DATA SCIENCE CAPSTONE PROJECT

KHA MINH NGUYEN

13TH AUGUST, 2021

OUTLINE

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

EXECUTIVE SUMMARY

- For this project, we will collect the data from Wikipedia and SpaceX API then pre-processing the data before using the prediction model to predict the launch outcome based on the input parameters.
- Out of all the utilized models, the Decision Tree Classifier yield the best accurary (0.9)

INTRODUCTION

- We are data scientist that represent SpaceY that want to compete with SpaceX by studying their launch data.
- The purpose of the study is to determine the launch cost by calculating if the first stage will be reused or not based on the input parameters of each launch.

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

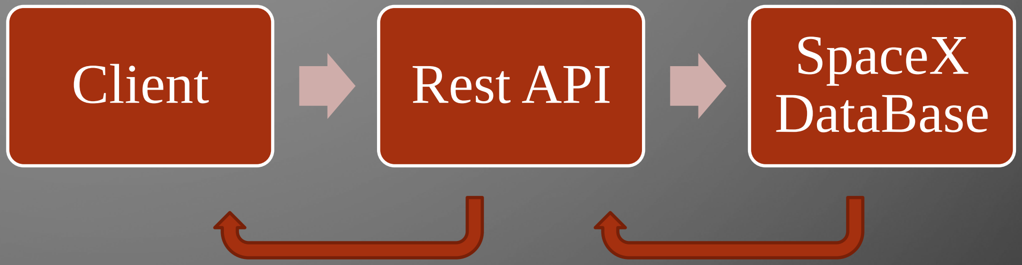
DATA COLLECTION

The data sets were collected by calling the SpaceX API and web scrapping information about Falcon 9 and Falcon Heavy launches on Wikipedia.

DATA COLLECTION – SPACEX API

Reference link:

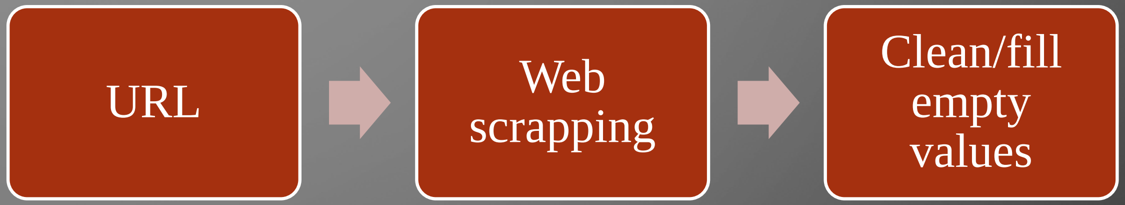
https://github.com/DeKhaos/Data-Science-Capstone-Project/blob/4acb46ddc61d2a9f89392829d74276db2e28d019/1.jupyter-labs-spacex-data-collection-api.ipynb



DATA COLLECTION – WEB SCRAPING

Reference link:

https://github.com/DeKhaos/Data-Science-Capstone-Project/blob/4acb46ddc61d2a9f89392829d74276db2e28d019/2.jupyter-labs-webscraping.ipynb



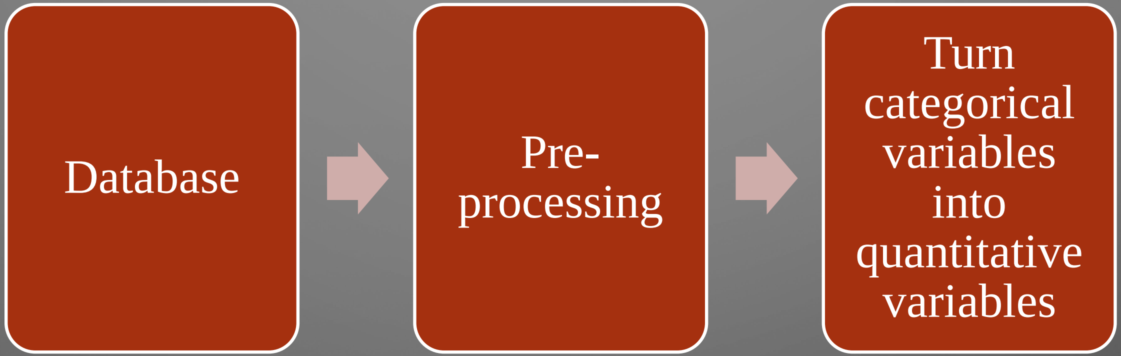
URL

Web scrapping

Clean/fill empty values

DATA WRANGLING

Since we are interested in the outcome of the launch and the prediction model can't process string values, we need to change the Launch Outcome result to binary value (0 for failure, 1 for success).



Database

Pre-processing

Turn categorical variables into quantitative variables

Reference link:

https://github.com/DeKhaos/Data-Science-Capstone-Project/blob/4acb46ddc61d2a9f89392829d74276db2e28d019/3.labs-jupyter-spacex-Data%20wrangling.ipynb

EDA WITH DATA VISUALIZATION

Line plot, bar plot and scatter plot were used to find the pattern between parameters which can be used for building model prediction.

Reference link:

https://github.com/DeKhaos/Data-Science-Capstone-Project/blob/4acb46ddc61d2a9f89392829d74276db2e28d019/5.jupyter-labs-eda-dataviz.ipynb

EDA WITH SQL

Summarize performed SQL queries using bullet points:

- 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 succesful 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 records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015
- Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.
- Reference link:

https://github.com/DeKhaos/Data-Science-Capstone-Project/blob/4acb46ddc61d2a9f89392829d74276db2e28d019/4.jupyter-labs-eda-sql-coursera.ipynb

BUILD AN INTERACTIVE MAP WITH FOLIUM

- Circles, markers and popups where added to the folium map.
- Those objects were used in order to display the locations of the launches as well as whether the launches were failure or success.

Reference link:

https://github.com/DeKhaos/Data-Science-Capstone-Project/blob/4acb46ddc61d2a9f89392829d74276db2e28d019/6.lab_jupyter_launch_site_location.ipynb

BUILD A DASHBOARD WITH PLOTLY DASH

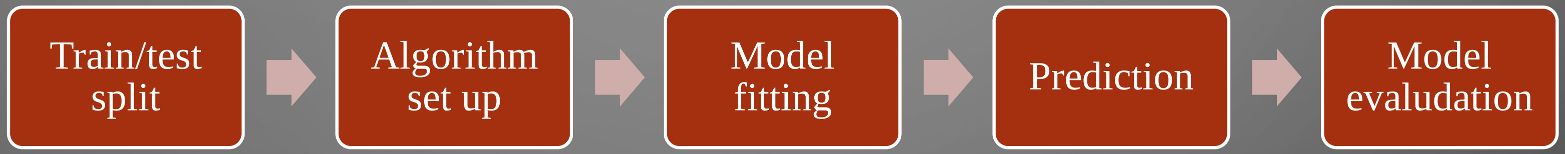
- Pie chart, scatter plot and RangeSlider were used to show an interactive dashboard which user can interact.
- Pie chart was used to display the success/failure rate at each launch site. Scatter plot was used to show which booster versions tend to success/fail at a specific launch site.

Reference link:

https://github.com/DeKhaos/Data-Science-Capstone-Project/blob/4acb46ddc61d2a9f89392829d74276db2e28d019/7.Interactive%20Dashboard%20with%20Ploty%20Dash.ipynb

PREDICTIVE ANALYSIS (CLASSIFICATION)

- First the data is divided into train/test data. Next we use the train data to build the models. And finally we use the test data to test the accuracy of each model to find the best model.



Train/test split

Algorithm set up

Model fitting

Prediction

Model evaludation

Reference link:

https://github.com/DeKhaos/Data-Science-Capstone-Project/blob/4acb46ddc61d2a9f89392829d74276db2e28d019/7.Interactive%20Dashboard%20with%20Ploty%20Dash.ipynb

