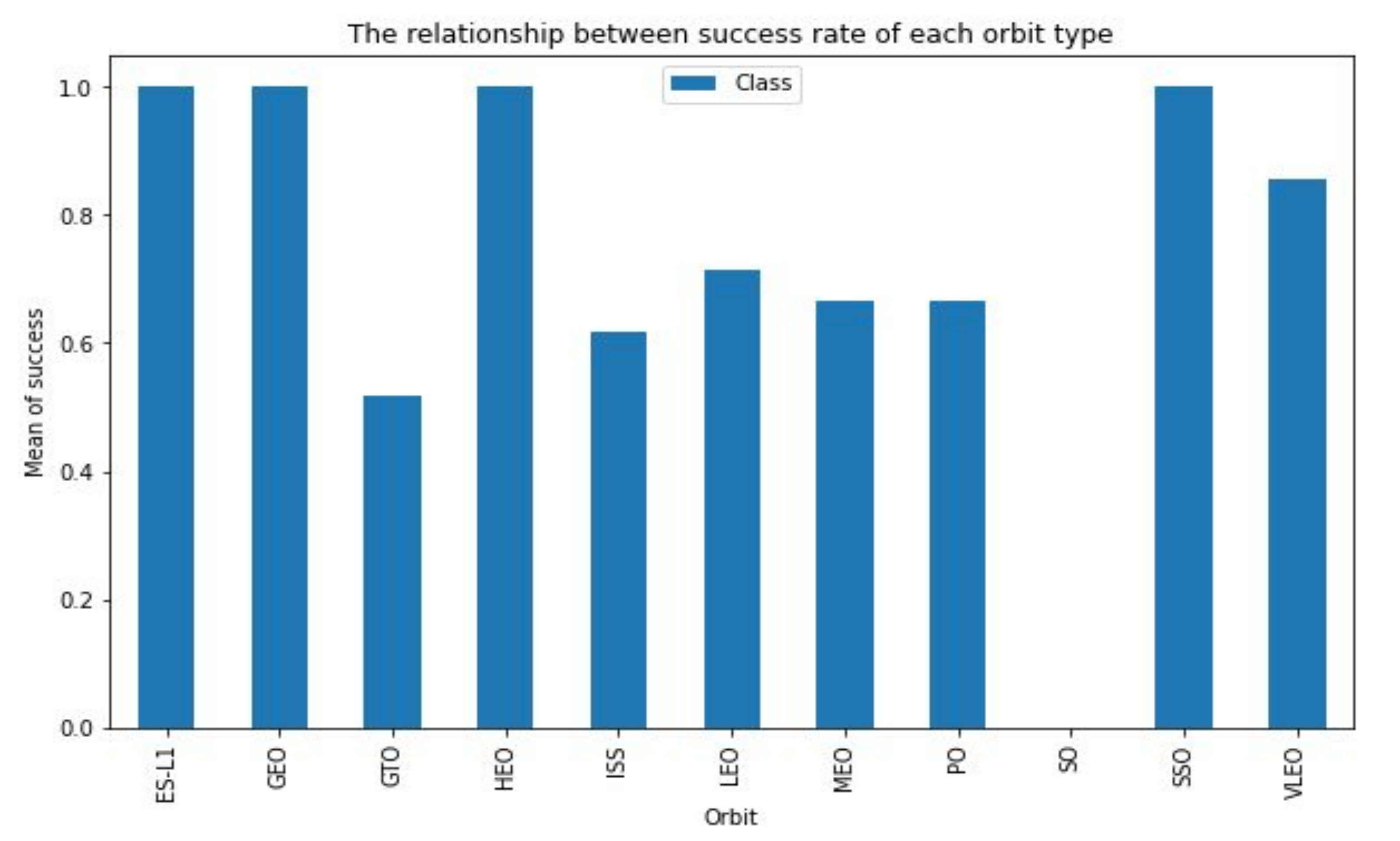
- there are no rockets launched for heavy payload mass(greater than 10000) for the VAFB-SLC launch site
- For the CCAFS SLC 40 launch site, higher payload indicates higher success rate
- For KSC LC 39A launch site, lower payload (<5000) indicates higher success rate, Otherwise there is no correlation between the 2 variables

```
plt.scatter(x='PayloadMass',y='LaunchSite', data=df,c="Class")
plt.xlabel("Flight PayloadMass",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



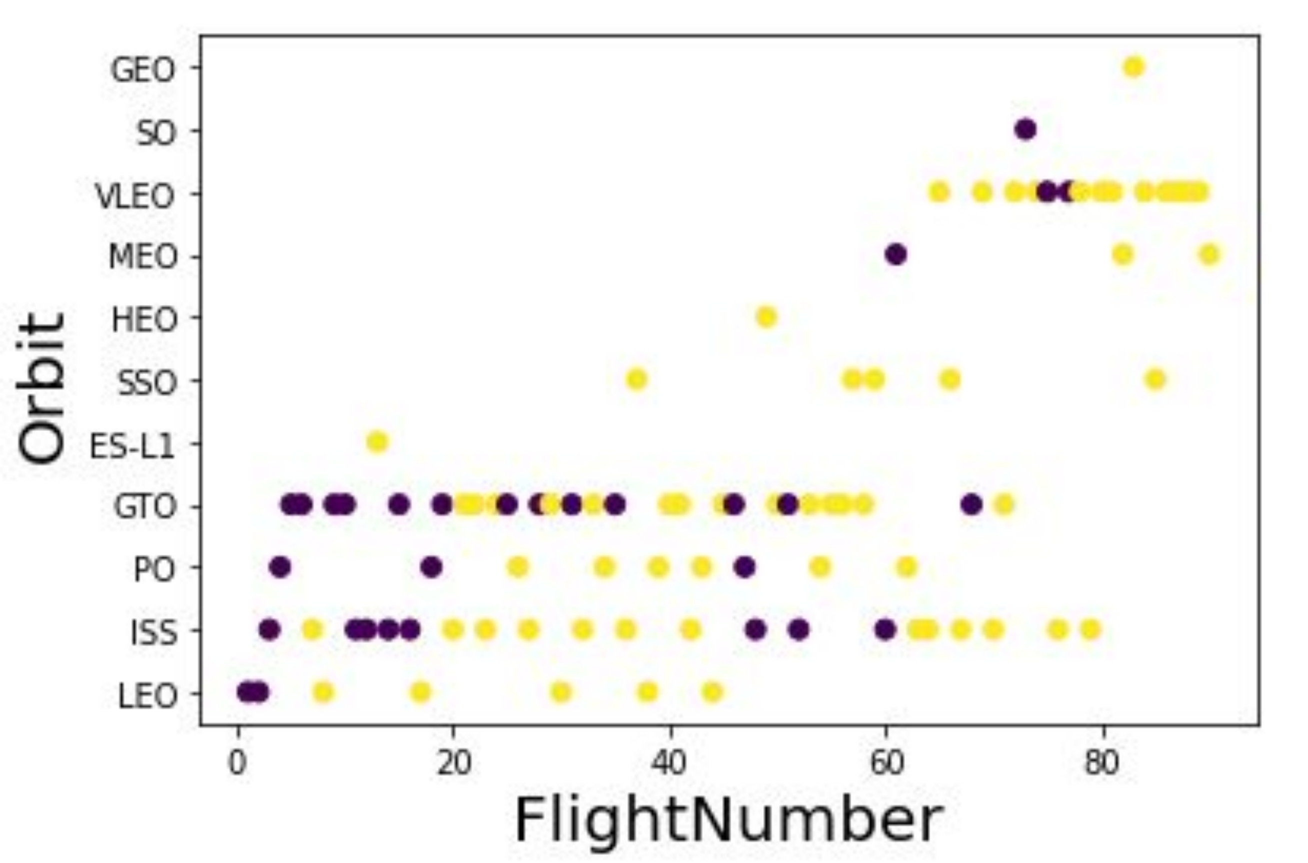
Success Rate vs. Orbit Type

- The SO orbit usage implies a direct fail since none of its launch was successful