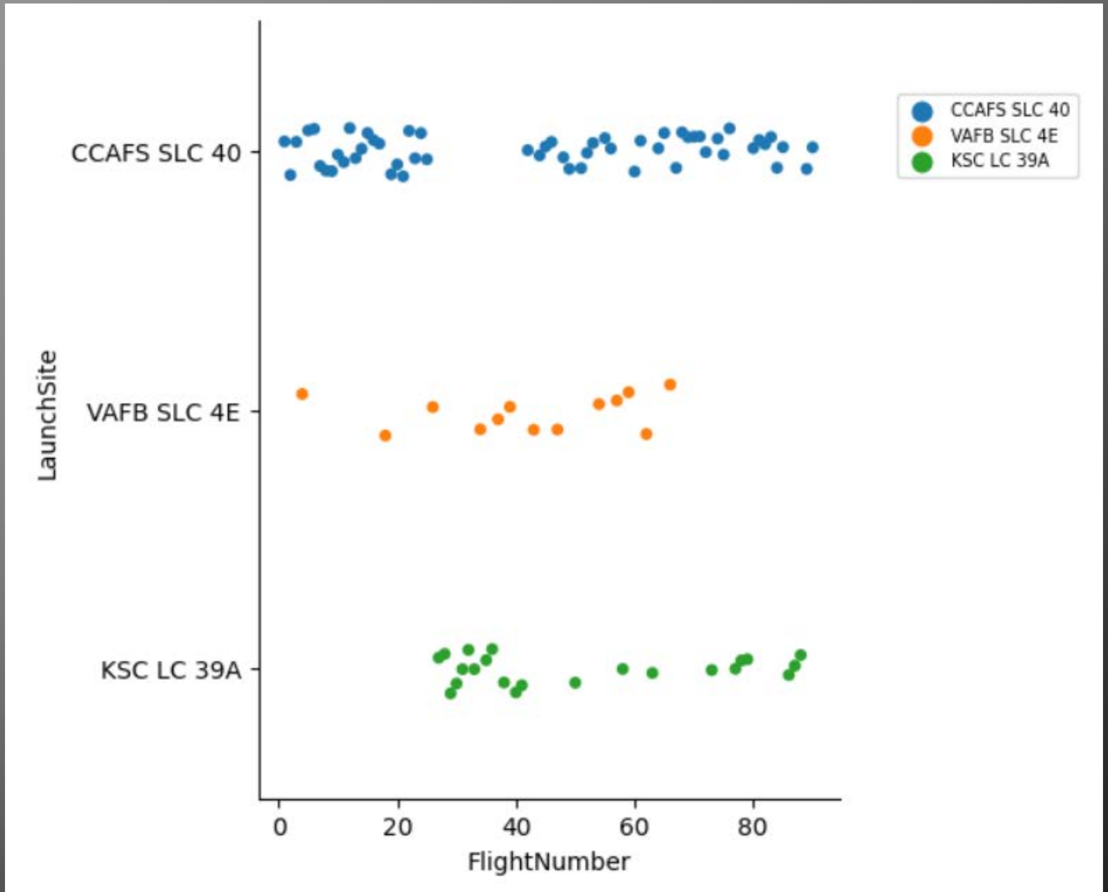
RESULTS

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

EDA WITH VISUALIZATION

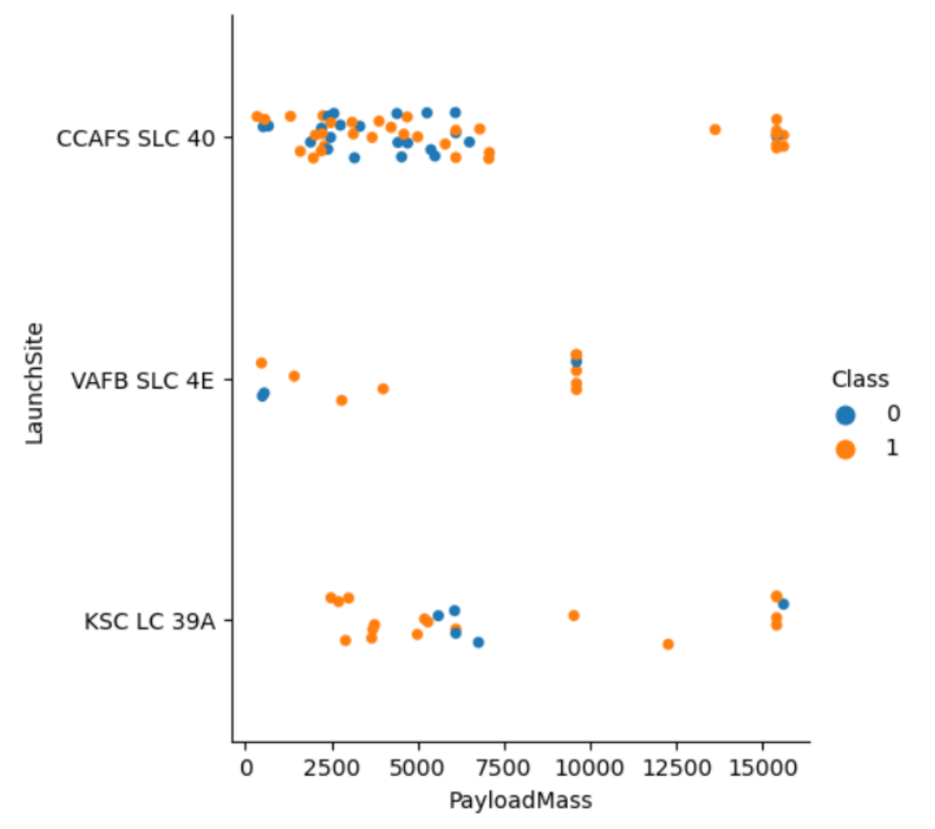
FLIGHT NUMBER VS. LAUNCH SITE

The scatter plot shows the distribution of number of launch at each launch site as well as which site is preferred than others.



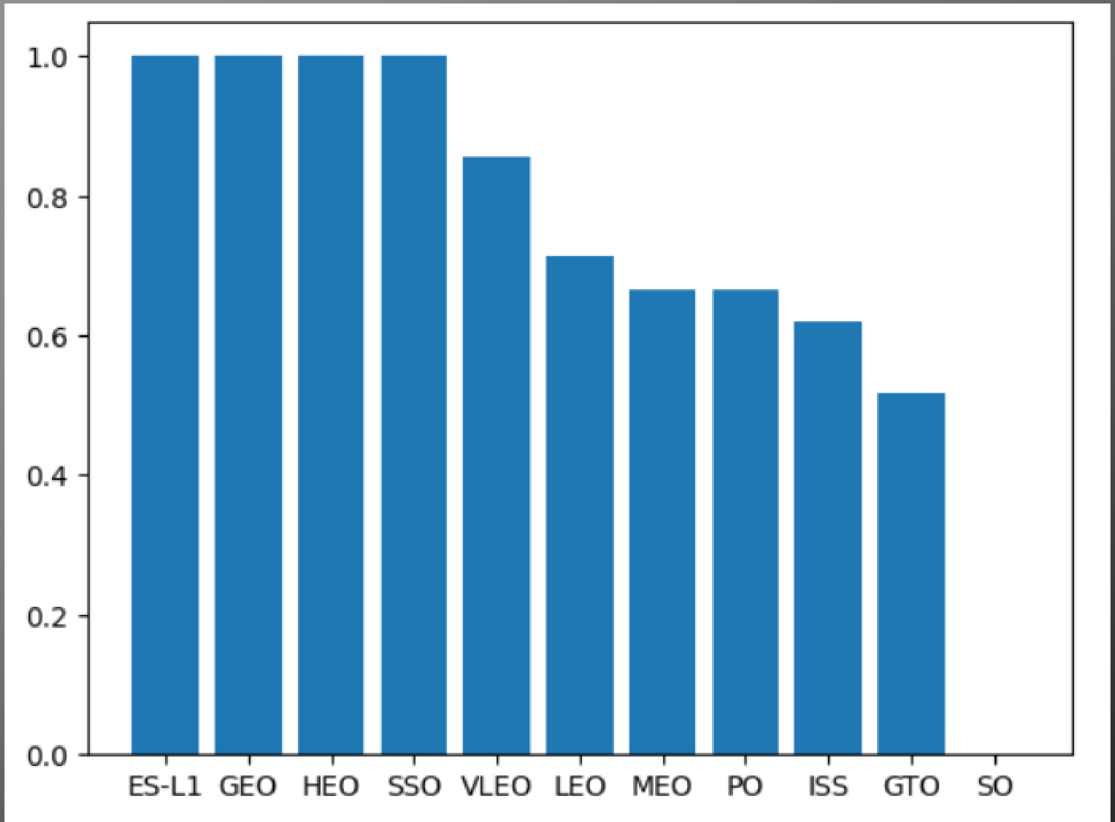
PAYLOAD VS. LAUNCH SITE

As we can see, with bigger payload, there are more success launches. For CCAFS SLC 40 launch site, low payload tend to cause launch failure.



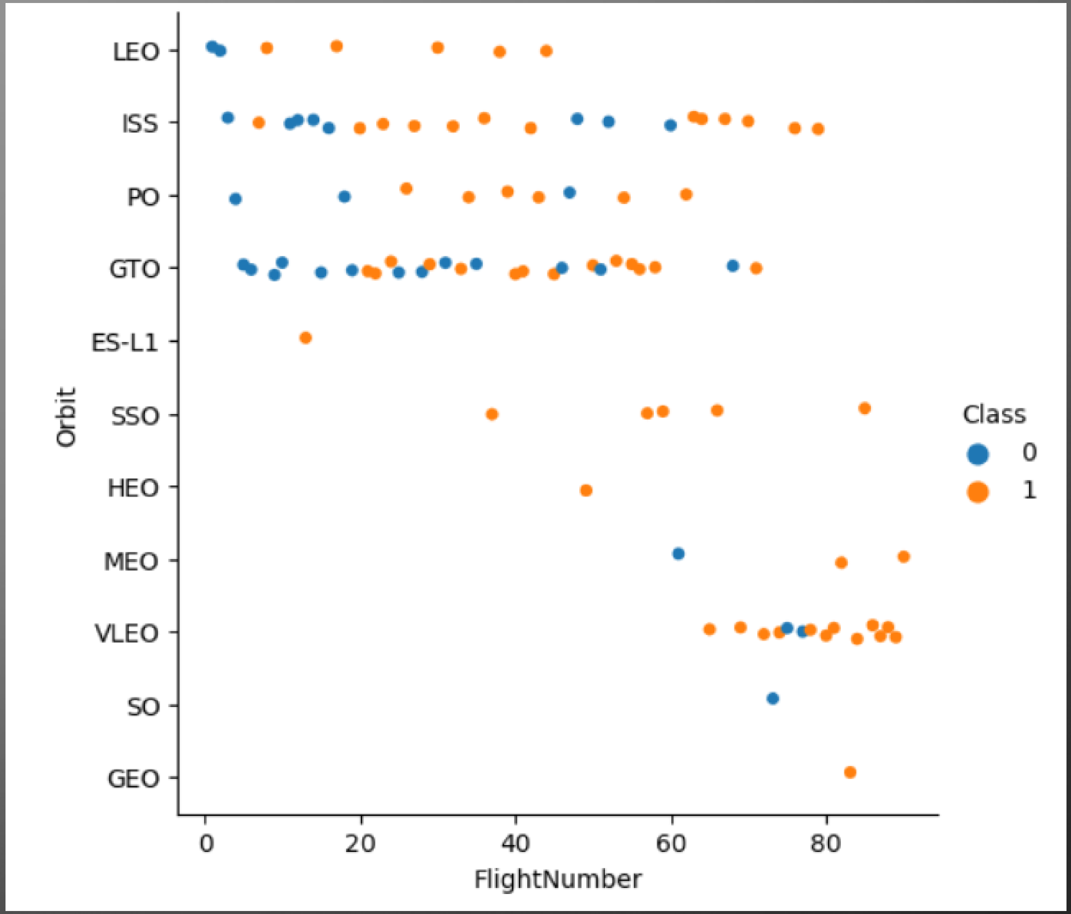
SUCCESS RATE VS. ORBIT TYPE

ELS-L1, GEO, HEO, SSO orbit type has 100% success launch rate (which can due to small sample), while others have less than 70% success rate.



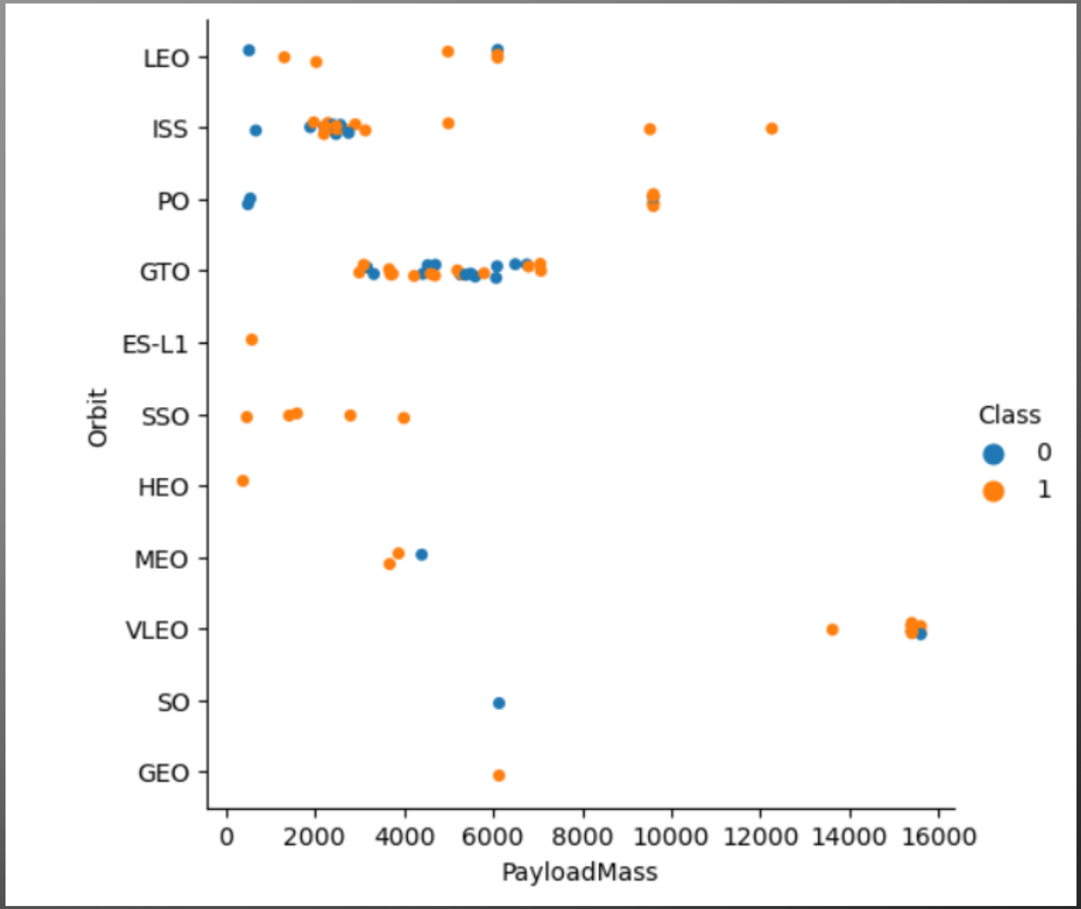
FLIGHT NUMBER VS. ORBIT TYPE

LEO, ISS, PO, GTO are the frequent orbits which use for launching. VLEO orbit are getting more frequent recently.



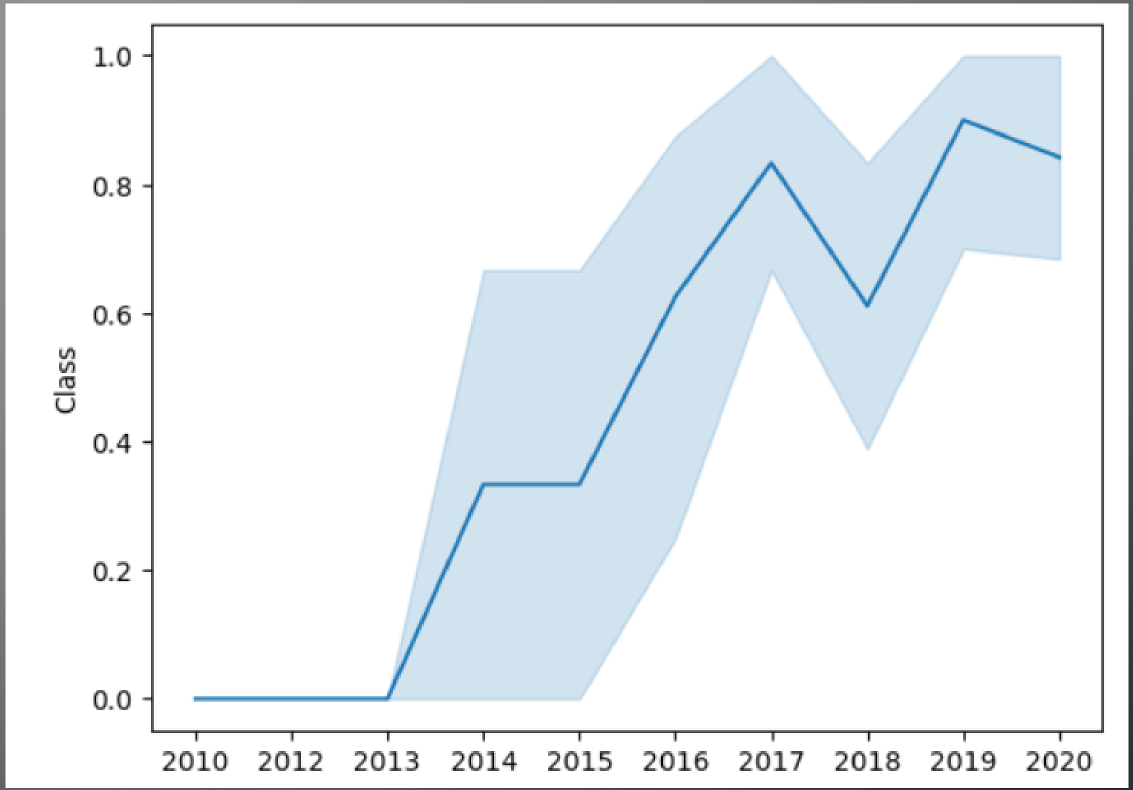
PAYLOAD VS. ORBIT TYPE

Low payload have a negative impact on GTO and ISS orbit and positive effect on SSO orbit.



LAUNCH SUCCESS YEARLY TREND

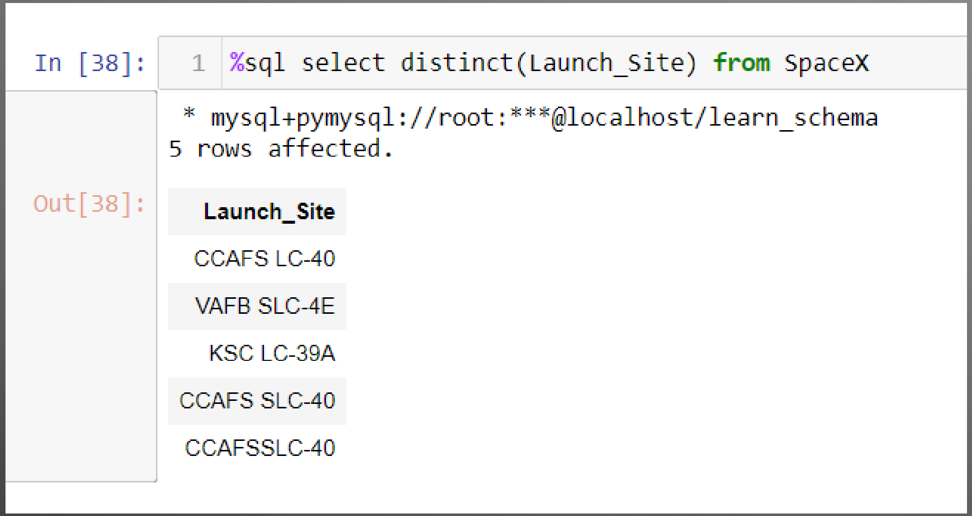
The success rate increases with recent years. This can be due to changes in technology or launch experience over the years.



EDA WITH SQL

ALL LAUNCH SITE NAMES

- Find the names of the unique launch sites



```
In [38]: %sql select distinct(Launch_Site) from SpaceX
```

```
 * mysql+pymysql://root:***@localhost/learn_schema
5 rows affected.
```

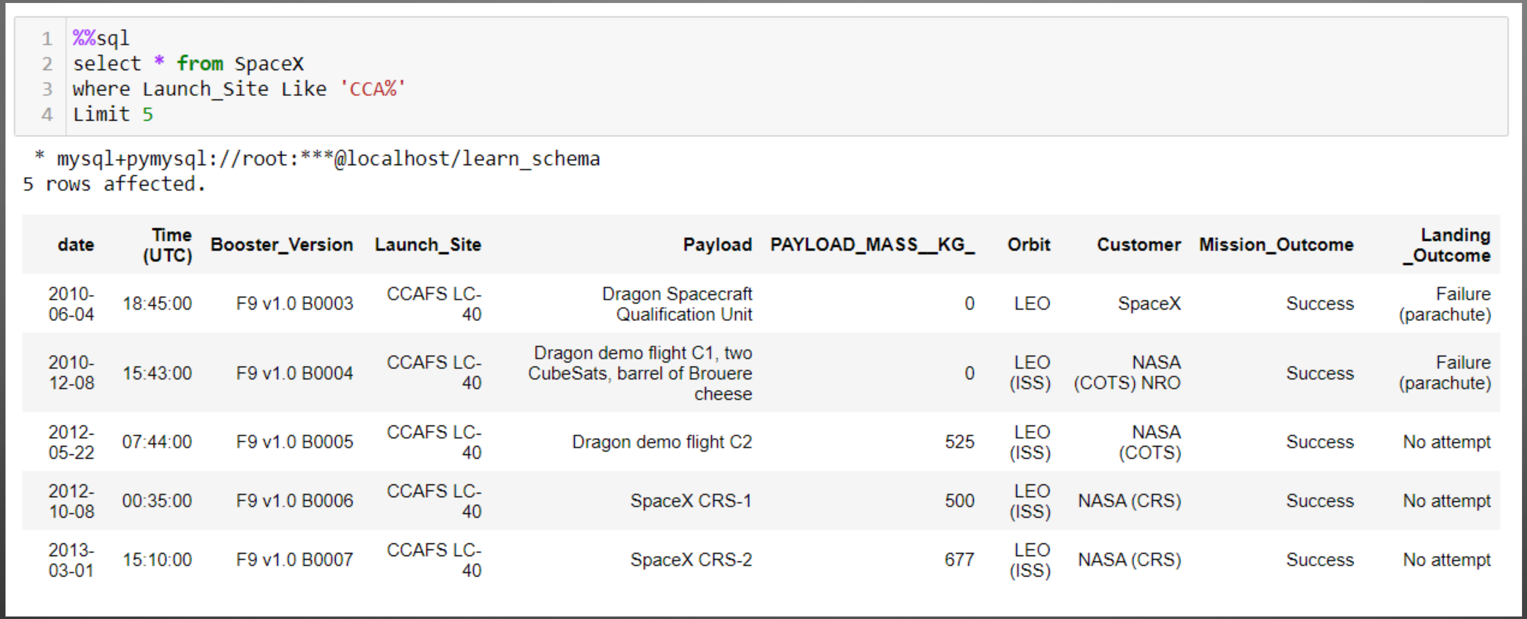
Out[38]:

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
CCAFSSLC-40

- We use DISTRICT to return unique launch site

LAUNCH SITE NAMES BEGIN WITH `CCA`

- Find all launch sites begin with `CCA`



```
%%sql
select * from SpaceX
where Launch_Site Like 'CCA%'
Limit 5
```

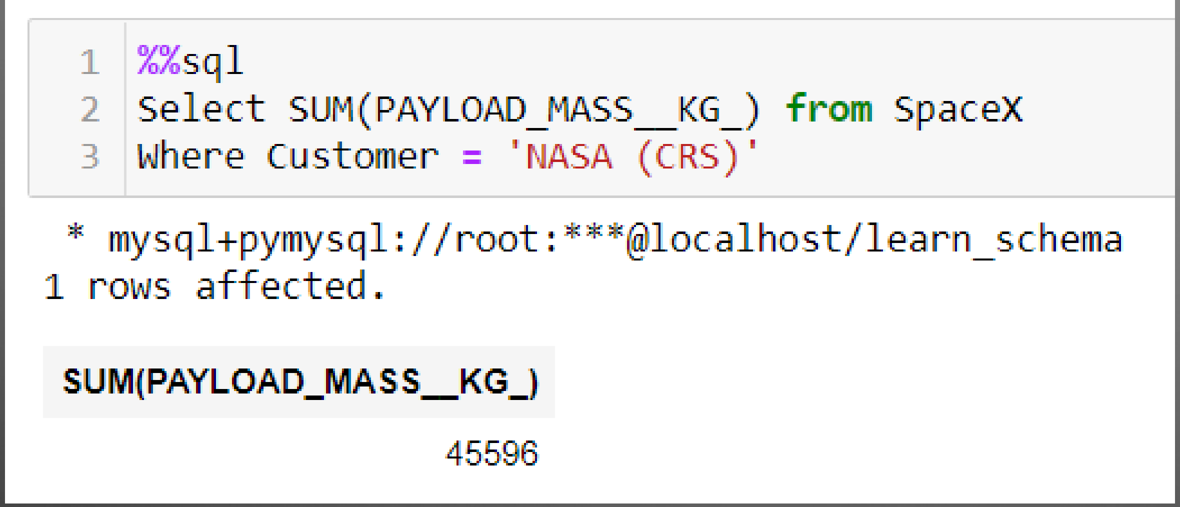
```
 * mysql+pymysql://root:***@localhost/learn_schema
5 rows affected.
```

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

- We use wildcard symbol % to return all string with the starting letters ‘CCA’

TOTAL PAYLOAD MASS

- Calculate the total payload carried by boosters from NASA



```
%%sql
Select SUM(PAYLOAD_MASS__KG_) from SpaceX
Where Customer = 'NASA (CRS)'
```

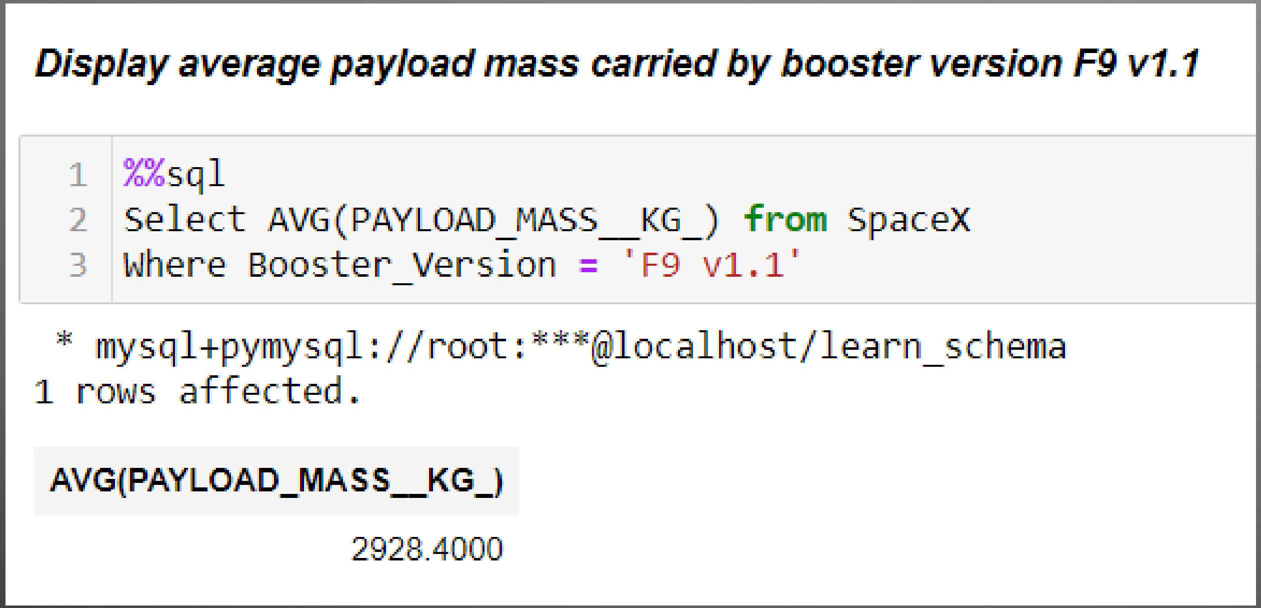
```
 * mysql+pymysql://root:***@localhost/learn_schema
1 rows affected.
```

SUM(PAYLOAD_MASS__KG_)
45596

- We use WHERE clause to define which index will be summed.

AVERAGE PAYLOAD MASS BY F9 V1.1

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



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

```
%%sql
Select AVG(PAYLOAD_MASS__KG_) from SpaceX
Where Booster_Version = 'F9 v1.1'
```

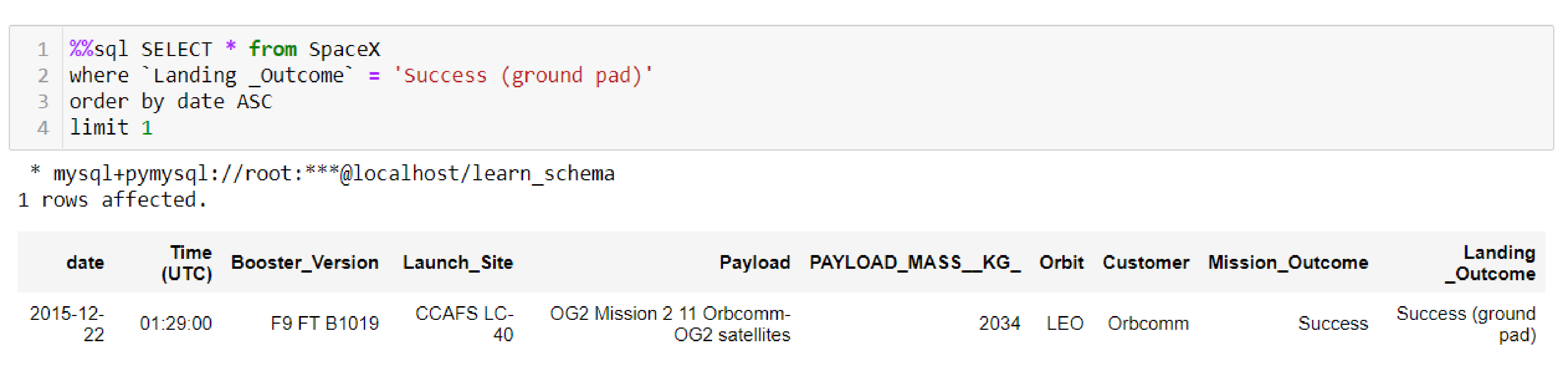
```
 * mysql+pymysql://root:***@localhost/learn_schema
1 rows affected.
```

AVG(PAYLOAD_MASS__KG_)
2928.4000

- We use where clause and AVG function to return the average payload mass by F9 v1.1

FIRST SUCCESSFUL GROUND LANDING DATE

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



```
%%sql SELECT * from SpaceX
where `Landing _Outcome` = 'Success (ground pad)'
order by date ASC
limit 1
```

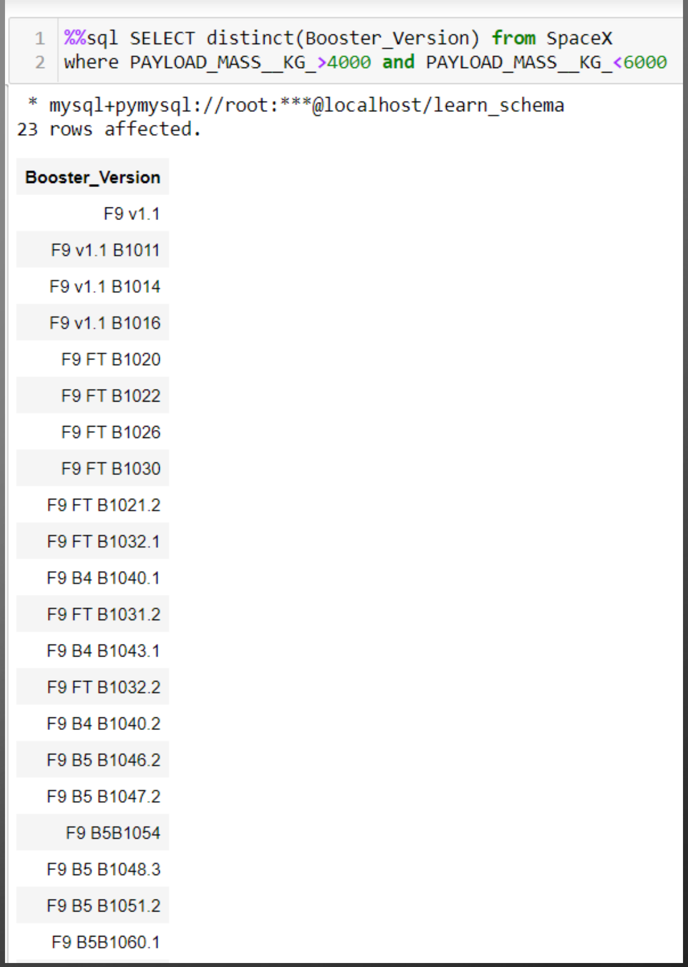
```
 * mysql+pymysql://root:***@localhost/learn_schema
1 rows affected.
```

date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
2015-12-22	01:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)

- We use sub-query and where clause, order by to return the first successful landing outcome

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
- We use distinct and where clause to return the list of Booster_Version