- The **ES-L1, GEO,HEO and SSO** have a success rate of 1 which mean all of the launch from there were successful



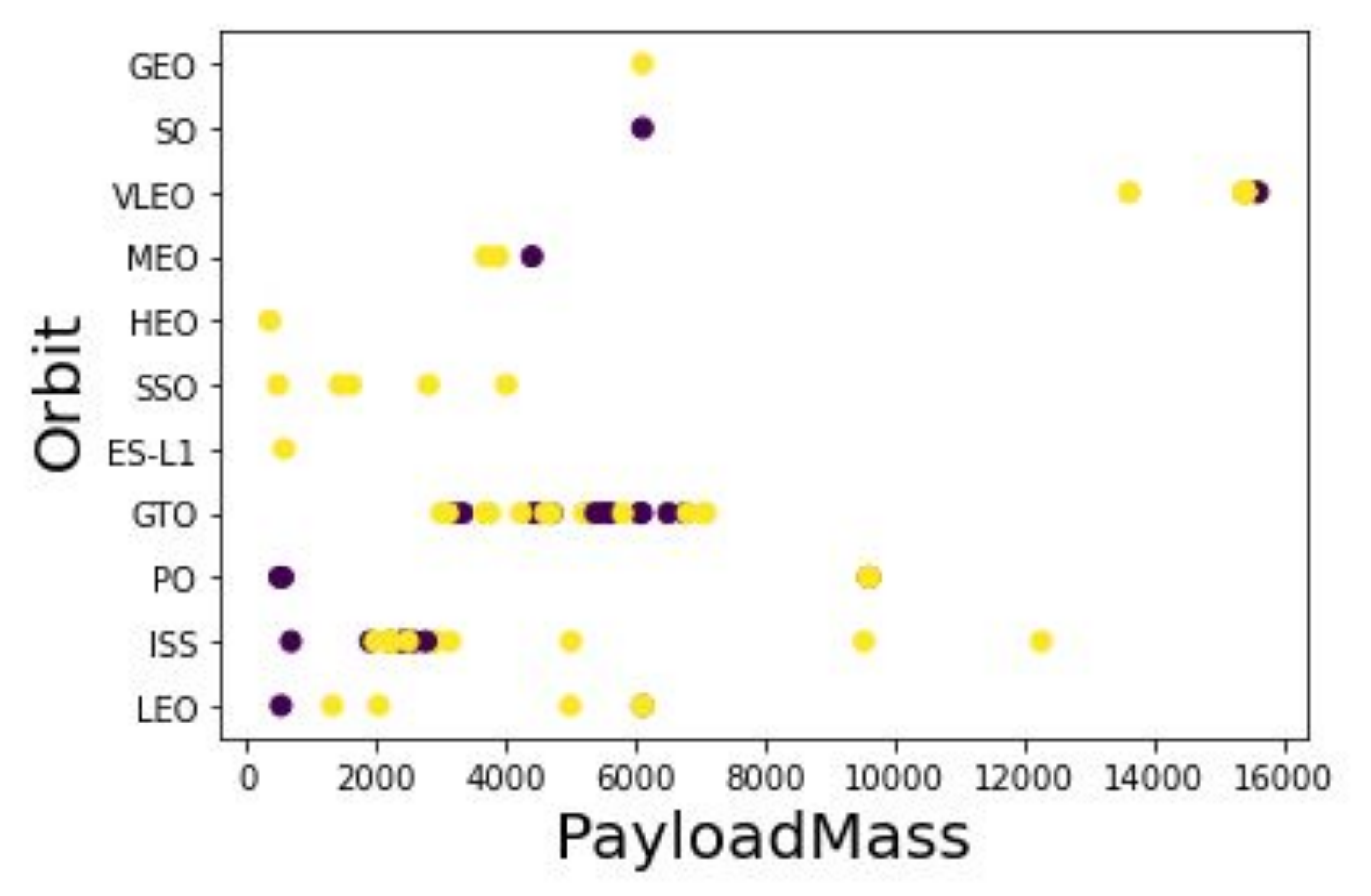
Flight Number vs. Orbit Type

- The newer the launch, the bigger its success rate for almost all the orbits
- The GTO: no correlation between the flight number and the success rate)
- The LEO orbit shows a perfect correlation between the tow variables
- Results confirms the previous graph



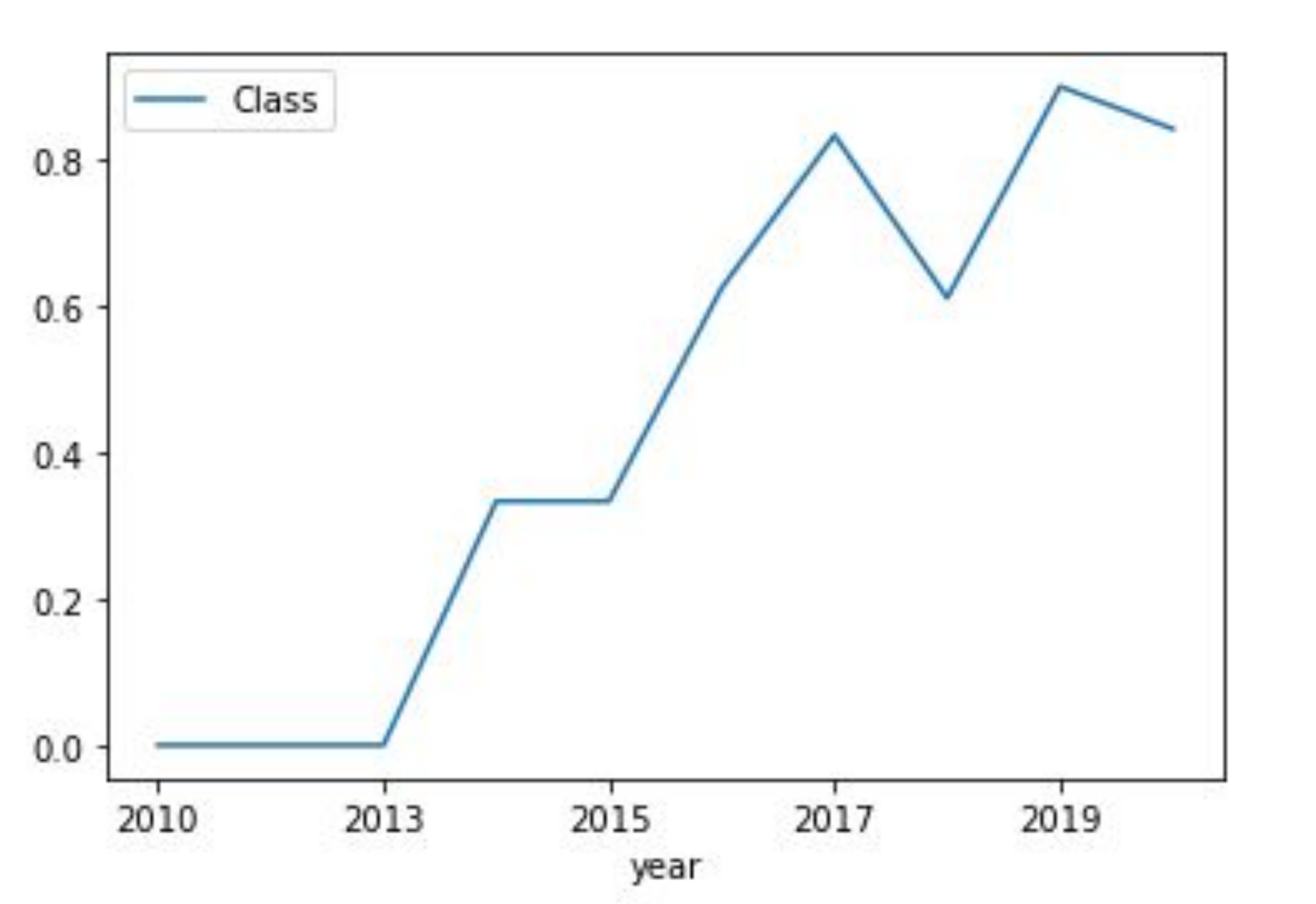
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- 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

- Since 2013, the success rate kept increasing
- An interesting decrease in 2018 must be justified



All Launch Site Names

Using the **distinct** SQL query shows 4 launch sites