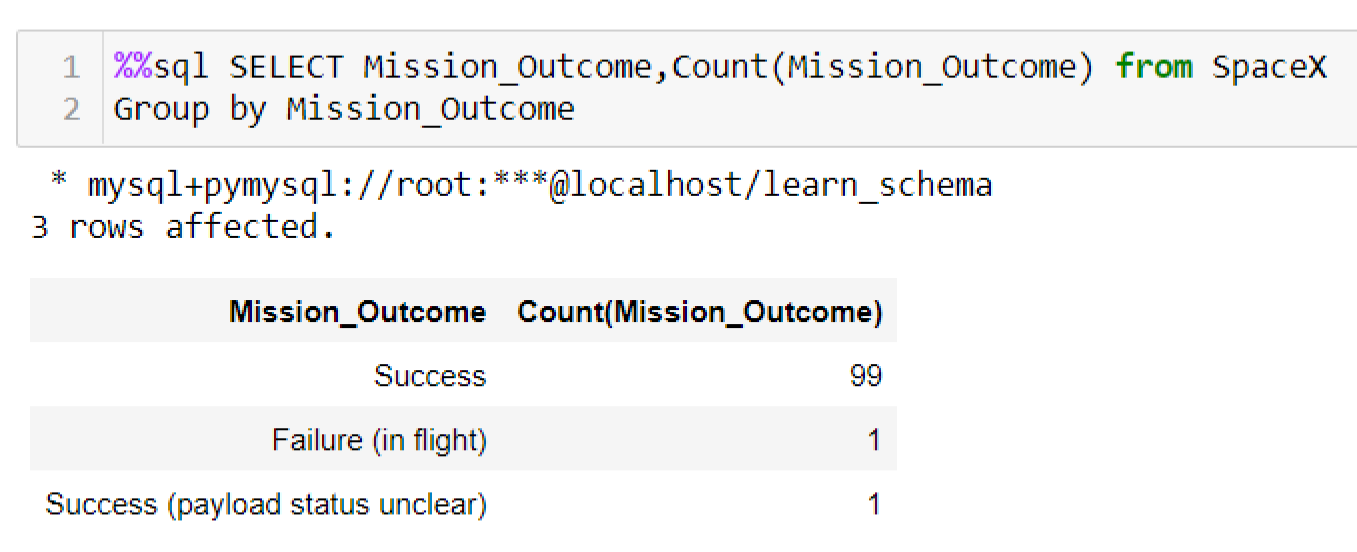
```
%%sql SELECT distinct(Booster_Version) from SpaceX
where PAYLOAD_MASS__KG_>4000 and PAYLOAD_MASS__KG_<6000
```

```
 * mysql+pymysql://root:***@localhost/learn_schema
23 rows affected.
```

Booster_Version
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1014
F9 v1.1 B1016
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1030
F9 FT B1021.2
F9 FT B1032.1
F9 B4 B1040.1
F9 FT B1031.2
F9 B4 B1043.1
F9 FT B1032.2
F9 B4 B1040.2
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5B1054
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5B1060.1

TOTAL NUMBER OF SUCCESSFUL AND FAILURE MISSION OUTCOMES

- Calculate the total number of successful and failure mission outcomes



```
%%sql SELECT Mission_Outcome,Count(Mission_Outcome) from SpaceX
Group by Mission_Outcome
```

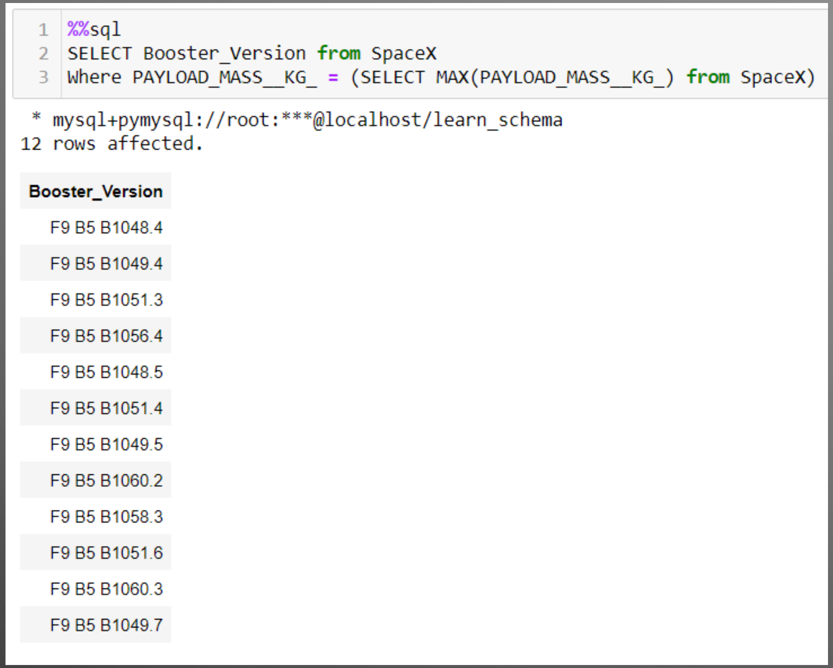
```
 * mysql+pymysql://root:***@localhost/learn_schema
3 rows affected.
```

Mission_Outcome	Count(Mission_Outcome)
Success	99
Failure (in flight)	1
Success (payload status unclear)	1

- We use group by and Count function to return the result

BOOSTERS CARRIED MAXIMUM PAYLOAD

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



```
%%sql
SELECT Booster_Version from SpaceX
Where PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) from SpaceX)
```

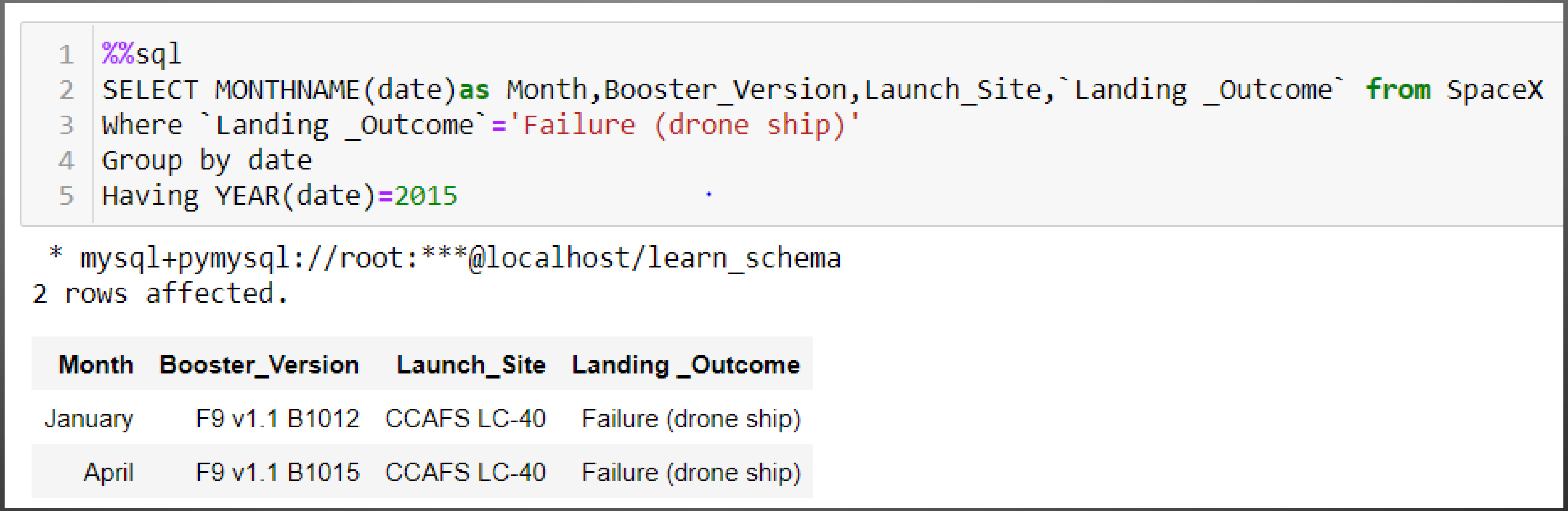
```
 * mysql+pymysql://root:***@localhost/learn_schema
12 rows affected.
```

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

- We use subquery in WHERE clause to return the result.

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



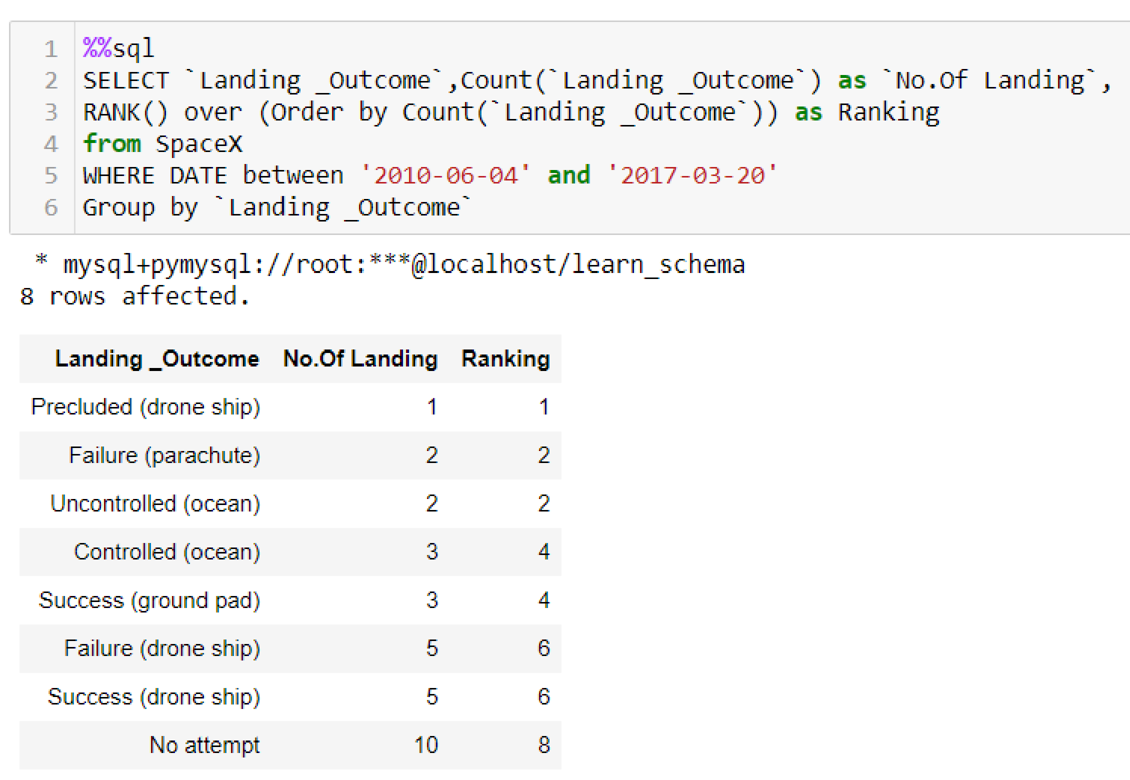
```
%%sql
SELECT MONTHNAME(date)as Month,Booster_Version,Launch_Site,`Landing _Outcome` from SpaceX
Where `Landing _Outcome`='Failure (drone ship)'
Group by date
Having YEAR(date)=2015
```

```
 * mysql+pymysql://root:***@localhost/learn_schema
2 rows affected.
```