```
%sql select distinct launch_site from SPACEXDATASET
```

```
 * ibm_db_sa://ghs21078:***@3883e7e4-18f5-4afe-be8c-f
Done.
```

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

Launch Site Names Begin with 'CCA'

- Using both the **Like** and the **limit** statements we found that the first 5 launch sites beginning with 'CCA' are as fellow

```
%%sql
select * from SPACEXDATASET
where launch_site like 'CCA%'
limit 5
```

```
 * ibm_db_sa://ghs21078:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/bludb
Done.
```

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

- Using both the **SUM** aggregation function and the **where** clause, the asked total mass load is:

```
%%sql
select sum(payload_mass__kg_) from SPACEXDATASET
where customer='NASA (CRS)'
```

```
 * ibm_db_sa://ghs21078:***@3883e7e4-18f5-4afe-be8c-fa3
Done.
```

1
45596

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 is obtained using the **AVG** aggregation function and the **where** clause

```
%%sql
select avg(payload_mass__kg_) from SPACEXDATASET

where booster_version='F9 v1.1'
```

```
 * ibm_db_sa://ghs21078:***@3883e7e4-18f5-4afe-be8c
Done.
```

1
2928

First Successful Ground Landing Date

- Using both the **MIN** aggregation function and the **where** clause, the result is as follow:

```
%%sql
select  min(date) from SPACEXDATASET

where landing__outcome='Success (ground pad)'
```

```
 * ibm_db_sa://ghs21078:***@3883e7e4-18f5-4afe-
Done.
```

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Using the **distinct**, **between**, **where** clauses shows these booster versions

```
%%sql
select  distinct booster_version from SPACEXDATASET
where landing__outcome='Success (drone ship)'
and payload_mass__kg_ between 4000 and 6000
```

```
 * ibm_db_sa://ghs21078:***@3883e7e4-18f5-4afe-be8c-f
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

- Using the **Count** aggregation function along with the **group by** statement, we found the following output:

```
%sql select  Mission_outcome,count(*)  from SPACEXDATASET group by Mission_outcome
```

```
 * ibm_db_sa://ghs21078:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.d
Done.
```

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

Boosters Carried Maximum Payload

- Using a **subquery** and the aggregation function **MAX**, we found the following result

```
%%sql
select  distinct booster_version from SPACEXDATASET
where payload_mass__kg_=(select max(payload_mass__kg_) from SPACEXDATASET  )
```

```
 * ibm_db_sa://ghs21078:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kq
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

- Using the where clause, the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015 are:

```
%%sql
select landing__outcome,booster_version,launch_site from SPACEXDATASET
where landing__outcome='Failure (drone ship)'
and year(date)=2015
```

```
 * ibm_db_sa://ghs21078:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io9
Done.
```

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- Using the where, between, group by, order by clauses along with the count aggregation function, the ranking of the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order is:

```
%%sql
select landing__outcome,count(*) from SPACEXDATASET
where  date between '2010-06-04' and '2017-03-20'
group by landing__outcome
order by 2 desc
```

```
 * ibm_db_sa://ghs21078:***@3883e7e4-18f5-4afe-be8c-fa
Done.
```

landing__outcome	2
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

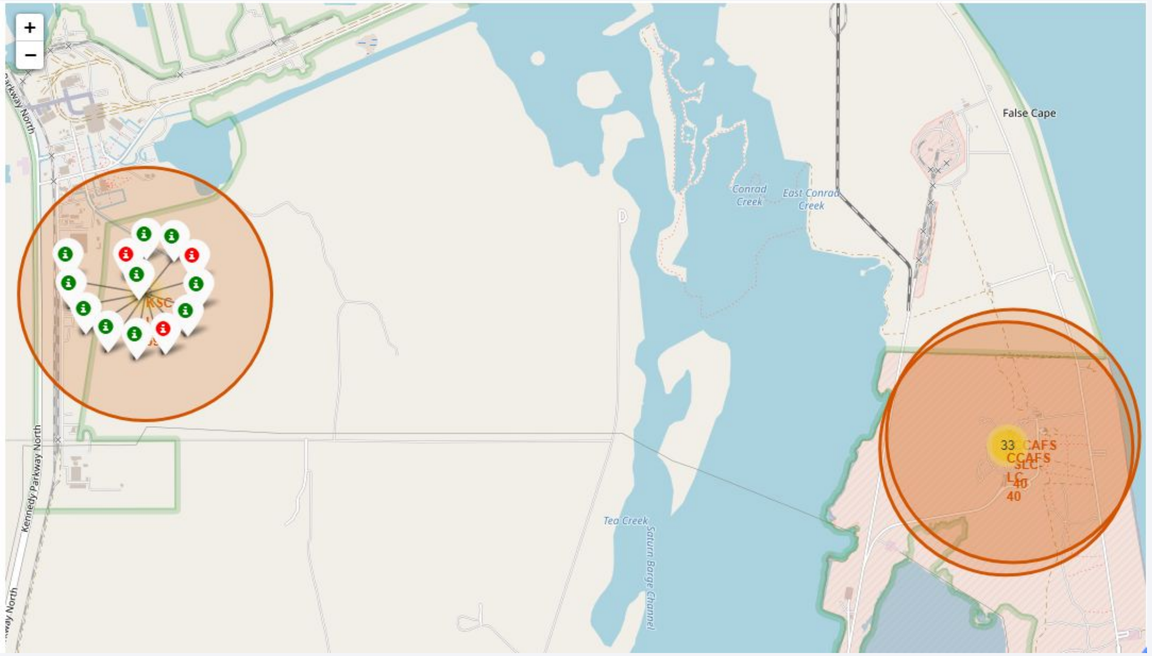
Mark all launch sites on a map

- Using the **folium.Circle** and **folium.Marker** objects, we marked all the 4 launch sites on the **site_map.**
- We can see clearly that 3 of them are located in the eastern south while the other one is in the western south

The success/failed launches' distribution (Folium)

- To give more details, using marker clusters we added iteratively a circle marker object to highlight the output of each launch using the built marker_color column
- We can see that now, and while zooming, we can distinguish between the successful launches and the failed one for each of the four sites.