Month	Booster_Version	Launch_Site	Landing _Outcome
January	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
April	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

RANK SUCCESS COUNT BETWEEN 2010-06-04 AND 2017-03-20

- Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.
- We use rank() function to rank the count of Landing_Outcome



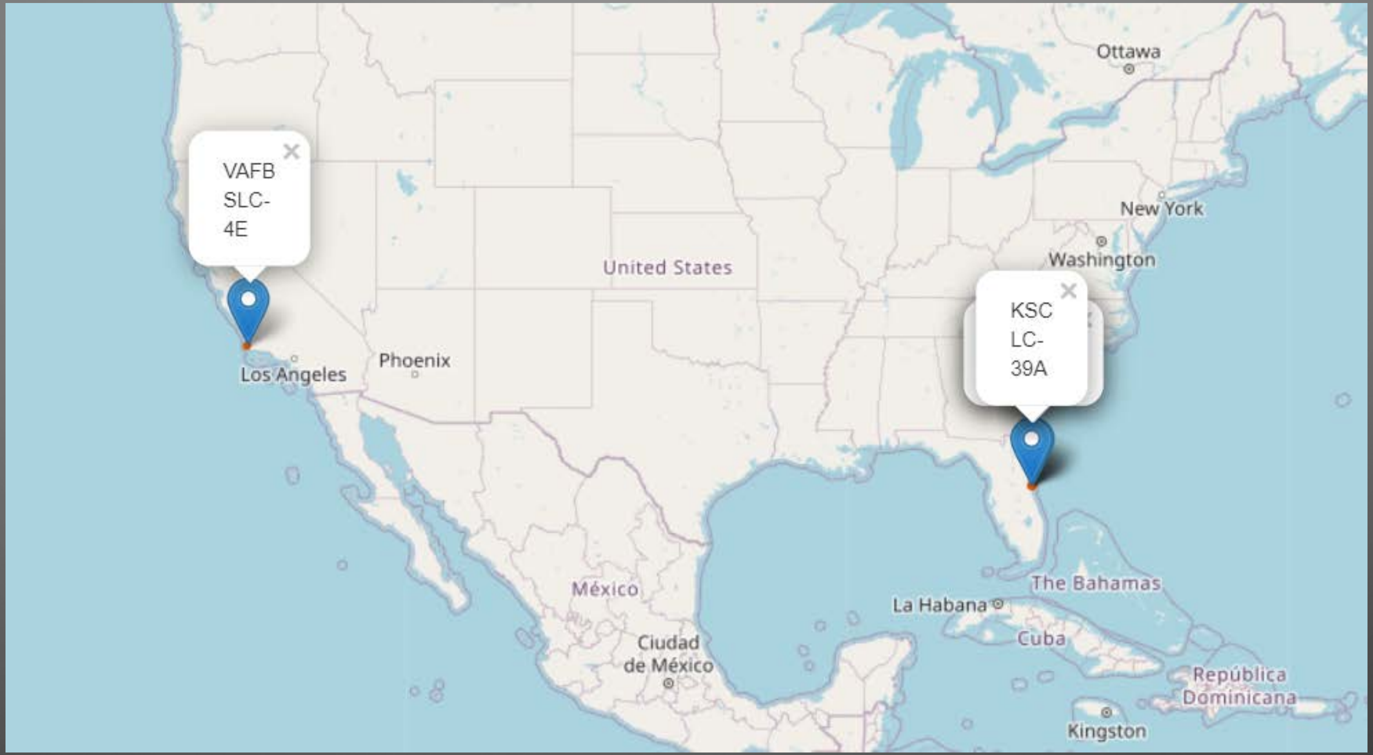
```
%%sql
SELECT `Landing _Outcome`,Count(`Landing _Outcome`) as `No.Of Landing`,
RANK() over (Order by Count(`Landing _Outcome`)) as Ranking
from SpaceX
WHERE DATE between '2010-06-04' and '2017-03-20'
Group by `Landing _Outcome`
```

```
 * mysql+pymysql://root:***@localhost/learn_schema
8 rows affected.
```

Landing _Outcome	No.Of Landing	Ranking
Precluded (drone ship)	1	1
Failure (parachute)	2	2
Uncontrolled (ocean)	2	2
Controlled (ocean)	3	4
Success (ground pad)	3	4
Failure (drone ship)	5	6
Success (drone ship)	5	6
No attempt	10	8

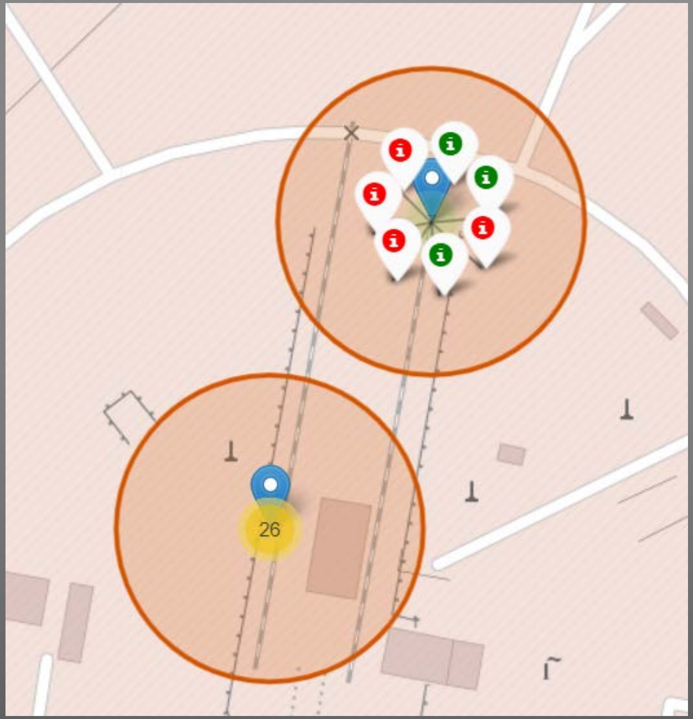
INTERACTIVE MAP WITH FOLIUM

SPACEX LAUNCH SITE



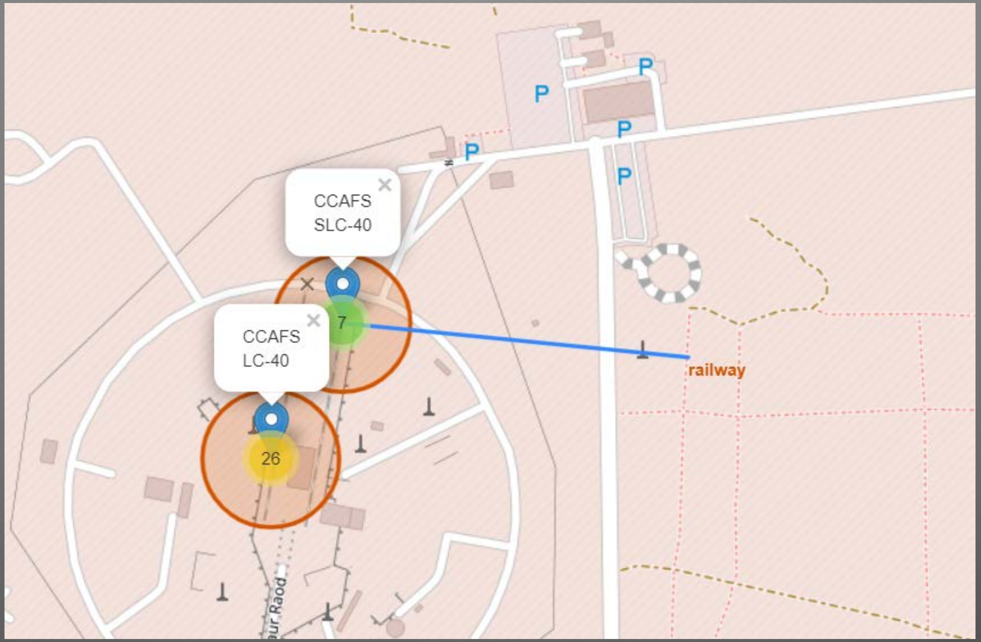
- All locations are chosen near the seashore in order to control the launch in case of failure without impacting residents.

LAUNCH RESULT BY LAUCH SITES



- Most lauch are carried on the east site of US. Each cluster marker in a launch site consist of all failure/success launch mark by color.

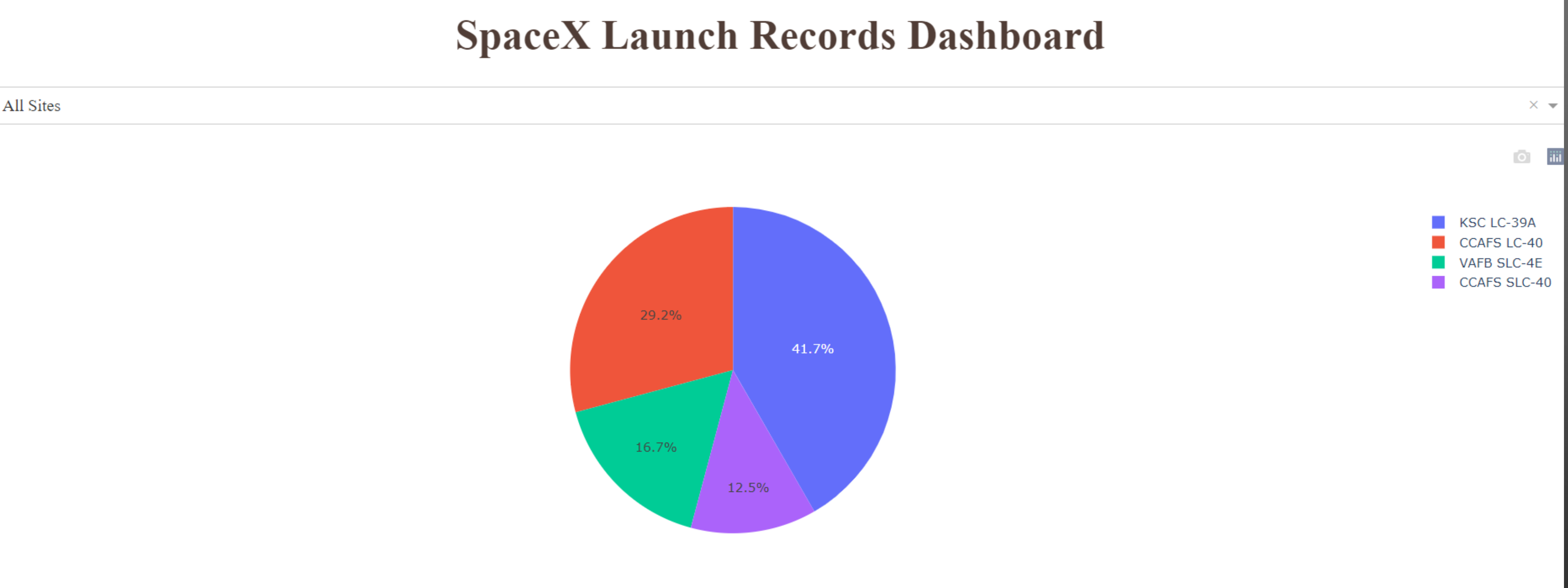
DISTANCE CALCULATION FROM LAUNCH SITES



- By using calculation equation, we can calculate the distance from the launch site to essential locations and populated areas.

BUILD A DASHBOARD WITH PLOTLY DASH

LAUNCH RECORDS BY LAUNCH SITES

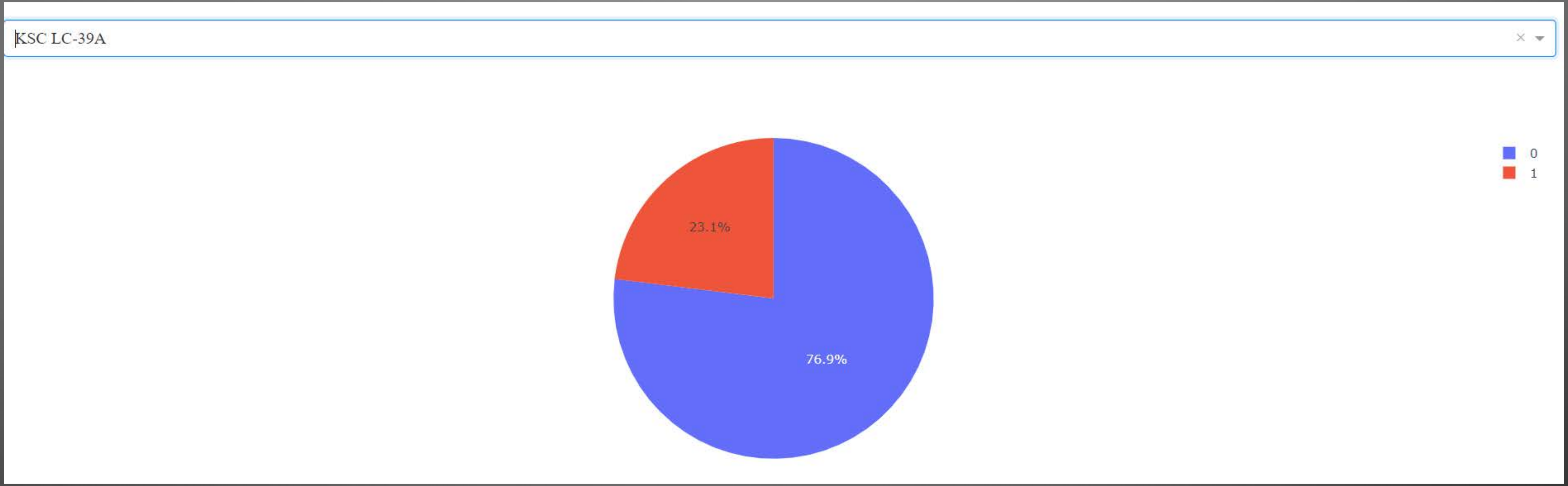


SpaceX Launch Records Dashboard

All Sites

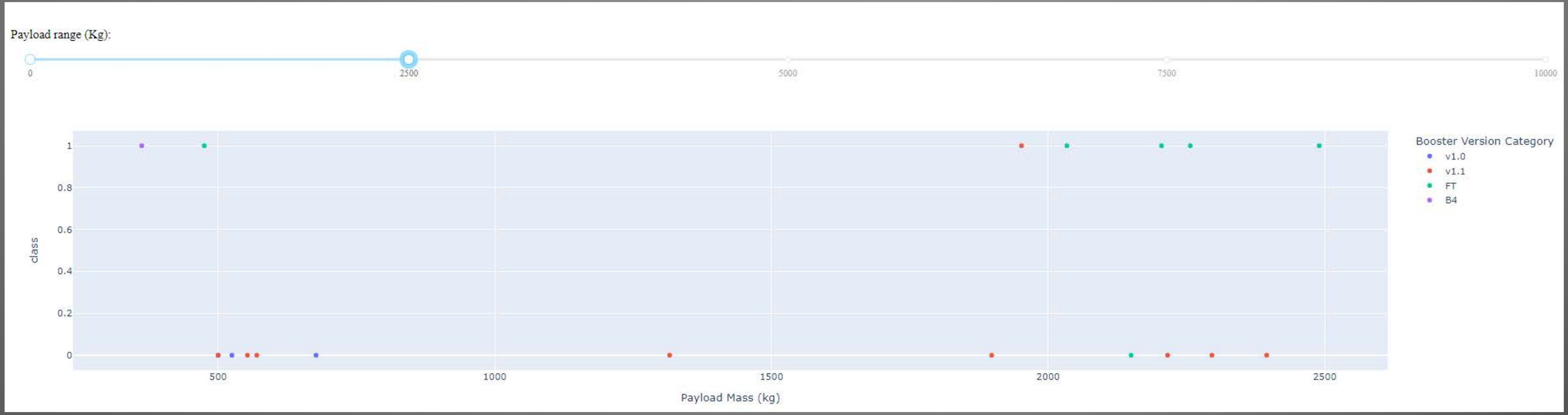
- Most launches are carried out at KSC LC-39A and CCAFS LC-40.

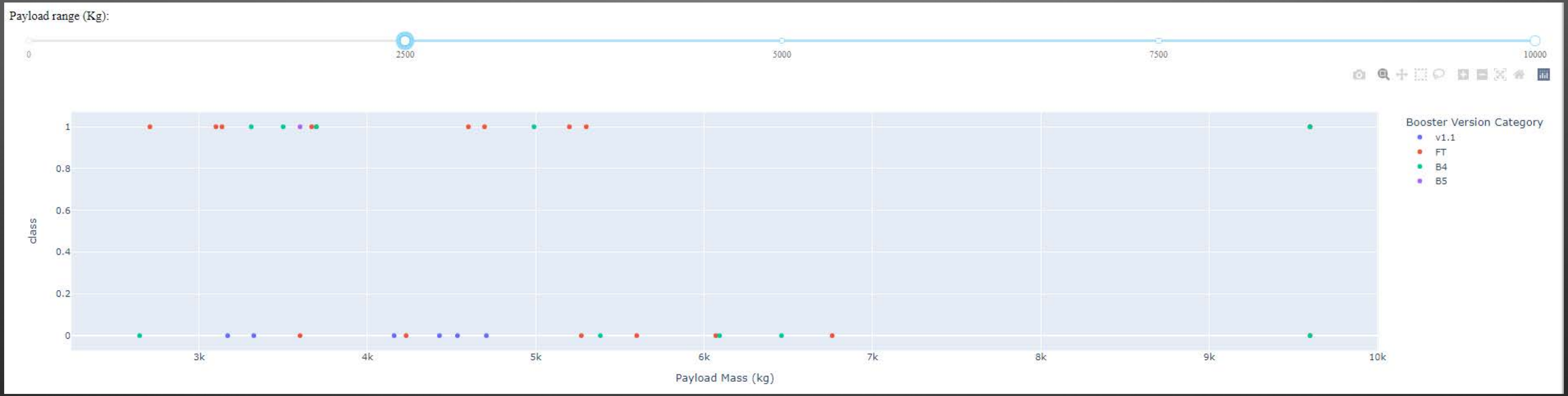
HIGHEST LAUNCH SUCCESS RATE



- After analysis through all launch sites, KSC LC-39A has the highest launch success rate.

PAYLOAD VS. LAUNCH OUTCOME PLOT



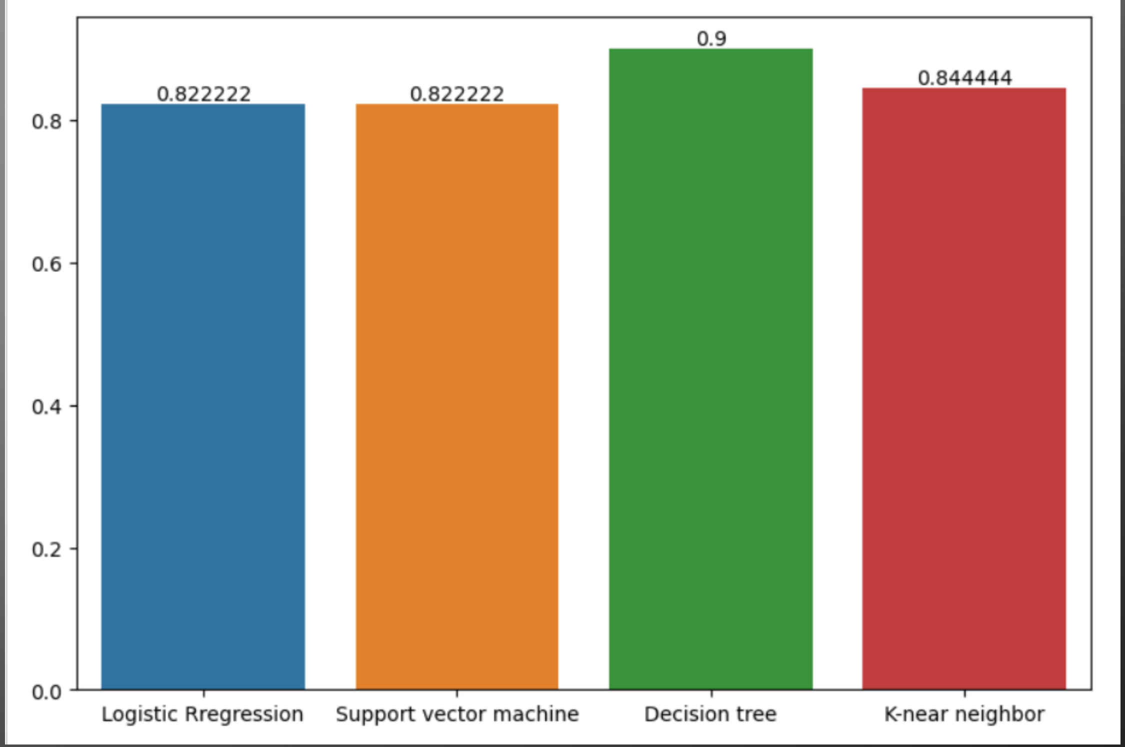


Most success launch cases are payload from 2000 to 6000 kg.