```python
for index, record in spacex_df.iterrows():
    # TODO: Create and add a Marker cluster to the site map
    # marker = folium.Marker(...)
    marker=folium.Marker (
            location=[record[1], record[2]],
            icon=folium.Icon(color='white', icon_color=record[4]),
            popup=record[0],
        )
    marker_cluster.add_child(marker)
```



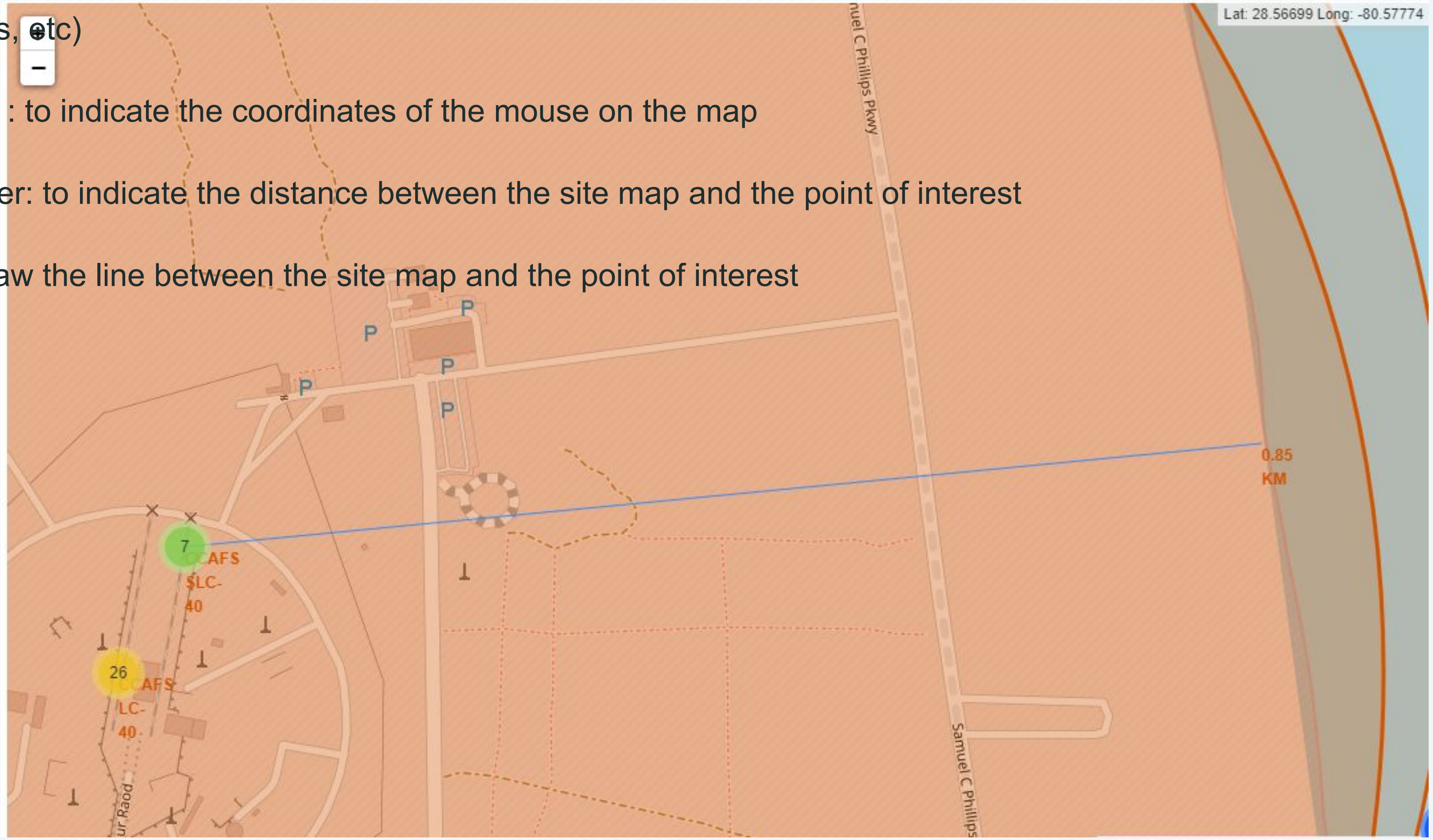
Mark the distances between a launch site to its proximities

- We added multiple new objects to our map to get the distance between a given launch site and the nearby points of interest (Coastlines, highways, railways, etc)

  -MousePosition : to indicate the coordinates of the mouse on the map

  -distance_marker: to indicate the distance between the site map and the point of interest

  -PolyLine: to draw the line between the site map and the point of interest

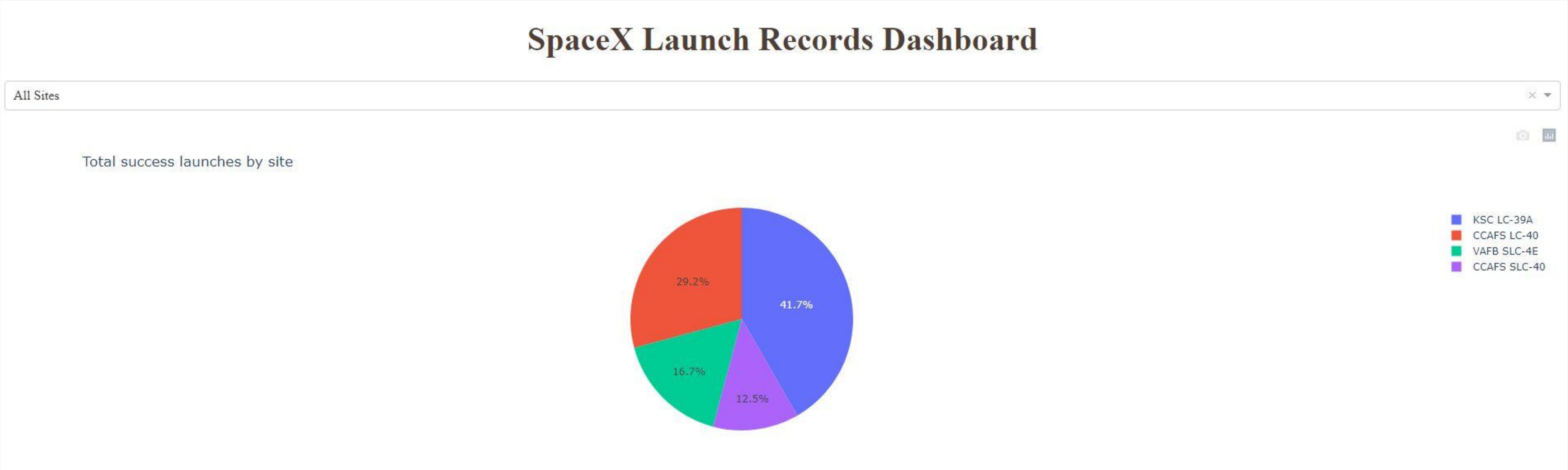


Section 5

Build a Dashboard with Plotly Dash

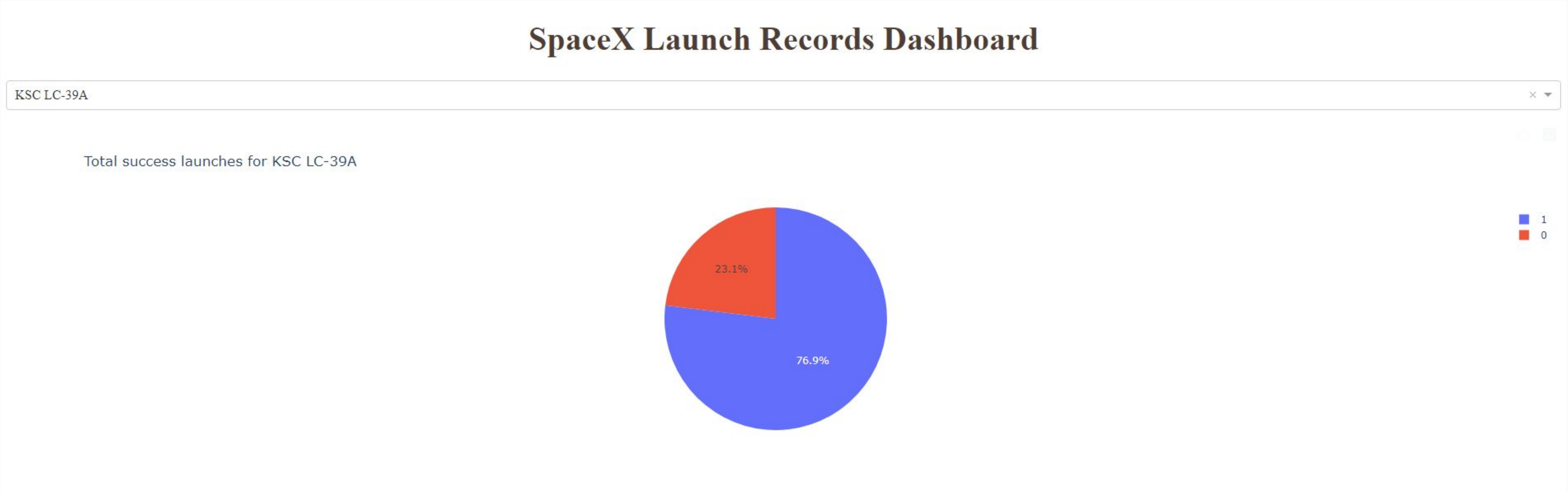
Total success launches by site