PREDICTIVE ANALYSIS (CLASSIFICATION)

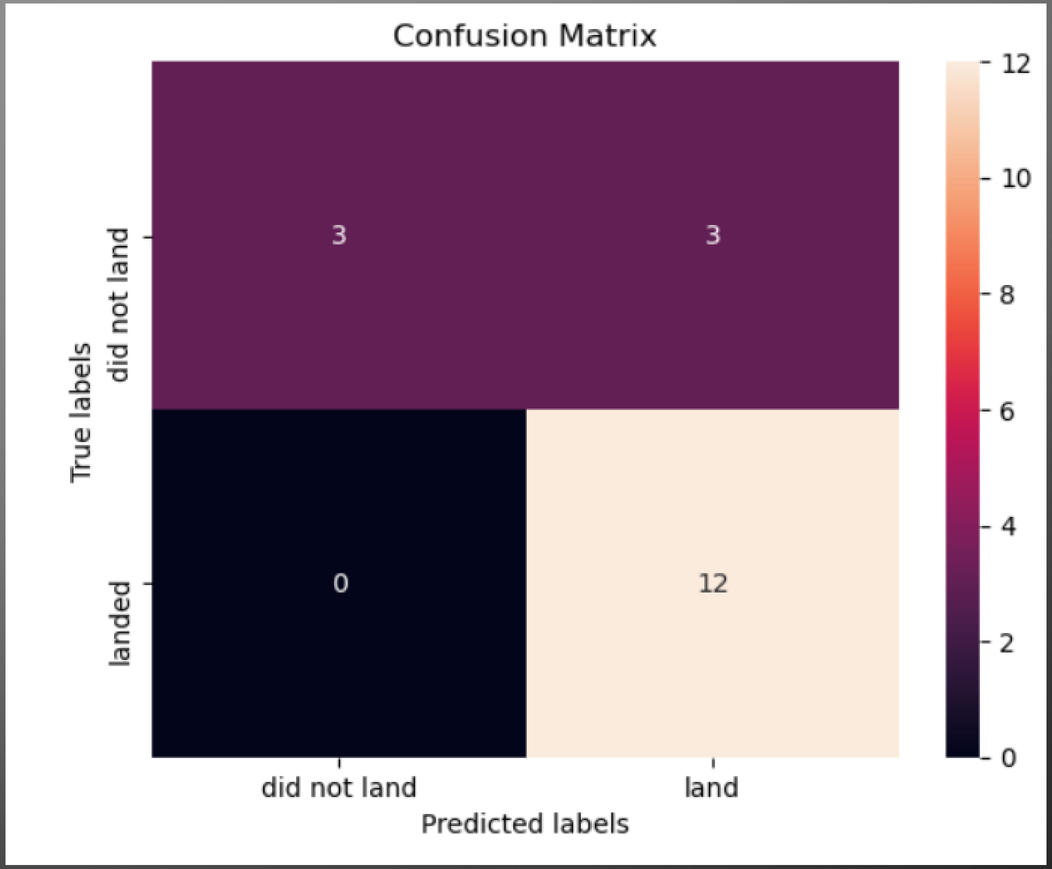
CLASSIFICATION ACCURACY

Decision tree model has the best accuracy (0.9)



CONFUSION MATRIX

The image is the confusion matrix of decision tree classification. As we can see, most problem is with the false positive values.

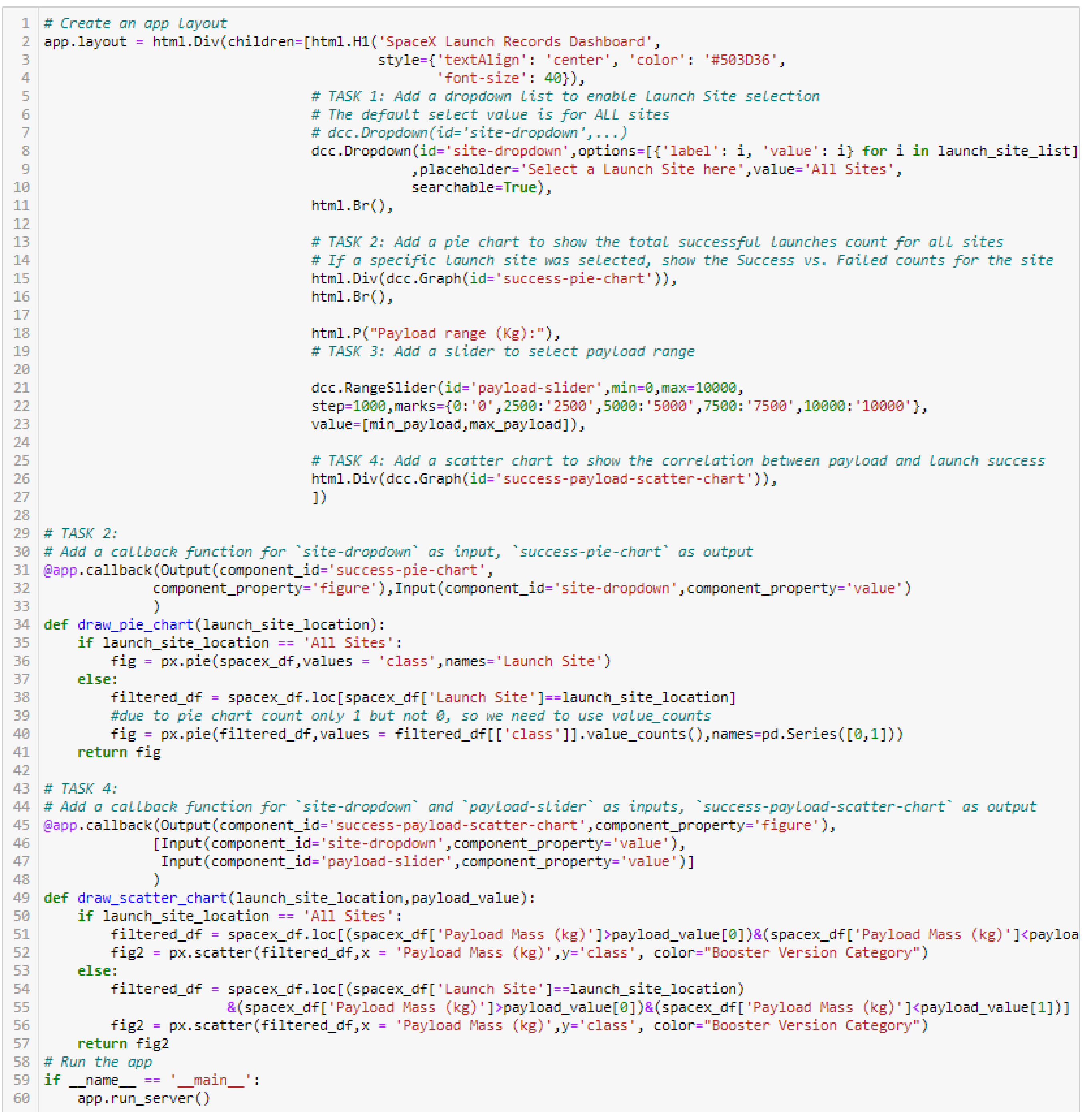


CONCLUSION

- Models with good accuracy were built based on the SpaceX launch database.
- Further components price and calculation will need to be carried out in order to determine the launch price.
- The sample contains roughly about 100 index. More information for other spacecraft company might be needed to gain more accuracy prediction.
- SpaceY company might want to re-assess their strategy based on the findings to find their strong point to compete with SpaceX, not just by using launch database.

APPENDIX

- This is the code that I created for visualization by using Dash.



```python
# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'color': '#503D36',
                                               'font-size': 40}),
                                # TASK 1: Add a dropdown list to enable Launch Site selection
                                # The default select value is for ALL sites
                                # dcc.Dropdown(id='site-dropdown',...)
                                dcc.Dropdown(id='site-dropdown',options=[{'label': i, 'value': i} for i in launch_site_list]
                                            ,placeholder='Select a Launch Site here',value='All Sites',
                                            searchable=True),
                                html.Br(),

                                # TASK 2: Add a pie chart to show the total successful launches count for all sites
                                # If a specific launch site was selected, show the Success vs. Failed counts for the site
                                html.Div(dcc.Graph(id='success-pie-chart')),
                                html.Br(),

                                html.P("Payload range (Kg):"),
                                # TASK 3: Add a slider to select payload range

                                dcc.RangeSlider(id='payload-slider',min=0,max=10000,
                                step=1000,marks={0:'0',2500:'2500',5000:'5000',7500:'7500',10000:'10000'},
                                value=[min_payload,max_payload]),

                                # TASK 4: Add a scatter chart to show the correlation between payload and launch success
                                html.Div(dcc.Graph(id='success-payload-scatter-chart')),
                                ])

# 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 draw_pie_chart(launch_site_location):
    if launch_site_location == 'All Sites':
        fig = px.pie(spacex_df,values = 'class',names='Launch Site')
    else:
        filtered_df = spacex_df.loc[spacex_df['Launch Site']==launch_site_location]
        #due to pie chart count only 1 but not 0, so we need to use value_counts
        fig = px.pie(filtered_df,values = filtered_df[['class']].value_counts(),names=pd.Series([0,1]))
    return fig

# 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 draw_scatter_chart(launch_site_location,payload_value):
    if launch_site_location == 'All Sites':
        filtered_df = spacex_df.loc[(spacex_df['Payload Mass (kg)']>payload_value[0])&(spacex_df['Payload Mass (kg)']<payloa
        fig2 = px.scatter(filtered_df,x = 'Payload Mass (kg)',y='class', color="Booster Version Category")
    else:
        filtered_df = spacex_df.loc[(spacex_df['Launch Site']==launch_site_location)
                      &(spacex_df['Payload Mass (kg)']>payload_value[0])&(spacex_df['Payload Mass (kg)']<payload_value[1])]
        fig2 = px.scatter(filtered_df,x = 'Payload Mass (kg)',y='class', color="Booster Version Category")
    return fig2
# Run the app
if __name__ == '__main__':
    app.run_server()
```