- Screenshot demonstrate that the “KSC LC-39A” has the highest successful launches among all the launches (41.7%)
- The “CCAFS SLC-40” has the lowest successful launches among all the launches (12.5%)



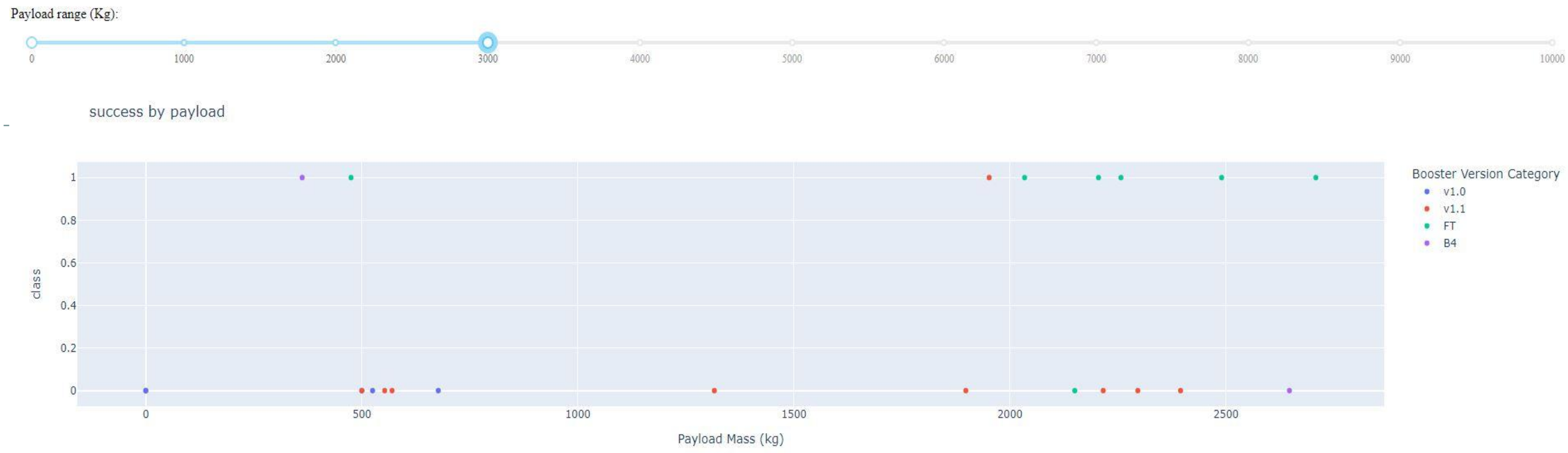
Total success launches for “KSC LC-39A”

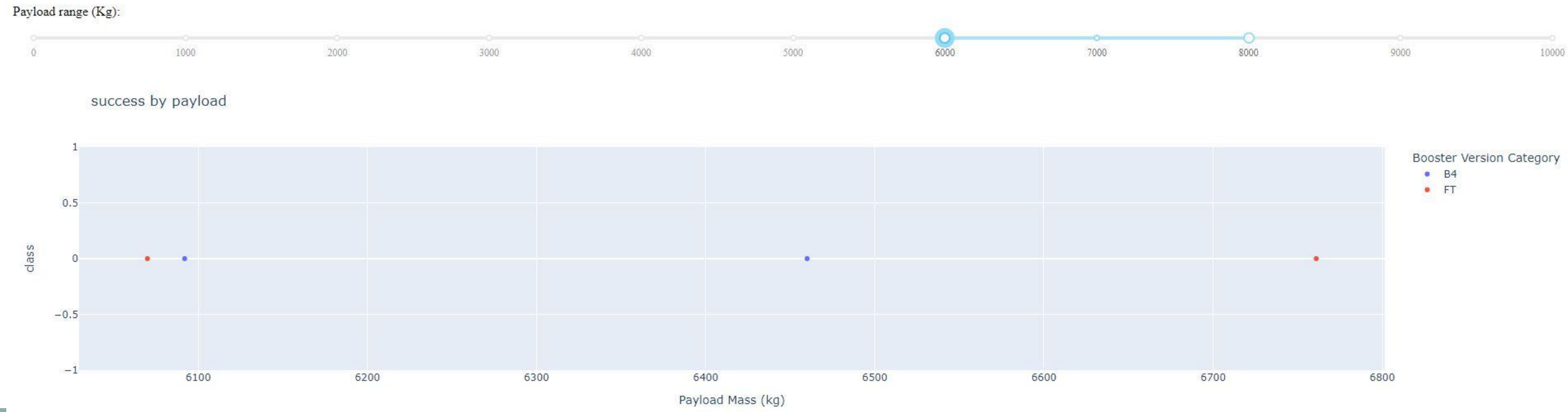
As shown below, ¾ of the launches are successful. Thus choosing this site will increase the probability of a successful first stage launching.



Success rate by payload and booster version

- The FT booster version have the highest successful rate in the overall payloads


Payload range (Kg):
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
success by payload
class
Payload Mass (kg)
Booster Version Category
v1.0
v1.1
FT
B4

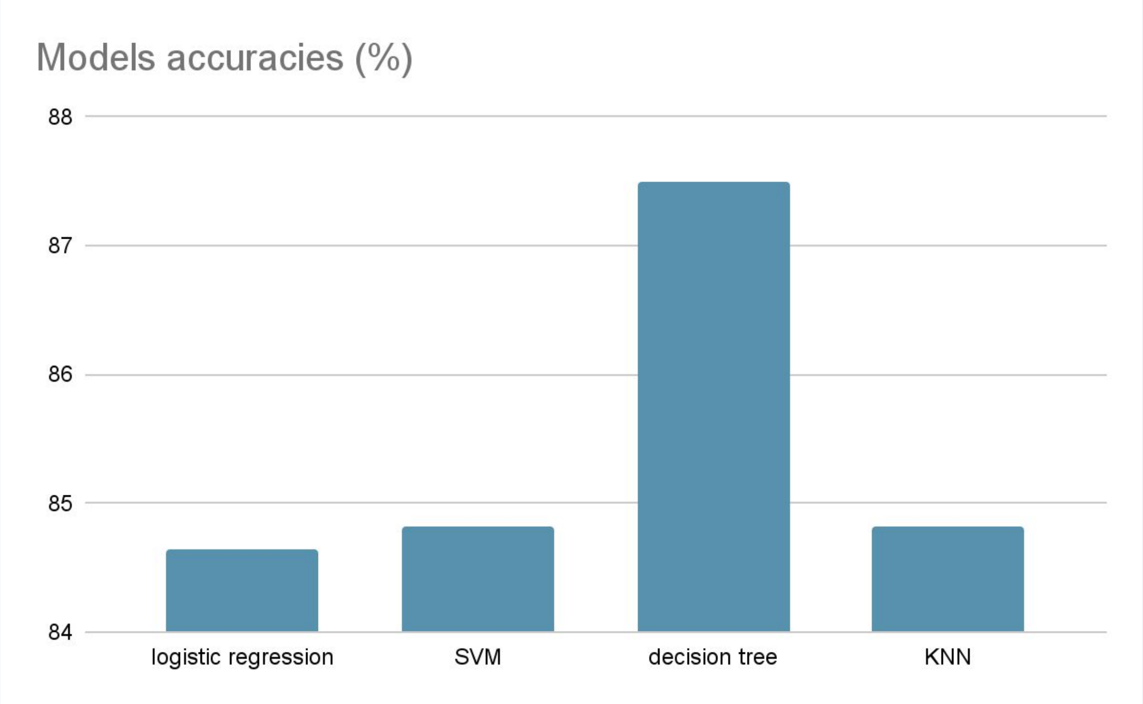

Payload range (Kg):
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
success by payload
class
Payload Mass (kg)
Booster Version Category
B4
FT

Section 6

Predictive Analysis (Classification)

Classification Accuracy

- As we saw previously in the predictive analyse results, all the models had the same accuracy on the test data set. As a result, we are going to draw a graph based on the gridsearch given scores.

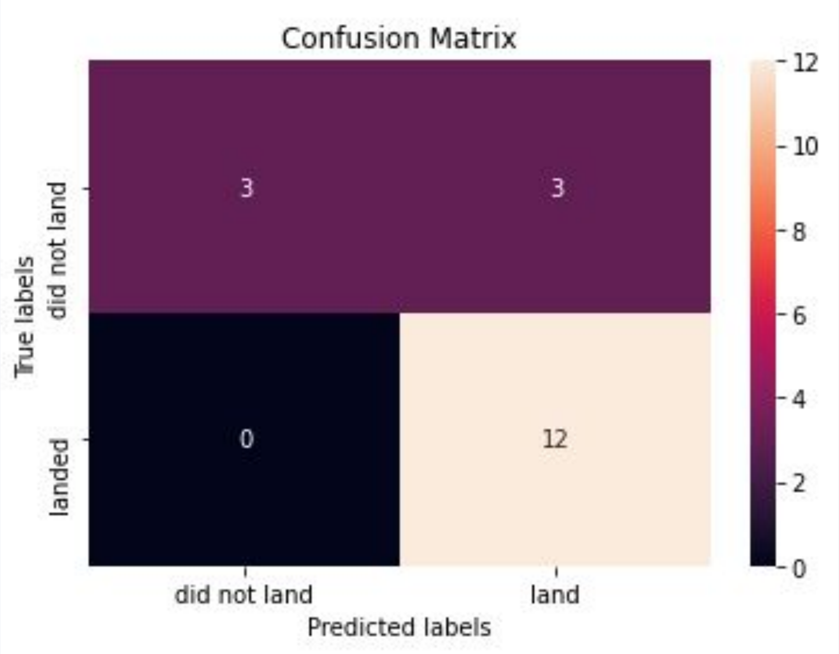


	Algorithm	jaccard	F1_score	LogLoss
0	KNN	0.5	0.814815	0.366219
1	Decision Tree	0.5	0.814815	0.287438
2	SVM	0.5	0.814815	NaN
3	LogisticRegression	0.5	0.814815	0.478667

⇒ Both results shows clearly that the decision tree model outperformed the other models (having the highest accuracy and the least logloss)

Confusion Matrix

- The confusion matrix of the decision tree model is:



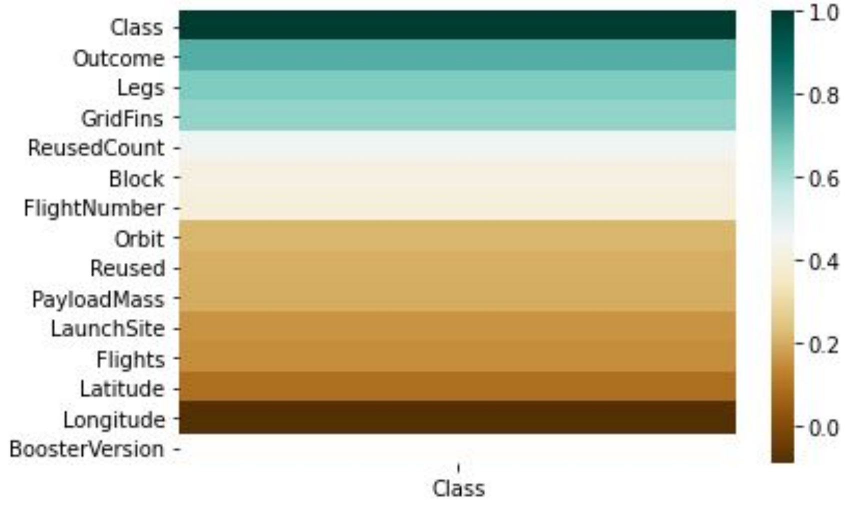
- Examining the confusion matrix, we see that the decision tree can distinguish between the different classes. We see that the major problem is false positives.

Innovative sights

- We can see that there other features that are highly correlated with the class and were not studied during the labs such as the “Outcome”,”Legs”, and the “GridFins”.
- Further works on the EDA parts could show the direct effect of these in enhancing our predictions.

```
data2=data.copy()
cat_columns = ['BoosterVersion','Orbit','LaunchSite','Outcome']
for col in cat_columns:
    data2[col] = data2[col].astype('category')
data2[cat_columns] = data2[cat_columns].apply(lambda x: x.cat.codes)
sns.heatmap(data2.corr()[['Class']].sort_values(by='Class', ascending=False),cmap='BrBG')
```

```
2]:  <AxesSubplot:>
```



Conclusions

- The launch success depends on many features (Launch site, payload, orbit etc)
- The newer the launch the more successful the first stage is
- The choice of orbit is vital since some of them have 100% success rate while others have 0%
- Payload can affect directly the success rate despite choosing the right orbit
- Launching cost can be reduced if we predicted correctly the success of a given launch
- Multiple predictive models was built and evaluated. The one which outperformed the others using different metrics is the decision tree.

Appendix

- Code snippet for the interactive dashboard

```python
# TASK 4:
# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-payload-scatter-chart` as output
@app.callback(Output(component_id='success-payload-scatter-chart', component_property='figure'),
              [Input(component_id='site-dropdown', component_property='value'), Input(component_id="payload-slider",
              component_property="value")])
def get_scatter_chart(entered_site,ranges):
    filtered_df = spacex_df [(spacex_df['Payload Mass (kg)']>= ranges[0]) & (spacex_df['Payload Mass (kg)']<=ranges [1]) ]
    if entered_site == 'ALL':
        fig = px.scatter(filtered_df, x='Payload Mass (kg)',y='class',
        color='Booster Version Category',
        title='success by payload')
        return fig
    else:
        filtered_df = filtered_df[filtered_df['Launch Site'] ==entered_site]
        fig = px.scatter(filtered_df, x='Payload Mass (kg)',y='class',
        color='Booster Version Category',
        title='success by payload for '+entered_site)
        return fig
```

```python
# TASK 2:
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output
@app.callback(Output(component_id='success-pie-chart', component_property='figure'),
              Input(component_id='site-dropdown', component_property='value'))
def get_pie_chart(entered_site):
    filtered_df = spacex_df
    if entered_site == 'ALL':
        fig = px.pie(filtered_df, values='class',
        names='Launch Site',
        title='Total success launches by site')
        return fig
    else:
        filtered_df = spacex_df[spacex_df['Launch Site'] ==entered_site]
        pie_df=filtered_df.groupby(['class'])['Launch Site'].count().reset_index()
        fig = px.pie(pie_df, values='Launch Site',
        names='class',
        title='Total success launches for '+entered_site)
        return fig
```


Thank you!